Esense: Communication through Energy Sensing

Kameswari Chebrolu
Department of Computer Science
IIT Bombay, Mumbai, India
chebrolu@cse.iitb.ac.in

Ashutosh Dhekne
Department of Computer Science
IIT Bombay, Mumbai, India
ashudhekne@gmail.com

ABSTRACT

In this paper, we present Esense: a new paradigm of communication between devices that have fundamentally different physical layers. The same communication framework also works between devices that have the same physical layer, which are out of communication range but within carrier-sense range. 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 build an "alphabet set": a set of signature packet sizes which can be used for Esense. For this, we take a measurement based approach by considering WiFi traces from actual deployments. We then analyze the channel activity resulting from these traces and build an appropriate alphabet set for Esense communication. Our results show that we could potentially construct an alphabet of size as high as 100; such a large alphabet size promises efficient Esense communication. We also validate this alphabet set via a prototype implementation, and show that effective communication is indeed feasible even when both sides use different physical layers.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Design, Algorithms, Experimentation, Performance

Keywords

802.11, 802.15.4, Communication, Energy Sensing

MobiCom'09, September 20–25, 2009, Beijing, China.

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 [4], IEEE 802.15.4 [3], Bluetooth [2] 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 since 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** since 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. The three problems and solution approaches are illustrated in Fig. 1, and summarized in Table. 1; we describe these below.

- **Coordinated Coexistence:** With more standards emerging for operation within limited spectrum, coexistence between the standards becomes a very important issue. Current practices tend to mitigate the problem by frequency hopping or assessing the amount of interference in a given channel and choosing a channel that has minimal interference. While this helps to some extent, performance suffers based on the extent of interference. 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. Such explicit communication can also permit nodes in the WPAN to duty-cycle, saving energy till their next activity period.

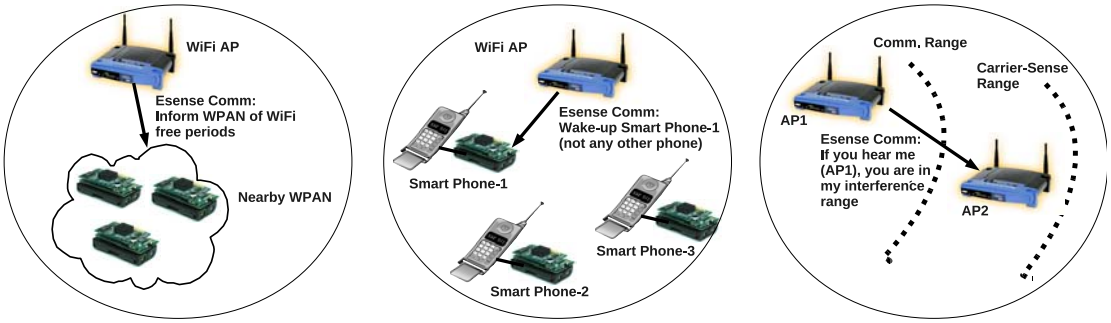

Figure 1: Esense application in three example scenarios

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

Table 1: Esense communication details for the three example scenarios

- **Energy Management:** WiFi enabled devices such as smart phones, laptops 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. Turning the interface or the entire device off naturally conserves more power, however it makes the task of waking up the device when necessary (e.g. incoming VoIP call, forwarding traffic) that much more difficult. A solution that helps in this setting is to employ an extremely low power secondary radio (such as 802.15.4) that operates in the same frequency band as WiFi but consumes significantly lesser power than WiFi. This secondary radio can then listen for WiFi communication from the other end, interpret any incoming WiFi communication, and then turn on the WiFi device 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 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 since it cannot interpret the incoming packets. The Esense framework of communication can help in these settings by clearly identifying the nodes responsible for this interference.

Our energy based framework achieves its task of communication by constructing an *alphabet set* of packet sizes; i.e., mapping an Esense message (or sequence of bits) to an appropriate energy burst duration. With this approach, 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 in this paper; as a proof of concept of the Esense framework, we enable uni-directional communication from an IEEE 802.11 radio to an 802.15.4 radio. However we emphasize that Esense is equally applicable to enable bi-directional 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 uni-directional 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).

Analysis of the packet size traces of many WiFi installations reveals a packet size distribution that is mostly bi-modal. Given this observation, a possible mechanism to distinguish between the Esense and regular packets is to assign those packet sizes to Esense packets that do not normally occur in practice.

