Impulsive Interference Avoidance in Dense Wireless Sensor Networks

Nicholas M. Boers
Dept. of Computing Science
University of Alberta
2-21 Athabasca Hall
Edmonton, Alberta T6G 1K7
Canada
nboers@ualberta.ca

Ioanis Nikolaidis
Dept. of Computing Science
University of Alberta
2-21 Athabasca Hall
Edmonton, Alberta T6G 1K7
Canada
nikolaidis@ualberta.ca

Pawel Gburzynski
Olsonet Communications Corporation
51 Wycliffe Street
Ottawa, Ontario K2G 5L9
Canada
pawel@olsonet.com

ABSTRACT
As with all wireless communication devices, wireless sensor networks (WSNs) are subject to interference from other users of the radio-frequency (RF) medium. Such interference is practically never random: originating in applications generally performing some practical and sensible activities, it naturally exhibits various regularities amounting to perceptible patterns, e.g., regularly-spaced short-duration impulses that correlate among multiple WSN nodes. If those nodes can recognize the interference pattern, they can benefit from steering their transmissions around it. This possibility has stirred some interest among researchers involved in cognitive radios, where special hardware has been postulated to circumvent non-random interference. Our goal is to explore ways of enhancing medium access control (MAC) schemes operating within the framework of traditional off-the-shelf RF modules applicable in low-cost WSN motes, such that they can detect interference patterns in the neighbourhood and creatively respond to them mitigating their negative impact on the packet reception rate. In this paper, we describe (a) a method for the post-deployment dynamic characterization of a channel aimed at identifying spiky interference patterns, (b) a way to incorporate interference models into an existing WSN emulator, and (c) the subsequent evaluation of a proof-of-concept MAC technique for circumventing the interference. We found that an interference-aware MAC can improve the packet delivery rates in these environments at the cost of increased latency. Notably, the latter is quite acceptable in the vast majority of WSN applications.

Categories and Subject Descriptors
C.2.1 [**Network Architecture and Design**]: Wireless communication; I.5.5 [**Pattern Recognition**]: Implementation—*special architectures*

General Terms
Experimentation, Performance, Measurement, Algorithms

Keywords
Classification, interference, sampling, wireless sensor networks, channel modelling, medium access control

1. INTRODUCTION
Wireless sensor networks (WSNs) typically operate in the industrial, scientific, and medical (ISM) unlicensed radio bands centred at 433.92 MHz, 915 MHz, and 2.450 GHz. Their nodes must be particularly resilient to interference because the ISM bands are heavily used, particularly in dense urban environments [6]. The users of them are quite varied, too, with some examples including cordless telephones/headphones, wireless local area networks (WLANs), and microwave ovens.

Even though external interference is an unfortunate reality in many environments, few papers explicitly address it. Instead, researchers typically base their performance studies on over-simplistic environmental models assuming that the only disturbance to the "proper" signal from a transmitting node at the receiver comes from white Gaussian background noise plus possible interference from peer devices (members of the same networked wireless system). The two types of disturbance have received considerable attention in research under the umbrellas of channel modelling and MAC protocol design, respectively. The third type of disturbance, namely external interference from a different wireless system, has been much overlooked. This is unfortunate, considering that the incessantly growing number of wireless applications, combined with the limited spectrum available to them, will make the impact of external interference more and more pronounced. Based on our experience, external interference is already the predominant source of communication problems in many WSN systems, especially those deployed in densely populated urban areas.

When wirelessly receiving an RF signal, the probability of misinterpreting it depends on the signal to interference-plus-noise ratio (SINR) [20]. A higher ratio indicates greater distinction of the signal from the interfering components and increased likelihood that the receiver can decode it. Devices external to the WSN can contribute to the interference component and have a significant and detrimental effect on a receiver's performance.

The potential effect of external interference became bla-

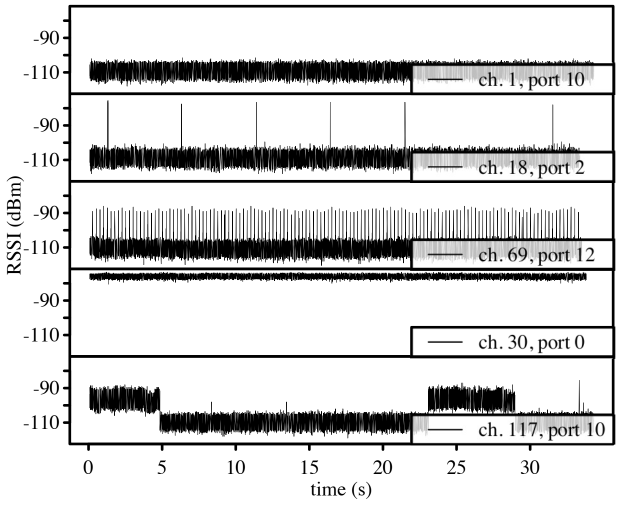

Figure 1: The different primary interference classes identified in our RSSI traces. The middle pattern, representing frequent impulses of short duration, is the focus of this work.

tantly obvious to us in our 2008 deployment of the Smart Condo – a network to passively monitor an independent living environment [2, 26]. As soon as its simple transceivers (RF Monolithics TR8100 [21]) began their operation (at 916.5 MHz), we noticed significant packet losses even over short distances and with the obvious lack of interference from peers. Those losses disappeared when the same set of motes was moved to another environment (several blocks away) for an in-lab study of their poor performance. Having thus confirmed that the environment itself was the culprit, we returned to it with another WSN comprised of 16 motes, whose intention was to assess the character of the external interference. We specifically wanted that assessment to be carried out by a WSN (as opposed to some specialized and sophisticated spectrum analyzing equipment), because one of its objectives was to make the network aware of the interference via its own means, such that it could analyze the problem and respond to it all by itself. The nodes of the new WSN were equipped with more flexible RF modules, namely Texas Instruments CC1100 [28], capable of collecting digitized samples of the received signal strength indicator (RSSI) at high rates over varying channels. Using that network we took an extensive collection of RSSI traces at 5000 Hz on 256 channels ranging from 904 to 954 MHz [4]. After plotting those traces as time series data, we immediately identified a number of recurrent interference patterns, including the one that caused our original alarming packet losses (Figure 1, middle). Reflecting back on that negative experience, although the TR8100 transmitted at reasonably powerful levels (10 dBm), it used a very simple encoding scheme (on-off keying) that is particularly susceptible to interference [29, 17].

