Esense: Energy Sensing-Based Cross-Technology Communication

Kameswari Chebrolu and Ashutosh Dhekne

Abstract—In this paper, we present Esense, a new paradigm of communication between devices that have fundamentally different physical layers. Esense is based on sensing and interpreting energy profiles. While our ideas are generic enough to be applicable in a variety of contexts, we illustrate the usefulness of our ideas by presenting novel solutions to existing problems in three distinct research domains. As part of these solutions, we demonstrate the ability to communicate between devices that follow two different standards: IEEE 802.11 and 802.15.4. We consider two scenarios here: 1) where there is no background traffic and 2) where there is background 802.11 traffic. In each case, we build an "alphabet set": a set of signature packet sizes that can be used for Esense communication. Specifically for the second case, we take a measurement-based alphabet set construction by considering WiFi traces from actual deployments. Based on practical observations and experiments, we theoretically quantify the maximum achievable transmission rate when using Esense. With background traffic, we could potentially construct an alphabet of size as high as 100. Such a large alphabet size promises efficient Esense communication. We show via a prototype implementation that effective communication is indeed feasible even when both sides use different physical layers.

Index Terms—Cross-technology communication, energy sensing

1 Introduction

A variety of standards have evolved to facilitate communication in different settings. A consequence of this is that in the wireless domain, quite a few standards have evolved in the same spectrum. For example, the Industry, Scientific and Medical (ISM) band supports a variety of standards such as IEEE 802.11b/g [1], IEEE 802.15.4 [2], Bluetooth [3], and cordless phones. The devices that implement these standards have very different physical layers (modulation, frequency band) and cannot interpret the bits that constitute a packet generated by the other standards. But because they operate in the same spectrum, they do interfere with each other.

In this paper, we present a new paradigm of communication between devices that have fundamentally different physical layers. This same paradigm can also be used for communication between devices belonging to the same standard, which are out of communication range but within carrier-sense range. We term our scheme *Esense* because it is based on energy sensing. Our scheme can enable communication in any setting as long as the two end points can sense each others energy. We illustrate the applicability of this framework by proposing novel solutions to existing problems in three distinct research domains, as described below. The three problems and solution approaches are also illustrated in Fig. 1, and summarized in Table 1. A detailed related work comparison is in Section 2:

- *Coordinated Coexistence*. With more standards emerging for operation within limited spectrum, coexistence between the standards becomes a very important issue. Current practices are mainly uncoordinated. The problem can be much more effectively solved by explicit communication, and coordination between the different entities operating in the same spectrum. They can then potentially share the spectrum in a time multiplexed fashion. For example, a WLAN access point (AP) can coordinate with nearby WPAN nodes operating within the same spectrum, by scheduling the latter's transmissions as part of the AP's contention-free period.
- *Energy Management*. WiFi enabled devices such as smartphones, laptops, access points, and router boards often employ a variety of techniques to conserve battery power. These techniques include enabling the PSM mode of 802.11, disabling the WiFi interface or turning the entire device off. In the past, some solutions [4], [5] have considered the use of a separate low-power secondary-radio to wake up the primary WiFi interface. In these cases, the communication conveying the wake-up message happens using the standard followed by the secondary radio. This results in range-mismatch, intermediate infrastructure support, and so on. These problems can be overcome easily if the secondary low-power radio can interpret WiFi communication, and then turn its associated WiFi node on only when required.
- *Network Management*. Debugging performance anomalies in deployed WiFi networks (infrastructure or long-distance community networks) needs a good interference map of the network. Such a map specifies which node's transmissions interfere with which other nodes. An interference map can help plan the networks better through proper transmit

- *K. Chebrolu is with the Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Powai, Mumbai 400076, India. E-mail: chebrolu@cse.iitb.ac.in.*
- *A. Dhekne is with Securifi Embedded Systems, Madhapur, Hyderabad 500081, India. E-mail: ashudhekne@gmail.com.*

Manuscript received 8 Dec. 2011; revised 26 Apr. 2012; accepted 23 Aug. 2012; published online 12 Sept. 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-12-0663. Digital Object Identifier no. 10.1109/TMC.2012.195.

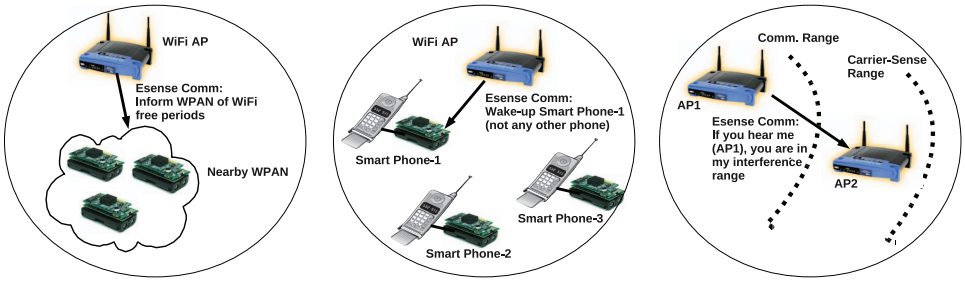


Fig. 1. Esense application in three example scenarios.

power or channel allocation. However, interference maps are hard to construct since often the interference range exceeds the transmission range. A node may be able to sense interference but is often clueless as to who is responsible for it because it cannot interpret the incoming packets. Solutions in the past [6] have relied on time-synchronized transmissions and network downtime to create such a map. The Esense framework of communication can avoid these limitations and help in these settings by clearly identifying the nodes responsible for this interference, by embedding their ids into energy profiles.

Our Esense framework achieves its task of communication by constructing an *alphabet set* of packet sizes. An Esense message (sequence of bits) is mapped to an alphabet and a packet corresponding to that size is transmitted (at a predetermined rate). This results in an appropriate energy burst duration, which the receiver measures and maps it to the appropriate alphabet and, hence, corresponding bits. We consider two settings here, one where there is no background traffic competing with Esense message exchange. This happens, for example, in the coordinated coexistence case. Since we are coordinating channel access, we can ensure that when the Esense messages are exchanged, no one else transmits. In the second case, we consider competing background traffic. This happens in energy management and network management, where Esense communication has to happen alongside with regular 802.11 transmissions. In this case, one has to tackle the following challenge. How does one distinguish these message modulated energy bursts (henceforth referred to as energy-sense or Esense packets), from other packets that are exchanged (henceforth referred to as regular packets). The mechanism to make this distinction is dependent on the communication patterns of the original system. We explore this issue in depth.

As a proof of concept of the Esense framework, we enable unidirectional communication from an IEEE 802.11 radio to an 802.15.4 radio. However, we emphasize that Esense is equally applicable to enable bidirectional communication as well as communication across other wireless standards. The 802.11 to 802.15.4 communication we illustrate is directly applicable in the first two of the three scenarios presented above: the communication is unidirectional and originates from the WiFi domain, with the receiver being an 802.15.4 radio. In the third scenario, the receiver is an 802.11 radio, but here too we could use an 802.15.4 radio to overcome practical limitations in off-the-shelf 802.11 radios (off-the-shelf WiFi radios do not export the carrier-sense interface readily).

We take a measurement-based approach to enable communication between an 802.11 radio and 802.15.4 radio. We first characterize the accuracy of the 802.15.4 radio in detecting channel occupancy. Commodity 802.15.4 hardware, such as the CC2420 chip platform we use, have limitations on the resolution of energy detection. This limitation, combined with very high data rates that can be employed by 802.11g (up to 54 Mbps) make the problem of energy detection *practically* challenging. This detection places certain restrictions on the alphabet size, i.e., we cannot use energy durations that cannot be detected accurately by the 802.15.4 hardware. Specific to the case, which involves competing background traffic, we analyze the packet size traces of many WiFi installations to determine the communication pattern. This analysis reveals a packet size distribution that is mostly bimodal. Given this observation, we distinguish between the Esense and regular packets by assigning those packet sizes to Esense packets that do not normally occur in practice. Thus, our alphabet construction algorithm is designed taking into account both the practical limitations of the hardware and the intended application scenario.

We evaluate our overall Esense framework, both theoretically and via implementation. Theoretically, we quantify the maximum achievable throughput that can be achieved by Esense in different settings and illustrate how this throughput varies with load. For the case, where there is no background traffic, our framework achieves a

TABLE 1
Esense Communication Details for the Three Example Scenarios

Scenario	*Sender*	*Intended Receiver*	*Possible Message Content*
Coordinated Coexistence	WiFi Node	All nodes (or Master) of a given WPAN	Identity of the WPAN, next CFP start time (relative to reception of this message)
Energy Savings	WiFi Node	Secondary radio of the WiFi node it wants to wake-up	Identity of the secondary radio (or receiver WiFi)
Interference Map	WiFi Node	Broadcast	Identity of the sender WiFi node

throughput of about 5 kbps. In the scenario involving background traffic, we validate the use of the constructed alphabet set using the prototype implementation and WiFi traces (different from those used for training) to model background traffic. Our framework achieves low false-positive and negative rates (under 5 percent).

Our overall contributions in this paper are threefold. First, we propose a new framework for communication based on energy and show how it can provide new solutions to problems in three distinct research domains. Second, we propose a methodology for enabling such communication factoring in the limitations of receiver hardware. Third, we thoroughly evaluate and validate our methodology both via theoretical analysis and an actual implementation that supports a unidirectional communication from 802.11 to 802.15.4 devices.