Since the frequently occurring packet sizes (in regular packets) are few in number, it appears on first glance that is possible to assign all the remaining packet sizes to Esense's alphabet set. However this is not as straightforward in practice. 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 (upto 54 Mbps) make the problem of energy detection **practically** challenging.

In addressing this challenge, we take a measurement based training approach to construct the alphabet set corresponding to Esense packets. For the training, we use publicly available traces of WiFi deployments. We then characterize

the channel activity such traces create on 802.15.4 hardware. We use a prototype implementation of Esense, on the CC2420-based Tmote Sky platform. The use of the prototype during the training process itself, naturally incorporates the limitations of the 802.15.4 hardware. From the observed channel activity profile, we exclude channel busy run lengths that occur at high frequency and use the remaining run length space to construct the alphabet set for the Esense packets. We then validate the use of the constructed alphabet set, using the prototype implementation and further WiFi traces (different from those used for training).

Our overall contributions in this paper are three fold. 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 via an actual implementation which supports a uni-directional communication from 802.11 to 802.15.4 devices.

The rest of the paper is organized as follows. In the next section (Sec. 2), we describe related work. Sec. 3 presents the background for the considered problem along with the detailed solution approach. We then present results of our evaluation in Sec. 4. Sec. 5 presents a few points of discussion and Sec. 6 concludes the paper.

2. RELATED WORK

To our knowledge, the framework of communication through energy sensing is quite novel. Some aspects of Esense resemble steganography, where information is concealed in otherwise normal looking messages. Steganography has been applied to networking protocols [12], 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.

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 (ISM) band is used by a variety of standards such as IEEE 802.11b/g [4], IEEE 802.15.4 [3] and Bluetooth [2]. Coexistence studies [21] have shown that there could be considerable interference between the different standards resulting in as high as 90% frame loss rate. The most common solution [17, 22, 11] 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 (22MHz) as opposed to WPAN channels (5MHz), it would be difficult to get away from interference. 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 feasible.

Energy conservation in WiFi based devices has received considerable attention. The IEEE 802.11 standard [4] supports a power save mode (PSM) which permits the WiFi interface to duty-cycle. There are other MAC layer techniques [13] that build on the PSM to achieve further energy-savings. However, as shown in [8], 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 [20, 9] 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 [14] is another solution that 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 since the router board itself consumes considerable energy when idle (5W). 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 [8] 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 smart phones. 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 WiFi interface is off or the entire end-device is off. Second, it does not suffer from the range mismatch problem since the secondary radio directly senses the WiFi-energy. This is possible because the receiver sensitivity of the secondary radio (-95dBm) is better than the necessary WiFi received signal strength at the end device (typically -90dBm @1 Mbps). Third, our solution needs software changes at the WiFi sender side and integration of a very low cost ($10), low power (60mW) 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 [16, 10, 15] in this domain rely on building the map by conducting individual and pair-wise broadcast measurements of order O(N^2), where N is the number of nodes in the network. [18] reduces the number of measurements to O(N) by considering only individual broadcast measure-

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

ments 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 down-time to conduct the experiments ([16] 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 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) and any receiver node R can process this identity information to figure out the number of interfering nodes and at what energy 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 etc.

3. THE ESENSE COMMUNICATION FRAMEWORK

We now present the overall Esense framework. We first develop the basic idea, then present practical limitations, and subsequently our approach to Esense given the limitations.

3.1 Esense: overall approach

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, since it is highly dependent on the external environment which could be dynamic. Similarly, the gap between energy bursts is also not a good parameter since 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 since it can be controlled at a given sender. In this paper, we develop a framework for communication based on modulating the energy duration.

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. For any message exchange, we need a base alphabet, and a set size of two is a minimum requirement for building a vocabulary (i.e. sequences of alphabets).

Now, the larger the alphabet set, the more efficient and less complex the communication framework would be. For example, in the energy-savings scenario described in Sec. 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. 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. Further one needs to allow for the media access latency of each of the individual packets. Hence there is merit in seeking a large alphabet set.