Needless to say, it would be highly presumptuous to claim that any interference patterns that we observed in a particular environment and on a particular day should be immediately generalized into blanket rules applicable to all wireless systems. However, the very fact that we clearly saw a small number of simple and easily discernible patterns (some of which, as we argue later, can be circumvented in software run at the motes) hints at the potential benefits of playing the same set of tricks in other circumstances. Then, it is highly unlikely that what we saw was specific to the one environment. In particular, the spiky patterns, which are most interesting from the viewpoint of navigating around them with packet transmissions, are intuitively natural and expected to occur in many (otherwise unknown) wireless systems. For one thing, they may be representative of a typical (generic) WSN operating in the same area. Although unlikely to be the case in our environment, one can reasonably predict that an increasing proliferation of WSNs will bring about spiky interference patterns. Circumventing them can be viewed as solving a slightly augmented medium access control problem, whereby a certain subset of "peers" follow an unknown and presumably non-responsive (but nonetheless systematic) schedule of transmissions. Consequently, the problem appears general and interesting enough to warrant further studies.

In this paper, we explore the avoidance of impulsive (spiky) interference in dense wireless sensor networks (Figure 1, middle). We first review work related to the general characterization of channels (Section 2), and then focus on a technique capable of identifying this particular pattern (Section 3). In Section 4, we describe the extension of an existing simulator with this characterization. After modelling the interference, we incorporate the classifier and a proof-of-concept MAC into a WSN application (Section 5) and present the results from simulating it (Section 6). Finally, in Section 7, we present some concluding remarks.

2. RELATED WORK

When exploring interference, some researchers have focused on the interaction of specific protocols, e.g., IEEE 802.11b (WLAN), 802.15.1 (Bluetooth), and 802.15.4 (ZigBee) [24]. Similarly, others have concentrated their efforts on specific expected interferers, e.g., Chandra [5] used a spectrum analyzer in a 3-story building to explore the noise generated by electronic equipment in a workshop, a photocopier, elevator, and fluorescent tubes. In this section, we describe the small body of work that addresses interference more generally.

Using sensor platforms, Srinivasan, Dutta, Tavakoli, and Levis [23] studied packet delivery performance. With nodes synchronized, they encountered strong, spatially-correlated impulses (up to -35 dBm or higher) in their traces. Given the high correlation, they concluded that the spikes originated externally to the nodes.

Researchers working on closest-fit pattern matching (CPM) sampled noise in (a) WLAN-enabled buildings, (b) WLAN-enabled outdoor areas, (c) outdoor quiet areas, and (d) controlled areas [12, 22]. They sampled channels both overlapping and non-overlapping IEEE 802.11b channels and observed three characteristics: (a) spikes sometimes as strong as 40 dB above the noise floor, (b) many of the spikes were periodic, and (c) over time, the noise patterns changed. In their work, they offered little description of the patterns beyond what we summarize here. Instead of focusing on specific patterns, they developed a modelling approach that initially replays the recorded trace and then estimates future points based on computed probabilities.

More recently, Srinivasan, Dutta, Tavakoli, and Levis [24] expanded on much of their previous work. With six synchronized nodes, they sampled RSSI values at 128 Hz and explored the correlation in the noise traces. They observed 802.11b interference as high at 45 dB above the noise floor, and in their figures, this interference appears as periodic impulses at roughly 36 Hz.

In our recent work, we explored measurements from a grid of sixteen nodes in an indoor urban environment [4]. Within the 80 m^2 space, we deployed the grid with 1.83 m spacing and elevated each node 28 cm off of the floor. We connected all of the nodes to a single computer using USB and then proceeded to simultaneously measure each node's RSSI value at 5000 Hz.[1] Over a period of roughly 2.5 hours, we scanned the 256 available channels ranging from 904 to 954 MHz. Upon inspecting our high resolution traces, we identified the five recurrent patterns that we show in Figure 1.

3. CHARACTERIZATION

As shown by the representatives of the interference classes in Figure 1, each pattern class has its its own unique characteristics, ranging from fairly benign (the "quiet" pattern) to difficult to predict (random impulses). We are interested in putting such a characterization to constructive use, in particular with respect to how to ensure better packet loss performance. Some of the patterns, for example those containing relatively rare random impulses, have little effect on packet loss rates, and therefore warrant little attention. Other patterns, such as the random shifting-mean pattern (bottom of Figure 1) with long bursts of continuous interference, also provide little opportunity for exploitation. The pattern that appears to lend itself to exploitation is the rapid and periodic impulses, and coincidentally, this is the pattern that affected our original network. Towards this end, we first address the issue of how to properly characterize this particular class of interference with the limited resources of a WSN node. Note that a full-blown "machine learning" approach to classification is prohibitive for the limited processing and storage available at the nodes, hence we narrow our search to techniques amenable to small footprint implementations.

3.1 Simplified Periodic Impulsive Interference Characterization

To characterize channels with frequent impulsive interference, we start with the classic Lomb periodogram method of least-squares spectral analysis [14, 19] and simplify it to fit our resource constraints. In the original method, the Lomb periodogram for N data points is defined by

$$P_N(\omega) \equiv \frac{1}{2\sigma^2}\left\{ \frac{\left[\sum_j (h_j - \bar{h})\cos\omega(t_j - \tau)\right]^2}{\sum_j \cos^2\omega(t_j-\tau)} + \frac{\left[\sum_j (h_j - \bar{h})\sin\omega(t_j - \tau)\right]^2}{\sum_j \sin^2\omega(t_j-\tau)} \right\} \quad (1)$$

[1]To the best of our knowledge, this sampling rate has been unmatched so far in a WSN framework.

where $\omega = 2\pi f$, f is the frequency, h_j and t_j represent the magnitude and time of sample j (respectively), $\bar{h}$ and σ^2 are the mean and variance of all the samples (respectively), and τ is defined by the relation

$$\tan(2\omega\tau) = \frac{\sum_j \sin 2\omega t_j}{\sum_j \cos 2\omega t_j} \quad (2)$$

Unfortunately, this exact method is inappropriate for directly implementing on simple microcontrollers because it

1. requires the storage of all the samples to compute the mean and variance used in 1 and τ used in 2,
2. makes extensive use of the sine and cosine trigonometric functions,
3. uses floating-point values throughout, and
4. includes many floating-point multiplications and divisions.

