WIRELESS INTERFERENCE PREDICTION WITH DISCRETE WINDOWS

By

Michael Farcasin

Bachelor of Science
University of Tennessee
Knoxville, TN
USA
2010

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2017

WIRELESS INTERFERENCE PREDICTION WITH DISCRETE WINDOWS

Thesis Approved:

Dr. Eric Chan-Tin

Thesis Advisor

Dr. K. M. George

Committee Member

Dr. Christopher Crick

Committee Member

Acknowledgments

The research results discussed in this publication were made possible in total or in part by funding through the Health Research award for project number HR13-035, from the Oklahoma Center for the Advancement of Science and Technology.

Contents

List of Tables

List of Figures

Figure Page

Chapter 1

Abstract

Low-power devices that rely on ZigBee wireless signals, such as embedded health care devices, face the problem of being drowned out by higher-power signals like WiFi. When that happens, those devices must spend time and battery power retransmitting those signals. In order to reduce collisions with WiFi signals and increase throughput, we investigate using Hidden Markov Models (HMMs) with discrete timeslots to predict when interference would occur, and when it would not (a white space). We found that, when we used short timeslots, the HMMs made slower and less accurate predictions than existing schemes. However when we looked at longer timeslots, HMM predictions made better use of the available white space.

Chapter 2

Introduction

Health care is coming to peoples' homes. Embedded health care devices are seeing an increase in usage and it is predicted that their use and adoption will increase in the coming years. Embedded health care devices, such as pacemakers, hearing aids, and home sensors are a new, pervasive, and better way to monitor patients health. Their use can both reduce trips to the hospital and enable health care providers to monitor patients remotely. However, unlike similar technologies such as smartwatches and other smarthealth devices, these sensors use the ZigBee (IEEE 802.15.4) protocol for communication. Moreover, the low-power sensor can last for months or years on a single charge. These sensors communicate to a base station connected to the Internet.

ZigBee is typically used for low-power, short-range communication. ZigBee uses signals resembling Bluetooth and WiFi (802.11), and operates on the same frequency (2.4GHz). However, the fact that ZigBee is low-power means that other wireless signals such as WiFi and microwave oven drown out ZigBee signals. In order to communicate in the presence of WiFi signals, ZigBee devices have to avoid signal

interference by either moving to a different frequency, or transmitting at a different time. These times or locations where there are no signals are called "white spaces". An interference for a ZigBee sensor means loss of energy in sending data as all the data sent from the sensor are not received by the base station.

Our goal is thus to create an algorithm that allows a ZigBee device to predict whether a white space will occur next so that it can power up its antenna and start sending data. The algorithm has to run continuously because as soon as it predicts that a white space will not occur, that is, that interference will happen, the device has to stop sending data and wait for the next white space prediction. As more healthcare devices will be used in the homes and in public places, battery usage and wireless interference will become an issue. The motivation to develop this algorithm is to avoid wireless interference and conserve battery usage.

We propose an algorithm using a Hidden Markov Model (HMM) to predict white spaces based on recently seen data. The HMM algorithm relies on training data to predict when the next white space will occur. We vary the number of observations for the training data and the number of states in the HMM to obtain the accuracy of the prediction. To predict when the next white space will occur, the algorithm looks at a small "window" size and attempts to match that window to the past data. Once a match is found, whatever happened next in that past data is used as prediction

The rest of the document is organized as follows. In chapter 3, we review wireless sensor networks, Hidden Markov Models, and related work. In chapter 4 we detail our algorithm's design and implementation, as well as our experimental setup. In chapter 5, we review the results. We present our conclusions and discuss future work

in chapter 6.

Chapter 3

Background

3.1 Wireless Sensor Protocols

There are many protocols for wireless sensor networks such as B-MAC [1], S-MAC [2], and WISEMAC [3]. The current wireless protocol used by ZigBee sensors is B-MAC. B-MAC is a lightweight, energy efficient, scalable, simple, and easily configurable Medium Access Control protocol. B-MACs advantages over other wireless sensor network protocols are its small code footprint and easy configuration. B-MAC is energy-efficient as it uses low-power listening (LPL) to check for wireless activity. B-MAC also does not implement features to handle hidden terminals or fragmentation of data. B-MAC uses CSMA/CA.

3.2 Hidden Markov Models (HMM)

A Markov chain is a model for a system made up of internal states that can each produce all possible observations, usually with a different probability distribution for