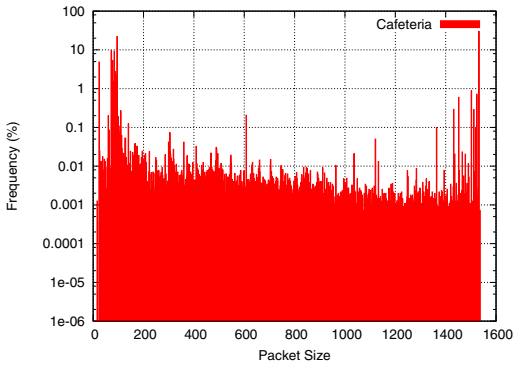

Figure 2: Packet size distribution of cafeteria trace

In an operational network, there may normally 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. That is, we may have a false positive. Thus, in choosing the Esense alphabet set, while we must seek a large set, we must also seek to minimize the false positive rate.

Packet sizes in WiFi traces: As proof of concept of the Esense framework, we enable uni-directional communication from an 802.11 radio to an 802.15.4 radio. As noted in Sec. 1, this hardware choice applies in the three scenarios listed. Hence, we attempt to construct the alphabet set based on the communication patterns predominant in WiFi networks. There are a variety of WiFi traces available in the public domain [1]. We identify five representative traces to study the communication patterns; these are traces from various WiFi infrastructure mode deployments [2]. The five traces are: Cafeteria (Powells), Library, PSU-CSE department (all from [7]), OSDI Conference [6], Stanford CSE department [5].

An analysis of the packet size distribution for two of the five traces is depicted in Fig. 2 and 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 ACKs, beacons, management frames. Or they are around 1500 byte packets corresponding to the Ethernet MTU. Such a bimodal distribution is a well know observation in the Internet as well; we confirm that the same applies in WiFi networks too.

A possible approach for Esense: The above observation suggests a possible approach for Esense. As far the 802.15.4 receiver node is concerned, it detects the energy

[2]We were not able to procure a trace of an operational mesh network since these networks are still mainly used for experimental purposes.

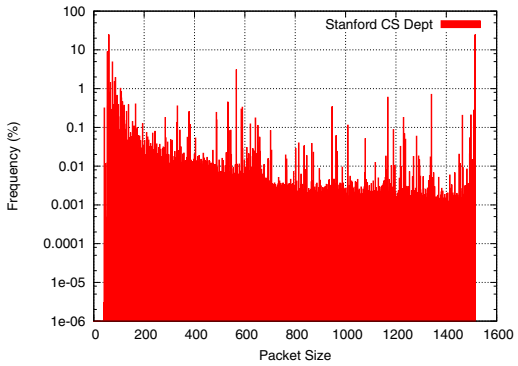

Figure 3: Packet size distribution of Stanford trace

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 and further to make sense of the message content of the Esense packets. Suppose we assume that the 802.15.4 node can detect channel occupancy with high accuracy i.e. it can detect any packet size upto 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 since 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, since the underlying technology has a maximum packet size (MTU). So, we bound the alphabet set by the maximum packet size feasible in 802.11 i.e. 2304 bytes.

Suppose we choose a threshold percentage of 1%. From the traces, if we were to count the packet sizes with frequency greater than 1%, this number turns out to be under 16 for all the traces. Thus the base alphabet for the Esense packets turns out to be 2288, which is quite large.

Detecting Esense alphabets: In this scheme, the task of the receiving 802.15.4 node would be to measure the duration of energy on the channel and subsequently map it to a corresponding packet size. If it sees a packet size that does not belong to the alphabet set, it considers it as a regular packet and rejects it. If it sees a packet that belongs to the alphabet set, it considers it as an Esense packet and the measured packet size corresponds to an Esense alphabet. Of course, in detecting the alphabets, we could have a small number of false positives; we characterize this in our evaluation.

In this base scheme, we can have as many Esense messages as the size of the alphabet set. We can enhance this by building a vocabulary (sequence of alphabets) from the alphabet set. This paper does not consider vocabulary construction in depth.

3.2 Practical Limitations

In reality, the above described scheme faces two main practical challenges. (1) The 802.15.4 Esense receiver can only sense the energy burst duration; mapping this to a packet length is not straightforward 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, upto 54 Mbps. (2b) Also, the inter-packet gaps in an operational system are unpredictable due to the very nature of the CSMA/CA protocol. These gaps could be as small as 50μs for 802.11b and 28μs for 802.11g.