Note that in the last case, one integer-float multiplication or division on the TI MSP430 could use over 400 instruction cycles, while a register add or shift operation requires only a single cycle [27]. Given that many microcontrollers have limited memory and lack of hardware support for floating-point arithmetic and trigonometric functions, we consider a suitable approximation.

We aim for constant memory usage regardless of the sample size, and instead, allow memory use to scale linearly with the number of analyzed frequencies. This setup allows us to collect the potentially large number of samples that may be required, and at the same time, it encourages us to reduce the memory footprint by searching only frequencies of interest. To make the construction of the algorithm feasible within mote-class devices, we make a number of compromises/approximations:.

- **Mean estimation:**
  The exact case's use of the mean $\bar{h}$ and variance σ^2 in 1 would require the logging of all samples. Instead of taking this approach, we (a) precompute an integer estimate of the mean and (b) calculate the variance while collecting samples. We evaluated different sample sizes for the mean calculation and 200 samples often proved sufficient. To calculate the variance, we build an integer sum while obtaining samples and avoid the multiplication by using a look-up table.

  The mean plays a very important role in the algorithm, which assumes a random distribution of non-periodic samples around the mean. To reduce the risks associated with deriving the mean from a non-representative sample, we constantly adjust it: a sample above it increases it by 1 (and vice versa for samples below it).

- **τ elimination:**
  The calculation of τ is another prerequisite for the exact periodogram. Lomb introduced this parameter to facilitate the statistical description of the least-squares spectrum [14]. For our approximation, however, we do

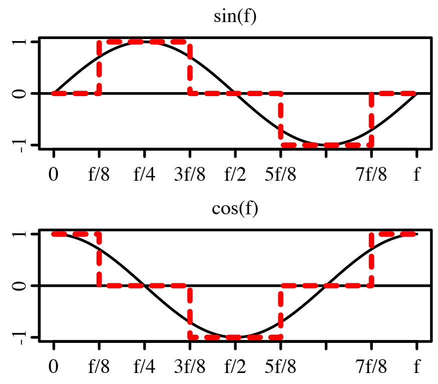

Figure 2: A three-level approximation of the sine and cosine functions used when calculating the Lomb periodogram.

not need such rigour, and intuitively, the periodogram changes little with shifts in time. We confirmed our suspicion by comparing several periodograms both before and after the removal of τ: we observed very little difference between the two cases.

- **Sine and cosine quantization:**
  The sines and cosines are the final problematic component of the exact formulation. Given samples normally distributed about the mean, i.e., samples lacking periodicity, the sums of the trigonometric products tend to zero. On the other hand, a pattern with a period matching either the wave frequency or a harmonic [16] will tend to a non-zero sum. We make a number of approximations here that maintain this fundamental property.

  To calculate the contribution of a new sample, we need to know the value of the sine and cosine waves corresponding to its time-stamp. A naïve approach might use the modulo operation to determine the offset; instead, we store an array of time-stamps that mark the start of each wave's next cycle. By knowing when the next wave begins for a given frequency, we can perform subtractions and additions to identify the offset into the current wave. The computational overhead of maintaining these time-stamps amounts to an integer comparison and typically zero or one addition for each wave, for each new sample. By quantizing the wave and making the discontinuities occur at opportune locations (bit-shifts of the frequency), we can easily compute the offset into a wave and avoid divisions.

  We have adopted a rather crude (but effective) approximations of the sine and cosine functions (Fig. 2) that allows us to use four comparisons to determine the correct offset. Given an offset, we need only add/subtract whole mean-adjusted samples, and we now only encounter zero and one in the denominator. In this case, we only require 4 short integer sums per frequency.

3.2 Simplification Results

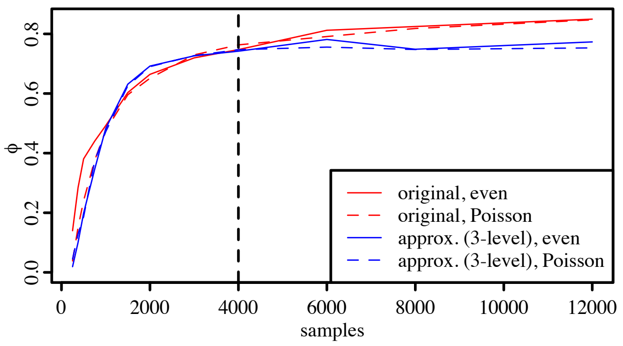

Figure 3: The performance of identifying frequent periodic impulses using only the Lomb periodogram. This graph compares using the original Lomb periodogram against the approximation based on our 3-level trigonometric approximation.

From the original 4096 traces (16 nodes $\times$ 256 channels), we randomly sampled 1024 traces and carefully hand-classified them for the presence of frequent periodic impulses. We encountered the pattern in 154 of the traces, and some of these traces contained other patterns as well.

Since each full trace consists of 175 000 points, we evaluate the two techniques on *subsamples* of the traces and compare both even and Poisson subsampling techniques. For each trace of subsamples, we record whether the periodogram indicates the presence of frequent periodic impulses. To compare the two techniques, we use the ϕ coefficient, which is actually a product-moment coefficient of correlation [11] and is also called the Matthews correlation coefficient [15]. ϕ, which ranges from -1 to 1, indicates the association between two variables. In this case, one variable is the hand classification and the other is the automated classification.

