University of California
Merced

Data-Driven Modeling of Phenomena in Wireless Sensor Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering and Computer Science

by

Ankur U. Kamthe

2012

The dissertation of Ankur U. Kamthe is approved.

David Noelle

Mani B. Srivastava

Miguel Á. Carreira-Perpiñán

Alberto E. Cerpa, Committee Chair

University of California, Merced

2012

Dedicated to my parents, Anuradha and Uday, and my wife, Gayatri.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my co-advisors Alberto Cerpa and Miguel Á. Carreira-Perpiñán for their support and advice throughout the last six years. I am also thankful for the support of my dissertation committee, David Noelle and Mani Srivastava whose advice and suggestions were very much appreciated.

A big thanks also go to my wife, Gayatri, for making this work possible and my time at UC Merced all the more enjoyable. My aunt, Archana and uncle, Shirish, alongwith their kids Priya, Maya and Comet for their loving support. Finally, I am also thankful for the delightful company of my colleagues, Lun Jiang, Tao Liu, Varick Erickson, Stefan Achleitner, Andreas Kolling, Benjamin Balaguer, Gorkem Erinc, Derek Burch, Chao Qin, Weiran Wang, Max Vladymyrov and Mohsen Farhadloo, and friends at UC Merced, especially Gosia Kolling (Skorek), Ryan Lucas, Alisa Keyser and Gokce Ugur.

Vita

1981	Born, Mumbai, India.
2002	Bachelor of Electronics Engineering, Mumbai University, India.
2005	Master of Science (Electrical Engineering), Auburn University.
2006	Master of Probability and Statistics, Auburn University.
2006–2012	Research Assistant, Electrical Engineering and Computer Science, University of California–Merced.

Publications

Ankur Kamthe, Varick L. Erickson, Miguel A. Carreira-Perpinan and Alberto E. Cerpa, Enabling building energy auditing using adapted occupancy models, In the Proceedings of the 3rd ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys 2011), November, 2011.

Ankur Kamthe and Soo-Young Lee, A stochastic approach to estimating earliest start times of nodes for scheduling DAGs on heterogeneous distributed computing systems, In Cluster Computing, Springer, 2011.

Ankur Kamthe, Miguel A. Carreira-Perpinan and Alberto E. Cerpa, Adaptation of a mixture of multivariate Bernoulli distributions, In the proceedings of

the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), Barcelona, Spain, 2011.

Ankur Kamthe, Miguel A. Carreira-Perpinan and Alberto E. Cerpa, M&M: Multilevel Markov Model for Wireless Link Simulation in Sensor Networks, In the Proceeding of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009), Berkeley, CA, November 2009.

Varick L. Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Alberto Cerpa, Michael D. Sohn and Satish Narayanan, Energy Efficient Building Environment Control Strategies Using Real-time Occupancy Measurements, In the Proceedings of the 1st ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys 2009), November, 2009.

Lun Jiang, Jyh-How Huang, Ankur Kamthe, Tao Liu, Ian Freeman, John Ledbetter, Shivakant Mishra, Richard Han and Alberto Cerpa, SenSearch: GPS and Witness Assisted Tracking for Delay Tolerant Sensor Networks, In the Proceedings of the Eighth International Conference on Ad hoc Networks and Wireless (ADHOCNOW 2009), Springer-Verlag, Murcia, Spain, September, 2009.

Tao Liu, Ankur Kamthe, Lun Jiang and Alberto Cerpa, Performance Evaluation of Link Quality Estimation Metrics for Static Multihop Wireless Sensor Networks, In the Proceedings of the Sixth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks(SECON 2009), IEEE, Rome, Italy, June, 2009.

Ankur Kamthe, Lun Jiang, Matt Dudys, Alberto Cerpa, SCOPES: Smart Cameras Object Position Estimation System, In the Proceedings of the 6th European Conference on Wireless Sensor Networks(EWSN 2009), Springer-Verlag, Cork, Ireland, February, 2009.

Ankur Kamthe and Soo-Young Lee, Stochastic Approach to Scheduling Multiple Divisible Tasks on a Heterogeneous Distributed Computing System, 21st International Parallel and Distributed Processing Symposium (IPDPS), Los Angeles CA, 2007.

Ankur Kamthe and Soo-Young Lee, A Stochastic Approach to Estimating Earliest Start Times of Nodes for Scheduling DAGs on Heterogeneous Distributed Computing Systems, 19th International Parallel and Distributed Processing Symposium (IPDPS), Denver CO, 2005.

Abstract of the Dissertation

Data-Driven Modeling of Phenomena in Wireless Sensor Networks

by

Ankur U. Kamthe

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Merced, 2012

Professor Alberto E. Cerpa, Chair

Wireless sensor networks (WSNs) is an emerging field with applications that span scientific, engineering, medical and other disciplines. Significant human and capital infrastructure is needed for testing behavior of WSNs in real-world deployments. Sensor network simulations have the advantage of facilitating testing without going through the rigors of a deployment, but require good simulation models. However, high-quality communication and phenomena models are extremely hard to come by and suffer from need of large quantity of training data to capture the time-varying nature of the underlying phenomenon. In my dissertation, I address issues in the modeling of wireless links (communication-related) and occupancy monitoring (application phenomenon-related). For wireless link modeling, I advocate a novel machine learning-based, data-driven approach involving Hidden Markov Models (HMMs) and Mixtures of Multivariate Bernoullis (MMBs) for modeling the long and short time scale behavior of links. For occupancy modeling, I propose SCOPES, a wireless camera sensor network for gathering building occupancy traces for aiding model creation. However, both problems require large quantity of training data to model the time varying nature

of the underlying phenomenon. For each instance, I solve the training data problem by using parameter-tying based model adaptation techniques that constrain the new model parameters through a nonlinear transformation of a pre-existing reference model. Using model adaptation, I showed that we can achieve significant reduction in training data requirement, thereby improving the simulation quality by enabling the construction of high-quality models