The rest of the paper is organized as follows: In the next section (Section 2), we describe related work. Section 3 presents the background for the considered problem along with the detailed solution approach. We then present our implementation details in Section 4. Results of our evaluation are presented in Section 6. Section 7 presents a few points of discussion and Section 8 concludes the paper.

2 Related Work

To our knowledge, the framework of communication through energy sensing is novel, there is no other similar work. Some aspects of Esense resemble steganography, where information is concealed in otherwise normal looking messages. Steganography has been applied to networking protocols [7], for instance, by using the reserved fields of protocol headers, or by manipulating the timings of Ethernet's CSMA backoff, to convey secret information. Esense resembles steganography in that we construct an information stream embedded within another information stream. However, unlike steganography, our goal is not to hide information; in fact, messages (Esense packets) are explicitly generated for the purpose of conveying the necessary information.

The mechanism where we employ 802.15.4 radios to sense WiFi transmissions, subsequent to our earlier publication [8], was used by others to provide interesting services like determining load of WiFi access points [9] and discovering WiFi access point's signatures [10]. All of these approaches do not focus on using the mechanism for communication.

We have given three example research domains in which our framework is applicable; there has been considerable prior work with respect to these. We present the same below.

The Industry, Scientific and Medical band is used by a variety of standards such as IEEE 802.11b/g [1], IEEE 802.15.4 [2], and Bluetooth [3]. Coexistence studies [11] have shown that there could be considerable interference between the different standards resulting in as high as 90 percent frame loss rate. The most common solution [12], [13], [14] to the above problem is to assess the amount of interference in a given channel and choose a channel that has no or minimum interference. This solution approach works provided there are enough channels that are relatively free of activity. With ISM band increasingly getting crowded coupled with the fact that the WLAN channels occupy much larger spectrum (22 MHz) as opposed to WPAN channels (5 MHz), it would be difficult to get away from interference. In the recent past, a few techniques have been suggested to aid coexistence, specifically in the context of 802.11 and 802.15.4 networks. In BuzzBuzz [15], multiple headers and FEC are used to overcome packet loss in 802.15.4 networks due to 802.11 transmissions. This comes with significant control overhead, high decoding/encoding costs. In TIMO [16], MIMO technology is employed to counter interference on 802.11 transmissions from a variety of interfering sources such as baby monitors, cordless phones, zigbee, and so on. This solution does not provide complete interference isolation, and can potentially be costly and complex for use in low-end technology platforms. In [17], the WiFi white spaces are exploited to time zigbee transmissions. The main problem with these approaches is that they are uncoordinated, they try to address the coexistence problem for a given technology, under the constraints imposed by its MAC. A better strategy is to coordinate and share the spectrum in a time-division multiplexed fashion. This would lead to better throughput, lesser delay, and energy efficiency. Our solution based on energy communication can make such coordination across technologies feasible.

Energy conservation in WiFi-based devices has also received considerable attention. The IEEE 802.11 standard [1] supports a power save mode (PSM) that permits the WiFi interface to duty cycle. There are other MAC layer techniques [18] that build on the PSM to achieve further energy savings. However, as shown in [5], the power consumption of the WiFi interface while idle and using this power save mode is still substantial ($\simeq$440 mW as measured on a smartphone). Given this high idle power consumption of WiFi, the work in [4], [19] has considered the approach of totally shutting down the WiFi-interface and using a low-power secondary radio. A WiFi sender wishing to wake-up a WiFi-receiver does so, by using its secondary radio to communicate with the secondary radio of the receiver. This solution, however, suffers from the disadvantage of range-mismatch: WiFi covers a much larger area than the secondary radio. The short range of the secondary radio makes it necessary to place multiple intermediate proxies and presence servers.

Wake-on-WLAN [8], a previously proposed solution of ours, employs a secondary radio and looks at the problem of energy savings in rural long-distance WiFi networks. Unlike the previous approaches, this solution advocates the need for turning the entire router (soekris board) including the WiFi interface off for conserving energy because the router board itself consumes considerable energy when idle (5 W). This solution, however, employs the secondary radio only on the receiver side (not sender) and avoids the range-mismatch problem by making the secondary radio directly sense the WiFi energy of the sender. The solution has the fundamental limitation that it will work only on point-to-point links. In point-to-multipoint links, or in a regular infrastructure setting, Wake-on-WLAN is incapable of telling exactly which receiver to wake-up.

Cell2notify [5] is another approach that also tries to overcome the range mismatch problem. It uses the cellular

radio as the secondary radio to bring up the WiFi interface. The solution has been specifically designed for enabling VoIP in WiFi-based smartphones. Further it needs infrastructure support in the form of a cell2notify server through which the calls are routed and the danger that the cellphone operators may block the server ID.

In contrast to the above solutions, our Esense-based solution has the following advantages. First, it can work in a variety of settings: enterprise WiFi or long-distance community networks; whether the end-device's WiFi interface is off or the entire end-device is off. Second, it does not suffer from the range mismatch problem because the secondary radio directly senses the WiFi-energy. This is possible because the receiver sensitivity of the secondary radio (-95 dBm) is better than the necessary WiFi received signal strength at the end device (typically -90 dBm @1 Mbps). Third, our solution needs software changes at the WiFi sender side and integration of a very low cost ($<\$10$), low-power (60 mW) secondary radio[1] at the receiver side. Specifically, it does not need any intermediate proxies or servers on the infrastructure side.

The third scenario in which our framework is applicable is in constructing the interference map of an 802.11 network. Prior work [20], [21], [22] in this domain build the maps by conducting individual and pairwise broadcast measurements of order $O(N^2)$, where N is the number of nodes in the network. Reis et al. [23] reduces the number of measurements to $O(N)$ by considering only individual broadcast measurements and relying on a physical layer model to derive the delivery ratio/throughput of a link when it operates in conjunction with others.

One of the major drawbacks of these approaches is that it requires network downtime to conduct the experiments ([20] reports 28 hrs are needed for these experiments on a 22 node testbed). In a dynamic environment where the interference profiles can change over time (due to environmental conditions), this is a serious limitation. Further, it requires careful coordination among the different nodes (to decide on who transmits when) to conduct the experiments apart from requiring a global view of the network.

Our solution to this problem, on the contrary does not require any downtime. The operating nodes need to just periodically broadcast their identity (and possibly the traffic load) via embedding this information in energy bursts and any receiver node R can process this identity information to figure out the number of interfering nodes and at what signal level they interfere with R. This information can then be used to perform appropriate transmit power/channel allocation, or TDMA-scheduling. The interference map may also be coupled with the traffic load information to derive the throughput achievable at this node based on an analytical model. These derivations can help with routing, call admission control, and so on.

3 The Esense Communication Framework

We now present the overall Esense framework. We first present the choices available for embedding information and then present how the alphabet set can be constructed from it.

1. CC2430 is a system on a chip that comes for under 10$ that can be used for this purpose.

3.1 Choices for Esense

In an energy-based communication system, the receiver can sense only the energy patterns on the channel. Based on this sensing, it can interpret three possible parameters: the intensity of the energy, the gap between energy bursts, and the duration of an energy burst. So, whatever information the sender wishes to convey has to be conveyed via these parameters. Among the three possibilities, the intensity of energy is not a good parameter, because it is highly dependent on the external environment, which could be dynamic. Similarly, the gap between energy bursts is also not a good parameter because it cannot be controlled at the sender, especially in distributed contention-based systems. This is because, before the sender can insert the next energy burst at the specified gap, some other node in the system can access the channel and transmit.

The third parameter, the duration of an energy burst, seems promising because it can be controlled at a given sender. In this paper, we develop a framework for communication based on modulating the energy duration.

3.2 Alphabet Set

Given that we wish to use the energy burst lengths for Esense communication, the immediate question then is: which of the possible energy burst lengths can be used? We term this set as the *alphabet set* for Esense. Transmission of an alphabet means transmission of an Esense packet of specific size at a predetermined rate that results in corresponding energy burst duration. For any message exchange, we need a base alphabet, and a set size of two (corresponding to bits 1 and 0) is a minimum requirement for building a vocabulary (i.e., sequences of alphabets). In this regard, alphabet is much like a symbol used at the physical layer. Typically, the size of the alphabet set is determined by practical considerations such as accuracy of the hardware in measuring channel busy times and presence or absence of background traffic.

If the alphabet set that could be built based on practical considerations is of size M, each alphabet basically conveys $log_2 M$ information bits. It is now easy to calculate the achievable transmission rate via Esense. Let $B_1, B_2, \ldots, B_M$ be the burst lengths corresponding to the alphabets, arranged in increasing order. Assuming all alphabets are equally likely, then the average burst length B is given by

$$\frac{\sum_{i=1}^{M} B_i}{M}.$$

Depending on the scenario, transmission of an Esense packet may be preceded by a channel access delay. For example, in the energy management scenarios, this delay could be due to regular 802.11 packet transmission times, *DIFS* and back off. If the average channel access delay prior to transmission of an alphabet is represented by A, the achievable transmission rate R of Esense is given by

$$R = \frac{log_2 M}{A + B}. \quad (1)$$

Larger the alphabet set, more the information that can be conveyed (numerator), but the average transmission time (B in the denominator) can also be high. Therefore, there is

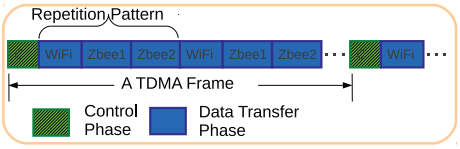


Fig. 2. Coordinated coexistence.

an optimal choice of the alphabet set that works best given the actual values of the burst lengths. Also note that for large access delays (A), it makes sense to use a larger alphabet set. This way, more information on average can be conveyed on each access (assuming A dominates B). This tradeoff is explored in detail in Section 6.1.