Fig. 3 compares the classification accuracy of the original versus 3-level trigonometric approximation. After four thousand samples, the performance of the original periodogram continues to slightly increase while the approximation remains constant. Overall, the approximations perform very well when compared with the exact periodogram. In the next step we introduce the particular interference behaviour as detected and classified here in a simulator.

4. SIMULATION

Our choice of network simulator follows from our choice of WSN operating system (OS). Although the most prominent OS described in the literature is quite possibly TinyOS [13], we use a mature and actively developed alternative named PicOS [1]. The latter has a number of advantages over the former, most notably:

1. All of the program dynamics available to the programmer are captured by PicOS's threads (finite state machines) rather than interrupt service routines (or callbacks). As all threads share the same (global) stack, PicOS's stack has absolutely no tendency to grow uncontrollably.

2. PicOS is incomparably more flexible with respect to memory. Dynamic memory allocation (even within devices with less than 1 KB of RAM) is its essential feature.

After the first paper introduced PicOS in 2003, an existing simulator gained support for simulating PicOS applications.

A locally-developed event-driven discrete-time network simulator named SIDE [8, 7] predates PicOS. This program evolved out of work done in the eighties, and over the years, it has been actively developed. Rather recently, it gained wireless channel support [9] and the ability to emulate PicOS applications at the level of their API (application programming interface) using a component named VUE^2 [3]. Given this close relationship between our chosen OS and a mature simulator, our decision to use SIDE was quite natural.

4.1 Interference Modelling

In this section, we describe the new extension to SIDE that adds the ability to define external impulsive interference. The extension consists of (a) a user-specified configuration, (b) a new "node" type within the simulator, (c) and threads running on those nodes to produce the specified interference.

Additional tags and attributes added to an existing XML (extensible markup language) configuration file provide the user-specified interference configuration. A new `interferers` attribute to the network tag indicates the number of interferers in the environment, e.g.,

```
<network nodes="40" interferers="3">
```

The user can use the new `<interferers>` tag to identify an interferer-specific block within the configuration akin to the existing `<nodes>` tag. Within this new section, the user can define the parameters for each interferer, e.g.,

```
<interferer number="0" type="impulsive">
  <location type="random">170.0 170.0</location>
  <pattern>
    R 0.245 s         ; random delay
    P                 ; start periodic portion
    O 0.0 dBm 3 dB    ; on at 0.0 dBm with 3 dB sd
    T 0.005 s         ; delay
    F                 ; off
    T 0.245 s         ; delay then implicit jump to P
  </pattern>
</interferer>
```

The attribute and value `type="random"` for the location causes SIDE to generate a new location every time the simulator starts (assuming a new random number generator seed). It uses the specified coordinates to bound the random values.

Internally, each interferer becomes an object within the simulation, not unlike what already occurs for nodes. For these new objects, the user can create a library of processes, each capable of producing a certain class of interference. For this work, we implemented an `Impulsive` process to simulate user-specified impulsive interference.

The body of the `<pattern>` tag essentially provides the script for the process `Impulsive` to follow. When SIDE processes the XML file, it extracts commands and arguments from the possibly verbose description of the pattern; it creates two arrays: one for the single-character command and one for the double-valued argument(s). For an impulsive interferer, SIDE supports following commands:

`R` delay for a random duration between 0 and the double argument (in seconds),

`T` delay for the specified duration (in seconds),

`O` generate interference at the specified power level (in dBm) with the specified standard deviation (in dB),

`F` stop the generation of interference, and

`P` mark the start of the periodic portion of the pattern.

Essentially, the `Interferer` process interprets (in a fetch-decode-execute style) the command sequence provided in the specification block. Once the end of the list of commands is reached, an implicit jump occurs to the command immediately following the `P` command.

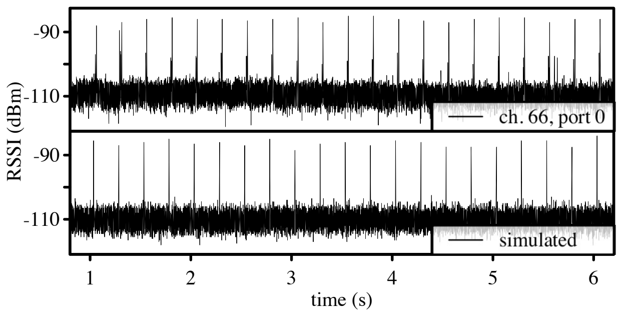

Figure 4: An actual trace (top) plotted with a simulated trace (bottom). We used the same application to collected both traces.

We placed a number of synchronized impulsive interferers in a virtual environment, and using our earlier sampling application [4], collected a number of virtual traces (e.g., Figure 4). With very little tweaking, we were able to make the simulated traces match the essence of the real traces. Upon close inspection, there are slight differences, e.g., the simulated traces lack some random non-periodic components, and with a little more work, we could include these in our model as well. That said, the existing detail suffices for the classification and medium access control techniques that we next implement.

5. EXPLOITING INTERFERENCE CLASSIFICATION IN A MAC PROTOCOL

PicOS and its accompanying VUE^2-extended SIDE simulator subscribe to a layer-less approach of networking. A central driver, named the Virtual NETwork Interface (VNETI), acts as the mediator between three primary components (Figure 5):

1. the application programming interface (API),

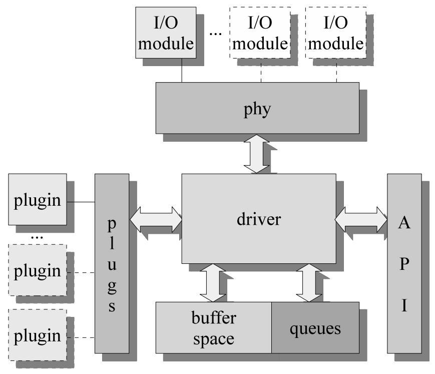

Figure 5: The Virtual NETwork Interface component in PicOS. This component serves as a mediator between the application programming interface (API), protocol plug-ins, and physical device drivers.

2. network protocol plug-ins, and
3. hardware drivers.