CHAPTER 1

Introduction

Wireless sensor networks (WSNs) are comprised of low power devices (nodes) equipped with limited computational and communication capability, and sensors for gathering data from the surrounding environment. Networks comprising of large numbers of these sensors have been deployed to gather data from the surrounding environment. WSNs have been used to monitor and study wildlife in their natural environment [92, 99], environmental phenomena such as volcanic eruptions [96], forest-fires [21] and for detection, classification and tracking of moving objects in outdoor [56, 39] and indoor [33, 54] environments. As the sensor nodes are battery-powered, there is no requirement for any additional infrastructure to power these nodes once they are deployed. This unobtrusive nature of the nodes enables autonomous sensing in conjunction with other nodes in their vicinity. Nodes communicate with each other using their low-power radios, which in turn facilitates node cooperation. Node cooperation enables each node to schedule its sensing cycles in a synchronized manner with respect to other nodes. This reduces the energy consumption of each node, resulting in an increase in the lifetime of the network. When the nodes are not sensing, they enter a low-power state and periodically awake to refresh the state (for eg: time synchronization, routing tables, etc.). Thus, the power of WSNs arises not from the capabilities of each node but of the network comprising of hundreds of nodes as a whole.

1.1 Problem Statement

The common denominator in all wireless sensor networks (WSNs), regardless of their underlying application, is the use of the radio to communicate information extracted from the sensed environment and, more importantly, to coordinate with other nodes. Consequently, radio communication and intelligent usage of the radio is a critical component of wireless distributed system in general and WSNs in particular. Due to the low power nature of WSNs, the radio used for communication is especially susceptible to changes in the quality of the wireless medium resulting in packet losses which can be attributed to limited transmission power levels as well as multipath effects resulting from lack of frequency diversity. Experiments [100] conducted with these low-power radio equipped sensor nodes have shown, using empirical measurements, that there exists a "gray area" within the communication range of sensor radios where the packet reception varies widely. Data collected using SCALE [13], led to the following conclusions: (i) no clear correlation between packet delivery and distance in an area of more than 50% of the communication range, (ii) temporal variations of packet delivery are correlated with mean reception rate of each link, and (iii) percentage of asymmetric links in a sensor network varies from 5% to 30%. These studies [100, 13, 14, 15, 91, 90] help confirm that low power wireless communication is hard to predict, is sensitive to changes in the environment and is known to significantly change over different time scales. Therefore, researchers have to extensively evaluate performance of their applications and protocols on physical testbeds; a time consuming process that otherwise could have been accelerated using good simulation models.

In systems research, a well-designed simulator provides users with the ability to test new ideas in an inexpensive manner. The simulator models the key elements of a given system, for example: hardware such as the CPU, network

interfaces, sensors, etc. and software such as the operating system. This allows the user to focus his attention on the design, testing and analysis of algorithms in a controlled and repeatable environment. Recent studies [75, 50] have indicated the presence of a wide chasm between the real world radio channel behavior and existing radio channel models in wireless simulators. This leads to significant differences in performance of a system in simulation as compared to a real world deployment. Improving wireless simulators by incorporating accurate and robust radio channel models will reduce the gap between simulation and real-world performance. Thus, a major goal of the proposed research is to create radio channel models that would help simulate more realistic packet losses for 802.15.4 wireless links.

1.2 Dissertation Goals

For wireless link modeling, I advocate a data-driven approach that requires collection of data traces of packet reception information over long periods of time at fine granularity. This data would be the seed for creating accurate, high-quality radio channel models that would help application designers increase the robustness of their applications by accounting in simulation for losses in the wireless medium. My approach is to model the long and short time scale behavior of links in wireless sensor networks using a multilevel approach. At a higher level, I intend to model the long-term evolution of a link and within each of these levels model the short term behavior capturing the correlations between successive packet receptions and losses.The objective is to have a parametrized model that captures the long and short time scale behavior from which one can sample wireless link traces. Such traces would capture the realistic variation observed in links from real-world environments.

However, one drawback of data-driven models with many parameters is the need for sufficiently large training sets to achieve reliable estimates. This means that for each link of a sensor network that one wants to model, the network developer must first record data for enough time (hours or days). This prevents quick setup of a new link and is costly in resources (e.g. testbed deployment, etc.), particularly for sensors in hard-to-reach locations (such as climate-sensing networks in Greenland). In these situations, it makes sense to use an existing model that has been trained with extensive data and adapt it to the new situation given a far smaller data trace than would be necessary to train a new model from scratch. Such model adaptation techniques can not only reduce the data requirement for suitably designed wireless link models but also help in the construction of accurate, high-quality wireless link models.

Furthermore, I will show that the same model adaptation techniques can be extended beyond link modeling to other application domains. In this dissertation, I consider the case of occupancy modeling in indoor environments using wireless sensor networks. Wireless sensor network infrastructure imposes limitations on the detection performance, latency and lifetime requirements of the system when used for counting occupancy. In this study, I propose using camera sensors for counting occupancy and using node coordination techniques to overcome the limitations. However, for creating high-quality occupancy models, it is required to collect data over long periods of time (weeks, months). Sensor networks using energy consuming sensors such as cameras cannot maintain such long lifetimes. To solve this problem, I propose using model adaptation techniques, similar to the one in the wireless link modeling case.

