Impulsive Interference Avoidance in Dense Wireless Sensor Networks

Nicholas M. Boers[1], Ioanis Nikolaidis[2], and Pawel Gburzynski[3]

[1] Department of Computer Science, Grant MacEwan University, 10700 104 Ave. NW, Edmonton, Alberta T5J 4S2, Canada, boersn@macewan.ca
[2] Department of Computing Science, University of Alberta, 2-21 Athabasca Hall, Edmonton, Alberta T6G 2E8, Canada, nikolaidis@ualberta.ca
[3] Olsonet Communications Corporation, 51 Wycliffe Street, Ottawa, Ontario K2G 5L9, Canada, pawel@olsonet.com

Abstract. Wireless sensor networks (WSNs) are subject to interference from other users of the radio-frequency (RF) medium. If the WSN 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, and based on previous work on the post-deployment characterization of a channel aimed at identifying "spiky" interference patterns, we describe (a) a way to incorporate interference models into an existing WSN emulator and (b) 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, but acceptable, latency.

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

1 Introduction

WSN nodes must be particularly resilient to interference because the ISM bands are heavily used, particularly in dense urban environments [1]. ISM sources are quite varied, including cordless telephones/headphones, wireless local area networks (WLANs), and microwave ovens. Most existing studies are based either on over-simplistic environmental models assuming Gaussian background noise, or on the assumption that interference arises 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 (possibly even from a system whose purpose is not data communication per se) has been much overlooked. Based on our empirical findings, external interference is already a major source of communication problems in WSN systems, especially those deployed in densely populated urban areas.

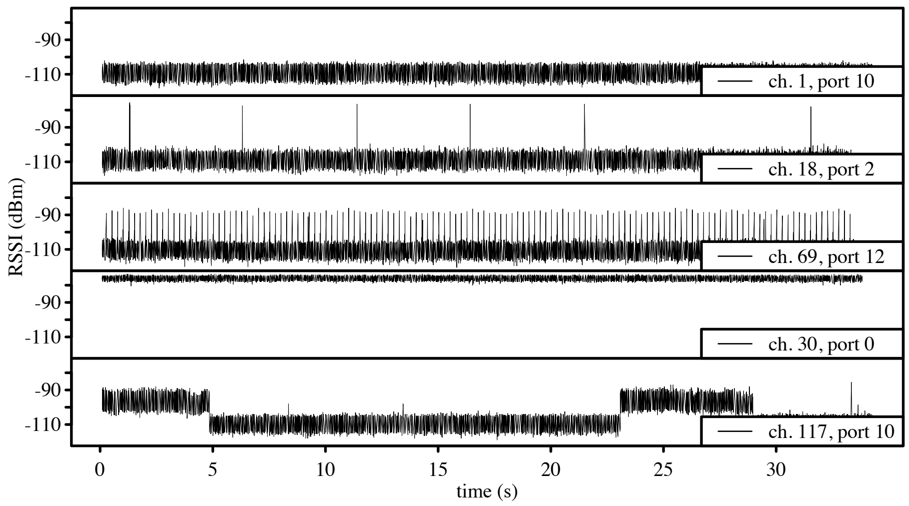

Fig. 1. The different primary interference classes identified in our RSSI traces. The middle pattern, representing frequent impulses of short duration, is the focus of this work.

Specifically, external interference became blatantly obvious to us in our 2008 deployment of the Smart Condo – a network to passively monitor an independent living environment [2, 3]. As soon as its simple transceivers (RF Monolithics TR8100) 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, more flexible, motes 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 we wanted to determine how WSNs could analyze and respond to interference problems on their own.

The nodes of the new WSN were equipped with the Texas Instruments CC1100, 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 sampled 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 [5, 6]. The analysis and the characterization and classification of interference patterns using WSN–suitable low-complexity techniques is described in two earlier publications [4, 7]. Here, we present just a summary of the main results and describe how we were able to (a) integrate interference sources in a high–quality simulation testbed and (b) evaluate a simple MAC protocol that takes advantage of particular interference patterns.

Specifically, 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 we then summarize our previously developed techniques capable of identifying this particular pattern. In Section 3, 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 4) and present the results from simulating it (Section 5). Finally, in Section 6, 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) [8]. Similarly, others have concentrated their efforts on specific expected interferers, e.g., Chandra [9] 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 [10] 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 [11, 12]. They sampled channels both overlapping and non-overlapping an IEEE 802.11b network 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 [8] 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 sampled at 5000 Hz. To the best of our knowledge, this sampling rate has been unmatched so far in a WSN framework. 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. Specifically, the patterns are: (1) quiet, (2) sparse (random) impulses, (3) frequent (strongly periodic) impulses, (4) high level interference, and (5) shifting-mean interference. 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 and that some of those patterns have been uncovered before strengthens our confidence that the set of patterns we observed can be considered representative.