VNETI's primary responsibility in this framework is buffer and queue management.

Since what we seek is a cross-layer interaction that involves classification algorithms at the top, packet queueing, as well as a particular form of opportunistic medium access control behaviour, the most appropriate location for implementing a scheme that combines all those facets within this layer-less architecture is at the location of an "I/O module" as shown in Figure 5. This particular location allows access to precise timing constraints and sampling dependencies, accessing in a direct non-abstracted fashion the underlying hardware. For the results presented in this work, we implemented both components in VUE^2's emulated radio driver, and they both make full use the PicOS finite-state machine paradigm.

5.1 On-line Classifier

To classify channels with regularly-spaced short-duration impulsive interference, we implement the approximated least-squares spectral analysis (LSSA) technique described in Section 3. In the transceiver's transmit FSM, we introduce three new states to accommodate the classifier:

`CLS_INIT` Initializes the variables required for classification and immediately advances to `CLS_MEANEST`.

`CLS_MEANEST` A visit to this state represents the measurement of a single RSSI sample to compute the mean RSSI estimate. It remains in this state for 200 iterations prior to transitioning to `CLS_SAMPLE` – a number of iterations that proved reasonable in our early tests. The delay between each iteration is uniformly randomly distributed between 0 and 7 ms.

`CLS_SAMPLE` A visit to this state represents obtaining a single RSSI sample for calculating the LSSA. It remains in this state for 5000 iterations which span approximately 17.5 s and then transitions to the (regular) MAC state. The delay between each iteration is uniformly randomly distributed between 0 and 7 ms. We used more iterations than the minimal 4000 identified earlier simply as a precaution.

The complete classification process lasts just over 18 s during which we prevent nodes from communicating. If the application wishes to transmit packets during the process, they are simply queued within VNETI until the classifier completes. It is implied that in a real deployment, the classification task is to be executed occasionally to assess the new levels and periods of any periodic impulse interference.

5.2 Pattern-aware Medium Access Control (PA-MAC)

The output from the classifier indicates the presence of periodic short-duration impulsive interference at any of the tested frequencies. Our proof-of-concept pattern-aware MAC (PA-MAC) then uses this output in its attempt to steer transmissions around the impulses. In fact, the protocol makes a virtue out of interference, because the periodic interference becomes well-defined time points around which to anchor transmissions (with some back-off of course as we will later see). Stretching definitions a bit, the interference becomes a means for implicit synchronization of the MAC transmissions across nodes.

In our approach, we make observations about the interference at a transmitting node and assume that they also hold for the receiving node, i.e., we assume a significant amount of correlation in the interference between nodes. Particularly in small dense deployments, we have found this assumption to hold, e.g., we observed significant correlation in the traces collected in the Smart Condo (Figure 6). Moreover, other researchers have observed significant correlations in packet losses [25]. Even in larger environments, this assumption may hold given either a particularly strong interferer or a collection of correlated interferers.

To implement PA-MAC, we introduce one further state to the transceiver's transmit FSM: `MAC_SEARCH` with the intention to use it as a way to track the impulse instants. Initially after the classification, and then again regularly after each transmission window, the process will enter this state to sample the channel to track the next impulse. Successfully finding an impulse causes the thread to (a) set a timer to mark the end of the next transmission window, i.e., the expected arrival of the next impulse and (b) delay for the expected duration of the currently identified impulse and then transition to the thread's preexisting primary state (`XM_LOOP`). Once in the main loop, the driver will retrieve outgoing packets from VNETI as they become available and transmit them until the expiration of the first timer. At that point, the process reenters `MAC_SEARCH` where it attempts to track the next impulse.

The existing PicOS MAC listens before transmitting (LBT) and resolves contention by using a random back-off. Given multiple transmitting nodes, this approach leaves little opportunity for them to synchronize – the random back-offs

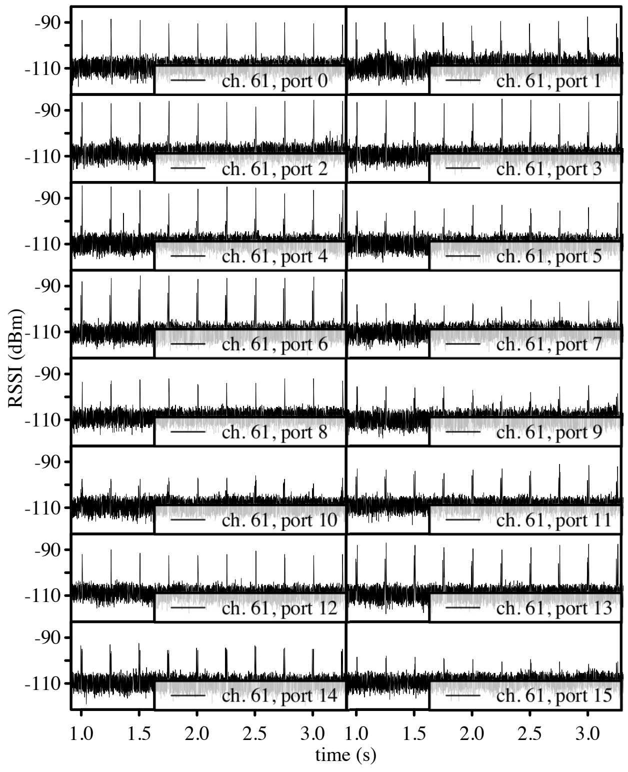

Figure 6: Traces collected on channel 61 (916.197 MHz) from 16 different locations simultaneously. Note that the impulses occur at all of the receivers.

effectively resolve contention. With PA-MAC, however, our regular tracking of the interference introduces a new opportunity for nodes to synchronize, which could ultimately cause a number of nodes to transmit at the same time.

In our initial experiments with PA-MAC, we overlooked this possibility and experienced high packet loss rates. Upon investigating those losses, the reason became blatantly obvious: a number of nodes would transmit immediately after an impulse, all at the same time. We eliminated this point of contention by introducing an additional random back-off, and we immediately saw the benefits in our results.