In summary, the specific goals of the research in this thesis are as follows:

1. To develop data-driven approaches for construction of high-quality wireless

link models.

2. To develop model adaptation techniques that can aid the estimation of parameters for wireless link models with little data.

3. Show that proposed model adaptation techniques are not specific to link models, but can generalize to other application domains, namely occupancy modeling.

Figure 1.1 shows the flow of the various parts of the dissertation.

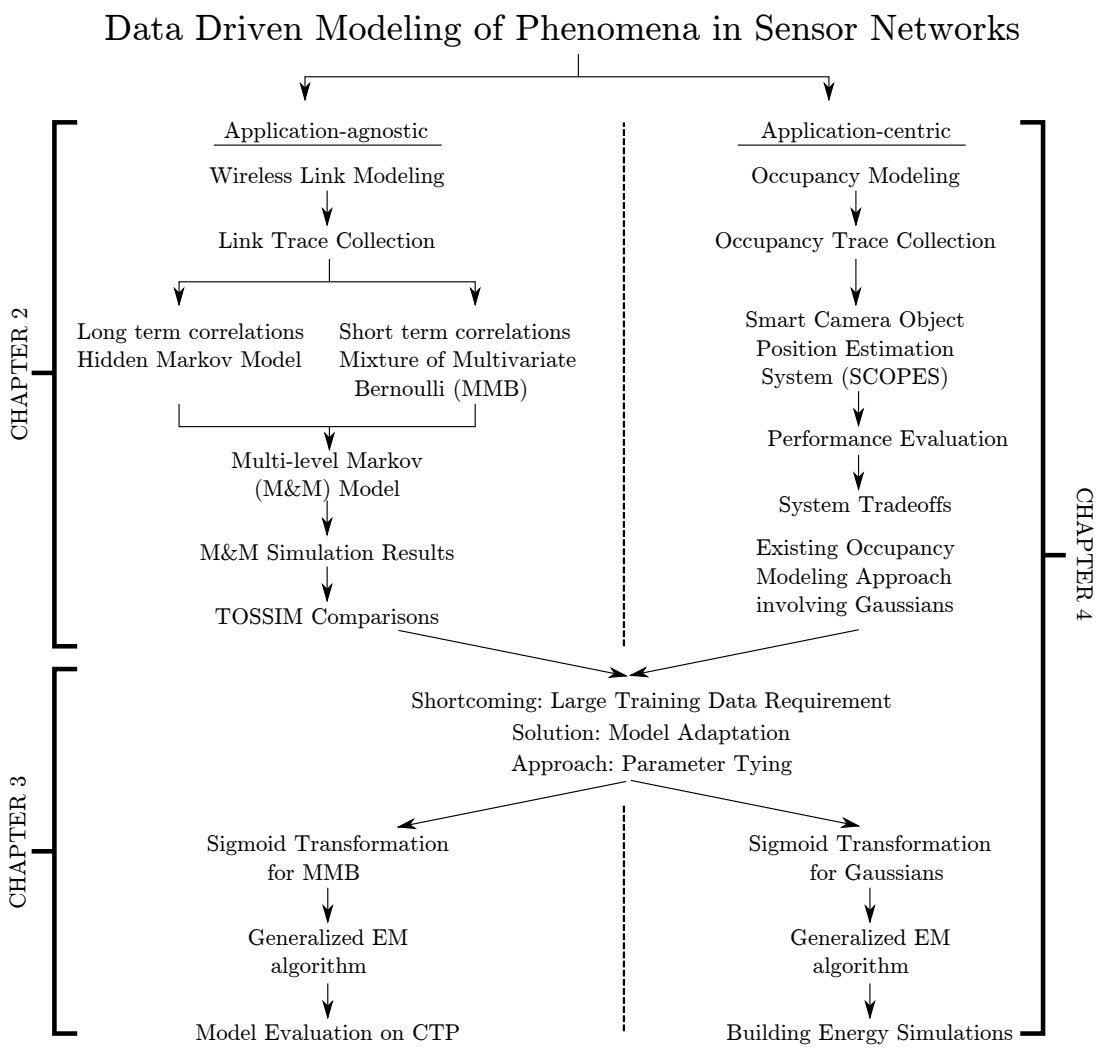


Figure 1.1: Dissertation Overview

CHAPTER 2

Building Data-Driven Communication Models

2.1 Background and Related Work

Models for characterizing the behavior of wireless links have been a widely studied area in networking literature [80]. These studies can be classified into radio propagation models and packet loss models.

2.1.1 Radio Propagation Models

In general, propagation models predict the average received signal strength and its variability at a given distance from the transmitter. They can be further subdivided into large-scale propagation models and small-scale propagation models. Large scale propagation models predict the mean signal strength for an given transmitter-receiver separation. Small-scale or fading models predict the variability in signal strength over small distances or very short time durations. Reflection, diffraction and scattering help to explain the propagation mechanisms in large and small scale models. Reflection occurs when the radio waves bounce off the surface of objects having large dimensions such as the surface of the earth, buildings and walls as compared to the wavelength of the signal. Diffraction occurs due to the signal changing direction because of sharp irregularities on the surface of objects between the transmitter and the receiver. Finally, scattering is

induced by the presence of small objects with irregular surfaces such as foliage, street signs and lamp-posts. Also, small scale models are affected by the fading effect which is caused by interference between multiple versions of the transmitted signal arriving at the receiver at slightly different times. This leads to constructive and destructive interference at the receiver resulting in the wide variation in amplitude and phase of the received signal. The waves are called multipath waves and the phenomenon is termed multipath.

