E-MiLi: Energy-Minimizing Idle Listening in Wireless Networks

Xinyu Zhang, *Student Member*, *IEEE*, and Kang G. Shin, *Fellow*, *IEEE*

Abstract—WiFi interface is known to be a primary energy consumer in mobile devices, and idle listening (IL) is the dominant source of energy consumption in WiFi. Most existing protocols, such as the 802.11 power-saving mode (PSM), attempt to reduce the *time* spent in IL by sleep scheduling. However, through an extensive analysis of real-world traffic, we found more than 60 percent of energy is consumed in IL, even with PSM enabled. To remedy this problem, we propose *E*nergy-*M*inimizing *i*dle *Li*stening (`E-MiLi`) that reduces the power consumption in IL, given that the time spent in IL has already been optimized by sleep scheduling. Observing that radio power consumption decreases proportionally to its clock rate, `E-MiLi` adaptively downclocks the radio during IL, and reverts to full clock rate when an incoming packet is detected or a packet has to be transmitted. `E-MiLi` incorporates *sampling rate invariant detection*, ensuring accurate packet detection and address filtering even when the receiver's sampling clock rate is much lower than the signal bandwidth. Further, it employs an opportunistic downclocking mechanism to optimize the efficiency of switching clock rate, based on a simple interface to existing MAC-layer scheduling protocols. We have implemented `E-MiLi` on the USRP software radio platform. Our experimental evaluation shows that `E-MiLi` can detect packets with close to 100 percent accuracy even with downclocking by a factor of 16. When integrated with 802.11, `E-MiLi` can reduce energy consumption by around 44 percent for 92 percent of users in real-world wireless networks.

Index Terms—Energy efficiency, CSMA wireless networks, idle listening, packet detection, adapting clock rate, dynamic frequency scaling.

1 Introduction

Continuing advances of physical-layer technologies have enabled WiFi to support high data rates at low cost and hence become widely deployed in networking infrastructures and mobile devices, such as laptops, smartphones, and tablet PCs. Despite its high performance and inexpensive availability, the energy efficiency of WiFi remains a challenging problem. For instance, WiFi accounts for more than 10 percent of the energy consumption in current laptops [1]. It may also raise a smartphone's power consumption 14 times even without packet transmissions [2].

WiFi's energy inefficiency comes from its intrinsic CSMA mechanism—the radio must perform *idle listening* (IL) continuously, in order to detect unpredictably arriving packets or assess a clear channel. The energy consumption of IL, unfortunately, is comparable to that of active transmission/reception [2], [3]. Even worse, WiFi clients tend to spend a large fraction of time in IL, due to MAC-level contention and network-level delay [4]. Therefore, minimizing the IL's energy consumption is crucial to WiFi's energy efficiency.

A natural way to reduce the IL's energy cost is sleep scheduling. In WiFi's power-saving mode (PSM) and its variants [1], [4], [5], [6], clients can sleep adaptively, and wake up only when they intend to transmit, or expect to receive packets. The AP buffers downlink packets and transmits only after the client wakes up. PSM essentially shapes the traffic by aggregating downlink packets, thereby reducing the receiver's wait time caused by the network-level latency. However, it cannot reduce the IL time associated with carrier sensing and contention. Through an extensive trace-based analysis of real WiFi networks (Section 3), we have found that IL still dominates the clients' energy consumption even with PSM enabled: it accounts for more than 80 percent of energy consumption for clients in a busy network and 60 percent in a relatively idle network.

Since the IL time cannot be reduced any further due to WiFi's CSMA, we exploit an additional dimension—reducing IL power consumption—in order to minimize its energy cost. Ideally, if the exact idle period is known, the radio could be powered off or put to sleep during IL, and wake up and process packets *on demand*. However, due to the distributed nature of CSMA, the idle time between packets varies widely and unpredictably. Underestimation of an idle interval will waste energy, while an overestimation causes the radio to drop all incoming packets during the sleep.

So, one may raise an important question: "is it possible to put the radio in a *subconscious* mode, where it consumes little power and can still respond to incoming packets promptly?" We answer this question by proposing *E*nergy-*M*inimizing *i*dle *Li*stening (`E-MiLi`) that reduces the clock rate of the radio during its IL period. The power consumption of digital devices is known to be proportional to their voltage square and clock rate [7], [8]. Theoretically, by reducing clock rate alone, `E-MiLi` reduces the IL's power consumption linearly.

• *The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, 2260 Hayward St., Ann Arbor, MI 48109-2121.*
E-mail: {xyzhang, kgshin}@eecs.umich.edu.

Manuscript received 12 Nov. 2011; revised 6 Feb. 2012; accepted 19 Apr. 2012; published online 1 May 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org and reference IEEECS Log Number TMCSI-2011-11-0607. Digital Object Identifier no. 10.1109/TMC.2012.112.

It is, however, nontrivial to ensure that packets can be received at a lower clock rate than required. To decode a packet, the receiver's sampling clock rate needs to be at least twice the bandwidth of the transmitted signal, following the Nyquist's Theorem. WiFi radios have already been optimized under this theorem by matching the receiver's clock rate with the Nyquist rate.

E-MiLi meets this challenge via a novel approach called *Sampling Rate Invariant Detection* (SRID). SRID separates the detection from the decoding of a packet. It adds a special preamble to each 802.11 packet, and incorporates a linear-time algorithm that can accurately detect the preamble even if the receiver's clock rate is much lower than the transmitter's. SRID embeds the destination address into the preamble, so that a receiver may only respond to packets destined for it. Upon detecting this special preamble, the receiver immediately switches to the full clock rate and then recovers the packet with a legacy 802.11 decoder.

E-MiLi allows SRID to be integrated into existing MAC or sleeping-scheduling protocols, using a simple *Opportunistic Downclocking* (ODoc) scheme. ODoc enables fine-grained, packet-level power management by adding a downclocked IL (dIL) mode into the radio's state machine. ODoc exploits the burstiness and correlation structure of real traffic to assess the potential benefit of downclocking, and then downclocks the radio only if it is unlikely to incur significant overhead.

We have implemented an E-MiLi prototype on the GNURadio/USRP platform [9]. Our experimental evaluation shows that E-MiLi can detect packets with close to 100 percent accuracy even if the radio operates at $\frac{1}{16}$ of the normal clock rate. Within a normal SNR range (>8 dB), E-MiLi performs comparably to a legacy 802.11 detector. Furthermore, from real traffic traces, we find that for the majority of clients, the overall energy saving with E-MiLi is close to that in pure IL mode with the maximum downclocking factor. According to our measurements, this corresponds to 47.5 percent for a typical WiFi card with a downclocking factor of 4, and 36.3 percent for a software radio with a downclocking factor of 8. Further, our packet-level simulation results show that E-MiLi reduces energy consumption consistently across different traffic patterns, without any noticeable performance degradation.

In summary, this paper makes the following contributions.

- Exploration of the feasibility and cost of fine-grained control of radio clock rate to improve energy efficiency.
- Design of SRID, a novel packet detection algorithm that makes it possible to detect packets even if the receivers are down clocked significantly.
- Introduction of ODoc, a generic approach to integrating SRID with existing MAC- and sleep-scheduling protocols.
- Implementation of E-MiLi on a software radio platform and validation of its performance with real traces and synthetic traffic.

The remainder of this paper is organized as follows: Section 2 analyzes the energy cost of IL in WiFi networks and describes the motivation behind E-MiLi. Section 3 presents a measurement study of the relation between energy consumption and clock rate in WiFi and software radio devices. Following an overview of E-MiLi (Section 4), Sections 5 and 6 present the detailed design of SRID and ODoc, respectively. Section 7 evaluates E-MiLi. Section 9 reviews related work and Section 10 concludes the paper.

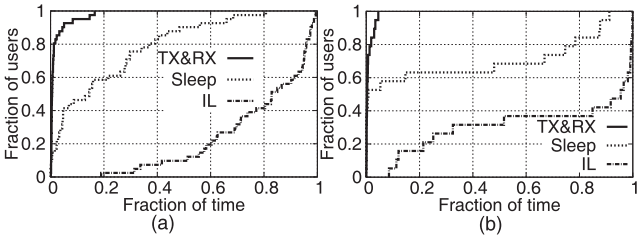


Fig. 1. CDF of the fraction of time spent in different modes for (a) SIGCOMM '08 trace and (b) PDX-Powell trace.

2 Why E-MiLi?

In this section, we motivate E-MiLi by showing a large fraction of time and energy spent in IL for real-world WiFi users. We also briefly discuss the reasons for the high power-consumption of IL by anatomizing a typical radio.

2.1 Cost of Idle Listening

We acquired packet-level WiFi traces from publicly available data sets: SIGCOMM'08 [10] and PDX-Powell [11]. The former was collected from a WLAN used for a conference session that has a peak (average) of 31 (7) clients. The latter was collected from a public hotspot at a university bookstore, with a peak (average) of 7 (3) clients. We built a simulator that can parse the traces and compute each client's sojourn time in different states, including:

- *TX&RX.* The client is transmitting or receiving a packet.
- *Sleep.* The client is put to sleep. A client sets the power-management field in its packet header to 1 if it intends to sleep after the current frame transmission and ACK [5].
- *Idle listening.* A state other than the above two. This includes sensing the channel, waiting for incoming packets, receiving packets not addressed to it, etc. We exclude the SIFS time, which is a short interval ($9 - 20\ \mu$s [5]) between two immediate packets (e.g., in between data/ACK). We also consider a client disconnected if it does not transmit/receive any unicast packets for 5 minutes or longer.

Fig. 1a plots the normalized fraction of time spent in the three modes, distributed among all the clients in the SIGCOMM '08 trace. More than 90 percent of clients enable power management and judiciously put their radios to sleep. However, clients spend most of the time in IL, rather than sleeping: the median IL time is 0.87, and is above 0.6 for more than 80 percent of clients. One may guess the reason for this to be the excessive contention in this busy network. However, even in the PDX-Powell trace (Fig. 1b), the IL time exceeds 0.52 for more than 70 percent of clients. In contrast, the actual TX&RX time is below 0.1 for more than 90 percent of clients in both networks. Since WiFi's PSM cannot eliminate MAC-layer contention and queuing

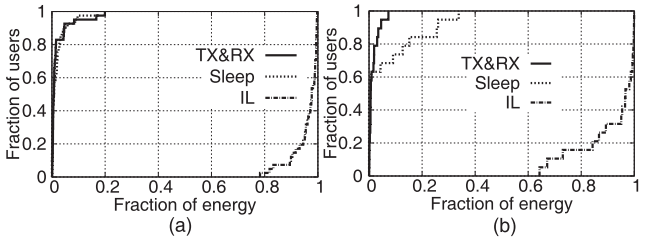


Fig. 2. CDF of the fraction of energy spent in different modes for (a) SIGCOMM 2008 trace and (b) PDX-Powell trace.

delays [6], the IL still dominates the TX&RX time by a significant margin.