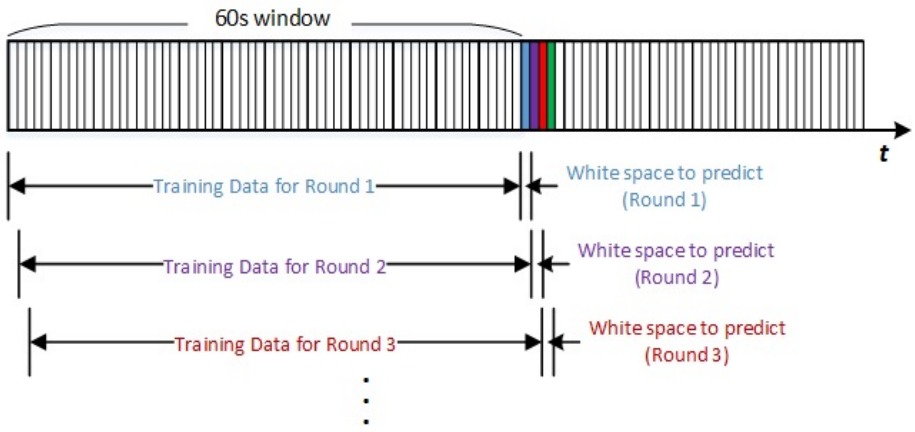

Figure 3.1: Periods and sliding windows used for training and prediction

each state, and all the states are observable. When training a Markov chain with a set of observations, the Markov chain computes a pair of probability matrices: the transition and emission probability matrices, which together form the Markov model. The transition matrix contains the probability of switching from one state to another, and the emission matrix contains the probability of producing a given observation from a given state. By combining the transition and emission matrices with a new sequence of observations, we can compute the probability of a particular output sequence, or the most likely next state and observation, as shown in Figure 3.1.

In a HMM, the internal states and their transitions are not directly observable (hence, "hidden") but their observations, whose probabilities are assumed to depend on the states, can be observed. Thus, a sequence of observations gives information about the sequence of the hidden internal states, and can be used to train a HMM. We show an example HMM in Figure 3.2.

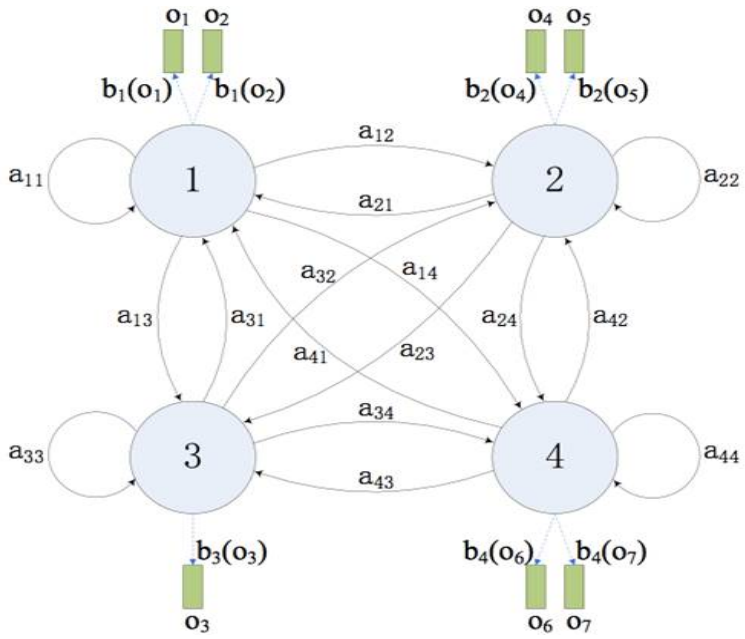

Figure 3.2: An example HMM with 4 states, 7 possible observations, a transition matrix A, and an emission matrix B.

3.3 Related Work

In the past, algorithms have been developed to detect interference and respond appropriately to combat negative affects. Yi et al. [4] detect interference and switch to clear channels. Hsu et al. [5] proposed a scheme that tries to optimize MAC layer packet length dynamically by adjusting the fragmentation level based on the packet error rate. Guo et al. [6] propose a real-time adaptive transmission (RAT) scheme to estimate Zigbee packet corruption to support the selection of appropriate error-correction coding to maximize throughput. The results of their extensive tests indicate that RAT significantly improves ZigBee throughput in the presence of relatively high WiFi throughput. Our work differs from these techniques as we use traffic observations to predict when the channel will be clear (a white space) before sending transmissions.

Other recent techniques use predictive methods to attempt to avoid experiencing interference. Frequency hopping communications limit continued interference for channels, and enhancements have been made to frequency hopping algorithms to improve performance under different types of interference [7, 8]. Hasan et al. [9] improved on previous work with a frequency hopping algorithm with a Gaussian distribution for choosing transmission frequencies. Srinivasan et al. [10] measured ZigBee reception and found that most intermediate links are bursty, that burstiness affects protocol performance, and that they could predict the effects. Dubay and Patel [11] sought a method to improve the performance of WiFi in presence of radio interference. Their work proposed a method to protect WiFi performance through

the HMM-driven White Space for WiFi protocol based on their characterization of the dynamic distribution of white space durations in the time domain using a Hidden Markov Model, and note its applicability to Zigbee, Bluetooth, and Wi-Max as well.