6. RESULTS

We evaluated the pattern-aware MAC within the interference-generating SIDE simulator. We used its built-in shadowing channel model, and we tweaked the simulator's parameters to represent our physical hardware.

We include results for both single- and multi-hop random topologies. For a given configuration, we average the measurements from 100 different topologies, each with its own traffic pattern, and plot the results with 95% confidence intervals.

6.1 Single-hop

The single-hop configurations consist of 19 source nodes, one destination node, and two interferers, and the simulator places them all randomly within an 18 m $\times$ 18 m field. Since the model neither includes obstructions nor considers radio irregularly [30], these dimensions guarantee that the destination is within the transmission range of every source node. Each node transmits at a rate of 10 kbps and a transmission power of -20 dBm. The interferers introduce 5 ms pulses of impulsive interference at 4 Hz and -30 dBm.

We first evaluated the effect of varying the packet length (Figure 7) on the packet reception rates (PRRs) and latency. Note that the destination node does not acknowledge received packets and nodes make no attempt to retransmit lost packets. We measure latency from the application perspective: the time that elapses between VNETI receiving the packet from the application (at the transmitter) and the application receiving the packet from VNETI (at the receiver). Note that the effect of varying the packet length should be seen relative to the frequency of impulsive interference. Alternatively, we could have kept the packet length the same and change the interference's period. We chose the former approach. In these tests, nodes generated new packets according to an exponential distribution with mean 50 s to reduce (if not practically eliminate) the effect of congestion. For each run of the simulation, we generate 500 s of input and allow the simulator to run for 600 s (in case of delayed packets).

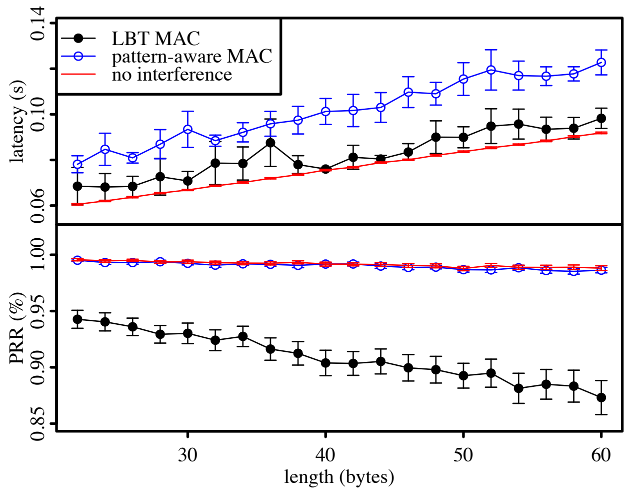

Figure 7: In a dense single-hop network, the effect of the packet length on the packet reception ratio and the latency.

When increasing the length, the PRR decreases for all configurations and the latency increases (as expected). In terms of PRRs, PA-MAC performs similarly to the quiet configuration because it successfully steers the transmissions around the interference. To obtain these PRRs, it ends up delaying transmissions that may collide with the interference, and the latency graph reflects this behaviour. The traditional LBT MAC's PRRs suffer at a greater rate than the other two configurations as more packets are lost to collisions than simply the non-zero bit error rate. The traditional LBT MAC shows higher latency than what is achieved by the quiet channel, demonstrating that the LBT MAC yields also occasionally to interference because it senses the medium as being busy.

We also evaluated the effect of varying the packet generation rate (Figure 8) on the PRRs and latency. In these experiments, we set the packet length to its maximum (60 bytes) in order to accentuate the variable's effect.

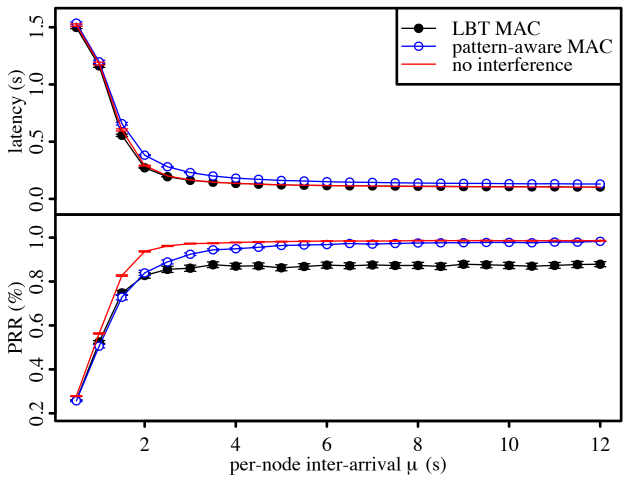

Figure 8: In a dense single-hop network, the effect of the mean latency between per-transmitter packet introductions on the packet reception ratio and the latency.

Under high congestion (mean packet inter-arrival time at each node $\mu < 2$ s), the packet reception rates drop significantly for all methods in this dense network, and the LBT MAC and PA-MAC perform very similarly. Since all nodes are within range of each other, all transmissions will generate interference, but that interference may not be sufficiently strong for a node to recognize the medium as busy. The PRRs are lower for both of the interference configurations because the MACs will sometimes yield to the interference, leaving less of a window for data transmission. At lower levels of congestion, the PA-MAC tends towards the performance of the quiet configuration.

6.2 Multi-hop

The multi-hop configurations consist of 39 source nodes, one destination node, and three interferers, and the simulator places them all randomly within a 170 m × 170 m field. As with the single-hop scenario, nodes transmit at a rate of 10 kbps and with transmission power -20 dBm. In this case, the three interferers produce a similar interference pattern to the single-hop case, but transmit at 0 dBm rather than -30 dBm. Given the larger field, we made this change to ensure the visibility of interferers across the network.

The transmitting nodes use the tiny ad hoc routing protocol, TARP [18], to deliver packets to the destination. TARP is a light-weight on-demand routing protocol that quickly converges to the shortest path in static networks. Because it lacks explicit control packets (minimal control information is present in the packet header) it does not inflate the overall traffic needed to support it. Although the application only demands one-way communication, the destination sends short 14-byte replies to each source node for the benefit of the routing protocol. Note that communication continues to be unacknowledged, and nodes make no attempt to retransmit lost packets.