We further analyze the energy cost of IL. Since information on the actual type of clients' WiFi cards is unavailable, we assume that their energy profile follows that of a typical Atheros card [12, Section 10.1.5] (TX: 127 mW, RX: 223.2 mW, IL: 219.6 mW, Sleep: 10.8 mW). Although their absolute power consumption differs, many widely used WiFi cards have consistent relative power consumption among different states [13]. Fig. 2a shows that in a busy network, for more than 92 percent of clients, 90 percent of energy is spent in IL, i.e., IL costs nine times more energy than TX&RX for most clients. Moreover, although the sleep time is substantial, the sleep power is negligible, whereas the IL power is comparable to the TX/RX power, so the majority of cost is still with IL. For a network with less contention (Fig. 2b), IL costs less, yet still accounts for more than 73 percent of energy cost for 90 percent of clients. Note that the sleep energy may exceed the TX&RX energy, due to the significant amount of sleep time.

The above evaluation reveals that IL accounts for the majority of a WiFi radio's energy cost, and optimizing the IL time alone using PSM is not enough. If the IL power can be reduced, it will clearly improve the energy efficiency of PSM-like sleep scheduling protocols. In addition, for real-time applications, the constant active mode (CAM) of WiFi is preferable, since PSM may incur an excessive delay and degrade the QoS [2]. By reducing IL power, even CAM can achieve high energy efficiency.

2.2 Why Is Idle Listening So Costly?

Intuitively, a radio should consume less power when it is not actively decoding or transmitting packets, but the IL power of commodity WiFi and other carrier-sensing wireless (e.g., ZigBee) devices is comparable to their TX&RX power [2], [12], [14]. In what follows, we briefly discuss the reason for this by anatomizing the radio hardware.

Fig. 3 illustrates the architecture of a typical WiFi receiver (based on an Atheros 802.11 chip [15]). An incoming signal is first passed through the RF and analog circuit, amplified and converted from RF (e.g., 2.4 GHz) to the baseband by a mixer. The analog baseband signal is sampled by an Analog-to-Digital Converter (ADC), and the resulting discrete samples are passed to the CPU (baseband and MAC processor), which decodes the signal and recovers the original bits in the data frame. The entire radio is driven by a 40 MHz crystal oscillator, which feeds two paths. The first is the frequency synthesizer that generates the center frequency used for the RF and analog mixer. The other is the Phase-Locked-Loop (PLL) that generates the

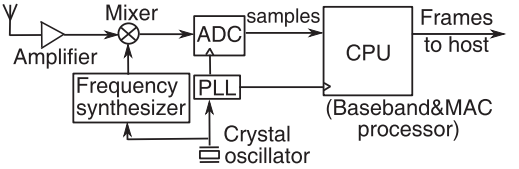


Fig. 3. Architecture of a WiFi receiver.

clocking signal for the digital circuit: the sampling clock for the ADC, as well as the main clock for the CPU.

Existing studies have shown the ADC and CPU to be the most power-hungry components of a receiver. In the Atheros 5001X chipset, for example, they account for 55.3 percent of the entire receiver power budget [16, Table 5]. ADC and CPU power consumptions are also similar (1.04:1 [17]). During IL, both the analog circuits and the ADC operate at full workload as in the receiving mode. Moreover, the decoding load of the CPU is alleviated, but it cannot be put into sleep—it needs to operate at full clock rate in order to perform carrier sensing and packet detection. This is the reason why IL power consumption is comparable to that of receiving packets.

A similar line of reasoning applies to other wireless transceivers such as software radios. In software radios, the ADC feeds the discrete samples to an FPGA, which may further decimate (downsample) the samples and then send them to a general processor that serves as the baseband CPU. The similarity in hardware components implies that software radios are likely to suffer from the same problem with IL. Considering the trend of software radios getting gradually integrated into mobile platforms to reduce the area cost [18], it is imperative to incorporate a mechanism to reduce its IL power.

3 IL Power versus Clock Rate

We propose to reduce the IL power by slowing down the clock that drives the digital circuitry in a radio. Modern digital circuits dissipate power when switching between logic levels, and their power consumption follows $P \propto V_{dd}^2 f$, where V_{dd} is the supply voltage and f the clock rate [7], [8]. Hence, a linear power reduction can be achieved by reducing clock rate. In practice, due to the analog peripherals, the actual reduction is less than ideal. For example, in the ADC used by an Atheros WiFi chip [19], halving the sampling clock rate results in a 31.4 percent power reduction. Here, using detailed measurements, we verify the actual effects of reducing the clock rate for both WiFi NIC and the USRP software radio.

3.1 WiFi Radio

According to IEEE 802.11-2007 [5], the OFDM-based PHY supports 2 downclocked operations with 10 MHz (half-clocked) and 5 MHz (quarter-clocked) sampling rate, in addition to the default full-clocked 20 MHz operation. We test these two modes on the LinkSys WPC55AG NIC (version 1.3, Atheros 5414 chipset), with a development version of Madwifi (trunk-r4132), which supports eight half-clocked and 18 quarter-clocked channels at the 5 GHz band. The downclocked modes can be enabled by activating the "USA with $\frac{1}{2}$ and $\frac{1}{4}$ width channels" regulatory domain on the NIC.

TABLE 1
Mean Power Consumption (in W)
of WiFi under Different Clock Rates

	rate = 1	rate = 1/2	rate = 1/4
Idle	1.22	0.78	0.64
RX	1.66	1.44	0.98
TX	1.71	1.46	1.21

As to measurement of the WiFi's power consumption, our approach is similar to that in [13]. We attach the NIC to a laptop (Dell 5410) powered with an external AC adapter, and use a passive current probe (HP1146A) and voltage probe (HP1160) together with a 1 Gsps oscilloscope (Agilent 54815A) to measure the power draw. The actual power consumption is the difference between the measured power level in different radio modes and the base level with the NIC removed. During the measurement, we tune the WiFi to a channel unused by ambient networks. The IL power is measured when the NIC is activated but not transmitting/receiving packets. The TX/RX power is measured when the WiFi is sending/receiving one-way ping-broadcast packets at the maximum rate (100 packets per second). The different clock modes are configured to use the same bit rate (6 Mbps) and packet size (1 KB). Table 1 shows the measurement results.

It can be seen that the power consumption decreases monotonically with clock rate. In particular, compared to a full-clocked radio, the IL power is reduced by 36 and 47.5 percent for half-clocked and quarter-clocked mode, respectively. The absolute reduction is found different from that reported in an existing measurement study [3]. We guess this discrepancy results from the use of a different WiFi card (Atheros 5212) in their experiment. As validated in [3], different NICs have very different power profiles at different clock rates. To confirm that the power consumption versus clock-rate relation is not limited to the WiFi radio, we have also conducted experiments with the USRP software radio.

3.2 Software Radio

The original USRP is driven by an internal 64 MHz clock, which is used by both the ADC and FPGA. We enabled the external clocking feature by resoldering the main clock circuit, following the instructions in [9]. We use the USRP E100 [9] as an external clock source, which has a programmable clock generator (AD9522) that produces reference clocks below 64 MHz.[1]

We mounted an XCVR2450 daughter board on the USRP, which was then connected to the PC host (a Dell E5410 laptop). The IL mode runs the standard 802.11a/g carrier sensing and packet detection algorithm (see Section 7 for the details of our implementation). The TX mode sends a continuous stream of samples prepended with 802.11 preambles. Since a complete 802.11 decoding module is unavailable, we only measure the IL and TX power. We measure the USRP power directly with the oscilloscope and current/voltage probes, and then add the power consumption of the external clock [20], which is 0.55 W and does not vary with clock rates. Note that the normal clock rate of USRP is 64 MHz, whereas the maximum signal bandwidth sent to the PC is 4 MHz since the FPGA downsamples (decimates) the signals. While reducing the clock rate, we ensure the signal bandwidth is decreased by the same ratio by adjusting the decimation rate.

1. The USRP E100 cannot be tuned to signals below 32 MHz. So, we used a signal generator to produce clock signals below 32 MHz, with the same configuration as those produced by the E100.

TABLE 2
Mean Power Consumption (in W)
of USRP under Different Clock Rates

	rate=1	rate=1/2	rate=1/4	rate=1/8	rate=1/16
IL	10.27	7.96	7.07	6.54	5.88
TX	6.36	5.69	5.18	4.70	4.47

Table 2 shows the measurement results. Similar to a WiFi radio, the USRP power consumption decreases monotonically with clock rate. A power reduction of 22.5 percent (36.3 percent) is achieved for a downclocking factor of 2 (8). We found that at a 4 MHz clock rate (a downclocking factor of 16), the USRP can no longer be tuned to the 2.4 GHz center frequency, but the ADC can still be tuned correctly to 4 MHz sampling rate, and power consumption decreases further.

Since the PC host consumes a negligible amount of power when processing the 4 MHz signal, we have omitted its power consumption in Table 2. Future mobile software radio systems may incorporate dedicated processors to process the baseband signals. By reducing the processors' clock rate in parallel with the ADC and FPGA, the entire software radio platform can achieve higher energy efficiency.

4 An Overview of E-MiLi

`E-MiLi` controls the radio clock rate on a fine-grained, per-packet basis, in order to reduce the energy consumption of IL. It opportunistically downclocks the radio during IL, and then restores it to full clock rate before transmitting or after detecting a packet. Fig. 4 illustrates the flow of core operations when `E-MiLi` receives and transmits packets.