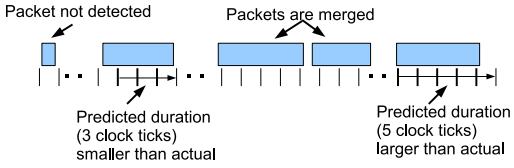

Figure 4: Illustration of 802.15.4 hardware limitations

The first issue implies that we cannot quite use packet sizes to encode Esense alphabets. The second issue, that of inaccuracy in channel occupancy estimation, may arise due to two reasons: (a) finite granularity of time measurement, and (b) finite sampling granularity. The inaccuracy issue manifests in many ways, illustrated in Fig. 4. First, when the channel occupancy is say x 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 μs (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.

We characterize the accuracy of 802.15.4 radio in detail in Sec. 4. For now, we outline the procedure we follow to address the above limitations.

3.3 Practical Alphabet Extraction Algorithm

We address the issue of multiple 802.11 data rates, by using energy burst lengths directly as Esense alphabets, instead of using packet sizes as alphabets [3]. We take a measurement based approach to determining the base alphabet for Esense.

The hardware we employ is the CC2420-based Tmote Sky. It uses a relatively impoverished microcontroller, the MSP430, but it is good for low-power operation. The microcontroller is only 8 MHz, and has 40 KB of ROM, and 10 KB of RAM.

The time measurement granularity on this platform is 1/32 KHz, or approximately 30.5 μs, since it uses a 32 KHz oscillator. We shall refer to this as one clock *tick* henceforth. In the Esense prototype based on this platform, we poll the

[3]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.

CCA pin of the CC2420 chip "continuously". We note that since the microcontroller is slow, the polling has a finite time granularity.

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. This process needs to address the following four 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 1500 byte packet produces a run length of about 8 ticks and the maximum size packet of 2304 byte produces a run length of 12 ticks. So, there are about 4 run lengths that can be considered as potential alphabets (assuming all packets larger than 1500 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 2304 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 2304 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.

[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.

2. *Building-in margins between alphabets:* Now, one could consider all the run lengths in the complement set as potential alphabets. But this is not a good idea since this does not account for the measurement inaccuracies of the 802.15.4 hardware. That is, a given packet size (at a given transmit rate) could be measured as any one of a set of run lengths. So, in our algorithms, we choose a *margin*, and ensure that for any run length chosen as an alphabet, there is a gap of at least the "margin" ticks between this alphabet and any adjacent alphabet. This gap is also maintained between an alphabet and a regular packet whose frequency exceeds the threshold percentage of frequency. The actual value of the margin is determined by 802.15.4 hardware characterization.

   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.

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.

4. EVALUATION

We first present our experimental setup, in Sec. 4.1. We then describe the accuracy characterization of the 802.15.4 platform in Sec. 4.2. This then leads us to experimentally determining the Esense alphabet set for 802.11 to 802.15.4 communication in Sec. 4.3. We finally validate the chosen Esense alphabet set and evaluate the effectiveness of Esense communication in Sec. 4.4.

4.1 Experimental Methodology

In all of our experiments, we use a WiFi radio setup as the sender and an 802.15.4 radio (on the Tmote Sky platform) as the 802.15.4 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", since this is the platform commonly used in many Wireless Sensor Network (WSN) applications.

The 802.15.4 receiver mote 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 changed. This log can then be used to calculate the run length in clock ticks of the channel busy time.

For our experiments to 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 2μs.

For our other experiments, we need to emulate WiFi activity as specified by the different traces. Now, 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 [5].

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. This is a worst case assumption as far as our setting is considered: in reality the spacing between the packets will be more than what we consider, and any extra inter-packet spacing can only help the mote detect the packets better.

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 since the mote's memory runs out if we continuously send packets.

4.2 Mote Accuracy Characterization

In order to characterize the accuracy of the mote, 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 5000 times, with inter-packet spacing set to a sufficiently large value of 50ms. 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.

Busy Time	*Value1*	*Value2*	*Value3*
123 μs	122 ; 8.84%	152.5 ; 74.5%	183 ; 16.62%
785 μs	762.5 ; 1.7%	793 ; 63.84%	823.5 ; 34.46%
4710 μs	4697 ; 26.6%	4727.5 ; 67.38%	4758 ; 5.94%

Table 2: Mote characterization: margin of error

Table. 2 shows the findings of the mote when the actual channel occupancy duration was set to 123, 785 and 4710μs. The columns represent the first, second and third highest