For now, we aim to build as large a Esense alphabet set as is feasible under the conditions. Large alphabet sets will also ease the complexity of communication. For example, in the energy-savings scenario described in Section 1, we need to have a different message for waking up each of the different WiFi receivers in the AP's range. If the WiFi receiver set is of smaller size than the Esense alphabet set, a single Esense packet is sufficient to convey the message (alphabet maps directly to the id of the receiver). However, if the message space required (i.e., the receiver set) is larger than the alphabet set, one needs to consider combinations of the base alphabet to convey the necessary information, which can lead to decoding complexity.

3.3 An Approach to Alphabet Set Construction

As mentioned earlier, which energy bursts constitute an alphabet set is highly depend on the capabilities of the hardware and the underlying application scenario which in turn dictates the presence/absence of competing background traffic. We consider two scenarios here, to explain how the construction differs in each case. We present an outline of our approach here, a more in-depth construction procedure is provided in Section 5.

Absence of Background Traffic. In application scenarios such as cross-technology coordination, by design, we can ensure that no competing background traffic flows during Esense communication. To understand why, we provide a brief explanation via a simple example scenario, more details can be found in [24].

Cross-technology coordination can be achieved by building an extremely lightweight centralized TDMA-based overlay MAC. This overlay MAC employs Esense communication and operates on top of individual MAC standards. Suppose there are three competing networks in the same geographical region, one based on Wifi and two based on zigbee, all wishing to time-share a given frequency band. And suppose each network is represented by a master (e.g., Access Point in WiFi network, sink nodes in sensor networks) and one of the masters also acts as the centralized *root* coordinator (e.g., WiFi AP). The TDMA MAC can be implemented via a control and data transfer phase (Fig. 2). During the control phase, Esense messages are broadcast by the root to time-sync the masters and to pass on the data schedule information (this information can be sent in as little as 20-30 bits). Data schedule information conveys which *networks* can access the channel in which data transfer slot. The master nodes then make use of this schedule information to coordinate their network channel access. During the allocated time (data transfer slot), all the nodes in the network employ their individual MACs to access the channel (e.g., WiFi nodes can employ CSMA).[2]

Given the above explanation, it is easy to see that during Esense communication (control phase), by design no competing background traffic flows. The individual nodes can transmit only in their allocated data transfer slots.

Since during Esense communication, no competing background traffic flows, the burst lengths are dictated solely by the hardware capabilities of the receiver. If the hardware permits clear resolution between two energy bursts that differ in length by a minimum value of $X\mu s$, then the burst lengths that form the alphabet can take on the values $i * X, i = 1, \ldots, M$. Basically, in this case, two successive alphabets differ in length by X. M is dictated by the energy burst length resulting in the transmission of the maximum packet size (MTU) sent at the lowest possible rate.

Note that when there is no competing traffic, it is possible to encode information during channel idle times as well. For example, bit "1" can be encoded as a channel "on/busy" time of $a\mu s$ and a bit "0" as a channel "off/idle" time of $b\mu s$. This improves efficiency considerably but needs much fine-grained detection of run lengths of 1s and 0s. We tried implementing this technique on the 802.15.4 hardware, but found it difficult due to weighted averaging of the signal strength employed by the hardware. We will explore this option in the future. For now, we assume that each burst length is separated by a duration Y, i.e., channel idle time of Y. Y is again dictated by the capability of the hardware in resolving two successive bursts. For this scenario, in (1) of Section 3.2, Y would correspond to the channel access delay A, because before transmission of the next alphabet, we need to maintain a channel idle time of Y.

Presence of Background Traffic. In both the energy management and building interference maps scenarios, Esense communication may need to be supported along with regular 802.11 packet transmissions. With these regular transmissions, there may be energy bursts of different lengths, due to the various packet sizes and various transmission rates in use. Hence, there is a risk that a regular packet may be confused for an Esense alphabet by the Esense receiver, i.e., we may have a false positive. Thus, in choosing the Esense alphabet set, apart from hardware considerations, we must also seek to minimize the false positive rate.

There are a variety of WiFi traces available in the public domain [25]. We identify five representative traces to study the communication patterns of WiFi networks; these are traces from various WiFi infrastructure mode deployments. The five traces are: Cafeteria (Powells), Library, PSU-CSE department (all from [26]), OSDI Conference [27], Stanford CSE department [28].

An analysis of the packet size distribution of one of the five traces is depicted in Fig. 3. Note that the y-axis is log scale. Across all traces, the majority of the packets are either small packets (<140 bytes) corresponding to the

2. The above coexistence mechanism does require some MAC support, i.e., individual nodes in a network not transmit during other network's data slots. This can be achieved for example, in WiFi networks, by appropriately setting the Network Allocation Vector (NAV). In sensor networks, this can be achieved via polling-based contention-free operation.

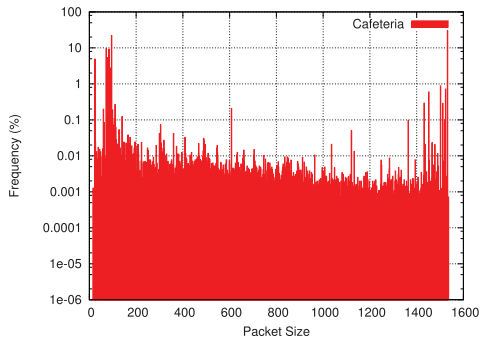


Fig. 3. Packet size distribution of cafeteria trace.

ACKs, beacons, management frames. Or they are around 1,500-byte packets corresponding to the Ethernet MTU. Such a bimodal distribution is a well-known observation in the Internet as well; we confirm that the same applies in WiFi networks too. The above observation suggests a possible approach for Esense. As far the 802.15.4 receiver node is concerned, it detects the energy of both regular packets (part of the WiFi network) and the Esense packets. It now needs a mechanism to distinguish between regular and Esense packets. For now, let us assume that the 802.15.4 node can detect channel occupancy with high accuracy, i.e., it can detect any packet size up to an accuracy of a byte. The solution then is to *exclude* all packet sizes whose frequency of occurrence in the WiFi traces is greater than a *threshold percentage*. And allocate the rest as alphabets for Esense. The higher we choose the threshold percentage, the larger the alphabet set because we would be excluding lesser number of packet sizes. However, increasing the threshold percentage can result in high false positives, where some of the regular packets that were not excluded get interpreted as Esense packets.

The base alphabet, thus, corresponds to the complement of the set corresponding to these packet sizes that exceed the threshold percentage. Note that the alphabet set needs to be bounded, because the underlying technology has a maximum packet size. So, we bound the alphabet set by the maximum packet size feasible in 802.11, i.e., 2,304 bytes. Note that in this case of background traffic, the burst durations in the alphabet set follow no mathematical formula.

Suppose we choose a threshold percentage of 1 percent. From the traces, if we were to count the packet sizes with frequency greater than 1 percent, this number turns out to be under 16 for all the traces. Thus, the base alphabet for the Esense packets turns out to be 2,288 (2,304-2,316), which is quite large. However, there are some practical limitations to achieving this alphabet set size. The 802.15.4 node does not have byte level accuracy and thereby we need to maintain a minimum difference of X between two successive energy burst lengths. This can cut down the size of the alphabet set significantly. The next section outlines these implementation constraints.

4 Esense Implementation Constraints

As proof of concept of the Esense framework, we enable unidirectional communication from an 802.11 radio to an 802.15.4 radio. As noted in Section 1, this hardware choice applies in the three scenarios listed. Before getting into details of Esense implementation, we first present our experimental methodology. We then present some practical limitations in achieving Esense communication and follow it up with accuracy characterization of the 802.15.4 platform. This section sets the context based on which we experimentally construct the Esense alphabet set for 802.11 to 802.15.4 communication (which is the focus of our next section, Section 5).

4.1 Experimental Methodology

In all of our experiments, we use a WiFi radio as the sender and an 802.15.4 radio (on the Tmote Sky platform) as the receiver. For the 802.11 sender, we use a laptop equipped with a 802.11b/g WiFi card (with the Linux open-source madwifi driver) as the transmitter. We term the 802.15.4 receiver platform as a "mote," because this is the platform commonly used in many Wireless Sensor Network (WSN) applications.

The 802.15.4 receiver mote is placed a few meters away from the 802.11 sender in an indoor lab environment and is connected to another laptop, which is used to log experimental data via the mote's serial interface. To reduce the memory requirement at the mote (mote has a limited memory of only 10 KB of RAM), we just log the time (clock tick) at which the channel state (busy/idle) changed. This log can then be used to calculate the run length in clock ticks of the channel busy time.