`E-MiLi` prepends to each 802.11 packet an additional preamble, called *M-preamble*. During its IL period, a downclocked receiver continuously senses the channel and looks for the M-preamble, using the SRID algorithm. Upon detecting an M-preamble, the receiver immediately switches back to full clock rate, and calls the legacy 802.11 decoder to recover the packet. The receiver leverages an implicit, PHY-layer addressing mechanism in SRID to filter

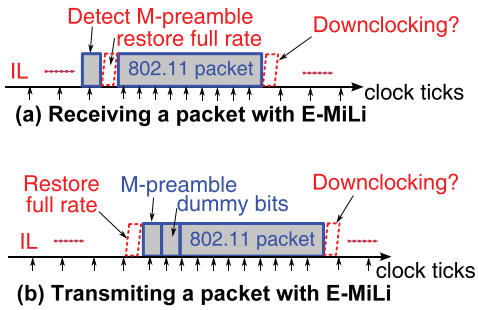


Fig. 4. Idle listening and RX/TX operations in E-MiLi.

the M-preamble intended for other nodes, and hence prevents unnecessary switching of clock rate.

A TX operations follow the legacy 802.11 MAC, except that the carrier sensing is done by SRID. If the radio is downclocked during carrier sensing and backoff, it needs to restore full clock rate before the actual transmission. The exact restoration time is scheduled by another component of E-MiLi, called *Opportunistic Downclocking*.

After completing an RX or TX operation, the radio cannot downclock greedily. As we will verify experimentally in Section 6, switching clock rate takes 9.5 to $151\,\mu s$ for a typical WiFi radio. During the switching, the clock is unstable, and packets cannot be detected even with SRID. To reduce the risk of packet loss, E-MiLi employs ODoc again to make a downclocking decision using a simple outage-prediction algorithm, which estimates if a packet is likely to arrive during the clock-rate switching.

In addition, after sending the M-preamble, a transmitter cannot wait silently during the receiver's switching period; it may otherwise lose the medium access and be preempted by other transmitters. To compensate for the switching gap, the transmitter inserts a sequence of dummy bits between the M-preamble and the 802.11 packet. The dummy bits cover the maximum switching period so that the channel is occupied continuously. Note that the transmitter always sends the M-preamble, dummy bits, and 802.11 packets at the full clock rate. It need not know the current clock rate of the receiver.

When multiple clients coexist, E-MiLi assigns a broadcast address as well as multiple unicast addresses, each with a unique feature. This feature is embedded in the M-preamble and detectable only by the intended receiver. To reduce the overhead of M-preamble, E-MiLi incorporates an optimization framework that allows multiple clients to share addresses at minimum cost.

In summary, E-MiLi always runs at full clock rate to transmit or decode packets, but downclocks the radio during IL to detect implicitly addressed packets, whenever possible. Next, we detail the design of components in E-MiLi.

5 Sample Rate Invariant Detection

To realize E-MiLi, its packet-detection algorithm must overcome the following challenges: 1) it must be resilient to the change of sampling clock rate; 2) it must be able to decode the address information directly at low sampling rates; and 3) due to unpredictable channel condition and node mobility, its decision rule should not be tuned at runtime, and hence must be resilient against the variation of SNR. We propose SRID to meet these challenges via a joint design of preamble construction and detection.

5.1 Construction of the M-Preamble

E-MiLi constructs the M-preamble to facilitate robust, sampling-rate invariant packet detection, while implicitly delivering the address information. An M-preamble comprises $C(C \geq 2)$ duplicated versions of a pseudorandom sequence, as shown in Fig. 5 (where $C = 3$).

Within the M-preamble duration, the channel remains relatively stable, and therefore the duplicated sequences sent by the transmitter maintain strong similarity at the receiver. Hence, a receiver can exploit the strong *self-correlation* between the C consecutive sequences to detect the M-preamble. More importantly, since radios sample signals at a constant rate, the receiver would obtain C similar sequences even if it down samples the M-preamble.

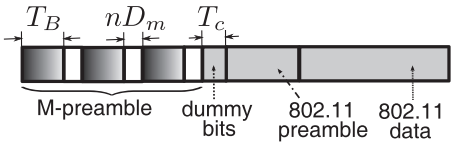


Fig. 5. M-preamble construction and integration with an 802.11 packet.

To enhance resilience to noise, the random sequence in M-preamble must have a strong self-correlation property—it should produce the best correlation output only when correlating with itself. The Gold sequence [21] satisfies this requirement. It outputs a peak magnitude only for perfectly aligned self-correlation, and correlating with any shifted version of itself results in a low, bounded magnitude. For a Gold sequence of length $L = 2^l - 1$ (l is an integer), the ratio between the magnitude of self-correlation peak and the secondary peak is at least $2^{\frac{l-1}{2}}$. The original Gold sequence is binary [21]. To make it amenable for WiFi transceivers, we construct a *complex Gold sequence* (CGS), in which the real and imaginary parts are shifted versions of the same Gold sequence generated by the standard approach [21].

In addition, we use the *length* of the CGS to implicitly convey address information. An address is an integer number n, and corresponds to a CGS of length $(T_B + nD_m)$, where D_m is the maximum downclocking factor of the radio hardware. T_B is the minimum length of the CGS used for the preamble, also referred to as *base length*. To detect its own address (e.g., n), at each sampling point t, the client simply self-correlates the latest T_B samples with the previous T_B samples offset by nD_m. When the client is downclocked by a factor of D, it scales down the base length to $T_B D^{-1}$ and offset to $nD_m D^{-1}$ accordingly. The nD_m value ensures that different addresses are offset by at least 1 sample, even if the CGS is downsampled by the maximum factor D_m.

One challenge related to the Gold sequence is that it only allows length of $L = 2^l - 1$. Hence, not all of the $(T_B + nD_m)$ samples can be exactly matched to a whole Gold sequence. We solve this problem by first generating a long CGS, and then assign the subsequence of length $(T_B + nD_m)$ to the nth address.

Clearly, to meet its design objectives, an ideal random sequence for M-preamble should have strong self-correlation even after it is *downsampled* and *truncated* (since we only use T_B of the $T_B + nD_m$ samples to perform self-correlation). We conjecture there does not exist such a sequence unless the sequence length is very large and the downsampling factor is small. We leave the theoretical investigation of this problem as our future work. In this paper, we will empirically verify that the CGS with a reasonable length suffices to achieve high detection accuracy in practical SNR ranges.

5.2 Detection of the Preamble

We formally derive the detection algorithm in SRID by modeling how the receiver down nsamples the M-preamble and identifies it via self-correlation.

Let $T = C(T_B + nD_m)$ be the total length of the M-preamble (Fig. 5), and $x(t), t \in [0, T)$, the transmitted samples corresponding to the M-preamble. For a full-clocked receiver, the received signals are

$$y_o(t) = e^{2\pi\Delta ft} h(t)x(t) + n(t), t \in [0, T), \tag{1}$$

where $n(t)$ is the noise, $h(t)$ the channel attenuation (a complex scalar representing amplitude and phase distortion), and Δf the frequency offset between the transmitter and the receiver. When a receiver operates at the clock rate of $\frac{1}{D}$ (i.e., with a downclocking factor of D), the received signals become

$$z(k) = e^{2\pi\Delta fkD} h(kD)x(kD) + n(kD), 0 \le k < \left\lfloor \frac{T}{D} \right\rfloor.$$

Here, D must be an integer divisor of the base length T_B of the CGS, i.e., $\lfloor \frac{T_B}{D} \rfloor = \frac{T_B}{D} \triangleq T_1$. To detect M-preamble, at each sampling point k, the receiver with address n performs self-correlation between the latest T_1 samples and the previous T_1 samples offset by nD_mD^{-1}, resulting in

$$R(k) = \sum_{i=k}^{k+T_1-1} z(i)z^*(i - T_1 - nD_mD^{-1}) \tag{2}$$

$$\approx \sum_{i=k}^{k+T_1-1} e^{2\pi\Delta fiD} h(iD)x(iD)\left[e^{2\pi\Delta f(iD-T_B-nD_m)} h(iD - T_B - nD_m)x(iD - T_B - nD_m)\right]^* \tag{3}$$

$$\approx e^{T_B+nD_m}|h(kD)|^2 \sum_{i=k}^{k+T_1-1} |x(iD)|^2, \tag{4}$$

where $(\cdot)^*$ denotes the complex conjugate operator.

Equation (3) is derived based on the fact that the signal level is usually much higher than the noise. Equation (4) is based on the fact that 1) the random sequence $x(t)$ preserves similarity with its predecessor sequence, even though it is downsampled; and 2) the channel remains relatively stable over its *coherence time*, which is much longer than the preamble duration. To see this, we note that the coherence time can be gauged as $T_o = \frac{\lambda}{\sqrt{2}\pi v}$, where λ and v denote the wavelength of the signal and the relative speed between the transmitter and the receiver [22]. At a walking speed of 1 m/s, T_o equals 28.8 milliseconds, whereas the M-preamble duration lasts for tens of microseconds (see Section 5.3.1).

Meanwhile, the energy level of T_1 samples is calculated as

$$E(k) = \sum_{i=k}^{k+T_1-1} |z(i)|^2 \approx |h(kD)|^2 \sum_{i=k}^{k+T_1-1} |x(iD)|^2. \tag{5}$$

From (4) and (5), we get $|R(t)| \approx E(t)$. By contrast, if no M-preamble presents or an M-preamble with a different address a is transmitted, then the self-correlation yields

$$|R(k)| \approx |h(kD)|^2 \left| \sum_{i=k}^{k+T_1-1} x(iD)x(iD - T_B - aD_m)^* \right| \approx 0.$$

This is because the sequence $x(iD), i \in [k, k + T_1 - 1]$ is a truncated CGS and has strong correlation only with itself.

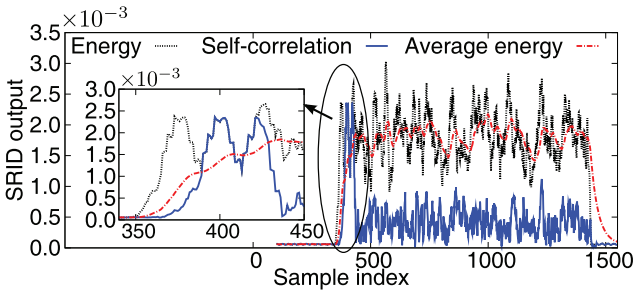


Fig. 6. Detecting M-preamble using SRID (clock rate $= 1/4$).