Given random node locations, we need to take precautions to ensure that each source node has a path to the destination node. Immediately after generating a random layout, the simulator will search for a path from every source to the single destination while ensuring that each hop is less than the maximum transmission range. If the procedure finds a disconnected node, the simulator will generates a completely new node placement until such a path exists.

Like in the single-hop case, we first evaluate the effect on varying the packet length on the PRRs and latency (Figure 9). To reduce congestion in the multi-hop environment given the high initial number of retransmissions, we lower the packet generation rate to follow an exponential distribution with mean of 200 s. Given the lower packet generation rate, we generate 2000 s of input and allow the simulator to run for 2100 s.

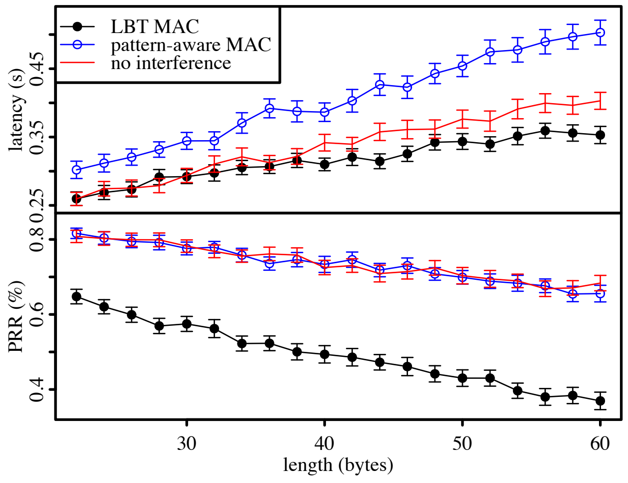

Figure 9: In a connected multi-hop network, the effect of the packet length on the packet reception ratio and the latency.

As with the single-hop case, we notice decreasing PRRs and increasing latencies as the length increases, and PA-MAC again follows the PRR of the quiet configuration. However, unlike in the single-hop case, we notice that the quiet configuration no longer provides the baseline for delay. To explore this phenomenon, we investigate the hop lengths compared to packet lengths (Figure 10).

Since the network is static, we would expect little change in the the expected number of hops as the packet length increases. However, we notice that the expected number of hops decreases for the LBT MAC as the packet length increases. The significant number of packet losses cause this behaviour: packets are more likely to be lost on the long paths, and these lost packets will not factor into the latency calculations.

Our final graph shows the effect of varying the packet generation rate on the PRRs and latency (Figure 11). In these

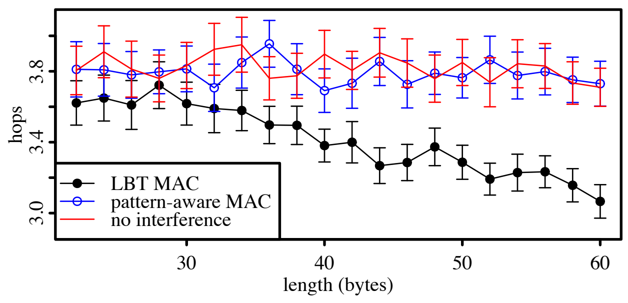

Figure 10: In a connected multi-hop network, the effect of the packet length on the mean number of hops.

experiments, we set the packet length to its maximum (60 bytes) in order to accentuate the variable's effect.

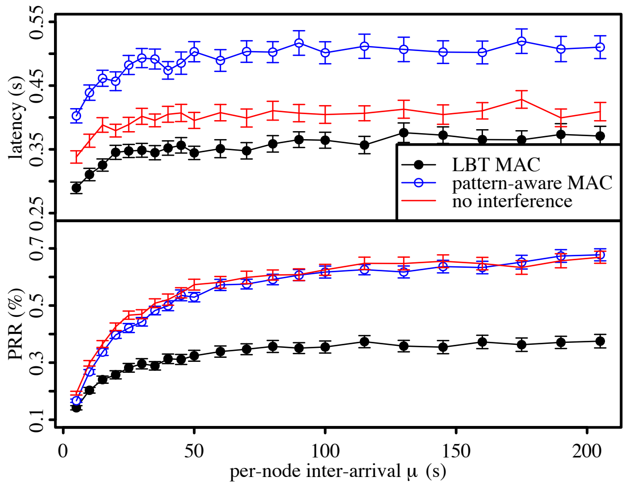

Figure 11: In a connected multi-hop network, the effect of the mean latency between per-transmitter packet introductions on the packet reception ratio and the latency.

Here, the PRR rate follows a similar trend to the single-hop case just at significantly lower levels. Unlike with the single-hop case, the latency curve again increases as we slow the rate of packet generation. As with the packet lengths, less congestion results in an increased number of the long paths succeeding which subsequently increase the latency.

In summary, the results demonstrate the benefits of using interference in a constructive manner. The benefits are evident even if used to augment a trivial MAC protocol, such as a rudimentary LBT. Naturally, more elaborate schemes can be devised. Suffice is to say that the impulse interference is the basis of synchronization around which a self-organizing TDMA-like MAC protocol could eventually be constructed.

7. CONCLUSION

In this paper, we first described a simplification of the Lomb periodogram for the post-deployment identification of frequent impulsive interference. Estimating the mean, calculating the variance at run-time, and eliminating τ reduced its memory requirements enough to fit in a typical WSN-node. Quantizing its trigonometric functions then reduced its computational complexity to a suitable level. Finally, we compared the exact periodogram with the simplification, and given enough samples, the latter performed well.

We then extended the SIDE simulator with a flexible interface for the production of impulsive interference. By using its existing configuration files, we outlined new syntax that will allow users to describe their desired patterns.

Finally, we incorporated the classifier and a proof-of-concept pattern-aware MAC (PA-MAC) into SIDE's emulated radio driver. After simulating a variety of different configurations, we found that PA-MAC could improve the packet reception rates in both single- and multi-hop environments at the cost of increased latency.