For the experiments that characterize the accuracy of the mote in detecting channel busy time, we need very fine-grained control over the spacing between adjacent packets. We achieve this control by modifying the madwifi driver. Specifically, we disable backoff and set cwmin, cwmax (contention window) to zero. We then use the AIFS feature (part of the 802.11e standard) to control the spacing between packets. For a given AIFS, the spacing between two packets is given by the formula $(10 + AIFS \times 20)\ \mu s$, where AIFS can take on values starting from 0. Hence we can achieve as low as $10\ \mu s$ spacing between the packets. We verified that this mechanism works correctly by connecting the WiFi card to a spectrum analyzer. We observed values as given by the formula with an error under $2\ \mu s$.

For our other experiments, we need to emulate background WiFi activity. For this, we use the same WiFi traces as described in Section 3.3 (under presence of background traffic). The timing interval between packets as captured by the traces is not a reflection of what actually happens on the channel since the timers are software based and not very accurate.[3] So, we just use the packet size distribution from the traces and consider that all these packets are backlogged at the single WiFi sender. The spacing between the packets is then dictated by the random backoff employed by the 802.11 standard. In reality, the spacing between the packets is often more than what we consider, and any extra interpacket spacing can help the mote detect the packets better. Note that this setting does not capture losses due to collision, which can result in alphabet loss.

3. While some trace files do provide high accuracy timing, not all of our traces do. To be consistent, we do not consider the timing information provided by the traces.

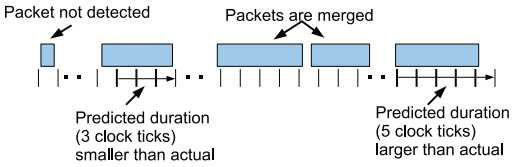


Fig. 4. Illustration of 802.15.4 hardware limitations.

We assume that collision losses are few and can be handled by our retransmission mechanism.

We implement this feature of backlogged queue in the madwifi driver taking care that no packets are actually dropped due to buffer overflow. We, however, do these experiments as batches of 500 backlogged packets because the mote's memory runs out if we continuously send packets.

4.2 Practical Limitations

There are a few practical challenges that need to be tackled to implement the Esense framework. 1) The 802.15.4 Esense receiver can only sense the energy burst duration. This implies that we cannot map packet sizes to alphabets since the WiFi sender could be sending at any of the multiple possible data rates. 2) The accuracy of commodity IEEE 802.15.4 radio hardware in detecting channel occupancy is limited. Further, the below two issues make this even harder. 2a) 802.11 operation supports multiple rates, with 802.11g mode supporting very high data rates, up to 54 Mbps. 2b) Also, the interpacket gaps in an operational system are unpredictable due to the very nature of the CSMA/CA protocol. These gaps could be as small as $50\ \mu s$ for 802.11b and $28\ \mu s$ for 802.11g.

The inaccuracy in channel occupancy estimation, may arise due to two reasons: 1) finite granularity of time measurement, and 2) finite sampling granularity. The inaccuracy issue manifests in many ways, illustrated in Fig. 4. First, when the channel occupancy is say L μs, the radio hardware may indicate a number greater or smaller than this. Second, the inherent inaccuracy combined with high data rate operation may mean that an entire packet is missed by the Esense receiver. For example, when operating at say 54 Mbps, a 40-byte packet's transmission time is roughly $31\ \mu s$ (including PLCP overhead). The 802.15.4 radio may not even be able to detect this packet. Third, if the time separation between two adjacent packets is very small (this interval is dictated by the random backoff of the 802.11 standard), the radio may club two or more packets together and interpret it as one large packet.

4.3 Mote Accuracy Characterization

To characterize the accuracy of the mote hardware, we perform two experiments. In the first experiment, we try to capture what the mote reports when the channel is busy for a specified duration. For this experiment, we send a given sized packet 5,000 times, with interpacket spacing set to a sufficiently large value of 50 ms. We measure the channel occupancy as reported by the mote and compare it with the actual value. The actual value is measured using a spectrum analyzer.

Table 2 shows the findings of the mote when the actual channel occupancy duration was set to 123, 785, and $4{,}710\ \mu s$. The columns represent the first, second, and third highest frequency components of the mote measurements. Note that the mote reports its findings in clock ticks. We multiple this with $30.5\ \mu s$ to arrive at the numbers in the table.

TABLE 2
Mote Characterization: Margin of Error

Busy Time	*Mode-1*	*Mode-2*	*Mode-3*
123 μs	152.5 ; 74.5%	183 ; 16.62%	122 ; 8.84%
785 μs	793 ; 63.84%	823.5 ; 34.46%	762.5 ; 1.7%
4710 μs	4727.5 ; 67.38%	4697 ; 26.6%	4758 ; 5.94%

As can be seen from the table, the mote often reports a value higher than the actual channel occupancy. And the margin of error is a maximum of about two clock ticks, i.e., $61\ \mu s$. We attribute the inaccuracies to the coarse time granularity as well as the system delays involved in polling the CCA pin.

The goal of the second experiment in the mote characterization is to determine the spacing between packets at which a mote can successfully distinguish one packet from the other. For this experiment, we send a periodic stream of packets, all of the same size, separated by a given interval. We choose a packet size that results in channel occupancy time of $785\ \mu s$. We consider different spacings: 10, 30, 50, 70, and $90\ \mu s$ by appropriately varying the AIFS value. Fig. 5 presents the results for the 50-μs case.

At 50-μs interpacket gap, the mote does not clearly distinguish between the packets. In a good number of cases, it tends to merge packets and reports them as one big packet. This is evident from the periodic spikes seen in the figure. These spikes occur at multiples of the packet transmission time. The largest run length that we observe in this experiment corresponds to one where 15 packets got merged, though this occurs very rarely at 0.1 percent. At $70\ \mu s$, the largest run length we observe corresponds to one, where five packets got merged and this happens at a frequency of 0.5 percent. At $90\ \mu s$, we do not observe any merging, the mote is able to clearly separate out the individual packets.

In summary, the mote reports channel occupancy time with a margin of error which is ± 2 clock ticks (i.e., $\pm 61\ \mu s$). And it can separate out the packets clearly only when they are separated by at least $90\ \mu s$.

5 Alphabet Extraction

We now focus our attention on extracting the alphabet based on above practical limitations and mote characterization experiments.

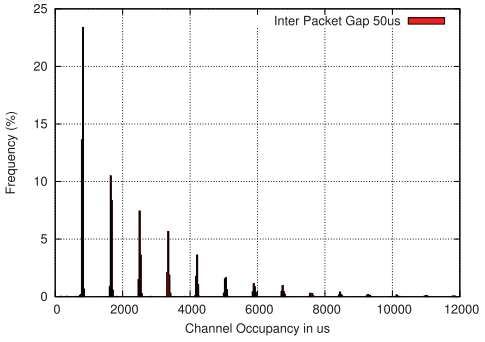


Fig. 5. Channel occupancy as reported by the mote at $50\ \mu s$ interpacket gap (actual channel occupancy: $785\ \mu s$).

We address the first limitation related to multiple 802.11 data rates, by using energy burst lengths directly as Esense alphabets, instead of using packet sizes. The 802.11 sender needs to figure out the packet size and rate at which to send this packet, such that the resulting channel activity results in the correct energy burst length.

Regarding the other limitation involving mote accuracy, we handle it as follows for the two cases:

5.1 Absence of Background Traffic

For this case, if we wish to ensure proper decoding of burst lengths, we need to maintain a difference of 2 clock ticks, i.e., $61\ \mu$s both above and below a given burst length. The given burst length could take on values anywhere in this range due to mote hardware inaccuracies. This would mean a minimum difference of 4 clock ticks between any two bursts in the alphabet set, i.e., a value of X is given by $122\ \mu$s. Thus, the alphabet set takes on values $122, 244, 366, \ldots, 18{,}544$, where the last value is dictated by the maximum packet size sent at 1 Mbps. Similarly, when transmitting on air, for proper decoding of the packet energy lengths, a channel idle time of $90\ \mu\text{s}\ (=\text{A})$ needs to be maintained. In this case, because we assume that a *root* coordinator (see Section 3.3) has full control over the channel, maintaining the required channel idle time is not difficult.

5.2 Presence of Background Traffic

In this case, we first present the design of the alphabet extraction algorithm, followed by the alphabet set resulting from its application.

Algorithm Design. This scenarios is more complex and we take a measurement-based approach to determining the base alphabet for Esense. We factor in the limitations of the commodity 802.15.4 hardware by actually using the hardware during the process of Esense alphabet construction itself. We measure the channel occupancy when a real WiFi node transmits packets of sizes as dictated by the traces and packet intervals as dictated by the standard. For a given packet, the 802.15.4 hardware measures the channel occupancy as a run length of clock ticks. Much like what we have done earlier for packet sizes, we calculate the frequency of occurrence of a given run length. We extract our alphabets from the complement set of run lengths that exceed a threshold percentage of frequency.

We consider snapshots of the traces, each snapshot consisting of 500 packets in the trace. For the CSE-PSU, Library, and Cafeteria traces (which were shorter), we took 20 different 500-packet snapshots from different positions in the trace, resulting in a total of 10,000 packets. And for the CSE-Stanford and OSDI traces (which were larger), we took 50 different snapshots, resulting in a total of 25,000 packets. We verified that the distribution of the extracted trace is similar to the original trace.