Fig. 6 shows a snapshot of $|R(t)|$ and $E(t)$ when receiving a packet prepended with M-preamble. $|R(t)|$ aligns almost perfectly with $E(t)$ in an M-preamble, even though the receiver is downclocked. In contrast, $|R(t)|$ differs from $E(t)$ significantly if noise or uncorrelated signals are present.

Based on the above findings, SRID uses the following basic decision rule to determine the presence of an M-preamble:

$$H < |R(k)| \cdot [E(k)]^{-1} < H^{-1}, \tag{6}$$

where H is a threshold such that $H \lessapprox 1$. This decision rule has several key advantages. First, it normalizes the self-correlation with the energy level, so H need not be changed according to the signal strength. We will show experimentally (Section 7) that a fixed value of $H = 0.9$ is robust across a wide range of SNR. Second, it does not require estimation of the channel parameters or calibration of the frequency offset, and hence can be used in dynamic WLANs with user churn and mobility.

For further enhancement of resilience to noise, note that the decision rule (6) is likely to be satisfied at all the sampling points from the second to the Cth CGS (Fig. 5). There are $\frac{(C-1)(T_B+nD_m)}{D} \triangleq T_2$ such points at a downclocking factor D, which can offer high diversity in a noisy or fading environment. To exploit this advantage, at each sampling point k, SRID stores the decision for the past T_2 samples in a FIFO queue, and then apply the following enhanced rule: for $k - T_2 < i \le k$, the number of sampling points satisfying (6) $\ge H_1T_2$, where H_1 is a tolerance threshold and $H_1 \in (0, 1]$.

In addition, during idle periods (i.e., when no signal is present), both the self-correlation and the energy level may be close to 0 and close to each other, and hence the decision rule (6) may be falsely triggered. To prevent such false alarms, we added an *SNR squelch*, which maintains a moving average of incoming signals' energy level, with the window size equal to T_1:

$$E_a(k) = T_1^{-1}E(k) + \left(1 - T_1^{-1}\right)E_a(k - 1). \tag{7}$$

The SNR squelch passes a sampling point to the self-correlator only if its SNR exceeds a threshold H_s, which corresponds to the minimum detectable SNR (set to 4 dB for SRID). Since an idle period (noise floor) usually precedes the M-preamble (with length TD^{-1}) due to the MAC-layer contention, the SNR level can be estimated as

$$SNR = 10 \log_{10} \frac{E_a(t)}{E_a(t - T)}. \tag{8}$$

Algorithm 1 summarizes the detection of M-preamble in SRID. For each time stamp (sampling point), both the self-correlation in (2) and the energy level in (5) can be computed by a single-step operation, which updates the metrics with an incoming signal and subtracts the obsolete signal. Hence, the algorithm has linear complexity with respect to the number of samples, and is well suited for implementation on an actual baseband signal processor.

Algorithm 1. Detecting the M-preamble using SRID.

1. **Input:** new sample $z(k+T_1-1)$ at sampling point $k+T_1-1$
2. **Output:** packet detection decision at sampling point k
3. /*Update energy level of past T_1 samples*/
4. $E(k) \leftarrow E(k-1) + |z(k+T_1-1)|^2 - |z(k-1)|^2$
5. /*Update average energy level*/
6. $E_a(k) \leftarrow T_1^{-1}E(k) + (1-T_1^{-1})E_a(k-1)$
7. /*Update self-correlation with predecessor sequence*/
8. $R(k) \leftarrow R(k-1) + z(k+T_1-1)z(k-nD_mD^{-1}-1)^*$
9. $\quad -z(k-1)z(k-1-T_1-nD_mD^{-1})^*$
10. /*Apply SNR squelch and self-correlation decision*/
11. **if** $10\log_{10}\frac{E_a(k)}{E_a(k-TD^{-1})} > H_s$ && $H < \frac{|R(k)|}{E(k)} < H^{-1}$
12. $\quad$ **then** decisionQ $\leftarrow$ push 1
13. **else** decisionQ $\leftarrow$ push 0
14. **fi**
15. **if** sum(decisionQ) $> H_1 \cdot \frac{(C-1)(T_B+nD_m)}{D}$
16. $\quad$ **then return** 1
17. **fi**
18. **return** 0

5.3 Address Allocation

5.3.1 Minimum-Cost Address Sharing

Since M-preamble uses sequence length to convey address information, the addressing overhead increases linearly with network size. For a network with N nodes, the M-preamble has a maximum length of $C(T_B+ND_m)$. In our implementation, the base length $T_B = 64$, and CGS repetition $C = 3$. For a medium-sized network, say $N = 5$, and a maximum downclocking factor $D_m = 4$, the entire M-preamble would have a length of 252. When transmitted at a 20 MHz sampling rate, the M-preamble only takes $\frac{252}{2\times10^7}s = 12.6\ \mu s$ channel time, which is comparable to the $16\ \mu s$ overhead of the 802.11a/g preamble [5]. However, for a large network, e.g., $N = 50$, the M-preamble overhead increases to $69.6\ \mu s$, which may be overly large, especially for short packets.

To reduce the addressing overhead, `E-MiLi` allows multiple clients to share a limited number of addresses. Address sharing, however, introduces side effects: clients may unnecessarily trigger each other, thus incurring extra energy consumption. `E-MiLi` makes a tradeoff by carefully allocating addresses according to clients' relative channel usage, i.e., the ratio of each client's TX&RX time to the total TX&RX time of the WLAN. The intuition behind this is that a client that transmits/receives packets more frequently should share his address with a fewer number of other clients, so as to minimize the cost of sharing.

We formalize this intuition with an optimization framework. Given the number of clients N, and the maximum address K_m, we seek the optimal address allocation that minimizes the overhead of `E-MiLi`, as follows:

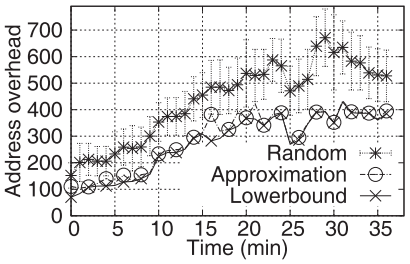


Fig. 7. Performance of address sharing algorithms.

$$\min \quad \sum_{k=1}^{K_m} L_k \left[\left(\sum_{i=1}^{N} p_i u_{ik} \right) \sum_{i=1}^{N} u_{ik} \right] \tag{9}$$

$$s.t. \quad \sum_{k=1}^{K_m} u_{ik} = 1, \quad \forall i \in [1,N], \tag{10}$$

$$u_{ik} \in \{0,1\}, \quad \forall i \in [1,N], \quad \forall k \in [1,K_m], \tag{11}$$

where L_k is the overhead when the address k is used. p_i is client i's relative channel usage, and u_{ik} a binary variable indicating whether or not client i uses address k. Intuitively, the objective function (9) represents the sum of the overhead of each address, weighted by sum of the channel usages of all clients sharing that address and further multiplied by the number of such clients. The multiplication is necessary because a packet with address k triggers all clients with address k. Equation (10) enforces the constraint that each client uses only one address.

This optimization problem is a nonlinear integer program, which is NP-hard in general. In our actual implementation, we approximate the solution by relaxing the integer constraint (11) to $0 \le u_{ik} \le 1$, solving the resulting quadratic optimization program, and then rounding the resulting u_{ik} back to its integer value. To implement the address sharing algorithm, the AP needs to periodically (e.g., every 1 minute) compute the relative channel usage p_i, and then broadcast the new allocation to all clients.

To test the effectiveness of the approximation, we run the address sharing algorithm on the SIGCOMM '08 trace (assuming $K_m = 5$ and $L_k = kD_m$) and plot the total address overhead of `E-MiLi` in Fig. 7. We observe that the integer-rounding-based solution closely approximates the lower bound enforced by the quadratic optimization over $0 \le u_{ik} \le 1$. On average, the approximate solution exceeds the lower bound by only 1.8 percent. Fig. 7 also shows the mean overhead of an algorithm that randomly assigns an address for each client (error bar shows standard deviation over 20 runs). We observe that the approximation algorithm can save more than 50 percent of overhead over the random allocation.

5.3.2 The Broadcast Address

In addition to the address designed for each node, `E-MiLi` assigns a broadcast address known to the AP and all clients. It corresponds to an M-preamble with address $n = 0$. Therefore, each node needs to maintain a self-correlator with offset $nD_m = 0$, in addition to the one with its own address.

For the carrier sensing purpose, a node also needs to identify the existence of packets from other transmitters. Similar to the original 802.11, SRID can perform both energy sensing and preamble detection. The former is achieved by following (7). When downclocked by a factor of D, a node can only sense D^{-1} of the energy compared with a full-clocked receiver. Hence, it reduces the energy detection threshold to D^{-1} of the original. When preamble-based carrier sensing is necessary, it can be realized by prepending an additional broadcast preamble. When this *first preamble* is detected, the node determines the channel to be busy, and continues to track the energy level of the entire packet. However, it will restore full clock rate only when it detects a *second preamble*, which is either addressed to it or is another broadcast preamble.

`E-MiLi` can coexist with 802.11a/g clients even in the preamble detection mode. The 802.11a/g [5] employs self-correlation to detect a short preamble, which corresponds to a random sequence in the frequency domain, and a periodic sequence (period 16, with 10 repetitions) in the time domain. It can be considered as a subset of SRID, with base length $T_B = 16$, sequence repetition $C = 10$, node address 0 and no downclocking, and thus can be easily detected by `E-MiLi` clients. On the other hand, by replacing the first preamble with an 802.11 preamble, `E-MiLi` nodes can be detected by legacy 802.11 as well.

6 Opportunistic Downclocking

We now present the ODoc module, which schedules the downclocking to balance its overhead and maintain compatibility with existing MAC and sleep scheduling protocols. We start by inspecting the overhead in switching clock-rates.

6.1 Delay in Switching Clock Rates

When switching to a new clock rate, the radio needs to be stabilized before transmitting/receiving signals. Since the frequency synthesizer and analog circuit's center frequency remain the same, the time cost mainly comes from stabilizing the digital PLL (driving the ADC and CPU). This is only several microseconds in state-of-the-art WiFi radios. For example, the PLL takes less than $8\ \mu s$ to stablize itself in the MAXIM 2831 RF transceiver [23, p. 17], and the digital circuit (ADC and CPU) needs only $1.5\ \mu s$ to reset [24, Section 7.1], so the total switching time is below $9.5\ \mu s$.