[5]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

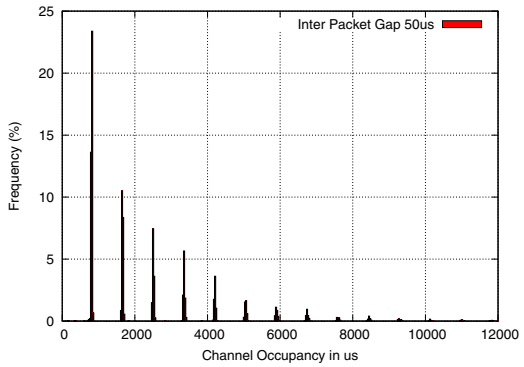

Figure 5: Channel occupancy as reported by the mote at 50μs inter-packet gap (actual channel occupancy: 785μs)

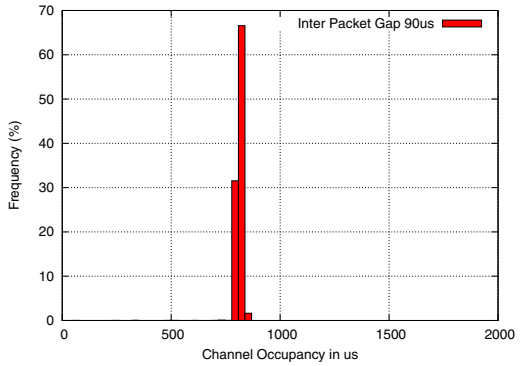

Figure 6: Channel occupancy as reported by the mote at 90μs inter-packet gap (actual channel occupancy: 785μs)

frequency components of the mote measurements. Note that the mote reports its findings in clock ticks. We multiple this with 30.5μs to arrive at the numbers in the table.

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 2 clock ticks i.e. 61μs. 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 785μs. We consider different spacings: 10, 30, 50, 70 and 90 μs by appropriately varying the AIFS value. Fig. 5 and Fig. 6 present the results for the 50μs and 90μs cases respectively. Note that x-axis are different in the two cases.

At 50μs inter-packet 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 ex-

periment corresponds to one where 15 packets got merged, though this occurs very rarely at 0.1%. At 70μs (not shown in the figure), the largest run length we observe corresponds to one where 5 packets got merged and this happens at a frequency of 0.5%. At 90μs the mote is able to clearly separate out the individual packet as is evident from the figure.

According to the 802.11b standard, the minimum separation between the packets is at least 50μs. With backlogged queues, the average separation would be about $50+16\times20 = 370\mu s$ (assuming a CWMin of 32). So, in most cases the mote can separate out the packets. With respect to 802.11g mode, if it has to support legacy 802.11b nodes, the same parameters as above apply. However it can also operate with a smaller slot time of 9μs (as opposed to 20μs in 802.11b). This then gives a minimum separation of 28μs and average separation of 172μs. So, the mote can still perform well in this setting.

In summary, the mote reports channel occupancy time with a margin of error which is ±2 clock ticks (i.e. $\pm61\mu s$). And it can separate out the packets clearly only when they are separated by at least $90\mu s$.

4.3 Alphabet Extraction

We now focus our attention on extracting the alphabet based on detailed measurements carried out with the traces. 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. Now, 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 auto-rate adaptation.

Fig. 7 shows the findings of the mote for the Cafeteria trace at 4 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 bi-modal distribution which we saw in the packet lengths, is preserved in the run-length distributions, upto 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 etc) 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 since we assume backlogged packets in our emulation.

Third, if we were to consider only run lengths with frequency greater than 1% 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