In this paper, we are interested in exploiting the pattern of periodic impulse "spikes." 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, and because we are interested in a small set of samples (to conserve the amount of energy spent to sampling the medium), we *subsampled* the traces using even and Poisson subsampling techniques. For each trace of subsamples, we record whether the periodogram indicates the presence of frequent periodic impulses. Furthermore, in the periodogram calculation, we simplified the computation of the (co)sine by approximating it with values from a lookup table which contained quantized approximations of the (co)sine function. As reported in [7], we found that the automated classification yielded the same results as the hand classification (i.e., our ground truth) in the vast majority of cases, and hence the classifier was deemed adequate for our purposes. We also found that the performance of the classifier did not improve significantly past the point where 4000 samples were used.

3 Simulation

One need arising in the study of networks under interference patterns observed in traces collected from real networks is that, in order to produce repeatable simulation experiments, it is essential to model the interference sources (diverse as they might be) in a manner that is both general and easily implementable on a simulator. An ideal approach, advocated in this paper, would be to support a form of "scripted" interference source behaviours. Since our platform of choice for application development and for emulation and simulation is PicOS [13], we crafted simulated interference patterns in the idiom of PicOS threads. PicOS is conceptually derived from the SMURPH/SIDE simulator. Rather recently, PicOS gained wireless channel support [14] and the ability to emulate PicOS applications at the level of their API (application programming interface) using a component named VUE^2 [15] which leverages SIDE to provide an accurate simulation environment on which pre-deployment evaluation of protocols and systems can be conducted. Given this close relationship between our chosen OS and a mature simulator, our decision to use SIDE was quite natural.

We extended SIDE by adding the ability to define scripted external impulsive interference. The extension consists of (a) a user-specified configuration, (b) a new "node" type within the simulator, and (c) 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 *interference behaviour script* for the process `Impulsive` to follow. 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.

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 2). With very little tweaking, we were able to make the simulated traces match the substance 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 implement next.

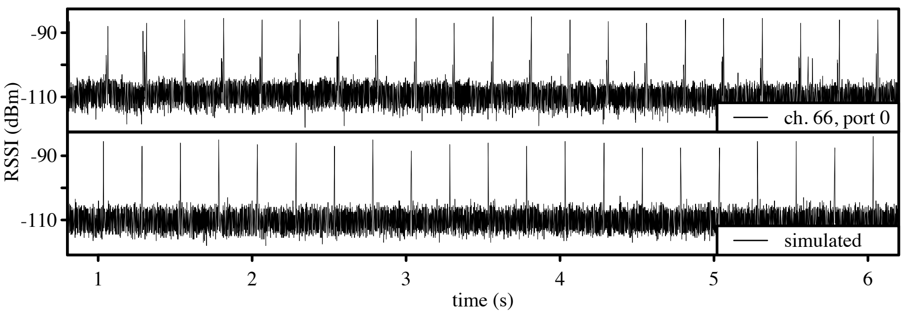

Fig. 2. An actual trace (top) plotted with a simulated trace (bottom). We used the same application to collect both traces.

4 Exploiting Interference Classification in a MAC Protocol

4.1 On-line Classifier

To classify channels with regularly-spaced short-duration impulsive interference, we implement the approximate least-squares spectral analysis (LSSA) technique described in [7]. In the transceiver's transmit state machine, 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 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.

4.2 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. Moreover, other researchers have observed significant correlations in packet losses [16].

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 state machine: `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 transceiver 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 primary state (`XM_LOOP`). Once in the main loop, the driver will retrieve outgoing packets 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.

For the sake of comparison, a simple baseline MAC protocol is *listen before transmitting* (LBT), which, other than sensing prior to attempting transmission, resolves contention using a random back-off. Given multiple transmitting nodes, the random back-off leaves little opportunity for nodes to synchronize. 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. We eliminated this point of contention by introducing an additional random back-off.

5 Results

We evaluated the pattern-aware MAC within the interference-generating simulator, using the 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.

5.1 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 irregularity [17], 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 a period of 4 Hz and -30 dBm.

We first evaluated the effect of varying the packet length (Figure 3) 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 receiving the packet from the application (at the transmitter) and the application receiving the packet (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 changed 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).