Propagation models can also be separated on the basis of path distance into outdoor and indoor propagation models. Whereas outdoor propagation models are affected by factors such as terrain and climate, indoor propagation models are affected by objects in the environment, layout of buildings and construction material. Since most models assume prior knowledge of factors affecting signal propagation, a naive assumption, in practice, radio propagation models are created from a combination of analytical and empirical methods based on actual field measurements. The field measurements would incorporate the effects of known and unknown factors in the measured environment. The validity of empirical models is extended to other environments by conducting additional experiments for measuring signal variation.

2.1.1.1 Free Space Propagation Model

Given a clear line of sight (LOS), the Friis free space propagation model [80] is used to infer the relationship between the transmitted and received signal strength . The free space power at receiver antenna $P_r(d)$, which is separated from a transmitter by a distance d, is given by the Friis free space equation as follows:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L}$$

where P_t is the transmitted power, G_t is the transmitter antenna gain, G_r is the receiver antenna gain, L is system loss factor and λ is the wavelength (in meter).

The path loss, which represents signal attenuation as a positive quantity measured in dB, is defined as the difference (in dB) between the effective transmitted power, and the received power, is given as follows:

$$PL(dB) = 10\log\frac{P_t}{P_r} = -10\log[\frac{G_t G_r \lambda^2}{(4\pi)^2 d^2}]$$

The Friis free space model is only a valid predictor for P_r for values of d which are in the far-field of the transmitting antenna. The far-field, or Fraunhofer region of a transmitting antenna is defined as the region beyond the far field distance d_f, which is related to the largest linear dimension of the transmitter antenna aperture and the carrier wavelength. The Fraunhofer distance is given by

$$d_f = \frac{2D^2}{\lambda}$$

where D is the largest physical dimension of the antenna.

2.1.1.2 Ground Reflection (2-ray) Model

The 2-ray ground propagation model [80] is based on geometric optics. It considers the direct path and a ground reflected propagation path between the transmitter and receiver. The 2-ray model is used in mobile radio systems, where the transmitter and receiver are separated by several kilometers and the height of the transmission tower allows for a LOS path and a ground reflected path to the

receiver. The power received by a receiver antenna $P_r(d)$ which is separated from a transmitter by a distance d, is as follows:

$$P_r(d) = P_t G_t G_r \frac{{h_t}^2 {h_r}^2}{d^4}$$

where, h_t and h_r are transmitter and receiver heights, respectively. According to this model, the received power decreases with distance raised to the fourth power, which is worse than the free space model. The path loss for the 2-ray model with antenna gain is given by

$$PL(dB) = 10\log\frac{P_t}{P_r} = 40\log d - (10\log G_t + 10\log G_r + 20\log h_t + 20\log h_r)$$

2.1.1.3 Log Distance Path Loss Model

Theoretical and measurement based studies have shown that the average received signal strength decreases logarithmically with distance in indoor and outdoor environments. The log distance path loss model [80] expresses the path loss as a function of distance by using a path loss exponent as follows:

$$PL(dB) = PL(d_0) + 10n\log(\frac{d}{d_0})$$

where n is the path loss exponent which indicates the rate at which the path loss increases with distance and it depends on the environment, d_0 is the reference distance between the transmitter and the receiver and d is the actual transmitter-receiver separation. This expression for path loss was extended to indoor propagation models by incorporating the effect of building type. This model, called the attenuation factor model, is as follows:

$$PL(dB) = PL(d_0) + 10n_{SF}\log(\frac{d}{d_0}) + \text{FAF}$$

where n_{SF} represents the exponent value for the "same floor" measurement and FAF is the floor attenuation factor (in dB). Alternately, path loss is also given as

$$PL(d)[dB] = PL(d_0) + 10n_{MF}\log(\frac{d}{d_0})$$

where n_{MF} represents the path loss exponent value through "multiple floors".

2.1.1.4 Log-normal Shadowing

The phenomenon due to which the received signal strength varies due to presence of objects along the propagation path is called shadowing. The shadowing effect can be described using a log normal distribution, where the logarithmic value of the average signal strength at a distance follows a normal distribution [80]. The path loss in the presence of log normal shadowing is given as follows:

$$PL_d(dB) = \overline{PL_d} + X_\sigma = \overline{PL_{d0}} + 10n.log(\frac{d}{d_0}) + X_\sigma$$

where, X_σ is a zero-mean Gaussian distributed random variable with standard deviation σ (both on dB). The log normal distribution accounts for random shadowing effects due to different levels of clutter on the propagation path. In practice, the path loss exponent (n) and standard deviation σ are computed from field measurements.

2.1.2 Packet Loss Models

In contrast, packet loss models try to discover the underlying bursty packet loss distribution. Errors in packet reception can be attributed to causes such as interference in the channel and fading effects which lead to irrecoverable bit errors. Packet loss models can be broadly classified into two areas: (a) packet delivery function estimation approaches and (b) Markovian approaches.