Further, Huang et al. [12] introduced a Pareto model that characterizes the white space of WiFi traffic and introduced a frame control protocol called WISE (White Space-aware Frame Adaptation) for ZigBee networks. WISE predicts the length of white space in WiFi traffic based on the Pareto model, and intelligently adapts frame size to maximize the throughput efficiency and provide assured levels of performance. Similarly, Ananthanarayanan and Stoica [13] presented a system that predicts the availability of the WiFi connectivity by using a combination of Bluetooth contact-patterns and cell-tower information. This allows a device to intelligently switch the WiFi interface online only when there is WiFi connectivity available, thus avoiding the long periods in idle state and significantly reducing the number of scans for discovery. We also use a Hidden Markov Model, but show through real dataset experiments that our algorithms had lower overhead with acceptable accuracy.

Yuan et al. [14] suggested that a Hidden Markov Model (HMM) accounts for white spaces better than a Pareto model, and that an algorithm incorporating this model can provide better throughput for ZigBee packets than existing protocols. They proposed the HMM-driven Smart White-space-aware Frame Control Protocol based on their HMM model of white spaces, and showed some experimental results supporting the effectiveness of the protocol in protecting communication performance. Our work significantly improves on their work as we show extensive experimental results based on real datasets.

Finally, in [15], we initially experimented with HMMs to predict and avoid WiFi white space. We found our model produced lower overhead than earlier models in a simulated environment. However, our results were limited to a single dataset, and didn't directly compare battery usage with and without the models. Therefore in this paper we perform experiments on a new dataset and simulate battery usage by calculating the genuine match rate (GMR) for white space in our models' predictions.

Chapter 4

Experimental Setup

In all our experiments, we used Matlab to store data about our models and make predictions. We chose Matlab because it was an easy interface with which to create simulations, especially because it contained built-in functions to create HMMs.

4.1 Datasets

For our experiments, we used wireless data from the crawdad web site [16]. Crawdad provided pcap files containing the frame length, channel frequency, RSSI, and microsecond capture time for wireless packets. We chose two datasets from the site: the first was the same dataset we used in [15] from OSDI [17] (initial version), and the second was from SIGCOMM [18].

For the OSDI dataset, the authors "gathered traces of wireless traffic at several monitoring nodes distributed across the conference floor and breakout areas [at OSDI 2006]. In addition, we gathered traces on the wired switch to which the wireless access points connect." [17]

For the SIGCOMM dataset, the authors "collected a trace of wireless network activity at SIGCOMM 2008. The subjects of the traced network chose to participate by joining the traced SSID." [18]

For our experiments, we converted the data in the pcap files of each dataset into discrete, 1-millisecond timeslots (or windows). Then for each window, we stored the start time of the window and the number of frames observed during that window. In the rest of this paper, we will refer to frames as interference, and to windows without any frames as white spaces. We chose 1 ms windows because that was the smallest time period that was large enough for a sensor to turn its antenna online for sensing or sending data. Finally, we stored the converted data — hereafter referred to as "window data" — in Matlab (.mat) files, with one Matlab file to one pcap file.

When making predictions in our experiments, we iterated through the windows in the window data and used the window data as the ground truth for our observations. This allowed us to abstract out the time required to sense for transmissions and simply measure the time required to compute a prediction with one of our models.

4.2 Frequency Tables

To perform an initial measurement of our data and provide a basis for training our prediction models, we created 3x3 frequency tables for our datasets. To do this, for each dataset, we scanned all the windows in each file in the dataset and compiled the data in a single table. Tables 4.1 and 4.2 show the frequency tables (with row and column headers) for the two datasets.

	white space	interference	%
white space	9,390,000,000	7,007,200	71.16
interference	7,007,181	3,800,862,000	28.84
%	71.16	28.84	

Table 4.1: The frequency table for the OSDI data.

	white space	interference	%
white space	315,050,000	6,114,100	50.21
interference	6,114,146	312,386,300	49.79
%	50.21	49.79	

Table 4.2: The frequency table for the SIGCOMM data.

In each table, the first row after the headers shows the number of white space windows, and the second row shows the number of windows with interference. The first column after the headers shows the number of white space windows that followed a window from the given row, and the second column shows the number of windows with interference that followed a window from the given row. For example, in the OSDI dataset, there were 9,390,000,000 windows that were white space and were followed by a white space, whereas there were 7,007,200 windows that were white space but followed by interference. The last row and column in show the percentage of windows in that row or column, respectively, of the total.

In addition to providing a base to build our models, these tables showed us two things. First, we saw the distribution of white space to interference in our datasets:

Testing dataset	Total Accuracy (%)			White-space GMR (%)		
	mean	median	std. dev.	mean	median	std. dev.
OSDI	99.99	99.99	0.00	99.91	99.91	0.03
SIGCOMM	98.99	99.27	0.60	97.99	98.00	1.09

Table 4.3: Accuracy and timing results for the sense-and-send model.

71:29 for the OSDI dataset, and 50:50 for the SIGCOMM dataset. Second, we saw that the datasets were extremely bursty when we used 1-ms windows: in both datasets, white space almost always preceded another white space, and interference almost always preceded more interference.