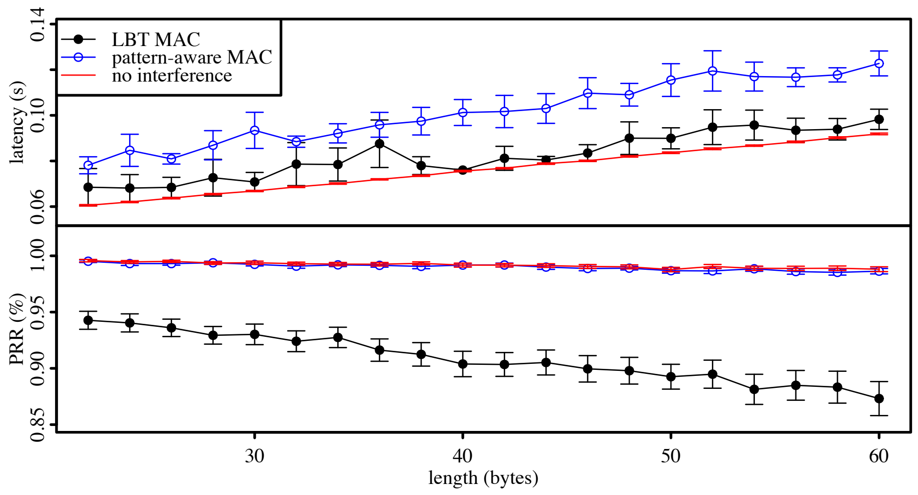

Fig. 3. In a dense single-hop network, the effect of the packet length on the packet reception ratio and the latency.

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 4) 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.

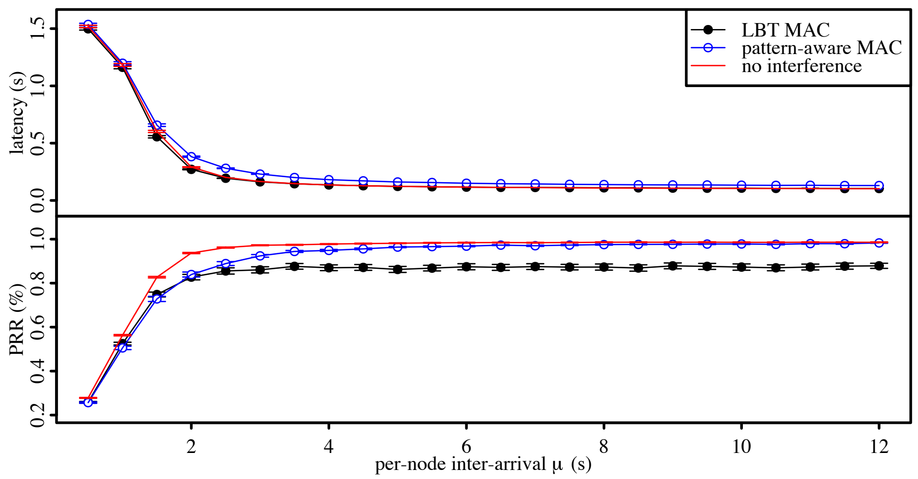

Fig. 4. In a dense single-hop network, the effect of the mean latency between per-transmitter packet introductions on the packet reception ratio and the latency.

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.

5.2 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 $\times$ 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 generate a completely new node placement until a path exists between every pair of nodes, i.e., until the communication graph is connected.

Like in the single-hop case, we first evaluate the effect on varying the packet length on the PRRs and latency (Figure 5). 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.

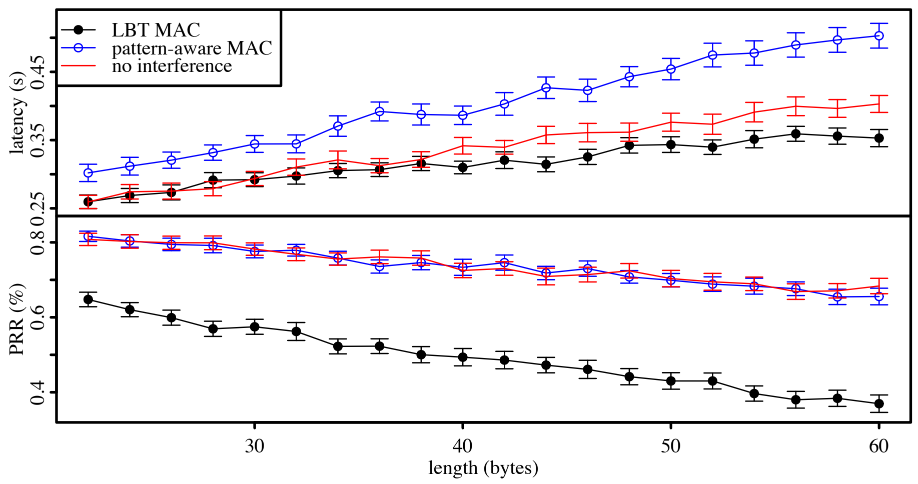

Fig. 5. In a connected multi-hop network, the effect of the packet length on the packet reception ratio and the latency.