We have also measured the switching delay of the Atheros 5414 NIC. We modified the ath5k driver that can directly access the hardware register and reset the clock rate. After changing the clock-rate register, we repeatedly check a *baseband testing function* until it returns 1 (a conventional way of verifying if the ADC and baseband processor have become ready to receive packets in ath5k), and then record the duration of this procedure.

According to our experimental results, switching between clock-rate 1 and $\frac{1}{4}$ takes 139 to 151 μs, whereas switching between 1 and $\frac{1}{2}$ takes 120 to 128 μs. We note that this is a conservative estimation of the actual switching delay. To switch to a new rate, the Atheros NIC needs to reset not just the PLL, but also all registers for the OFDM decoding and MAC blocks in the CPU, so that the entire receiver chain can run a valid 802.11 mode. In contrast, `E-MiLi` only needs to reset the PLL, while keeping the registers in the CPU intact. In addition, the latency induced by the baseband testing function and its interface to the PC host is unknown, but is included in the switching delay in our measurement.

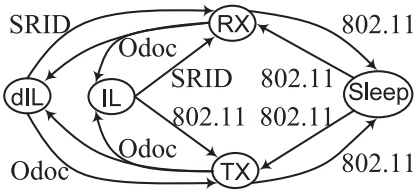


Fig. 8. Radio state-transition when integrating E-MiLi with 802.11.

We will henceforth use the $9.5\ \mu s$ switching delay for the MAXIM 2831 chip as a lower bound, and use the measurement result for Atheros 5414 as an upper bound, although the ODoc module is not restricted to these bounds.

6.2 Scheduling of Downclocking

6.2.1 Control Flow

`E-MiLi` interacts with the WiFi MAC/PHY using a simple interface. On the one hand, WiFi calls `E-MiLi` (the SRID module) to assess the channel availability. On the other hand, `E-MiLi` obtains the radio's state machine from the WiFi MAC and the sleep scheduler. Whenever the radio transits to IL, `E-MiLi` calls its ODoc module to determine whether and when to switch clock rate.

Fig. 8 illustrates the state machine of `E-MiLi`. In dIL mode, the radio runs SRID continuously, and switches to the full-clocked RX mode immediately upon detection of an M-preamble. When there are packets to be transmitted, carrier sensing is performed by SRID, but the MAC schedule strictly follows the 802.11 CSMA/CA algorithm. ODoc continuously queries the 802.11 backoff counter, and reverts the radio to full clock rate when the countdown value of the backoff counter is less than $T_c + SIFS$, where T_c is the maximum switching delay, and *SIFS* is the short interframe space defined in 802.11 [5]. ODoc mandates the radio to perform carrier sensing within this *SIFS* interval after switching to full-clock rate, in order to ensure the channel remains idle after switching. Otherwise, it needs to continue carrier sensing and backoff according to 802.11.

The state-transitions $\mathrm{TX} \leftrightarrow \mathrm{Sleep}$ and $\mathrm{RX} \leftrightarrow \mathrm{Sleep}$ are managed by 802.11 or other sleep-scheduling protocols. Whenever a TX or RX completes and the radio is not put to sleep, ODoc decides whether to switch to dIL or the normal IL mode. It makes this decision using an outage prediction scheme, as detailed next.

6.2.2 Outage Prediction

ODoc's outage prediction mechanism decides if the next packet is likely to arrive before the radio is stabilized to a new clock rate (referred to as an *outage* event). It first checks if there will be a deterministic operation, i.e., an immediate response of the previous operation. For example, CTS, DATA, and ACK packets are all deterministic operations to follow an RTS. Such packets are separated only by an *SIFS*, which is usually shorter than or comparable to the switching time, so the radio must remain at full rate in between.

When a series of deterministic operations end, ODoc checks if an outage occurred recently. It maintains a binary history for each nondeterministic packet arrival, with "1" representing that the interpacket interval is shorter than T_c, and "0" otherwise. It asserts that an outage is likely to occur and remains at full clock rate, if the recent history contains a "1." The key intuition lies in the burstiness of WiFi traffic—a short interval implies an ongoing transmission of certain data, and is likely to continue multiple short intervals until the transmission completes.

An important parameter in ODoc is the size of history. A large history size may predict an outage when it does not occur, thus missing an opportunity of saving energy by downclocking. On the other hand, a small history size results in frequent misdetection of packets arriving within T_c. Fortunately, *a misdetection causes only one more retransmission*, because a missed packet will be detected in its next retransmission, when the receiver has already been stabilized. Therefore, a small history size is always preferred when energy efficiency is of high priority. As will be clarified in our experimental study, a history size of between 1 and 10 is sufficient to balance the tradeoff between false prediction and misdetection.

7 Evaluation

In this section, we present a detailed experimental evaluation of `E-MiLi`. Our experiments center around two questions: 1) How accurate can `E-MiLi` detect packets in a real wireless environment, and with different downclocking rates? 2) How much of energy can `E-MiLi` save for real-world WiFi devices and at what cost?

To answer these questions, we have implemented `E-MiLi` on software radios and network-level simulators as follows:

- We have implemented the SRID algorithm, including the M-preamble construction and detection, on the GNURadio platform and verify it on a USRP testbed. As a performance benchmark, we have also implemented the 802.11 OFDM preamble encoding/detection algorithm (Section 5.3.2).
- `E-MiLi`'s energy efficiency depends on the relative time of IL, which, in turn, depends on network delay and contention, and hence, we leverage real WiFi traces again to evaluate the energy efficiency of `E-MiLi`. We implemented the ODoc framework and address allocation algorithm by extending the trace-based simulator (Section 3), and then integrating results from the SRID experiments.
- We have also implemented ODoc in ns-2.34, which can be used to verify the performance of `E-MiLi` with synthetic traffic patterns (e.g., HTTP and FTP) independently.

7.1 Packet-Detection Performance

We test the detection performance of SRID under different SNR levels and downclocking factors. The SNR is estimated as $SNR = \frac{E_s - E_N}{E_N}$, where E_s is the average energy level of incoming samples when a packet is present, and E_N is the noise floor, both smoothed using a moving average with the window size equal to the length of the M-preamble. Note

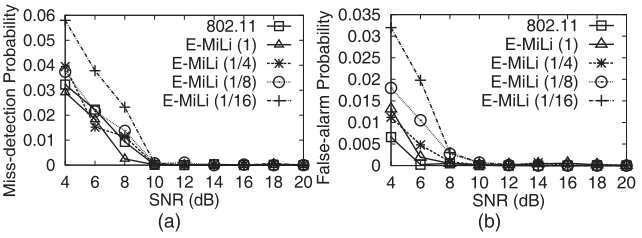


Fig. 9. SRID performance for a single link.

that this SNR value overestimates the actual SNR experienced by the *decoder*, since the decoding modules will raise the noise level by around 3.5 dB [12]. Given that 802.11 needs at least 9.7 dB SNR to decode packets [17], SRID must be able to detect packets accurately above 9.7 dB SNR.

We set the base length of SRID's CGS to $T_B = 64$, and maximum downclocking factor $D_m = 16$. We fix the self-correlation threshold $H = 0.9$, and the tolerance threshold $H_1 = 0.6$ (Section 5). We will show that these thresholds are robust across different experiment settings.

7.1.1 Single Link

We first test SRID on a single link consisting of two USRP nodes within Line-of-Sight (LOS). We downclock the receiver by different factors, and vary the link's SNR by adjusting the transmit power and link length/distance. Since the USRP fails to work when the external clock is downclocked to $\frac{1}{16}$, we scale its FPGA decimation rate by 16, which is equivalent to downsampling the signals by a factor of 16. Under each SNR/clock-rate setting, the transmitter sends 10^6 packets at full clock rate with constant interarrival time. The misdetection probability (P_m) is calculated by the fraction of time stamps where a packet is expected to arrive but fails to be detected, and vice versa, for the false-alarm probability (P_f).

Fig. 9 plots P_m and P_f as a function of a link's time-averaged SNR (rounded to integer values). P_m drops sharply as SNR increases, and approaches 0 as SNR grows above 8 dB. It tends to be higher under a high downclocking factor, mainly because fewer sampling points are available that satisfy the decision rule (6) and thus, SRID is more susceptible to noise. When $\mathrm{SNR} = 4\,\mathrm{dB}$ and $D = 16$, P_m grows up to 6 percent. Under practical SNR ranges (above 9.7 dB), however, P_m is consistently below 1 percent for all the clock rates. In addition, SRID shows a comparable detection performance with 802.11. In fact, it may have lower P_m when the down-clocking factor D is below 16. This is because SRID uses a longer self-correlation sequence than 802.11 (64 versus 16), which increases its robustness to noise. The false-alarm probability P_f in Fig. 9b shows a trend similar to P_m.

Recall SRID uses nD_m, the spacing between repetitive CGS to convey address n. A natural question is: how large can n be to ensure a high detection accuracy? Fig. 10 plots the detection performance as n increases. For a stationary link, both P_m and P_f remain relatively stable. This is because even for the address $n = 100$, two self-correlation sequences are separated by 1,600 samples, corresponding to $400\ \mu$s at the 4 MHz signal bandwidth of USRP, which is well below the channel's coherence time. For a mobile client

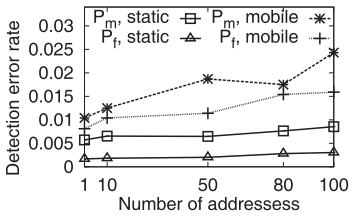


Fig. 10. Detection performance versus the number of unique addresses.

(created by moving the USRP receiver around the transmitter at walking speed), the detection performance is only slightly affected by the address length, since the low mobility causes SNR variations, but does not change the coherence time significantly.

7.1.2 Testbed

We proceed to evaluate SRID on a testbed consisting of nine USRP2 nodes (one AP and eight clients) deployed in a laboratory environment with metal/wood shelves and glass walls. Fig. 11 shows a map of the node locations. Node D is moving between point D and E at walking speed, and all others are stationary. This testbed enables the evaluation of SRID in a real wireless environment subject to effects of multipath fading, mobility, and NLOS obstruction. More importantly, it allows testing the false-alarm rate due to cross correlation between different node addresses.