For a given extracted trace, the 802.11 sender generates packets of size as specified in the trace and sends them back-to-back. On air, these packets get separated out by a gap as dictated by the random backoff of 802.11 standard. There is the question of what 802.11 data rate to use to send these packets. The traces unfortunately do not provide this information. Hence, we consider a variety of data rates including random rates to emulate nodes that employ autorate adaptation.

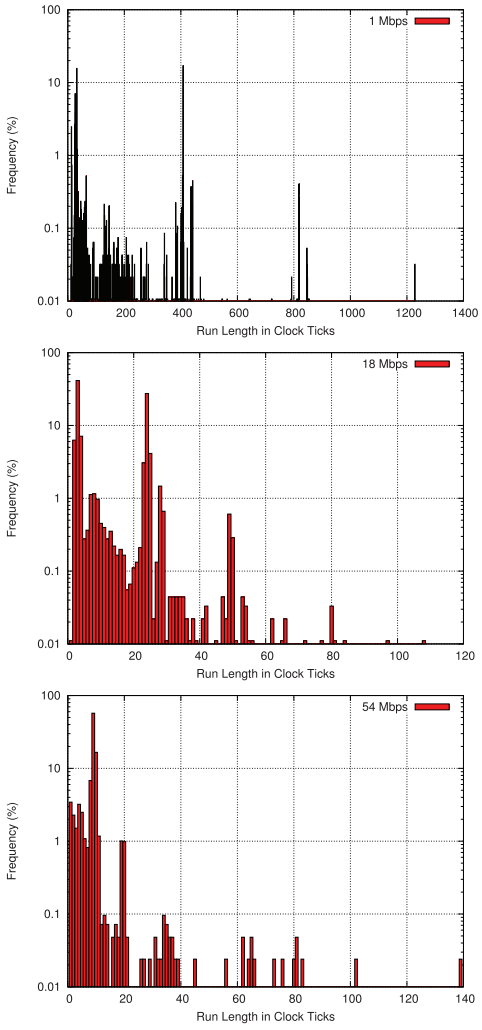


Fig. 6. Cafeteria trace: Run lengths at 1, 18, 54 Mbps.

Fig. 6 shows the findings of the mote for the Cafeteria trace at three different rates. Note that the y-axis is log scale and the x-axis range is different for the different rates.

We make three observations from the figure. One, the bimodal distribution that we saw in the packet lengths is preserved in the run-length distributions, up to 18 Mbps and after that it starts becoming unimodal. This is because at high data rates, the shorter length packets (corresponding to beacons, acks, management frames, and so on) are often not detected by the mote. Two, one can observe peaks at positions corresponding to multiples of the predominant run lengths. This is due to the mote merging adjacent packets. This may not be as predominant in reality because we assume backlogged packets in our emulation.

Third, if we were to consider only run lengths with frequency greater than 1 percent for elimination, prior to alphabet extraction, we can see one chunk of run lengths available between the two predominant modes. This chunk starts to become smaller beyond 18 Mbps and disappears altogether at 54 Mbps. For the various data rates, there is another chunk of run lengths available to the right of the second highest frequency component. This chunk starts at 410 ticks @ 1 Mbps, 28 ticks @ 18 Mbps, and 19 ticks @ 54 Mbps. If we assume that all Esense alphabets are sent at 1 Mbps, the maximum packet size

corresponds to a run length of about 610 ticks. So, there is considerable space available in all cases to construct alphabets in spite of the mote's limitations.

While we have presented Fig. 6 only for the Cafeteria trace, we plotted similar graphs for the other traces too, and observed similar findings.

We extract our alphabets from the complement set of run lengths that exceed a threshold percentage of frequency. This process needs to address the following three issues:

1. What is the run length we should use to bound the complement set?
2. Do we consider all run lengths in the complement set as alphabets?
3. How do we handle false negatives resulting from merging of packets by the low-resolution hardware?
4. How do we handle false positives resulting from regular packets being confused as Esense packets?

We address the above issues in the following manner:

1. *Bounding the complement set*. The first issue is that of determining the maximum run length to use for an alphabet. This run length could correspond to that which occurs when the maximum sized packet (MTU corresponding to that standard) is transmitted at the configured data rate of the sender, i.e., the data rate the sender uses to send regular packets. While this may give enough alphabets at low data rates, at high data rates, the alphabet set will be very limited. For example, at 54 Mbps, a 1,500-byte packet produces a run length of about 8 ticks and the maximum size packet of 2,304 byte produces a run length of 12 ticks. So, there are about four run lengths that can be considered as potential alphabets (assuming all packets larger than 1,500 bytes occur at a frequency less than the threshold), which is very limited. We can overcome this limitation by considering the lowest rate at which an 802.11 node can operate. If an 802.11 node has been configured to operate in the g mode only, we bound the alphabet by the run length that occurs when a 2,304-byte packet is sent at 6 Mbps (lowest data rate in the g mode). If the 802.11 node has been configured to operate in the b mode, we bound the alphabet by the run length that occurs when a 2,304-byte packet is sent at 1 Mbps (lowest data rate in the b mode).[4] Note that regular data packet transmissions except management frames happen at high data rates, but we restrict that the Esense packets are always sent at the lowest possible data rate at the sender.
2. *Building-in margins between alphabets*. One cannot consider all the run lengths in the complement set as potential alphabets because of the measurement inaccuracies of the 802.15.4 hardware. Much like that was done in the case of alphabet extraction in absence of background traffic, we maintain a margin (4 clock ticks) between two successive burst lengths. This gap is also maintained between an alphabet and a regular packet whose frequency exceeds the threshold percentage of frequency. So, not all run length in the complement set get chosen, only a subset. Once we determine the alphabet set, we map the run length to appropriate packet size considering the lowest data rate of operation possible in that setting (since we have stipulated that Esense packets are always sent at the lowest data rate).
3. *Handling false positives and negatives*. Problems 3 and 4 have a common solution. To reduce the occurrence of false positives and negatives, we consider the use of repetitions. We essentially send the Esense alphabet (packet) multiple times in a small time window. And at the receiver, we deem that an Esense alphabet (packet) has been received only if it detects some threshold number of instances of the Esense alphabet within the chosen time window.

4. A 802.11 node in g mode can potentially transmit at 1 Mbps. Doing so can increase its alphabet space further but we make the worst case assumption that its configuration prevents it from doing the same.

TABLE 3
Alphabet Size

Rate (Mbps)	*PSU*	*Library*	*Cafeteria*	*Stanford*	*OSDI*	*All* Traces
1	109	112	115	112	110	104
11	115	116	115	115	117	113
6	116(15)	117(15)	118(15)	117(14)	113(15)	112(10)
18	118(16)	118(14)	118(14)	116(14)	118(16)	115(13)
36	119(15)	117(15)	117(15)	117(15)	199(17)	116(14)
54	118(13)	117(14)	118(15)	118(14)	119(11)	117(15)
All Rates	107(12)	110(12)	110(11)	109(12)	110(11)	100(10)

To summarize our design, a WiFi node wishing to send an Esense packet does so, by selecting a run length (and, therefore, a corresponding packet size at the lowest rate of operation) in the alphabet that corresponds to the message it wishes to communicate. It then sends this packet multiple times at the lowest possible data rate. The 802.15.4 receiver concludes it as an Esense packet if it observes a channel occupancy run length corresponding to an alphabet. This should further occur a specified number of times within a given time window.

Alphabet Set. For the actual alphabet extraction, for each of the five traces, we considered six different data rates at which the packets get sent: 1, 11, 6, 18, 36, 54 Mbps. This gives a total of 30 run-length logs. For each log, we extract the alphabet based on the algorithm explained above. We use the value of 1 percent for the threshold frequency of occurrence, beyond which we eliminate a run length from consideration for the Esense alphabet set. This threshold provides a good tradeoff between alphabet size and false-positive rate.

The algorithm takes as input another parameter: the mode of operation (b or g). This specifies the rate at which the Esense alphabet is sent. For the b and g mode, the alphabet is sent at 1 and 6 Mbps, respectively. Note that the b mode should not be interpreted to mean that the data rates are restricted to under 11 Mbps. It just means that it uses a data rate for alphabets that is available only in that mode.

Table 3 shows the size of the extracted alphabet set from the above-mentioned 30 logs: five traces along columns, six data rates along rows. For the 6, 18, 36, and 54 Mbps rows, the values without brackets correspond to the b mode of operation and values within correspond to the g mode of operation.

TABLE 4
Cafeteria Trace: All Data Rates

802.11b Data Rates			
1mbps	2Mbps	5.5Mbps	11Mbps
115	115	115	115
802.11g Data Rates			
6mbps	9Mbps	12Mbps	18Mbps
118(18)	118(15)	118(15)	118(14)
802.11g Data Rates			
24mbps	36Mbps	48Mbps	54Mbps
116(13)	117(15)	118(16)	118 (15)
All rates			
105 (6)			

In the table, we have a last column labeled "All traces," and a last row labeled "All Rates." The last column corresponds to alphabet extraction for the case where we eliminated run lengths that exceeded the threshold percentage of 1 percent in *any* of the traces (i.e., in at least one of the traces). Similarly, the last row corresponds to alphabet extraction where we eliminated run lengths, which exceeded the 1 percent frequency threshold in *any* of the six considered rates of operation. The last row and the last column, thus, correspond to conservative choices of the alphabet set; the last entry in the table, thus, represents the most conservative choice. (Conservative with respect to data rate implies: if we do not know the rate at which a high-frequency regular packet size would be sent, eliminate all the run lengths corresponding to all the possible data rates).