4.3 Creating a Baseline for Comparison

As a comparison for our model, we wanted to get an idea of the accuracy for a device that sent data without any prediction model, but which followed the CSMA-CA algorithm. In other words, a "sense-and-send" device that simply sensed the line and sent its data when it detected a white space. We simulated such a device with a program that scanned the window data and always "predicted" the next window would match the current one. In a sense, we were testing the sparsity of white spaces in the datasets. The timing and accuracy results for this model are shown in Table 4.3.

We found that the sense-and-send model was both fast and accurate. It required on average less than 0.0001 ms per prediction for the OSDI dataset, and less than

0.003 ms per prediction for the SIGCOMM dataset. The table shows that the model produced a white-space genuine match rate (GMR) of 99% and 97%, respectively, for the OSDI and SIGCOMM datasets, with nearly 100% and 98% total accuracy. We calculated white-space GMR as $1 - \frac{\text{\# predicted collisions}}{\text{\# predicted white space}}$, i.e. 1− the false match rate.

These were the results we expected. Since we chose windows that only lasted 1 ms, the OSDI dataset had on approximately 100% consecutive white space, and the SIGCOMM dataset had approximately 98% consecutive white space — the same as their white-space GMR. We discuss experiments with larger windows in chapter 4.7.2.

We knew that because the sense-and-send model only changed its predictions when it first sensed white space or interference, it would always make the wrong prediction for that window. Our goal then was to produce a model with fewer collisions or greater white-space usage than the sense-and-send model. We calculated white-space usage as $\frac{\text{\# correctly-predicted white space}}{\text{\# predicted white space}}$

4.4 Prediction With Direct Probability

The simplest model we considered was one that would make predictions based solely on the data in the frequency tables. Our underlying assumption for this model was that although the frequency tables were created from a single dataset, they would still apply to to other datasets.

Therefore, we created a model that simply checked the chance of white space in the frequency table and then randomly predicted white space with that probability

for each window. The pseudocode for these predictions is shown in Algorithm 1.

Note that the direct-probability model differs from sense-and-send model in 2 important respects:

I. The sense-and-send model senses for interference, whereas the direct-probability model does not.

II. The direct-probability model makes probabilistic predictions based on the frequency table, whereas the sense-and-send model deterministically predicts that the next window will be the same as the current one.

Algorithm 1 Making predictions with the direct-probability model

```
Get P(white space) from the frequency table
Choose a random number between 0 and 1
if the number was less than P(white space) then
    Predict white space for the next window
else
    Predict interference for the next window
end if
```

4.5 Prediction with Bayesian Probability

For a slightly-more-complex model, we used Bayesian probability to predict whether a window would be white space or not. Bayesian probability follows the equation:

$$P(a|b) = \frac{P(b|a) \cdot P(a)}{P(b)}$$

Therefore, we created a model that checked whether the current window was busy or not, then checked the frequency table to obtain the parameters:

- a = an observation of white space
- b = the observation for the current window (whether it was white space or interference)

The pseudocode for these predictions is shown in Algorithm 2.

Algorithm 2 Making predictions with the Bayesian probability model

Get the observation for the current window
Use the frequency table to calculate P(b | a) = (the count of the current observation following a white space) / (total # of observations)
Use the frequency table to calculate P(a) = the probability of a white space
Use the frequency table to calculate P(b) = the probability of the current observation following any other observation
Calculate P(a | b) = P(b | a) * P(a) / P(b)
Choose a random number between 0 and 1
if the number was less than P(a | b) **then**
  Predict white space for the next window
else
  Predict interference for the next window
end if

4.6 Prediction with HMMs

We used a different method than [15] in an attempt to create a HMM that achieved better results:

I. We used the frequency tables for our dataset to create initial estimates for our transition and emission matrices, as shown in Figure 4.4.

II. We selected the first 1000 observations from the first file in our dataset to use for training.[1]

III. We used those estimates with Matlab's hmmtrain function [19] to get the trained transition and emission matrices.

Transition estimate

P(white space)	1 - P(white space)
1 - P(white space)	P(white space)

Emission estimate

P(consecutive white space)	1 - P(consecutive white space)
1 - P(consecutive interference)	P(consecutive interference)

Table 4.4: The initial estimates for the transition (above) and emission (below) estimates. Note the probabilities above come from the frequency table we created before.

[1]We tested 100, 1000, 10000, and 100000 observations, and found that the algorithm converged to the same result when using 1000 or more.

0.99961	0.00039297
0.0014683	0.99853

1	1.2815E-17
7.9241E-14	1

Table 4.5: The transition (left) and emission (right) matrices for the HMM trained on the OSDI dataset.

At the end of these steps, the hmmtrain function converged. We performed these steps multiple times (more information below), and achieved roughly the same models each time. An example of our transition and emission matrices for each dataset is shown in Tables 4.5 and 4.6. The tables for each are close to the identity matrix, implying that our HMM would almost always predict the same state it started with.

When making predictions, for each window, we used Matlab's hmmdecode function [20] to first compute the the most-likely state for the window based on our observation for that window. Then we computed the most-likely state for the next window, and its observation. We used that observation as our next prediction. The pseudocode for these predictions is shown in Algorithm 3.