Due to the limited number of external clocks, we create the effect of downclocking by changing the USRP2's decimation rate, so that the receiver's sampling rate becomes 1 to $\frac{1}{16}$ of the transmitter's. We allow the AP to send 10^6 packets to each client in sequence. Fig. 12a shows that, depending on node locations, P_m varies greatly. In general, nodes farther away (e.g., H) or obstructed by walls (e.g., F) from the AP has higher P_m. The mobile node D may have higher P_m than a node farther from the AP but is stationary (e.g., node E). Consistent with the single link experiment, the downclocking factor 4 results in comparable P_m with 802.11.

Fig. 12b shows the false-alarm probability due to cross correlation, i.e., the probability that a client detects packets addressed to others. The relative P_f for different clients shows a similar trend as P_m, depending on the location and mobility. Unlike the single link case, the P_f tends to be larger than P_m, because the cross correlation between sequences has stronger effects on P_f than pure noise. Remarkably, even for the worst link and with $D = 16$, P_f is below 0.04, implying negligible energy cost due to false triggering. We note that for 802.11, the address field must be decoded from the packet, so P_f here is not meaningful for it.

From the above experiments, we observe that SRID has close to 100 percent detection accuracy (and is comparable to

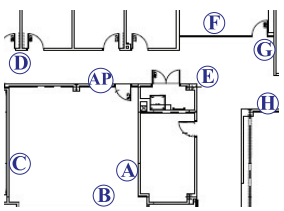


Fig. 11. Network topology for evaluating SRID in a testbed.

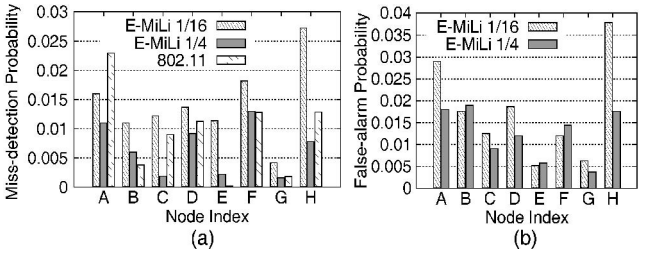


Fig. 12. SRID performance in a USRP testbed.

802.11) under practical SNR ranges and with downclocking rate up to 16. Hence, it can be used to realize `E-MiLi` in practical wireless networks.

7.2 Improving WiFi Energy Efficiency

7.2.1 Real WiFi Traffic

We now evaluate `E-MiLi`'s energy efficiency through trace-based simulation. We obtain WiFi and USRP power-consumption statistics from actual measurements (Section 3). We use the 151 μs switching time of the Atheros AR5414 NIC as the worst case estimate of switching delay, assuming the power consumption during clock switching is the same as in full-clocked mode. As we will clarify, an outage due to the switching delay occurs with a less than 4.2 percent probability, so we assume an outage event does not affect the WiFi traces except causing one retransmission. In addition, we adopt the P_m and P_f values at 8 dB as a conservative estimation of the packet loss or false alarm caused by SRID. Unless mentioned otherwise, 15 addresses are allocated and shared among all clients, and a history size of 5 is used in ODoc.

Energy savings. Fig. 13a illustrates the energy saving of `E-MiLi`, assuming clients are using WiFi devices with a maximum downclocking factor of 4. For a large network (SIGCOMM '08 traces [10]), the energy saving ranges from 41 to 47.3 percent. Its CDF is densely concentrated—for around 92 percent of clients, the energy saving ranges between 44 and 47.2 percent, which is close to the 47.5 percent energy saving when a client remains in downclocked IL mode (Section 3). In a small network (PDX-Powell traces [11]) with less contention, IL induces less energy cost, so the energy-saving ratio of `E-MiLi` is relatively low. However, since IL time still dominates, the median saving remains around 44 percent, and minimum 37.2 percent. Fig. 13b plots the results assuming clients' power consumption is the same as the USRP device with a maximum downclocking factor of 8. Again, the energy saving is concentrated near 36.3 percent, the saving in pure IL mode (Section 3).

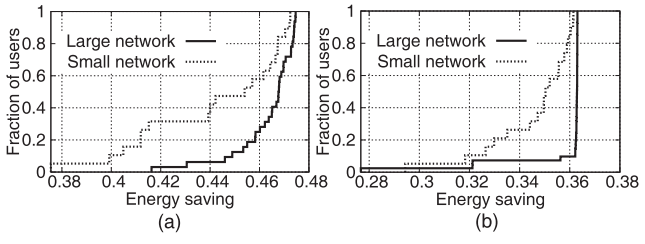


Fig. 13. Energy saving ratio for (a) WiFi, maximum downclocking factor of 4; (b) USRP, maximum downclocking factor of 8.

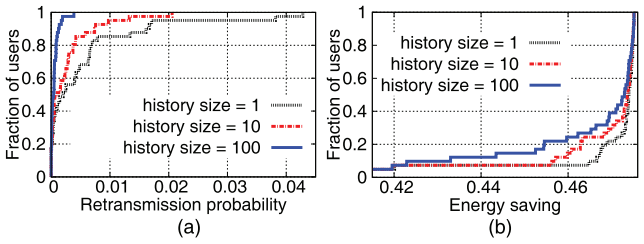


Fig. 14. Effects of history size (SIGCOMM '08 trace).

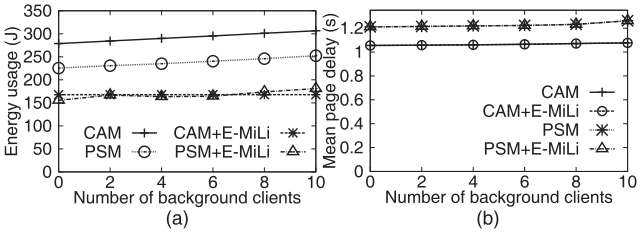


Fig. 15. Performance of a 5-minute web-browsing session.

These experiments reveal that E-MiLi can explore the majority of IL intervals to perform downclocking. Its energy-saving ratio can be roughly estimated as $\eta = \eta_c P_{IL}$, where η_c is the energy-savings ratio in pure IL mode using the maximum downclocking factor, and P_{IL} the percentage of idle listening energy during a radio's lifetime. Since P_{IL} is close to 1 for most clients, η is close to η_c.

Overhead of E-MiLi and effect of ODoc. The overhead of E-MiLi comes from misdetection (and retransmission) due to a packet arriving in between the switching time. Such events can be alleviated by ODoc's history-based outage prediction mechanism. In this experiment, we evaluate the cost of such outage and the effectiveness of ODoc in alleviating it. Fig. 14a shows that when history size equals 1, 4.2 percent packets may need to be retransmitted for some clients. With a history size of 10, retransmission is reduced to below 0.8 percent for 90 percent of clients. A further increase of the history size to 100 shows only a marginal improvement. On the other hand, Fig. 14b shows a small history size results in higher energy efficiency, implying that the energy savings from aggressive downclocking dwarfs the small waste due to retransmissions. Hence, a small history size is preferable for ODoc if energy efficiency is of high priority.

7.2.2 Synthetic Traffic Patterns

To further understand E-MiLi's benefits and cost under controllable network conditions, we implement and test it in ns-2.34. We compare performance of the legacy WiFi (including both CAM and PSM), and E-MiLi-enhanced WiFi (referred to as CAM + E-MiLi and PSM + E-MiLi). We modified the PHY/MAC parameters of ns-2 to be consistent with that in 802.11g, and fix the data rate to 6 Mbps. We implement the ODoc based on 802.11, and configure it in a similar manner to the trace-driven simulator. The PSM module builds on the 802.11 PSM extension to ns-2 [25], and the power consumption statistics follow our measurement of AR5414 (Section 3). We evaluate two applications: web browsing and FTP, which have different performance constraints.

Web browsing. We simulate a web-browsing application using the PackMIME http traffic generator in ns-2, which provides realistic stochastic models of HTTP flows. The network consists of one HTTP server connecting to a WLAN AP via an ADSL2 link, with 1.5 Mbps (0.5 Mbps) downlink (uplink) bandwidth and exponentially distributed delay with mean 15 ms. The AP serves one HTTP client (with mean page request interval of 30 s) and multiple background clients. Similar to [6], we study the effect of background traffic by running fixed-rate (200 Kbps, 512-byte packet size) UDP file transfer between the AP and the background clients.

Fig. 15a shows the energy usage of a 5-minute web-browsing session. PSM shows around 18 percent energy saving over CAM. CAM+E-MiLi saves 39.8 percent of energy over CAM without background traffic, and 47.1 percent when the number of background clients grows to 10. Since PSM optimizes the sleep schedule of clients, the ratio of IL time is less, compared to CAM, and thus PSM + E-MiLi achieves less energy saving (33 to 37.1 percent) than CAM + E-MiLi. Also, note that E-MiLi is relatively insensitive to background traffic, as it can enforce address filtering even at low clock rate.

Fig. 15b plots the average per-page delay during the web-browsing session. Clearly, E-MiLi incurs a negligible delay when integrated into legacy WiFi. Although the M-preamble and clock switching costs channel time, it is much shorter than the network and contention delay. Notably, PSM incurs a longer delay than CAM due to its sleep scheduling mechanism, and CAM + E-MiLi has a shorter delay, yet higher energy efficiency than PSM. We expect an even better energy-delay tradeoff to be achieved by jointly designing the PSM sleep scheduling algorithm and E-MiLi. We leave such an optimization as our future work.

FTP. We proceed to evaluate E-MiLi using the FTP traffic generator in ns-2, assuming a client downloads a 20 MB file (with packet size 1 KB) directly from the AP. Compared to the fixed-duration web browsing, the FTP's energy usage is more sensitive to the background traffic (Fig. 16a), because the downloading duration is prolonged by MAC-layer contention. PSM is found to consume 36.8 to 39.4 percent more energy than CAM, due to the fact that it may result in higher energy-per-bit than CAM [1]. In addition, although E-MiLi achieves a similar level of energy saving as in the web browsing, it may degrade the FTP throughput by up to 4.4 percent in the absence of background traffic (Fig. 16b). This is due mainly to its overhead, i.e., the switching delay, the extra channel time of the M-preamble, and the imperfect detector and outage predictor that incur MAC-layer retransmissions. Moreover, note that we assume no end-to-end delay and the throughput depends only on MAC contention, which zooms in the overhead from E-MiLi.