We observe that the "All traces" column is only slightly different from the columns for the five traces. This implies that the run length distributions are more or less the same across the five traces. In fact, we observed in our algorithm output that the alphabet set itself is not very different across the various traces.

Similarly, we observe that the "All rates" row is only slightly different from the columns for the six different rates. This means that there are only a few run lengths, which show a low frequency of occurrence at one rate, but show a high frequency of occurrence at another rate. This then means that rate adaptation will not significantly affect the size of the extracted alphabet set.

We observe in the table that across the six rates, as the data rate increases, the size of the alphabet set shows a slight increase. This is because at higher data rates, packet transmission times are much smaller and get clustered toward the shorter run length, leaving more empty space from which to extract alphabets.

As can be seen from the table, we get about 100 alphabets when alphabets are sent at 1 Mbps and about 10 when alphabets sent at 6 Mbps. This difference due to change of mode is because in g mode, the maximum run length that we can consider is only 100 ticks (corresponding to 2,304 bytes sent at 6 Mbps), while in b mode this turns out to be 610 ticks.

We have performed the above experiments for different threshold percentages. At 10 percent, the resulting alphabet set was 108 for b mode and 13 for the g mode (corresponding to the "All traces" row and "All rates" column). At 0.1 percent, the alphabet set falls sharply to 60 and 3, respectively. This sharp decline is mainly due to the conservative operation across the traces. This is because quite a few run lengths in all the traces exceed the small threshold of 0.1 percent. And further very few of these "frequent" run lengths are common across the traces. This results in lot of elimination resulting in smaller alphabet set. The conservative operation across data rates (the "All rates" column) for any trace still yields high alphabet set: around 90 and 6 for b and g, respectively.

In the above experiments, we have considered only a subset of the data rates supported by 802.11 b and g. For a given trace, there is not that much difference in the alphabet sets across the different rates. But for completeness, we consider all data rates for the Cafeteria trace. Table 4 shows the results of this experiment. The results are very similar to what we obtained earlier. The conservative "All rates" entry now reduces the alphabet set size to 105 (6) because we are considering intersection across a much larger set.

The above results in terms of alphabet size could improve considerably if we worked with an 802.15.4 hardware that is more accurate.

TABLE 5
Transmission Rate: No Background Traffic

Alphabet Size	2	4	8	16	32
Rate (kbps)	3.70	5.13	4.76	3.60	2.41

6 Evaluation

We now undertake both a theoretical and implementation-based evaluation of the Esense framework. We also consider application of Esense in rural mesh networks as a case study and briefly explain the performance improvement that it can provide.

6.1 Transmission Rate Analysis

As discussed in Section 3, the transmission rate is given by $R = \frac{log_2 M}{A+B}$, where M is the alphabet set size, B is the average Esense alphabet transmission time, and A is the average channel access delay. We consider two scenarios in the analysis: absence/presence of background traffic.

Absence of Background Traffic. For this case, the implementation-based analysis carried in Section 5 suggests burst lengths that are multiples of $122\ \mu$s $(=\mathrm{X})$, with an access delay of $A = 90\ \mu$s. Table 5 tabulates the variation of the transmission rate as a function of the alphabet size. Note that we considered alphabet size to be an exponent of 2, because it permits easy implementation, where bits can be mapped to alphabets easily. As can be seen, the rate initially increases with alphabet size and then decreases. This is because of the tradeoff between the numeration and denominator of the equation. The maximum transmission rate is achieved for an alphabet size of 4 and it is 5.13 kbps.

Presence of Background Traffic. In the case involving background traffic, the alphabet set available for use is a function of the mode of operation: 100 alphabets for 802.11b and 10 alphabets for 802.11g (refer to Table 3). Based on the alphabet set, the average transmission time B is straightforward to calculate. On the other hand, the average access delay is a function of the actual 802.11 MAC protocol operation and is governed by many factors such as number of active users, their average packet sizes, 802.11 MAC parameters such as retry limit, backoff exponent, and so on

TABLE 6
Transmission Rate: Background Traffic

No external Users: 802.11 b					
Alphabet Size	2	4	8	16	32
Rate (kbps)	0.39	0.71	0.93	1.03	0.92
No external Users: 802.11 g					
Alphabet Size	2	4	8	16	32
Rate (kbps)	0.80	1.43	1.63	NA	NA

[29]. Instead of undertaking a theoretical analysis of this average delay (which is outside the scope of this paper), we consider two cases: 1) We set the number of other active users to zero, i.e., only the node sending Esense packets is active. In this case, the access delay is dictated the average back off employed before sending the Esense packet. 2) We vary the access delay between 1 and 8 ms, representative of varying number of other active users. These values are typical in operational networks.

In the case involving no other users, the average access delay is given by $360\ \mu s$ for 802.11b and $95.5\ \mu s$ for 802.11g.[5] Table 6 shows the achieved data rate as a function of the alphabet size. For 802.11b mode of operation, an alphabet set size of 16 gives the best rate of 1.02 kbps. The rates fall down significantly compared to the no background traffic case, because we are paying a penalty in the form of access delay for each Esense transmission (A is $360\ \mu s$ and B is 3.5 ms for 16 alphabets). For 802.11g mode, eight alphabets leads to the best rate of 1.627 kbps. In this case, in reality we have only 10 alphabets, but an extrapolation for 16 alphabets, does not increase the rate. Hence, 1.627 kbps is indeed the maximum for this mode of operation.

Table 7 shows the results of the best transmission rate and the corresponding alphabet size for varying access delays (case two) for the 802.11b mode. As the access delay increases, as expected the achievable rate falls. As the access delay increases, the corresponding alphabet size increases due to reasons mentioned earlier, i.e., we can transfer more information on average on each access since A dominates B.

In all these cases that involve penalty of access delay, the data rates are much lower than the maximum feasible under no background traffic case. We note that this data rate is still quite adequate to carry out lightweight communication. Many of the application scenarios we described do not need very high rates and can manage quite well with these rates. That said, if higher date rates are needed, further optimization is possible through a practical hack. It is possible to reserve a NAV duration of channel time for Esense transmissions by sending a CTS packet prior to sending the Esense packets. The maximum NAV that can be realized in practice is 32 ms. For example, if average channel access time is 8 ms, we can get approximately $32^{*}5.13/(32+8) = 4.1$ kbps data rate as opposed to 0.38 kbps. On every access, we grab the channel for 32 ms and leave it for others for 8 ms before next access. During the 32-ms time, we send Esense packets much like the case where there is no background traffic, during which we get 5.13-kbps rate.

5. Average access delay is given by $\text{DIFS} + (CW/2) * \text{slot-time}$. For 802.11b, it is $50 + (31/2) * 20$ and for 802.11g, it is $(28 + (15/2) * 9)$.

TABLE 7
Transmission Rate as a Function of Access Delay

A (in ms)	1	2	3	4	5	6	7	8
Max R (in kbps)	0.88	0.72	0.62	0.55	0.49	0.45	0.41	0.38
Size	16	16	32	32	32	32	32	32

6.2 Implementation-Based Validation

We now validate our communication framework using the extracted alphabet (Table 3). We consider the alphabet set corresponding to the conservative calculation because this is the most situation independent; i.e., the set corresponding to the "All traces" row and "All rates" column. We consider both modes of operation, i.e., the b and g compatible modes. For the b mode of operation, we consider two fixed data rates of 11 and 18 Mbps and one variable data rate. For the fixed data rates, all regular packets with the exception of management packets are sent at the respective rates. The management and the Esense packets are sent at 1 Mbps. The variable data rate is supposed to model nodes that employ rate adaptation. For this case, the regular packets (with the exception of management packets) gets sent at a rate randomly chosen from the set (1, 11, 6, 18, 36, and 54 Mbps). For the g mode of operation, we consider one fixed rate of 18 Mbps and one variable rate. The management and Esense packets in this mode are sent at 6 Mbps. The other regular packets are sent at 18 Mbps for the fixed rate case and randomly chosen from the set (6, 18, 36, and 54) for the variable data rate case.

We perform our validation on two traces: Cafeteria and CSE-PSU. For each trace, we take 60 500-packet snapshots of the trace at different trace positions to generate an overall count of 30,000 packets. We ensure that these snapshots do not overlap with the snapshots we used to extract the alphabet, i.e., our validation trace is not the same as the training trace. We then insert 250 random alphabets into this trace. Recollect that to reduce the instances of false positives and negatives in our setting, it is desirable to send an alphabet multiple times within a time window. We consider four possible values for this: 1, 3, 5, and 10, i.e., each alphabet gets sent this many number of times. We refer to this as the sender repetition count, SRC. It is possible in reality that some regular packets get in between these alphabet packets due to other WiFi node transmission. We model this by inserting a random number of regular packets (maximum of 5) between the alphabets.

At the receiver, we measure the number of distinct alphabets detected. Since a given alphabet gets sent multiple times within a window, the receiver concludes that an alphabet is detected successfully if it observes the alphabet a specified number of times within the window. We term this expected repetition count as the receiver detection count, RDC.