2.1.2.1 Packet Delivery Function Estimation Approaches

Reis et al [82] noted the fact that in wireless networks, measurements of average behavior over even relatively short time periods tend to be stable, even for widely separated intervals. They exploit this to develop models for wireless delivery with interference using radio signal strength (RSSI) measurements. They model delivery probability as a function of interference, which the authors posit is the prime cause of variation in packet delivery. The receiver model is used with the RF profile to compute the probability a packet is correctly received in the presence of competing transmissions. In experimental evaluations, the approach has an root mean square (RMS) error of 0.5 and 0.3 for 802.11a and 802.11b networks, respectively, compared to a model that ignores interference. Similarly, Reddy and Riley [81] utilized RSSI and packet retry measurement values to create a radio propagation and packet error rate model, respectively. The radio propagation model is comprised of a CDF of the RSSI variation which enables simulation users to compute signal strength as a function of distance. The number of retries are used to compute the success of packet transmission given a retry count, datarate, packet size and distance.

Kashyap et al. [44] proposed a measurement-based approach to model the physical layer behavior, mainly, radio propagation, carrier sense and packet reception models. Radio propagation is modeled using a log normal shadowing model. A carrier sense model is created using a function fitted to measurements of received signal strength between a pair of nodes at different locations. Packet reception is modeled by curve-fitting of packet reception probability and signal to noise ratio data. In experiments conducted on a 12-node testbed, using the proposed approach, there was 10% difference between measured and predicted throughput capacity in contrast to 50% difference in more traditional simulation

models.

Lenders et al. [61] combine a physical receiver model with a medium access control (MAC) deferral and interference model. Here the physical receiver model models the effects of radio propagation, environmental noise and node mobility whereas the MAC model predicts packet delivery in the presence of carrier sense and interference from concurrent transmissions. The physical receiver model utilizes the windowed average at each time instance to compute the packet reception probability. The MAC model uses the packet reception probabilities from the physical receiver model to simulate carrier sense (channel deferral). This approach was evaluated using 5 mobile 802.11 nodes competing for the same channel under different environment and mobility conditions. Across various settings, the root mean square error (RMSE) of the estimated versus measured benchmark packet probability was below 12 percent. The error of the estimated throughput versus the effective observed throughput was below 10 percent, in contrast to up to > 50 percent for a naive model that ignores carrier sense and interference effects.

The common theme in these approaches is that a function relating physical layer characteristics such as RSSI or signal to interference noise ratio (SINR) to packet reception is computed from measurement data. Using this empirical pdf of packet reception, the packet delivery probability is computed. The problem with such approaches is that packet errors are assumed to be independent which has been proven incorrect in empirical studies [15, 91] showing temporal correlations between successful and failed packet receptions. Burstiness behavior in packet delivery where packet receptions and losses shows high correlation is not captured in these models. Also, all of the above models do not account for the fact that packet reception shows long periods of stability.

2.1.2.2 Markovian Approaches

The Gilbert model [30] is a probabilistic model for simulating burst noise in data transmission channels. In this model, a Hidden Markov model with two states is used to generate noise bursts, the first state has zero probability of encountering an error whereas the other state has a certain fixed non-zero probability for transmission errors. The transition probabilities control the amount of time spent in each state, thus, controlling the error patterns from a given set of parameters. Analysis of traces [72] for the AT&T Wavelan system concluded that loss behavior could not be accounted by the 2-state Markov model. They proposed a methodology to model the error-free and error traces using exponential and Pareto distributions to model segments of the trace. Yajnik et al. [95] modeled Internet packet loss traces using a Bernoulli model, 2-state Markov chain model and k^{th} order Markov chains. They checked the model accuracy with regards to loss estimation over 38 stationary trace segments. They concluded that all these models are inadequate as they could not accurately model losses in their dataset.

Markov-Based Trace Analysis (MTA) [47] and Multiple MTA [48] approaches propose modeling channel errors by decomposing a trace with non-stationary properties into a set of piecewise stationary traces consisting of "lossy" and "error free" states. Lossy states exhibit stationarity, where a sequence of lossy states can be modeled by a traditional discrete time Markov chain (DTMC). In [88], HMMs were proposed for modeling packet reception traces and choosing a model based on the likelihood criterion. Markov-based stochastic chains were proposed [46] to model 802.11b channel behavior for bit errors and packet losses. The study compared the performance of high order Markov chains, 2-state Hidden Markov Models and hierarchical Markov Models and concluded that Markov chains of order 9 (i.e., 2^9 states) are required for accurate models for the bit error process.

These studies helped reinforce the notion that for any modeling approach to simulate behavior of wireless links, the model needs to account for the long and short term variations in the link quality. Also, the model should be easy to train and show close correlation between the input and the simulated data traces.

2.1.3 TOSSIM

TOSSIM [62] is a discrete event simulator for sensor networks running on the TinyOS operating system. It allows users to write TinyOS code in a simulation environment that is scalable and bridges the gap between algorithm testing and application development. TOSSIM simulates behavior of the CPUs, radios and sensors in different sensor nodes, networking stacks and other OS primitives.

TOSSIM supports several radio models, namely the Simple Propagation model, the Link Layer model [102] and the Closest-fit Pattern Matching (CPM) model [57]. In the Simple Propagation model, every node can receive packets transmitted by any other node. The Link Layer model specifies the behavior of the wireless link depending on the radio and the channel characteristics for static and low-dynamic environments. CPM is based on a statistical model created from noise reading traces collected from the deployment environment. It computes the probability distribution of n_t given the noise readings $(n_{t-k}, n_{t-k+1}, ...n_{t-1})$, where k is the duration of noise history considered by the model. A $k = 0$ would make each noise value independent, while k equal to the length of the trace would provide an exact replay of the noise trace. In a recent paper [87], two approaches (Expected Value PMF and Average Signal Power Value) were proposed to estimate the signal power of missing packets in a packet reception trace, and, using this data the CPM algorithm models the variations in packet signal strength. These existing models require the modeling of two separate physical layer measurements,

namely, RSSI and noise/interference values to create a representative model of a real environment.