Algorithm 3 Making predictions with the HMM

Get the observation for the current window

Use hmmdecode with our HMM to get the most recent posterior state

Multiply the posterior state matrix by the transition matrix

Multiply the new matrix by the emission matrix

Choose as our next prediction the most likely emission from the new matrix

0.9847	0.015298
0.018629	0.98137

1	3.3993E-12
3.0646E-11	1

Table 4.6: The transition (left) and emission (right) matrices for the HMM trained on the SIGCOMM dataset.

4.7 Experiments

4.7.1 1-ms Windows

For each algorithm, we conducted 4 experiments:

I. We trained the model on the OSDI dataset and tested it on the OSDI dataset.

II. We trained the model on the OSDI dataset and tested it on the SIGCOMM dataset.

III. We trained the model on the SIGCOMM dataset and tested it on the OSDI dataset.

IV. We trained the model on the SIGCOMM dataset and tested it on the SIGCOMM dataset.

In each experiment, we measured the total accuracy of the algorithm and the time taken per prediction, then calculated the mean, median, and standard deviation of those values across all window data files.

We began by running 10 trials of each experiment. For HMMs, this included re-creating the transition and emission matrices in each trial.

4.7.2 Larger Windows

We saw that with 1 ms windows, we had hundreds and thousands of consecutive white space (and interference) windows, which not only resulted in artificially-high accuracy for the sense-and-send model, but also resulted in very simple HMMs. Although we had chosen the smallest possible window size to achieve the greatest accuracy with our models, we wanted to investigate the accuracy of our model with larger windows. We reasoned that if our models proved to be accurate on larger windows, we should use them with larger windows. That would increase efficiency by reducing the number of predictions needed to send ZigBee data.

We created 4 variations of our SIGCOMM dataset with larger windows:

- 5 ms
- 10 ms
- 50 ms
- 100 ms

Then we repeated the previous experiments, but used only the models trained on the OSDI dataset. We chose these models for two reasons:

I. We wanted to test our established models on a "new" dataset.

II. We saw in our initial results that training on the SIGCOMM dataset had only a small effect on the HMM compared to the OSDI dataset.

Chapter 5

Evaluation

5.1 Results With 1-ms Windows

We present the mean accuracy results for our experiments in Figure 5.1 and Tables 5.7, 5.8, and 5.9. In the plots, 'Accuracy' means the total accuracy measured as the number of correct predictions over the total (regardless of whether they were white space or interference), 'GMR' means the white-space GMR, calculated as above, and 'Usage' means the white-space usage, also calculated as above.

We found that the results were almost identical across trials, with less than 0.01% or less difference in numbers between each trial. Therefore, each number on the table is the average of that measurement across all 10 trials.

We found that both the direct- and Bayesian-probability models performed significantly worse than the baseline sense-and-send model, with accuracy as low as 65%. On the other hand, the HMM produced approximately the same accuracy as the sense-and-send model, regardless of which dataset it was trained on, just as we expected.

In fact, the HMM predicted the same number of white spaces and collisions (i.e. when we predicted a white space and found a busy window), and achieved the same white-space GMR as the sense-and-send model.

However, the HMM took between 10-1000x longer to make predictions than the sense-and-send model. The sense-and-send model took approximately 0.00002 ms per prediction for the OSDI dataset and 0.003 ms for the SIGCOMM dataset, as did the direct- and Bayesian-probability models, while the HMM took 0.03 ms for the OSDI dataset and 0.06 ms for the SIGCOMM dataset.

By comparison, the direct probability model, while less accurate, was very fast. Since it doesn't rely on sensing the channel before making a prediction, it might be possible to save battery power with the direct model depending on how long it takes to sense the channel.

Having said that, HMM had higher white-space GMR and total accuracy than either the direct- or Bayesian-probability models, with as much as a 30% increase in accuracy on the SIGCOMM dataset. However, the direct-probability model ran almost as quickly as the sense-and-send model, and the Bayesian-probability model ran 10x faster than the HMM.

5.2 Results With Larger Windows

We show the results of our experiments with larger windows in Figure 5.2.