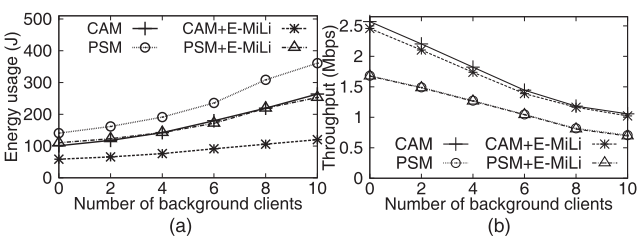


Fig. 16. Performance when downloading a 20 MB file using FTP.

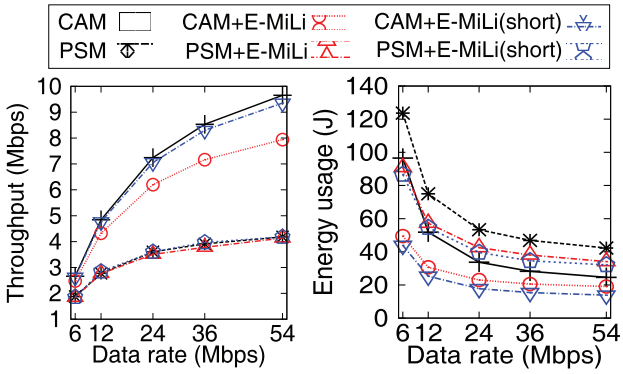


Fig. 17. FTP performance when data rate varies. "E-MiLi(short)" denotes E-MiLi with a short switching time ($9.5\ \mu s$).

Overhead in high-rate applications. One caveat to E-MiLi is that the duration of the M-preamble and the switching delay are fixed, whereas the channel time for transmission of useful data decreases as the data rate increases. The overhead of E-MiLi will thus be amplified at a high data rate. We illustrate this effect by varying the PHY-layer data rate for an FTP file transfer with the number of contending clients fixed at 6. Other parameter settings are the same as in the FTP experiments above. The corresponding M-preamble takes $10.1\ \mu s$ and the switching delay is again configured to $151\ \mu s$. Fig. 17 shows that as the data rate increases, CAM + E-MiLi causes CAM more throughput degradation, and the amount of energy saving decreases due to the longer time in transferring the data. When the data rate reaches 54 Mbps, CAM + E-MiLi degrades the throughput of CAM by 17.6 percent (the actual throughput is much lower than 54 Mbps due to inherent overhead and collision induced by TCP when running over 802.11 [26]), while saving 23.1 percent of energy. However, when taking advantage of the short switching delay of recent WiFi chipset (e.g., $9.5\ \mu s$ with MAXIM 2831), the throughput degradation is negligible, and the energy saving ratio is consistently around 40 percent for all data rates. In addition, E-MiLi sees no throughput degradation when integrated with PSM, and the resulting energy saving is kept around 30 percent.

Noted that the effect of fixed preamble overhead is an inherent problem of high data-rate 802.11 protocols, and can be resolved by standard solutions such as the packet aggregation in 802.11n. Further, the effects of overhead of E-MiLi becomes less severe in a busy network, where contention is high and the channel time consumed by preamble and switching overhead becomes negligible compared to the contention delay. In addition, throughput is a critical metric only for rate-intensive applications like FTP. Mobile wireless devices are more likely to be dominated by elastic traffic such as VoIP and HTTP. Such traffic patterns tend to incur a significant amount of idle listening time which overwhelms the channel time consumed by E-MiLi's preamble and switching delay. As already exemplified in our web-\textbackslash browsing experiments, they can make substantial energy saving by using E-MiLi.

8 Discussion

Scalability to MIMO radios. The overhead of E-MiLi is fixed even if the NIC were equipped with a MIMO transceiver. The overhead of E-MiLi mainly comes from the preamble and the clock switching delay. For MIMO systems such as 802.11n, all the RF chains of a receiver detect a single preamble embedded in each packet, and then uses different preambles for channel estimation. Similarly, when using E-MiLi, they can share the same M-preamble for packet detection. In addition, the clock switching delay depends on the PLL settling time of each RF chain. Modern MIMO transceivers may either allow the RF chains to share the same PLL [27], or equip each RF chain with a separate PLL [28]. In the former case, the switching delay is fixed and shared among all RF chains. In the latter case, the settling time of all RF chains is similar and can overlap with each other.

In summary, neither the preamble overhead nor the switching delay increases with the number of antennas in a MIMO system. Therefore, E-MiLi works for modern MIMO NICs without introducing any extra overhead compared to the case of SISO NICs.

Enabling virtual carrier sensing.[2] An E-MiLi receiver employs SRID to detect packets intended for itself, and is able to carrier sense other packets via energy detection. However, energy sensing alone may not be enough to address a pathological case, i.e., the hidden terminal problem. In IEEE 802.11, virtual carrier sensing is an optional solution, which requires an RTS/CTS handshake before the actual data transmission. The RTS/CTS packet piggy-backs a duration of the forthcoming data packet. Neighboring transmitters overhear the RTS/CTS and extend the channel's busy time by the corresponding duration.

In E-MiLi, virtual carrier sensing can be simply realized as follows: A transmitter/receiver prepends RTS/CTS with the broadcast preamble, so that all neighboring nodes can detect the RTS/CTS, restore full-clock rate and decode the duration field using a legacy 802.11 decoder. Then, as in the 802.11 virtual carrier sensing mechanism, if the forth-coming data packet is not intended for it, a node will enter the sleep mode and remain there throughout the packet duration. Since the data packets' duration is usually much longer than the RTS/CTS, the energy consumption in decoding RTS/CTS is dominated by the energy savings with sleep, and the savings in IL energy remain the same. Hence, with this simple mechanism, E-MiLi will retain its advantages over legacy approaches with virtual carrier sensing enabled.

Note that the RTS/CTS-based virtual carrier sensing is necessary only when a hidden terminal is a critical problem. It has been shown that a hidden terminal rarely occurs in WiFi networks when the rate adaptation is enabled [29] and even rarer under a light traffic load. Most WiFi devices disable the virtual carrier sensing by default, so as to save the overhead caused by RTS/CTS. Hence, E-MiLi without RTS/CTS is more preferable in practice.

Association process. When E-MiLi coexists with legacy WiFi, the AP needs to discriminate them and prepend the M-preamble only for packets destined for E-MiLi-capable clients. The discrimination should be initialized during the association process, when a newly joining E-MiLi client notifies the AP about its capability, and subsequently the AP runs the address allocation algorithm to assign an

2. This issue was brought up by Sunghyun Choi during the ACM MobiCom 2011, Las Vegas, Nevada.

address to it (and possibly reassign addresses to existing E-MiLi clients using the address allocation algorithm).

9 Related Work

Energy-efficient protocols for WiFi. Energy efficiency has long been a paramount concern for portable WiFi devices. Many MAC-level scheduling protocols have been proposed to reduce the energy wasted by IL. For example, NAPman [6] carefully isolates PSM clients' traffic using an energy-aware fair scheduler, so as to reduce unnecessary IL caused by background traffic. SleepWell [30] further isolates the traffic from different WLAN cells, by scheduling their wake-up time in a distributed TDMA manner. μPM [4] adopts a more fine-grained scheduler that aggressively puts clients to sleep even in between short packet intervals. E-MiLi can be integrated with these and other MAC-level energy-saving solutions, by adding the downclocked IL mode into their state machine (Section 6.2). E-MiLi can also work in CAM, thus overcoming the excessive delay typically seen in PSM-style protocols.

An alternative way of reducing the cost of IL is to wake up the receiver *on demand*. The wake-on-wireless scheme [31] augments a secondary low-power radio for packet detection, and triggers the primary receiver only when a new packet arrives. E-MiLi also adopts the philosophy of on-demand packet processing. Its energy saving may be less than wake-on-wireless, because it needs to keep the analog circuit active in IL. Its advantage is that no extra radio is required. In fact, it only requires a change of firmware to support the construction and detection of M-preamble, and adjustment of clock rate. E-MiLi can also be used with wake-on-wireless to optimize the power consumption of the secondary radio.

Low-power listening (LPL) in sensor networks. In sensor networks, a popular MAC-layer energy saving mechanism is LPL, which is used by S-MAC [32], B-MAC [33] and many derivatives. Since sensor networks typically run low-rate, small duty-cycle applications, LPL shifts more power consumption to the transmitter side, thus reducing the time spent in idle listening. Specifically, a receiver periodically wakes up to detect packets from the transmitter, and the transmitter uses a long preamble that spans that period to ensure detectability. Similar to the WiFi's PSM, LPL is a sleep scheduling mechanism that reduces the IL time, and can be enhanced by integrating with E-MiLi. For example, since E-MiLi reduces IL power, it can shorten the receiver's wake-up period, thereby shortening the transmitter's preamble length and lowering its power consumption.

Packet detection. The general idea of correlation-based packet detection is not new. As mentioned in Section 5.3.2, the 802.11 OFDM PHY incorporates a preamble that allows self-correlation-based detection. Its variants have also been used in other software-radio implementations [34]. In E-MiLi, we have designed a new preamble mechanism that preserves the self-correlation property even when it is downsampled. Cross-correlation-based packet detection (i.e., correlating the incoming signal with a known sequence) is an alternative way of detecting packets [35], [36], but cannot detect downsampled signals and is more susceptible to the frequency offset.

Dynamic voltage-frequency scaling (DVFS). DVFS is a mature technology used in microprocessor design [7]. It exploits the variance in processor load, lowering the voltage and clock rate when few tasks are pending, and raising it when the processor is heavily loaded. It has also been proposed for Gigabit wireline links [37], and for audio signal processing [8]. The key idea is to observe the peak frequency of the incoming workload, and then limit the processor's clock rate to that level.

DVFS has not been used for improving the energy efficiency for wireless radios, due mainly to a well-known paradox: the radio should be activated only after detecting a packet, but to detect the packet, the radio must always be active at its full sampling rate. We overcome this paradox by separating packet detection and decoding, and performing both at different rates. Our approach is partly inspired by the experiments by Chandra et al. [3], who found WiFi NIC's power consumption to scale linearly with the sampling bandwidth, and proposed the SampleWidth algorithm to adjust the bandwidth according to the traffic load. SampleWidth uses the same clock rate for detection and decoding, and can only adjust clock rate at a coarse-grained level, because the transmitter and the receiver must agree on the same clock rate before packet transmissions.