In terms of future work, we plan to explore protocols that would allow nodes to come to a consensus about the channel classification. An immediate result from this would be the weakening of our correlation assumption. Moreover, such a protocol would allow nodes to join the network without pausing communication while the evaluation occurs.

Many practical WSNs consist of nodes of different types with different power budgets. The simplest generic representative of such a system is the so-called *Tags and Pegs* involving a semi-infrastructure of immobile Pegs (possibly connected to power outlets) communicating with mobile Tags (powered from batteries) [10]. In such a network it may make sense to delegate the task of channel sampling solely to the Pegs (which can afford more wasteful duty cycles) providing for some way of disseminating the information about interference patterns to the Tags. Generally, the problem of optimal collaborative identification of interference patterns and selective dissemination of knowledge (not all nodes need to receive the same information) appears as an interesting topic for a further study.

8. ACKNOWLEDGMENTS

The authors thank the Natural Sciences and Engineering Research Council (NSERC), Alberta Advanced Education and Technology, and the Killam Trusts for financial support.

9. REFERENCES

[1] E. Akhmetshina, P. Gburzynski, and F. Vizeacoumar. PicOS: A tiny operating system for extremely small embedded platforms. In H. R. Arabnia and L. T. Yang, editors, *Embedded Systems and Applications*, pages 116–122. CSREA Press, 2003.

[2] N. M. Boers, D. Chodos, J. Huang, E. Stroulia, P. Gburzynski, and I. Nikolaidis. The Smart Condo: Visualizing independent living environments in a virtual world. In *PervasiveHealth '09: Proceedings from the 3rd International Conference on Pervasive Computing Technologies for Healthcare*, London, UK, Apr. 2009.

[3] N. M. Boers, P. Gburzynski, I. Nikolaidis, and W. Olesinski. Developing wireless sensor network applications in a virtual environment. *Telecommunication Systems*, 2010.

[4] N. M. Boers, I. Nikolaidis, and P. Gburzynski. Patterns in the RSSI traces from an indoor urban environment. In *CAMAD '10: IEEE 14th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*, Coconut Creek, FL, Dec. 3-4, 2010.

[5] A. Chandra. Measurements of radio impulsive noise from various sources in an indoor environment at 900 MHz and 1800 MHz. In *13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 2, pages 639–643, Sept. 2002.

[6] J. Do, D. Akos, and P. Enge. L and S bands spectrum survey in the San Francisco Bay area. In *PLANS 2004: Position Location and Navigation Symposium*, pages 566–572, 2004.

[7] W. Dobosiewicz and P. Gburzynski. *State-of-the-art in Performance Modeling and Simulation*, chapter Protocol design in SMURPH, pages 255–274. Gordon and Breach, 1997.

[8] P. Gburzynski. *Protocol Design for Local and Metropolitan Area Networks*. Prentice Hall PTR, Upper Saddle River, NJ, 1995.

[9] P. Gburzynski and I. Nikolaidis. Wireless network simulation extensions in SMURPH/SIDE. In *WSC'06: Proceedings of the 2006 Winter Simulation Conference*, Monterey, California, Dec. 2006.

[10] P. Gburzynski and W. Olesinski. On a practical approach to low-cost ad hoc wireless networking. *Journal of Telecommunications and Information Technology*, 2008(1):29–42, Jan. 2008.

[11] S. Kotz, N. L. Johnson, and C. B. Read, editors. *Encyclopedia of Statistical Sciences*. Wiley-Interscience, 2 edition, 2006.

[12] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 21–30, New York, NY, USA, 2007. ACM.

[13] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. *Ambient Intelligence*, pages 115–148, 2005.

[14] N. R. Lomb. Least-squares frequency analysis of unequally spaced data. *Astrophysics and Space Science*, 39:447–462, Feb. 1976.

[15] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451, 1975.

[16] H. J. Newton. Lecture 2: The periodogram. Lecture notes, STAT 685, Texas A&M University, 2007.

[17] J. Oetting. A comparison of modulation techniques for digital radio. *IEEE Transactions on Communications*, 27(12):1752 – 1762, Dec. 1979.

[18] W. Olesinski, A. Rahman, and P. Gburzynski. TARP: A tiny ad-hoc routing protocol for wireless networks. In *ATNAC '03: Proceedings of Australian Telecommunications Networks and Applications Conference*, Melbourne, Australia, Dec. 8–10, 2003.

[19] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, MA, 2 edition, 1992.

[20] A. Räisänen and A. Lehto. *Radio Engineering for Wireless Communication and Sensor Applications*. Artech House Publishers, 2003.

[21] RF Monolithics, Inc. TR8100: 916.50 MHz hybrid transceiver. Data sheet, 2006.

[22] T. Rusak and P. Levis. Physically-based models of low-power wireless links using signal power simulation. *Computer Networks*, 54(4):658 – 673, 2010.

[23] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. In *SenSys '06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pages 419–420, New York, NY, 2006. ACM.

[24] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *ACM Transactions on Sensor Networks*, 6(2):1–49, 2010.

[25] K. Srinivasan, M. Jain, J. I. Choi, T. Azim, E. S. Kim, P. Levis, and B. Krishnamachari. The κ-factor: Inferring protocol performance using inter-link reception correlation. In submission, 2010.

[26] E. Stroulia, D. Chodos, N. M. Boers, J. Huang, P. Gburzynski, and I. Nikolaidis. Software engineering for health education and care delivery systems: The Smart Condo project. In *SEHC '09: Proceedings from the 31st International Conference on Software Engineering*, Vancouver, Canada, 2009.

[27] Texas Instruments. Efficient multiplication and division using MSP430. Application Report SLAA329, Sept. 2006.

[28] Texas Instruments. Data sheet for CC1100: Low-power sub-1 GHz RF transceiver, Oct. 2009.

[29] M. Vieira, J. Coelho, C.N., J. da Silva, D.C., and J. da Mata. Survey on wireless sensor network devices. In *ETFA '03: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*, volume 1, pages 537 – 544, Sept. 2003.

[30] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, pages 125–138, New York, NY, USA, 2004. ACM.