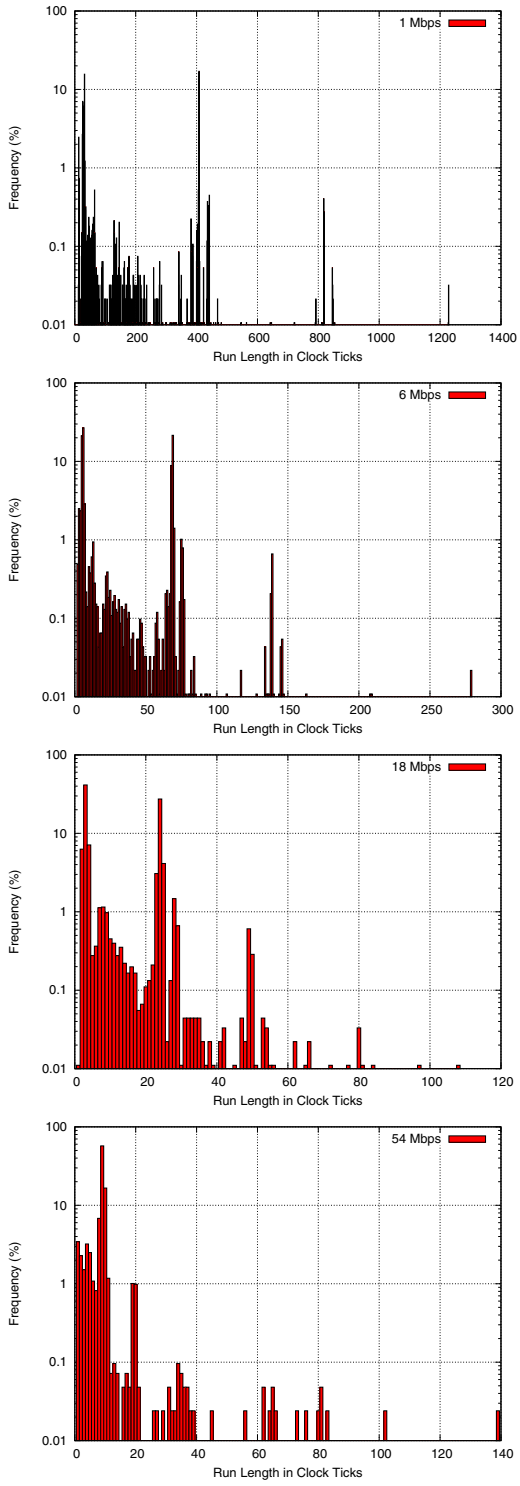

Figure 7: Cafeteria trace: Run Lengths at 1, 6, 18, 54 Mbps

chunk of run lengths available to the right of the second highest frequency component. This chunk starts at 410 ticks @ 1 Mbps, 70 ticks @ 6 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. 7 only for the Cafeteria trace, we plotted similar graphs for the other traces too, and observed similar findings.

Now, for the actual alphabet extraction, for each of the five traces, we considered six different data rates at which the packets were 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 in Sec. 3. The algorithm uses a margin of 2 ticks, obtained in our mote characterization experiments in Sec. 4.2. We use the value of 1% 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. From Fig. 7, most run lengths below 1% occur at a very low frequency.

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 Mbps 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: 5 traces along rows, 6 data rates along columns. For the 6, 18, 36, and 54 Mbps columns, the values without brackets correspond to the b mode of operation and values within correspond to the g mode of operation.

In the table, we have a last row labeled "All traces", and a last column labeled "All rates". The last row corresponds to alphabet extraction for the case where we eliminated run-lengths which exceeded the threshold percentage of 1% in *any* of the traces (i.e. in at least one of the traces). Similarly, the last column corresponds to alphabet extraction where we eliminated run-lengths which exceeded the 1% 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" row is only slightly different from the rows for the 5 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" column is only slightly different from the columns for the 6 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 towards 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 2304 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%, 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%, 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%. 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) since we are considering intersection across a much larger set.

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)			

Table 4: Cafeteria trace: all data rates

The above results in terms of alphabet size could improve considerably if we worked with an 802.15.4 hardware that is more accurate[6].

4.4 Validation

We now validate our communication framework using the above extracted alphabet. We consider the alphabet set corresponding to the conservative calculation since 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 Mbps and 18 Mbps and one variable data rate. For the fixed data rates, all regular packets with the exception

[6]Note that in the process of building more accurate hardware, one should not lose the original low-cost and low-power advantages of 802.15.4.

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

Table 3: Alphabet size

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. 5 shows the number of false positives and false negatives that occur in the various cases. 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 corresponds to the pair (SRC, RDC). We make the following observations from the table.

- For a given SRC, increasing RDC results in lesser number of false positives but higher number of false negatives. 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 trade-off in choosing 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% of the time) repeat at least 6 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 non-alphabet. 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 since 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.

5. DISCUSSION

5.1 Architectural Aspects

The architecture and implementation details of the various scenarios of Esense usage, are context specific and beyond the scope of this paper. However, we touch upon two important aspects. First, at what layer does Esense belong?

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
CSE-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

Table 5: False positives and negatives: 250 Alphabets, 30,000 Regular Packets

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 micro-controller (that implements the MAC) for this purpose. However, 802.11 drivers do not have access to such an interface from the radio in current off-the-shelf platforms.

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 co-existence, interference map), the alphabet set/meaning has to be decided apriori 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).