10 Conclusion

We have presented E-MiLi, a novel mechanism for reducing the energy cost of IL that dominates the energy consumption in WiFi networks. Our goal was to exercise fine-grained IL power control by adjusting clock rate without compromising packet-detection capability. We met this goal by devising a sampling-rate invariant packet detector, which enables a downclocked radio to detect packets with accuracy comparable to that of a full-clocked radio. We have also introduced an opportunistic down-clocking scheme to balance the overhead in changing clock rate and minimize its negative influence on network performance. Our experimental evaluation and trace-based simulation confirm the feasibility and effectiveness of E-MiLi in real WiFi networks with different traffic patterns.

E-MiLi has wider implications for wireless design than what we have explored in this paper. Its simple MAC/PHY interface facilitates its integration with other carrier sensing-based wireless networks, such as ZigBee sensor networks. In addition, we only explored the benefits of downclocking in E-MiLi due to hardware limitation. By changing the voltage along with clock rate, additional energy savings can be achieved. This is a matter of our future inquiry.

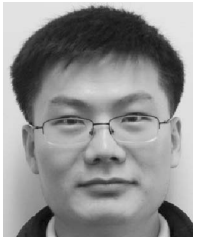
Acknowledgments

The work reported in this paper was supported in part by the US Defense Advanced Research Projects Agency (DARPA) under Grant HR0011-10-1-0081. The main ideas of E-MiLi are part of a pending US patent.

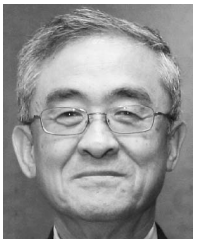
References

[1] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "802.11 Power-Saving Mode for Mobile Computing in Wi-Fi Hotspots: Limitations, Enhancements and Open Issues," *Wireless Networks*, vol. 14, no. 6, pp. 745-768, 2008.

[2] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless Wakeups Revisited: Energy Management for VoIP over Wi-Fi Smartphones," *Proc. ACM MobiSys,* 2007.
[3] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A Case for Adapting Channel Width in Wireless Networks," *Proc. ACM SIGCOMM,* 2008.
[4] J. Liu and L. Zhong, "Micro Power Management of Active 802.11 Interfaces," *Proc. ACM MobiSys,* 2008.
[5] *IEEE Std. 802.11, Wireless LAN Medium Access Control and Physical Layer Specifications,* IEEE, 2007.
[6] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, "NAPman: Network-Assisted Power Management for WiFi Devices," *Proc. ACM MobiSys,* 2010.
[7] K. Flautner, S. Reinhardt, and T. Mudge, "Automatic Performance Setting for Dynamic Voltage Scaling," *Proc. ACM MobiCom,* 2001.
[8] W.R. Dieter, S. Datta, and W.K. Kai, "Power Reduction by Varying Sampling Rate," *Proc. ACM/IEEE Int'l Symp. Low Power Electronics and Design (ISLPED '05),* 2005.
[9] Ettus Research, "Universal Software Radio Peripheral (USRP)," http://www.ettus.com, 2012.
[10] A. Schulman, D. Levin, and N. Spring, "CRAWDAD Data Set umd/sigcomm2008," 2008.
[11] C. Phillips and S. Singh, "CRAWDAD Data Set pdx/vwave," 2007.
[12] Atheros Comm., "AR5213 Preliminary Datasheet," 2004.
[13] Atheros Comm., "Power Consumption and Energy Efficiency of WLAN Products," 2004.
[14] TI, Inc., "CC2420 Preliminary Datasheet," 2004.
[15] M. Zargari et al., "A Dual-Band CMOS MIMO Radio SoC for IEEE 802.11n Wireless LAN," *IEEE J. Solid-State Circuits,* vol. 43, no. 12, pp. 2882-2895, Dec. 2008.
[16] B. McFarland, A. Shor, and A. Tabatabaei, "A 2.4 & 5 GHz Dual Band 802.11 WLAN Supporting Data Rates to 108 Mb/s," *Proc. 24th Ann. Technical Digest Gallium Arsenide Integrated Circuit (GaAs IC) Symp.,* 2002.
[17] J.T. Bevan et al., "An Integrated 802.11a Baseband and MAC Processor," *Proc. IEEE Int'l Solid-State Circuits Conf. (ISSCC) Digest of Technical Papers,* 2002.
[18] P. Dutta, Y.-S. Kuo, A. Ledeczi, T. Schmid, and P. Volgyesi, "Putting the Software Radio on a Low-Calorie Diet," *Proc. Ninth ACM SIGCOMM Workshop Hot Topics in Networks (HotNets '10),* 2010.
[19] A. Tabatabaei, K. Onodera, M. Zargari, H. Samavati, and D. Su, "A Dual Channel ΣΔ ADC with 40MHz Aggregate Signal Bandwidth," *Proc. IEEE Int'l Solid-State Circuits Conf. (ISSCC) Digest of Technical Papers,* 2003.
[20] Analog Devices, "AD9522 Data Sheet," 2008.
[21] P. Fan and M. Darnell, *Sequence Design for Communications Application.* Research Studies, 1996.
[22] J.K. Cavers, *Mobile Channel Characteristics.* Kluwer Academic, 2000.
[23] Maxim, "MAX2831/MAX2832 2.4 GHz to 2.5 GHz 802.11g/b RF Transceivers," 2010.
[24] Atheros Comm., "AR9280 Datasheet," 2009.
[25] M. Fujinami and T. Murakami, "PSM Extension for ns-2," http://nspme.sourceforge.net/index.html, 2012.
[26] S. Gopal, S. Paul, and D. Raychaudhuri, "Investigation of the TCP Simultaneous-Send Problem in 802.11 Wireless Local Area Networks," *Proc. IEEE Int'l Conf. Comm. (ICC),* 2005.
[27] S. Sankaran, M. Zargari, L. Nathawad, H. Samavati, S. Mehta, A. Kheirkhahi, P. Chen, K. Gong, B. Vakili-Amini, J. Hwang, S.-W. Chen, M. Terrovitis, B. Kaczynski, S. Limotyrakis, M. Mack, H. Gan, M. Lee, R. Chang, H. Dogan, S. Abdollahi-Alibeik, B. Baytekin, K. Onodera, S. Mendis, A. Chang, Y. Rajavi, S.-M. Jen, D. Su, and B. Wooley, "Design and Implementation of a CMOS 802.11n SoC," *IEEE Comm. Magazine,* vol. 47, no. 4, pp. 134 -143, Apr. 2009.
[28] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E.W. Knightly, "WARP: A Flexible Platform for Clean-Slate Wireless Medium Access Protocol Design," *SIGMOBILE Mobile Computing and Comm. Rev.,* vol. 12, pp. 56-58, 2008.
[29] M.Z. Brodsky and R.T. Morris, "In Defense of Wireless Carrier Sense," *Proc. ACM SIGCOMM,* 2009.
[30] J. Manweiler and R.R. Choudhury, "Avoiding the Rush Hours: WiFi Energy Management via Traffic Isolation," *Proc. ACM MobiSys,* 2011.
[31] E. Shih, P. Bahl, and M.J. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices," *Proc. ACM MobiCom,* 2002.
[32] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2002.
[33] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. ACM Second Int'l Conf. Embedded Networked Sensor Systems (SenSys),* 2004.
[34] K. Jamieson and H. Balakrishnan, "PPR: Partial Packet Recovery for Wireless Networks," *Proc. ACM SIGCOMM,* 2007.
[35] S. Sen, R.R. Choudhury, and S. Nelakuditi, "CSMA/CN: Carrier Sense Multiple Access with Collision Notification," *Proc. ACM MobiCom,* 2010.
[36] S. Sen, R.R. Choudhury, and B. Radunovic, "PHY-Assisted Energy Management for Mobile Devices," *Proc. ACM MobiSys Poster Session,* 2010.
[37] L. Shang, L.-S. Peh, and N.K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," *Proc. IEEE Int'l Symp. High Performance Computer Architecture (HPCA),* 2003.



Xinyu Zhang received the BE degree in 2005 from the Harbin Institute of Technology, China, and the MS degree in 2007 from the University of Toronto, Canada. He is currently working toward the PhD degree in the Department of Electrical Engineering and Computer Science, University of Michigan. His research interests are in the MAC/PHY codesign of wireless networks, with applications in WLANs, WPANs, mesh networks, and white-space networks. He was a recipient of the Best Paper Award at ACM MobiCom 2011. He is a student member of the IEEE.



Kang G. Shin is the Kevin & Nancy O'Connor professor of computer science in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. His current research focuses on computing systems and networks as well as on embedded real-time and cyber-physical systems, all with an emphasis on timeliness, security, and dependability. He has supervised the completion of 70 PhD degrees and authored/coauthored more than 770 technical articles (more than 270 of these are published in archival journals), one textbook, and more than 20 patents or invention disclosures. He has also received numerous awards, including Best Paper Awards from the 2011 ACM International Conference on Mobile Computing and Networking (MobiCom 2011), the 2011 IEEE International Conference on Autonomic Computing, the 2010 and 2000 USENIX Annual Technical Conferences, the 2003 IEEE Communications Society William R. Bennett Prize Paper Award, and the 1987 Outstanding *IEEE Transactions of Automatic Control* Paper Award. He received the Research Excellence Award in 1989, the Outstanding Achievement Award in 1999, the Distinguished Faculty Achievement Award in 2001, and the Stephen Attwood Award in 2004 from The University of Michigan (the highest honor bestowed to Michigan Engineering faculty), a Distinguished Alumni Award of the College of Engineering, Seoul National University, in 2002, the 2003 IEEE RTC Technical Achievement Award, and the 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers). He has chaired several major conferences, including ACM MobiCom 2009, IEEE SECON 2008, ACM/USENIX MobiSys 2005, IEEE RTAS 2000, and IEEE RTSS 1987. He has served on several editorial boards, including the *IEEE Transations on Parallel and Distributed Systems* and *ACM Transactions on Embedded Systems*. He has also served or is serving on numerous government committees, such as the US National Science Foundation Cyber-Physical Systems Executive Committee and the Korean Government R&D Strategy Advisory Committee. He has also cofounded a couple of startups. He is a fellow of the IEEE and ACM.