Table 8 shows the number (not percentage) of false positives and false negatives that occur in the various cases, when 250 random alphabets are embed in a trace of 30,000 regular packets. False positives correspond to the case where no alphabet was sent but the receiver detected one. False negatives correspond to the case where an alphabet was sent but was not detected. The first column in the table

TABLE 8
False Positives and Negatives:
250 Alphabets, 30,000 Regular Packets

Cafeteria Trace										
	b Compatible Mode						g Compatible Mode			
Param	11Mbps		18Mbps		R-Adapt		18Mbps		R-Adapt	
	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
(1,1)	24	268	11	142	13	413	8	73	2	63
(3,1)	3	332	3	185	0	419	0	242	0	162
(3,2)	17	67	22	22	10	26	13	23	9	10
(5,2)	0	24	1	14	1	19	0	20	1	16
(5,3)	5	4	3	0	5	2	1	2	3	4
(10,4)	0	6	0	1	0	2	0	0	0	2
(10,5)	0	0	0	0	0	0	0	0	0	0
(10,6)	1	0	6	0	1	0	1	0	0	0
PSU Trace										
(1,1)	17	545	18	340	21	787	20	92	13	262
(3,1)	2	626	0	414	1	808	0	100	1	290
(3,2)	24	130	20	80	14	97	17	14	10	48
(5,2)	1	126	0	67	1	109	1	14	0	50
(5,3)	5	40	7	29	7	32	4	3	2	15
(10,4)	0	24	0	20	0	21	0	2	0	8
(10,5)	0	12	0	9	0	15	0	0	0	6
(10,6)	0	8	2	5	2	8	1	0	3	1

corresponds to the pair (SRC, RDC). We make the following observations from the table:

- For a given SRC, as the RDC is increased, it results in lesser number of false positives but higher number of false negatives. This trend is evident in the table. Higher the RDC, lesser the chance of a regular packet being mistaken for an Esense packet. A regular packet has to repeat in a small time window as many times as the RDC to be mistaken for an Esense packet. But this also means that we cannot properly detect alphabets since the mote can merge packets. Hence, there is a fundamental tradeoff in the choice of RDC.
- The overall false positives as well as negatives are quite small especially for values beyond (5, 3) considering that there are 250 alphabets sent amidst 30,000 packets. Note that the false negatives should be compared relative to the alphabets sent and the false positives relative to the total packet count. The parameter set (10, 5) or (10, 4) seems to be a good operation point. It results in no false negatives and very small false positives.
- The fact that there are false positives even at (10, 6) seems to indicate that some regular packets (which occur less than 1 percent of the time) repeat at least six times in a small window. We looked at the individual traces and confirmed that it is indeed the case. However, a point to note in this context is that we have not used timing information from the traces. In reality, if a regular packet repeats multiple times but the interval between repetitions is large say above 100 ms, then it will never result in a false positive. In our evaluation, it will result in a false positive.
- A value of RDC below 2 can generate quite a few false positives. However, false positives can easily be tackled with a small change at the sender. Whenever the sender sees a regular packet that corresponds to an alphabet, it can either pad or truncate it to make it a nonalphabet. The false positive number can also be reduced if we choose a smaller threshold percentage when extracting the alphabet. This will, however, reduce the alphabet size.
- The g mode alphabet seems to be slightly more robust to false positives and negatives, at least for the PSC-CSE trace. Within a mode, the higher data rates seem to be more robust. This is to be expected because higher data rates and g mode operation gives a relatively uncluttered run length space from which to extract the alphabet.

In summary, the above results look encouraging. We are able to extract a sufficiently large sized alphabet set for communication. Our validation shows that communication can work effectively in practice with very few false positives and false negatives.

6.3 An Example Case Study: Energy Savings in Rural WiFi Mesh Networks

To illustrate the advantage of the Esense framework, we briefly outline its application and the performance improvement it achieves when applied in rural mesh networks.

Application Scenario. WiFi mesh networks have been employed in rural villages to provide Internet connectivity [30], [31]. Given the power scarcity in these settings, the WiFi nodes (routers) that form the mesh network often run on batteries, making the issue of energy savings very important. Esense can be effectively employed in these settings by equipping each WiFi router with an extremely low power, low-cost transceiver (like tmote SKY, CC2520 module [32]) that operates in the same frequency band as the WiFi router. In this case, the WiFi router can be put-to-sleep/powered-off when idle and remotely woken up by its WiFi neighboring when required via Esense messages, which are received on the low-power mote hardware, which in turn will wake-up the high power WiFi.

Protocol Description. In the mesh network, a given WiFi node (router) offers services to two types of nodes: 1) a client node, which is typically a user device who access the Internet via the WiFi node and 2) neighboring WiFi router, which routes its own client and downstream traffic multi-hop via the given WiFi node.

Any WiFi node is associated with an Esense identity, basically an Esense alphabet, which is unique among its carrier-sensing neighborhood. During initial handshake mechanisms, this node communicates its identity to its one-hop neighbors (clients or other WiFi routers). When a WiFi node detects no incoming traffic for a threshold period of time, it concludes itself to be idle, activates the low-power hardware and puts itself to sleep or powers-off.[6]

A neighbor, who wishes to use the services of the sleeping WiFi node, uses the Esense framework to wake it up, i.e., the neighbor sends the Esense energy burst corresponding to the sleeping node's id multiple times (dictated by SRC) to wake it up. The low-power hardware is configured to detect the id and on seeing the id a given number of times (RDC), wakes up its associated sleeping WiFi node.

Performance Improvement. We have implemented a prototype of this setup as shown in Fig. 7. The mote on sensing an Esense message uses its I/O pin to trigger the power

6. Some WiFi nodes do not permit sleep operation. If permitted, there is a tradeoff in energy savings versus wake-up delay.

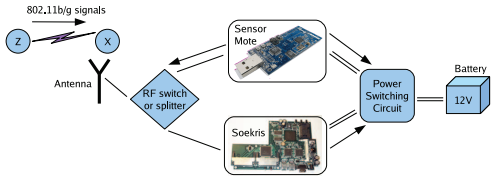


Fig. 7. Prototype implementation.

switching circuit to supply power to the WiFi router (Soekris Box). Validation of this platform is no different from the results obtained in Table 8. Hence, we do not elaborate on this aspect further.

The amount of energy savings that can be achieved in this setup is a function of the idle time in the network. Previous studies [4], [19], [5] have quantified such energy savings when employing secondary radios in WiFi infrastructure networks using traffic traces. Similar such traces are unavailable for rural settings for a detailed study. However, it is not difficult to see that the energy savings at a node is more or less given by the ratio of total time over active time of that node. For example, suppose a node is active serving traffic only 10 hours a day, i.e., it is idle remaining 14 hours. Also suppose that when powered-on it consumes 5 watts (for WiFi nodes, this value is typically between 5-10 W) and when powered-off it consumes 50 mW (typical of low power platforms). Then, without Esense framework, the energy consumed is $5 * 24$ watt-hour in a day. With Esense framework, the energy consumed is $5 * 10 + 0.05 * 14$ watt-hour. Thus, the energy saving ratio is $5 * 24/(5 * 10 + 0.05 * 14)$ or approximately $24/10 = 2.4$. Active times rarely exceed 10 hours in the rural settings based on our experience. Esense basically saves energy whenever feasible, thus achieving optimal savings.

Compared to the case where a WiFi node is on all the time, turning it off or putting it to sleep, takes a hit in terms of increased service delay (time taken to access the services of the node). This delay will be made up of time to convey the Esense message and time to boot/bring it out of sleep. If we were to send the Esense message 10 times, assuming a worst case access delay of 50 ms for each of these messages, the Esense delay accounts for 500 ms. The boot time can range from 10-30 seconds depending on the platform employed. Bringing a node out of sleep, typically takes less than a second.

Depending on the configuration (making node sleep or power-off), the service delay can range from under a second to few tens of seconds. Note that this delay comes into play only at the beginning of a connection and can be tolerated in most application settings in rural villages.

7 Discussion

7.1 Architectural Aspects

While we briefly touched upon the architecture and implementation of the various scenarios of Esense usage, the full details are context specific and beyond the scope of this paper. However, we mention two important aspects. First, at what layer does Esense belong? We view Esense mainly as a MAC layer functionality. At the sender side, driver level changes are adequate. Essentially, the sender needs to assemble the right sized Esense packet based on the alphabet to convey, and if necessary to pad regular packets to avoid generating false positives. At the receiver side, we need appropriate interrupts to the MAC layer from the physical layer indicating channel occupancy information. 802.15.4 radios expose a pin (CCA), which can be interfaced with a microcontroller (that implements the MAC) for this purpose. However, current 802.11b/g drivers do not have access to such an interface from the radio but 802.11n drivers do.

The second important aspect is to do with alphabet negotiation (what alphabet set to use?) and meaning attribution (what does an alphabet convey?). These are again very scenario specific. For some scenarios (such as coexistence, interference map), the alphabet set/meaning has to be decided a priori and configured (software level) into the various nodes that are part of the Esense framework. An online update procedure is possible, provided there is a secondary channel (say Ethernet) and the update frequency is small. For the energy saving scenarios, the alphabet update information can be conveyed over the regular 802.11 network (e.g., during association or before a node powers downs its WiFi interface).