As window size grows, the sense-and-send model tends to continue to have the

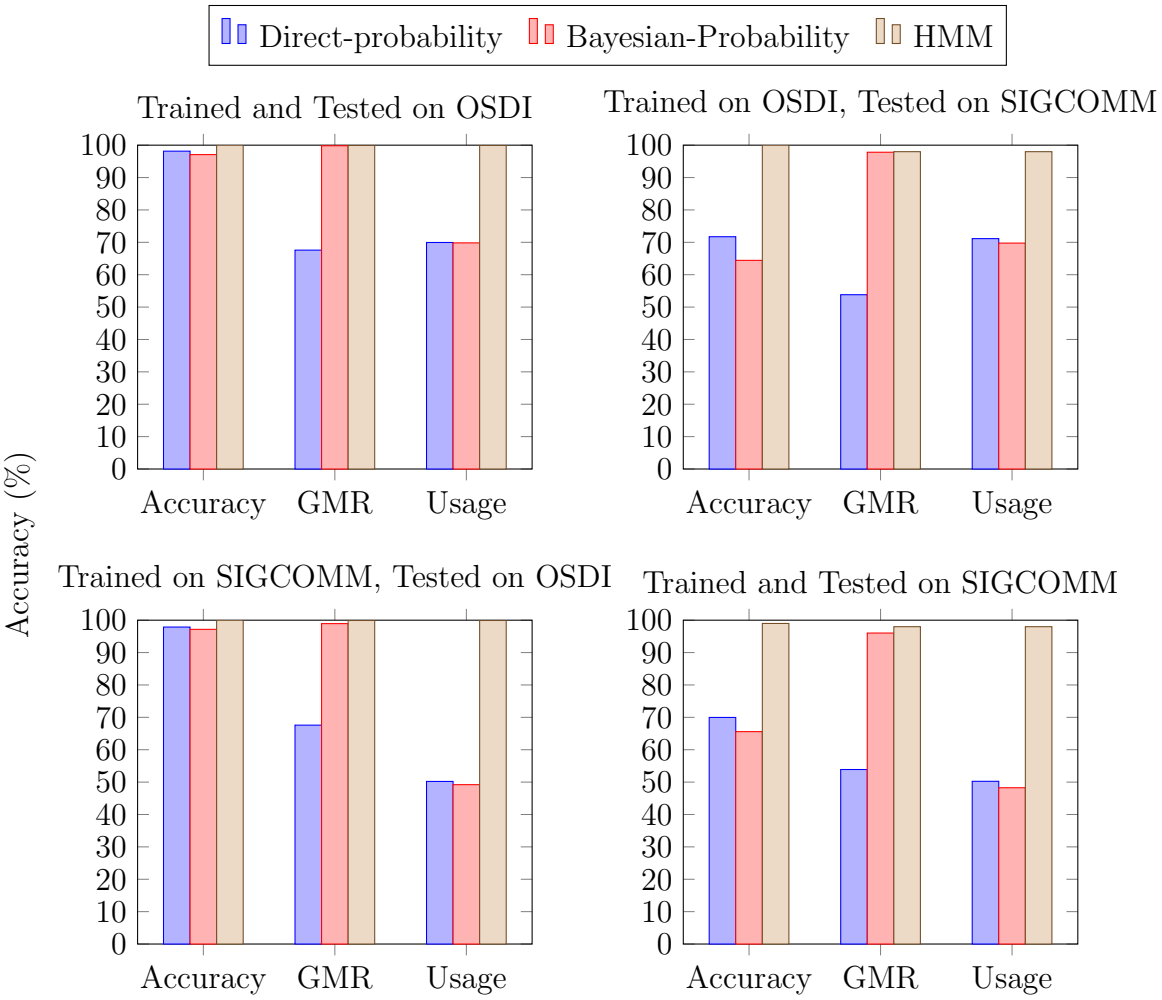

Figure 5.1: Accuracy results.

Training dataset	Testing dataset	Model	Total Accuracy (%)		
			mean	median	std. dev.
OSDI	OSDI	Direct	98.15	98.22	0.68
		Bayes	97.09	97.16	0.01
		HMM	**99.99**	99.99	0.00
	SIGCOMM	Direct	71.73	80.14	13.96
		Bayes	64.43	72.95	23.97
		HMM	**98.99**	99.27	0.60
SIGCOMM	OSDI	Direct	97.88	97.96	0.67
		Bayes	97.17	97.24	00.78
		HMM	**99.99**	99.99	0.00
	SIGCOMM	Direct	69.97	75.05	15.12
		Bayes	65.58	75.65	23.48
		HMM	**98.99**	99.27	0.60

Table 5.7: Accuracy results for the prediction models. The first column is the database used to train the models, and the second column is the database the models were tested on.

Training dataset	Testing dataset	Model	White-space GMR (%)		
			mean	median	std. dev.
OSDI	OSDI	Direct	67.59	69.16	8.07
		Bayes	99.82	99.83	0.07
		HMM	**99.91**	99.91	0.03
	SIGCOMM	Direct	53.82	49.89	18.80
		Bayes	97.82	97.83	1.21
		HMM	**97.99**	98.00	1.09
SIGCOMM	OSDI	Direct	67.59	69.16	8.07
		Bayes	98.94	99.04	0.41
		HMM	**99.91**	99.91	0.03
	SIGCOMM	Direct	53.89	49.84	18.80
		Bayes	96.03	96.01	2.35
		HMM	**97.99**	98.00	1.09

Table 5.8: White-space GMR for the prediction models.