5.2 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 be used with other standards, for bi-directional communications and possibly in other scenarios (including wired networks). For example, one could use this framework to achieve energy savings in 802.11a networks, construct interference maps of Wimax networks, explore coexistence between Bluetooth and 802.11. All that is needed is the capability of sensing energy at good accuracy in a specific frequency band [7]. One could make do with current implementations if they export such sensing functionality or design new hardware that does the job with adequate accuracy.

5.3 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. 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 [19]). We believe this is an interesting avenue for future exploration.

5.4 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 bi-modal, 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

[7]For frequency hopping systems, the receiver needs to pause on a specific channel for Esense communication. For its own internal network data communication, it can however hop.

the packet or pad the packet to make it a non-alphabet. Note that this procedure would have to be followed by all the WiFi-nodes that are part of the system.

5.5 Possible consideration for Esense in future standards

We believe that there is a strong case for having native support for Esense in future wireless standards. A wireless standard, when it is designed, cannot predict what other future standard is going to operate in the same spectrum space. It could incorporate support for Esense, for instance, by means of reserving certain packet lengths or certain energy run-lengths. This could then be used for any intelligent Esense-based coordination with any future wireless technology in the same spectrum. Further, the kind of scenarios we envision for Esense communication are very low data rate, hence this coordination should have minimal effect on capacity of individual networks.

And we believe that such coordination is often very low data rate (as evident in the three scenarious listed) that its affect on the capacity of the individual standards will be very minimal.

6. 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.

7. REFERENCES

[1] A Community Resource for Archiving Wireless Data At Dartmouth. `http://crawdad.cs.dartmouth.edu/`.

[2] Bluetooth Special Interest Group. `http://www.bluetooth.org/`.

[3] IEEE 802.15 WPAN Task Group 4 (TG4). `http://www.ieee802.org/15/pub/TG4.html`.

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

[5] Traces of the Stanford CS department's wireless network. `http://crawdad.cs.dartmouth.edu/stanford/gates`, 2003.

[6] Traces of network activity at OSDI 2006. `http://crawdad.cs.dartmouth.edu/microsoft/osdi2006`, 2006.

[7] Dataset of wireless LAN traffic around Portland, Oregon using a commercial sniffer VWave. `http://crawdad.cs.dartmouth.edu/pdx/vwave`, 2007.

[8] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless Wakeups Revisited: Energy Management for VoIP over Wi-Fi Smartphones. In *MobiSys*, 2007.

[9] Y. Agarwal, C. Schurgers, and R. Gupta. Dynamic Power Management Using On Demand Paging for Networked Embedded Systems. In *ASP-DAC*, 2005.

[10] S. M. Das, D. Koutsonikolas, Y. C. Hu, and D. Peroulis. Characterizing multi-way interference in wireless mesh networks. In *WINTECH*, 2006.

[11] S. Han, S. L. S. Lee, and Y. Kim. Channel Allocation Algorithms for Coexistence of LR-WPAN with WLAN. *IEICE Transactions on communications*, May 2008.

[12] T. G. Handel and M. T. Sandford. Hiding data in the OSI network model. In *First International Workshop on Information Hiding*, 1996.

[13] R. Krashinsky and H. Balakrishnan. Minimizing Energy for Wireless Web Access with Bounded Slowdown. In *MobiCom*, 2002.

[14] N. Mishra, K. Chebrolu, B. Raman, and A. Pathak. Wake-on-WLAN. In *WWW'06*, May 2006.

[15] D. Niculescu. Interference Map for 802.11 Networks. In *IMC*, 2007.

[16] J. Padhye, S. Agarwal, V. N. Padmanabhan, and L. Qiu. Estimation of link interference in static multi-hop wireless networks. In *IMC*, 2005.

[17] 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. In *CROWNCOM*, 2006.

[18] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *SIGCOMM*, 2006.

[19] L. J. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions and transpositions. *IEEE Transactions on Information Theory*, 1999.

[20] E. Shih, P. Bahl, and M. J. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *MobiCom*, 2002.

[21] A. Sikora. Compatibility of IEEE802.15.4 (Zigbee) with IEEE802.11 (WLAN), Bluetooth, and Microwave Ovens in 2.4 GHz ISM-Band. Technical report, Steibeis-Transfer Centre, Sep 2004. `http://www.ba-loerrach.de`.

[22] 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. In *VTC*, 2005.