In contrast, I propose the modeling of correlations between successive packet receptions and failures from a given packet reception trace as the packet reception traces are a direct indicator of the link quality.

2.2 Wireless Link Modeling

2.2.1 Collection of Packet Reception Traces

In order to create an accurate packet loss model, a comprehensive database of packet reception traces of links having different reception rates is required. For this task, I collected data from a 75 node MoteIV Tmote Sky testbed deployed along the ceiling of the Science and Engineering Building (SE testbed) at the University of California - Merced. Each mote is comprised of an ultra low power Texas Instruments MSP430 F1611 micro-controller featuring 10KB of RAM, 48KB of flash and a 802.15.4 compliant Chipcon CC2420 radio (channel 26) for wireless communication. The node locations are fixed for the duration of our experiments (refer Figure 2.1 for details). All the motes in a group are connected to a Linksys NSLU2 network storage device via an USB hub. The Linksys NSLU2 device is used to bridge serial communication between the motes and a central server over the local network.

A number of experiments were performed to collect packet reception traces from a diverse set of links (see Table 3.2). In each experiment, there was one fixed sender and multiple receivers. The sender sends 64 packets per second with an inter-packet interval of 16ms on channel 26 for durations of 1, 6 and 12 hours. The receivers record the sequence number, received signal strength (RSSI) and

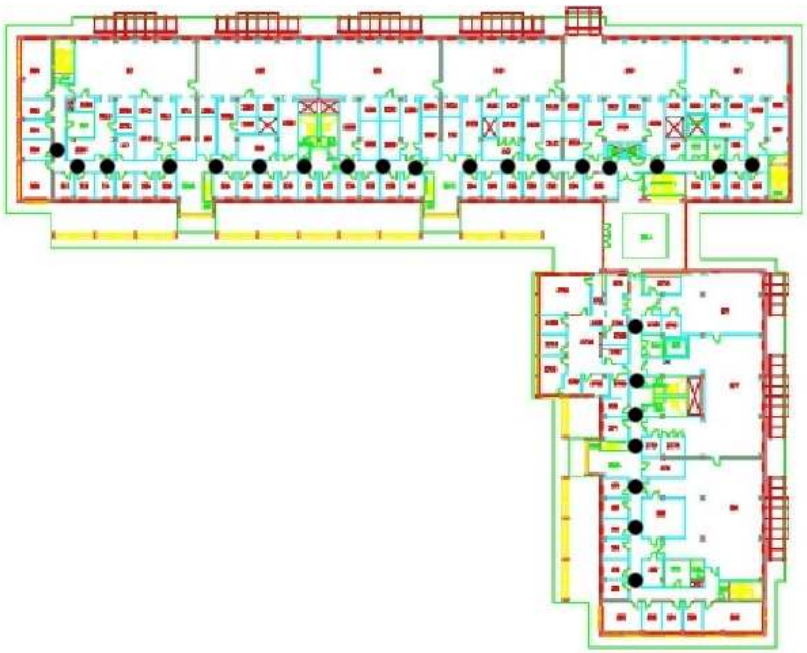


Figure 2.1: SE testbed: 25 groups of three nodes each separated by a distance of 40cm were deployed in the second floor of the Science and Engineering building at the University of California - Merced. Each black dot represents one group of nodes. Nodes are placed at fixed locations along the corridor ceiling of the building.

link quality indicator (LQI) values of each received packet. The same data was collected from the Motelab testbed [97] but the duration of each experiment was limited to 30 minutes due to storage concerns regarding the large amount of data generated in every run. After each experiment, records or traces of packet reception for each of the receiver nodes were created. In addition, noise data (channel 26) for all nodes using the *RssiSample* program on both the testbeds was gathered. The length of the noise traces is equivalent to the *meyer-heavy* trace collected in [57]. The noise traces are meant to be utilized for a faithful comparison between the TOSSIM simulation model and our proposed approach.

Testbed	Program	Num. Expts.	Duration	Num. Packets/Expt.	Tx power
SE	RssiDemo	9	1 hour	230400	1-31
SE	RssiDemo	1	6 hours	1382400	7
SE	RssiDemo	3	12 hours	2764800	8,9,11
SE	RssiSample	3	30 minutes	196608	-
MoteLab	RssiDemo	18	30 minutes	115200	31
MoteLab	RssiSample	3	30 minutes	196608	-

Table 2.1: Summary of experiments conducted on the MoteLab and SE testbeds (Note: 802.15.4 channel 26 is used in all experiments. Tx. power=Transmit power level).

2.2.2 Exploratory Data Analysis

In this section, issues that need to be addressed when modeling 802.15.4 wireless links are highlighted. Links having packet reception rate (PRR) $< 10\%$ are termed as bad or poor links, links having PRR between 10% and 90% as intermediate links and links having PRR greater than 90% as good links. Links having PRR $= 0\%$ are termed as inactive links.