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 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 6).

Since the network is static, we would expect little change in the expected number of hops as the packet length increases. However, we notice that the

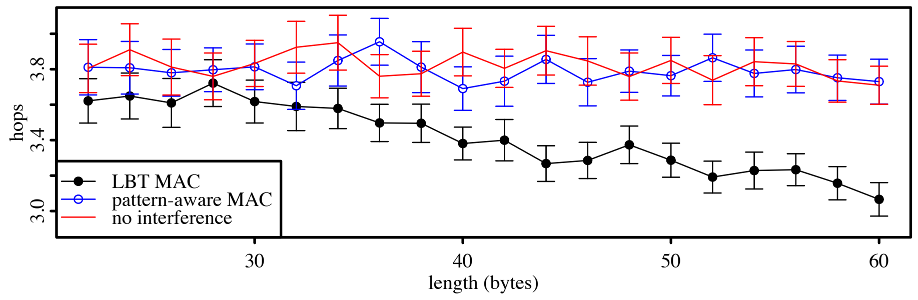

Fig. 6. In a connected multi-hop network, the effect of the packet length on the mean number of hops.

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

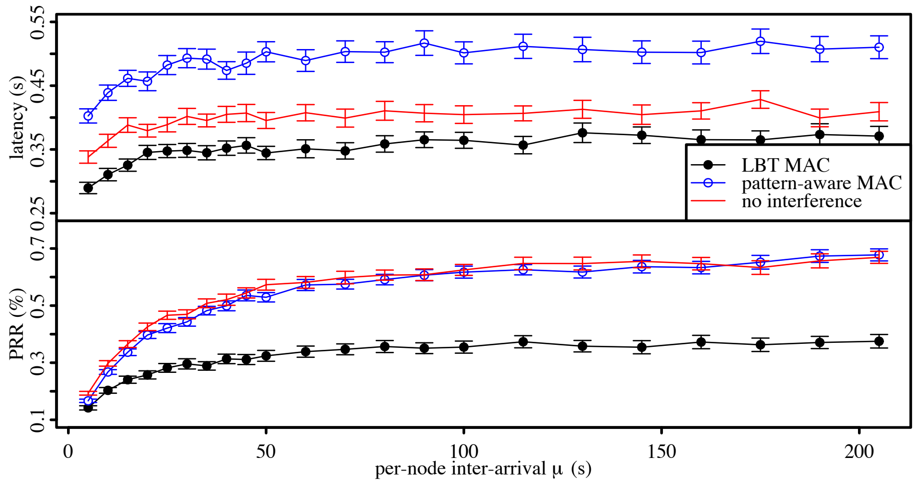

Fig. 7. In a connected multi-hop network, the effect of the mean latency between per-transmitter packet introductions on the packet reception ratio and the latency.

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.

6 Conclusion

In this paper, based on our previous work on simplification of the Lomb periodogram for the post-deployment identification of frequent impulsive interference, we extend an existing simulator with a flexible interface for the production of impulsive interference. Subsequently, we incorporated the impulse classifier and a proof-of-concept pattern-aware MAC (PA-MAC) into the simulator and simulated 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. Such a protocol would allow nodes to join the network without pausing communication while the evaluation occurs. Moreover, it may make sense to delegate the task of channel sampling solely to a subset of nodes (possibly the ones that have better energy reserves) while providing a way of disseminating the information about interference patterns to the rest of the network. 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.

References

1. 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.
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. 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.

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. 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.
6. J. Oetting. A comparison of modulation techniques for digital radio. *IEEE Transactions on Communications*, 27(12):1752 – 1762, Dec. 1979.
7. N. M. Boers, I. Nikolaidis, and P. Gburzynski. Sampling and classifying interference patterns in a wireless sensor network. *ACM Transactions on Sensor Networks*, (to appear).
8. 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.
9. 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.
10. 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.
11. 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.
12. T. Rusak and P. Levis. Physically-based models of low-power wireless links using signal power simulation. *Computer Networks*, 54(4):658 – 673, 2010.
13. 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.
14. 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.
15. N. M. Boers, P. Gburzynski, I. Nikolaidis, and W. Olesinski. Developing wireless sensor network applications in a virtual environment. *Telecommunication Systems*, 45(2):165 – 176, 2010.
16. 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 *MobiCom '10: Proceedings of the 16th Annual International Conference on Mobile Computing and Networking*, pages 317–328, Chicago, IL, USA, 2010. ACM.
17. 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.
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.