Training dataset	Testing dataset	Model	White-space Usage (%)		
			mean	median	std. dev.
OSDI	OSDI	Direct	69.95	69.95	0.00
		Bayes	69.83	69.83	0.02
		HMM	**99.91**	99.91	0.03
	SIGCOMM	Direct	71.14	71.16	0.43
		Bayes	69.76	69.62	1.01
		HMM	**97.99**	74.23	16.38
SIGCOMM	OSDI	Direct	50.21	50.21	0.00
		Bayes	49.21	49.21	0.02
		HMM	99.91	99.91	0.03
	SIGCOMM	Direct	50.25	50.20	0.61
		Bayes	48.26	48.21	0.70
		HMM	**97.99**	74.23	16.38

Table 5.9: White-space usage for the prediction models.

best total accuracy and white-space GMR. At 100-ms windows, it is overshadowed in white-space usage by our HMM, but in that scenario the HMM also has low white-space GMR — in other words, a high number of collisions.

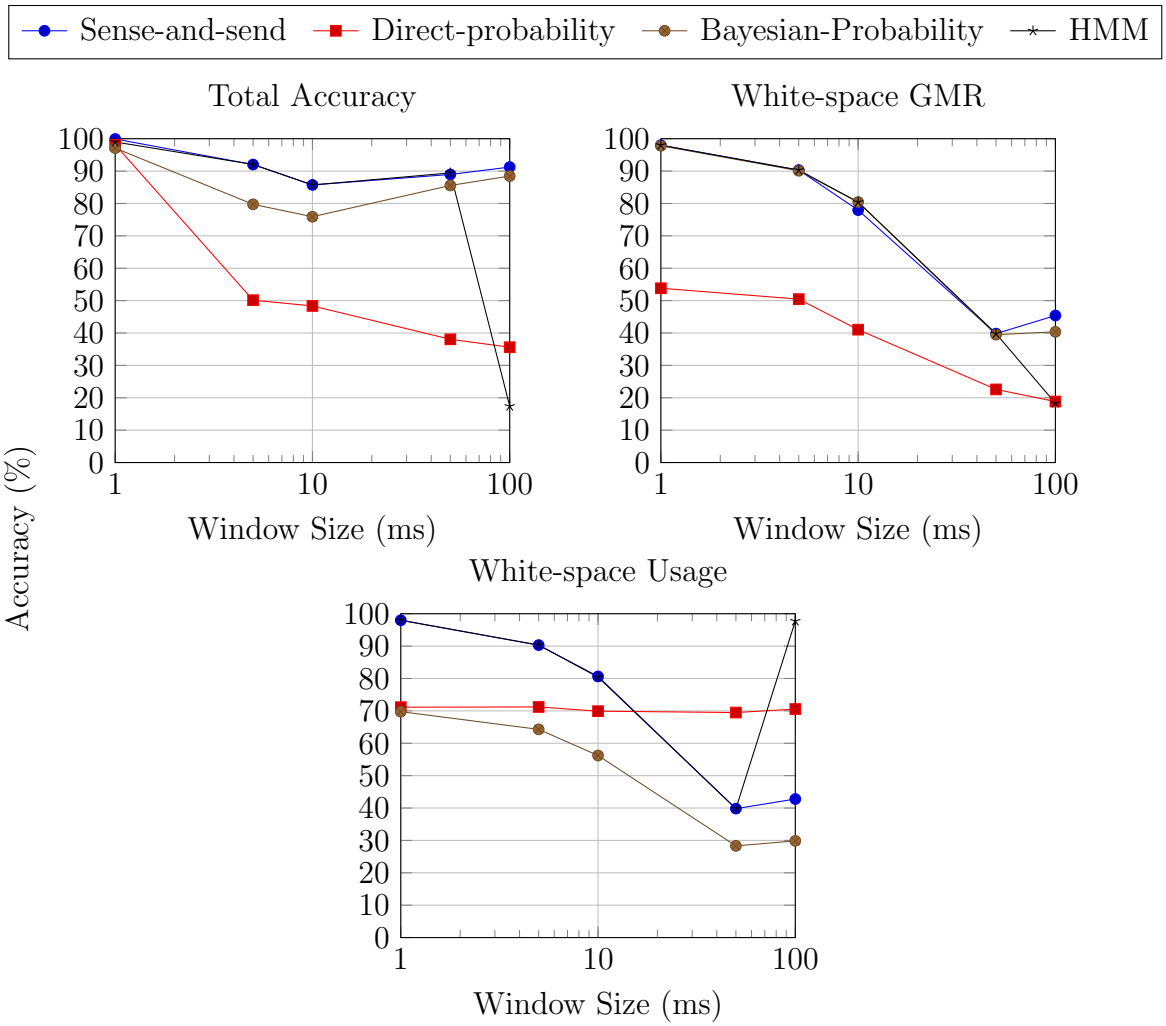

Figure 5.2: Results for the experiments with larger window sizes.

Chapter 6

Conclusion and Future Work

We investigated the use of HMMs in predicting white spaces in wireless traffic. We implemented a sense-and-send model to simulate CSMA-CA, and 3 predictive models: a direct-probability model, a Bayesian-probability model, and a HMM. We compared the models by total accuracy, white-space GMR, white-space usage, and time per prediction. We showed that our HMM produced better accuracy and white-space GMR than either the direct-probability or Bayesian-probability model, but no more so than sense-and-send. And the HMM required more time than any of the other models. On the other hand, direct-probability requiring the least of the predictive models, making it possibly faster than sense-and-send while still having significant accuracy.