Exp. #	CC2420 Tx. Lvl.	Good	Bad	Interm	Inactive
24	11	20(48%)	8(19%)	7(17%)	7(16%)
26	10	19(45%)	7(17%)	3(7%)	13(31%)
28	8	19(45%)	10(24%)	3(7%)	10(23%)

Table 2.2: Summary of variation of link quality in a network as a function of sender radio transmission power. CC2420 Tx. Lvl. = CC2420 transmit power level

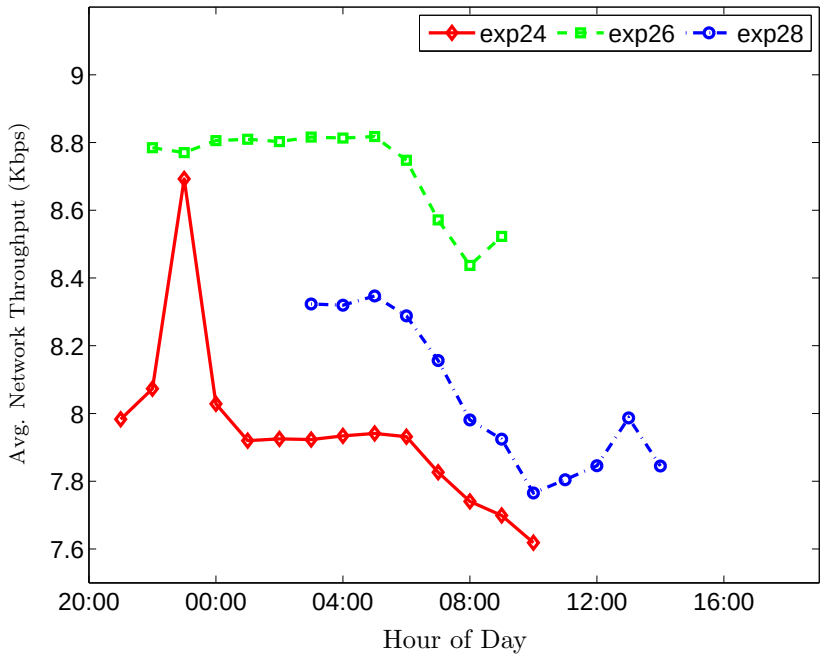
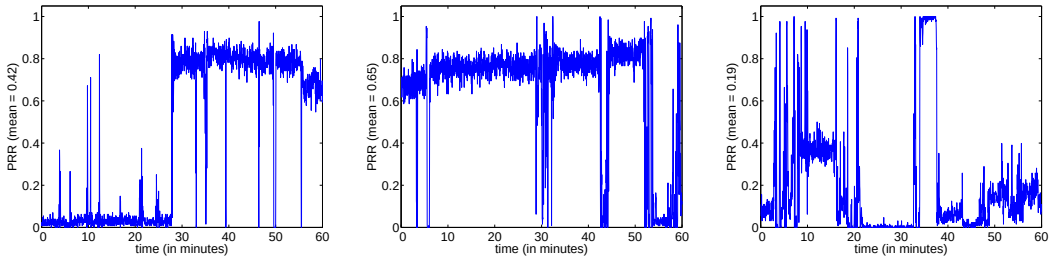


Figure 2.2: Variation of the average data throughput per hour for all good and intermediate links in the network.

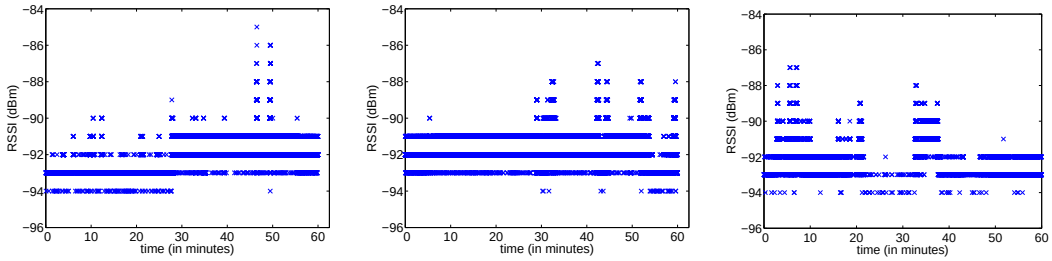
Prior studies have shown that 802.15.4 links can vary significantly over time [13, 15, 64]. In Figure 2.2, the average network throughput per hour averaged over all the links having $PRR > 10\%$ in the network is shown as a function of time of day for the 12 hour experiments. The figure clearly shows that the average network throughput is not constant, but fluctuates with time. This is a clear indication of variation of PRR across nodes in the network. The radio transmission power levels in experiments 24, 26 and 28, correspond to values 11, 10 and 8 in the CC2420 registers. This would lead one to think that, the throughput should be highest for experiment 24, followed by level 26 and 28, respectively. However, from the data, the throughput for experiment 24 is less than that of the others. This can be explained by the higher total number of intermediate links (see Table 2.2) compared to the other experiments. An interesting artifact of the environment

can be seen in Figure 2.2, which shows a fairly consistent decrease in throughput from midnight to midday in all three experiments. From our experimental data, the good and bad links are relatively stable over time whereas intermediate links show significant variation in link quality over time. This is consistent with previous findings reported in [13, 15]. In general, in simulation, it is easy to model good links as they do not show significant variation with time [67, 87]. On the other hand, there is a significant difference between the models of intermediate links in simulation and the real-world. If the accuracy of simulation models of these intermediate links were improved, then it is possible that WSN application simulations could show the potential benefits of using these intermediate links when their quality is high enough for transporting data instead of permanently ignoring or blacklisting them. In addition, it would help application designers to test performance of algorithms for the common case, and the corner cases that are one of the causes of protocol failure.