7.2 Applicability in Other Contexts

We have evaluated our ideas in the context of 802.11b/g and 802.15.4 standards and considered only unidirectional communication. We wish to emphasize that these ideas are general enough that they can potentially be used with other standards, for bidirectional communications and possibly in other scenarios (including wired networks). Esense does need some PHY layer support, i.e., a given technology should provide the capability of sensing energy at good accuracy in a specific frequency band. For example, one could use this framework to achieve energy savings in 802.11a networks, construct interference maps of Wimax networks.

7.3 Building Vocabulary

We have not looked at building vocabulary on top of the base alphabet. Such an approach may be necessary when the message space exceeds the base alphabet size. For example, if one were considering energy savings in 802.11g infrastructure mode, with the use of only the base alphabet set, it would be quite constraining. The alphabet size of 10 means that Esense-based wake-up via the secondary radio is possibly for only 10 clients at a time. Further, the false positive (negative) rate can be reduced significantly by considering an extended vocabulary: a specific pattern of alphabets will be more difficult to generate (miss) than a single alphabet. However, delineating word boundaries in presence of insertions/deletions of alphabets on the channel is not a straightforward task. The insertions on the channel correspond to regular packets being misinterpreted as Esense packets. The deletions on the channel correspond to the case where the hardware is unable to detect the alphabet due to merging of packets. Such insertion/deletions on channels have been looked in the context of error correction codes (see [33]). We believe this is an interesting avenue for future exploration.

7.4 Packet Size Distribution

The size of the alphabet has a strong correlation with the packet size distribution. Supposing the packet size distribution is no longer bimodal, which could be the case if the

traffic is dominated by video traffic. This can then potentially result in a very small alphabet set. This can be overcome by a couple of approaches. One approach would be to go for a higher accuracy hardware. A second approach would be to build a vocabulary on top of the limited alphabet. The third approach would be to still stick to a larger alphabet set but manipulate the size of the regular packets. For example, if a WiFi node sees a regular packet that has a size that corresponds to the alphabet, it can either fragment the packet or pad the packet to make it a nonalphabet.

8 Conclusions

In this paper, we consider a new method of communication between devices that cannot interpret the individual bits of the packet. However, we facilitate their communication by sensing and interpreting energy patterns on the air. We describe how this framework opens up new approaches to solving problems in at least three distinct research domains. We validate our mode of communication on a hardware platform using realistic traces from WiFi deployments. Our evaluation shows that our approach can lead to sufficiently large alphabet set that can facilitate effective communication. We believe that this framework has implications in other areas beyond the three example scenarios we have presented.

References

[1] "IEEE P802.11, The Working Group for Wireless LANs," http://grouper.ieee.org/groups/802/11, 2013.

[2] "IEEE 802.15 WPAN Task Group 4 (TG4)," http://www.ieee802.org/15/pub/TG4.html, 2013.

[3] "Bluetooth Special Interest Group," http://www.bluetooth.org, 2013.

[4] 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.

[5] 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.

[6] D. Niculescu, "Interference Map for 802.11 Networks," *Proc. Seventh ACM SIGCOMM Conf. Internet Measurement (IMC),* 2007.

[7] T.G. Handel and M.T. Sandford, "Hiding Data in the OSI Network Model," *Proc. First Int'l Workshop Information Hiding,* 1996.

[8] N. Mishra, K. Chebrolu, B. Raman, and A. Pathak, "Wake-on-WLAN," *Proc. Int'l Conf. World Wide Web (WWW '06),* May 2006.

[9] J. Choi and K.G. Shin, "Out-of-Band Sensing with ZigBee for Dynamic Channel Assignment in On-the-Move Hotspots," *Proc. IEEE Int'l Conf. Network Protocols (ICNP),* Oct. 2011.

[10] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "ZiFi: Wireless LAN Discovery via ZigBee Interference Signatures," *Proc. ACM MobiCom,* Sept. 2010.

[11] A. Sikora, "Compatibility of IEEE802.15.4 (ZigBee) with IEEE 802.11 (WLAN), Bluetooth, and Microwave Ovens in 2.4 GHz ISM-Band," technical report, Steibeis-Transfer Centre, http://www.ba-loerrach.de, Sept. 2004.

[12] S. Pollin, M. Ergen, A. Dejonghe, L.V. Perre, F. Catthoor, I. Moerman, and A. Bahai, "Distributed Cognitive Coexistence of 802.15.4 with 802.11," *Proc. First Int'l Conf. Cognitive Radio Oriented Wireless Networks Comm. (CROWNCOM),* 2006.

[13] C. Won, J.H. Youn, H.A. Sharif, and J. Deogun, "Adaptive Radio Channel Allocation for Supporting Coexistence of 802.15.4 and 802.11b," *Proc. IEEE Vehicular Technology Conf. (VTC),* 2005.

[14] S. Han, S.L.S. Lee, and Y. Kim, "Channel Allocation Algorithms for Coexistence of LR-WPAN with WLAN," *IEICE Trans. Comm.,* vol. 91, pp. 1627-1631, May 2008.

[15] C.-J.M. Liang, N.B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi Interference in Low Power ZigBee Networks," *Proc. Eighth ACM Conf. Embedded Networked Sensor Systems (Sensys),* Nov. 2010.

[16] S. Gollakota, F. Adib, D. Katabi, and S. Seshan, "Clearing the RF Smog: Making 802.11 Robust to Cross-Technology Interference," *Proc. ACM SIGCOMM,* 2011.

[17] J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond Co-Existence: Exploiting WiFi White Space for ZigBee Performance Assurance," *Proc. IEEE 18th Int'l Conf. Network Protocols (ICNP),* Oct. 2010.

[18] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown," *Proc. ACM MobiCom,* 2002.

[19] Y. Agarwal, C. Schurgers, and R. Gupta, "Dynamic Power Management Using On Demand Paging for Networked Embedded Systems," *Proc. Asia South Pacific Design Automation Conf. (ASP-DAC),* 2005.

[20] J. Padhye, S. Agarwal, V.N. Padmanabhan, and L. Qiu, "Estimation of Link Interference in Static Multi-Hop Wireless Networks," *Proc. ACM Fifth ACM SIGCOMM Conf. Internet Measurement (IMC),* 2005.

[21] S.M. Das, D. Koutsonikolas, Y.C. Hu, and D. Peroulis, "Characterizing Multi-Way Interference in Wireless Mesh Networks," *Proc. First Int'l Workshop Wireless Network Testbeds, Experimental Evaluation and Characterization (WINTECH),* 2006.

[22] D. Niculescu, "Interference Map for 802.11 Networks," *Proc. Seventh ACM SIGCOMM Conf. Internet Measurement (IMC),* 2007.

[23] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-Based Models of Delivery and Interference in Static Wireless Networks," *Proc. ACM SIGCOMM,* 2006.

[24] H. Yadav, "Energy Sense Based Coordinated Co-Existence," master's thesis, IIT Bombay, June 2011.

[25] "A Community Resource for Archiving Wireless Data At Dartmouth," http://crawdad.cs.dartmouth.edu, 2013.

[26] "Dataset of Wireless LAN Traffic around Portland, Oregon Using a Commercial Sniffer VWave," http://crawdad.cs.dartmouth.edu/pdx/vwave, 2007.

[27] "Traces of Network Activity at OSDI 2006," http://crawdad.cs.dartmouth.edu/microsoft/osdi2006, 2006.

[28] "Traces of the Stanford CS Department's Wireless Network," http://crawdad.cs.dartmouth.edu/stanford/gates, 2003.

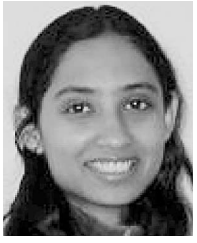
[29] T. Sakurai and H.L. Vu, "MAC Access Delay of IEEE 802.11 DCF," *IEEE Trans. Wireless Comm.,* vol. 6, no. 5, pp. 1702-1710, May 2007.

[30] B. Raman and K. Chebrolu, "Experiences in Using WiFi for Rural Internet in India," *IEEE Comm. Magazine,* Special Issue on New Directions in Networking Technologies in Emerging Economies, vol. 45, no. 1, pp. 104-110, Jan. 2007.

[31] K. Chebrolu and B. Raman, "FRACTEL: A Fresh Perspective on (Rural) Mesh Networks," *Proc. ACM Workshop Networked Systems for Developing Regions (NSDR),* Sept. 2007.

[32] "2.4 GHz ZigBee/IEEE 802.15.4 RF Transceiver," http://www.ti.com/product/cc2520, 2013.

[33] L.J. Schulman and D. Zuckerman, "Asymptotically Good Codes Correcting Insertions, Deletions and Transpositions," *IEEE Trans. Information Theory,* vol. 45, no. 7, pp. 2552-2557, Nov. 1999.



Kameswari Chebrolu received the MS and PhD degrees in electrical and computer engineering from the University of California at San Diego in 2001 and 2004, respectively. She is currently an associate professor with the Department of Computer Science and Engineering at the Indian Institute of Technology, Bombay, India. Her research interests are in the areas of wireless network architecture, protocol design, and analysis. She is specifically interested in technology for developing regions.

Ashutosh Dhekne received the Master of Technology degree in computer science and engineering from IIT Bombay in 2009. He currently works for Securifi Embedded Systems at Hyderabad and is working on WiFi and home automation. He is interested in various aspects of wireless communications from protocols to practical applications.