As we scaled the size of the windows, sense-and-send continued to have the best performance, with the 3 predictive models producing mixed results.

In the future, we would like to build a continuous HMM using the original data from the pcap files in our datasets to predict when the next transmission will occur, rather than whether the next 1-millisecond window will contain a transmission or

not. We believe the bursty nature of network traffic that we observed would lend itself well to this type of model.

Bibliography

[1] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 95–107, ACM, 2004.

[2] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1567–1576, IEEE, 2002.

[3] A. El-Hoiydi and J.-D. Decotignie, "Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks," in *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, vol. 1, pp. 244–251, IEEE, 2004.

[4] P. Yi, A. Iwayemi, and C. Zhou, "Developing zigbee deployment guideline under wifi interference for smart grid applications," *Smart Grid, IEEE Transactions on*, vol. 2, no. 1, pp. 110–120, 2011.

[5] A. C.-C. Hsu, D. S. Wei, and C.-C. J. Kuo, "Coexistence mechanism using dynamic fragmentation for interference mitigation between wi-fi bluetooth," in

Military Communications Conference, 2006. MILCOM 2006. IEEE, pp. 1–7, IEEE, 2006.

[6] P. Guo, J. Cao, K. Zhang, and X. Liu, "Enhancing zigbee throughput under wifi interference using real-time adaptive coding," in *INFOCOM, 2014 Proceedings IEEE*, pp. 2858–2866, IEEE, 2014.

[7] A.-C. Hsu, D. S. Wei, C.-C. Kuo, N. Shiratori, and C.-J. Chang, "Enhanced adaptive frequency hopping for wireless personal area networks in a coexistence environment," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pp. 668–672, IEEE, 2007.

[8] Z. Jiang, V. C. Leung, and V. W. Wong, "Reducing collisions between bluetooth piconets by orthogonal hop set partitioning," in *Radio and Wireless Conference, 2003. RAWCON'03. Proceedings*, pp. 229–232, IEEE, 2003.

[9] M. M. Hasan, R. Prakash, and J. P. Jue, "Dynamic coexistence of frequency hopping networks using parallel and gaussian allocations," in *Communications, 2009. ICC'09. IEEE International Conference on*, pp. 1–5, IEEE, 2009.

[10] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: measuring wireless link burstiness," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 29–42, ACM, 2008.

[11] N. Dubay and V. Patel, "Hmm driven white space for ieee 802.11," *International Journal of Engineering Trends and Technology*, pp. 217–219, 2014.

[12] J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond co-existence: Exploiting wifi white space for zigbee performance assurance," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pp. 305–314, IEEE, 2010.

[13] G. Ananthanarayanan and I. Stoica, "Blue-fi: enhancing wi-fi performance using bluetooth signals," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pp. 249–262, ACM, 2009.

[14] J. Yuan, T. Ward, S. Honarvar, T. Chen, and J. Thomas, "Hmm-driven smart white-space-aware frame control protocol for coexistence of zigbee and wifi," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pp. 348–351, IEEE, 2013.

[15] J. Yu, M. Farcasin, and E. Chan-Tin, "Wireless interference prediction for embedded health devices," in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pp. 1–6, IEEE, 2017.

[16] "Crawdad." `http://crawdad.org`. Accessed: 2017-10-18.

[17] R. Chandra, R. Mahajan, V. Padmanabhan, and M. Zhang, "CRAWDAD dataset microsoft/osdi2006 (v. 2007-05-23)." Downloaded from `https://crawdad.org/microsoft/osdi2006/20070523`, May 2007.

[18] A. Schulman, D. Levin, and N. Spring, "CRAWDAD dataset umd/sigcomm2008 (v. 2009-03-02)." Downloaded from `http://crawdad.org/umd/sigcomm2008/20090302`, Mar. 2009.

[19] "Hidden markov model parameter estimates from emissions - matlab hmmtrain." https://www.mathworks.com/help/stats/hmmtrain.html. Accessed: 2017-10-18.

[20] "Hidden markov model posterior state probabilities - matlab hmmdecode." https://www.mathworks.com/help/stats/hmmdecode.html. Accessed: 2017-10-19.

VITA

Michael Farcasin

Candidate for the Degree of

MASTER OF SCIENCE

Thesis: WIRELESS INTERFERENCE PREDICTION WITH DISCRETE WINDOWS

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Bachelor of Science at University of Tennessee, Knoxville, TN in May 2010

Experience:

Teaching Assistant, Oklahoma State University (Aug 2012—Present)

For the past 4 years, I have been responsible for grading homework, teaching classes, and providing additional help to students outside of classes.

Professional Memberships:

Association of Computing Machinery

Information Security and Assurance Club