To emphasize this point, the variation in PRR and RSSI of a representative intermediate link was plotted. For this link (see Figure 2.3), the PRR is plotted as a function of time, where each PRR value is calculated for a two second interval (i.e., for 128 consecutive packets at a time). Figure 2.3 also shows the corresponding variation in RSSI values of the received packets. From Figure 2.3, the average PRR of link 1 is 42%, 65% and 19% for hours 1, 2 and 3, respectively and the corresponding average RSSI values are -91.85 dBm, -91.8 dBm and -91.67 dBm, respectively. In each hour, the PRR and RSSI values fluctuates widely, cycling between good, intermediate and bad states. In each state, the link is relatively stable for a given period of time before a significant change in link quality. A closer look at the sequence of received packets within few tens of seconds reveals that packet receptions and losses are not independent i.e., intermediate links show significant bursty behavior. This shows that links



(a) Link1-Hour1, PRR=42% (b) Link1-Hour2, PRR=65% (c) Link1-Hour3, PRR=19%



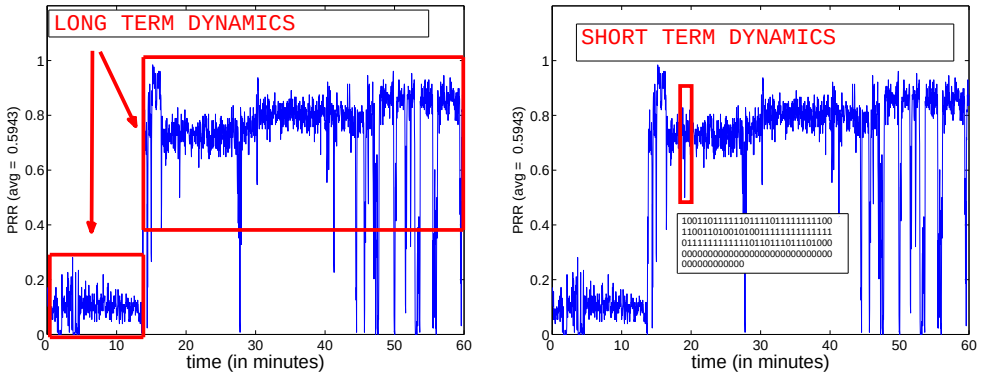
(d) Link1-Hour1, PRR=42% (e) Link1-Hour2, PRR=65% (f) Link1-Hour3, PRR=19%

Figure 2.3: Variation in PRR and RSSI by hour for a typical intermediate quality link. Figure shows that PRR and RSSI values are stable for short periods of time.

of intermediate quality manifest highly dynamic behavior over time at different time scales, thus highlighting the non-trivial nature of the modeling problem for such links.

2.2.3 Our Modeling Approach

Consider the observed data as binary sequences where 1 indicates successful packet reception and 0 indicates lost or corrupted packets. (We will also consider a sequence of continuous values, namely the reception rates in $[0, 1]$ indicating the average over a binary window.) The fundamental motivation for our modeling approach is that observed traces display structure at different temporal scales. In Figure 2.4, for example, one can see that over a period of minutes the link seems to switch between two states: one with PRR ≈ 0.1 and the other with PRR



(a) (b)

Figure 2.4: Illustration of long and short term dynamics in an empirical trace (avg. PRR=59.43%). (a) Long term dynamics are periods of nearly constant PRR (10% and 70%, respectively) which persist for periods in the order of minutes. (b) Short term dynamics is the burstiness observed in packet delivery over a period of seconds indicating that packet receptions and losses are highly correlated.

≈ 0.7. This is called the *long-term dynamics*. In a period of seconds, however, while the PRR may stay roughly constant, it is more likely to observe a bursty sequence `0000111111` than a wildly oscillating sequence `1010101101`. This is termed the *short-term dynamics*. In order to simulate realistically the behavior of links, a model that is flexible enough to replicate this multiscale structure is required. Also, we want to estimate its parameters (which determine its typical PRRs or its local burstiness) from observed traces. The details of our model, the **Multi-level Markov (M&M) model**, are described next; appendices A.1–A.2 give an overview of hidden Markov models and mixtures of multivariate Bernoulli distributions.

2.3 The Multi-level Markov (M&M) Model

We model a possibly infinite binary sequence (the data trace) as a *sequence of binary strings* (windows) $\mathbf{x}_t$ of length W, as shown in Figure 2.5. A *level-1 hidden Markov model (L1–HMM)* with Q_1 different states $q = 1, \ldots, Q_1$ models transitions between long-term states, and has Q_1^2 tunable parameters a_{ij} (the transition probabilities $p(q = j|q = i)$). Each long-term state q has its own distribution $p(\mathbf{x}|q)$ of emitting binary W-windows, which captures the short-term behavior of the link in that state—that is, the dynamics of the variations in consecutive packet reception successes or failures that has its own parameters (described below). Thus, W controls the tradeoff of short vs. long term. We have studied two types of short-term, or level-2, models $p(\mathbf{x}|q)$:

- A *hidden Markov model (L2–HMM)*. This has (1) a set of Q_2 short-term states (different from those of the L1–HMM) and its own Q_2^2 transition probabilities, and (2) a (univariate) Bernoulli emission distribution with parameter p. Thus, this is a sequential model: to emit a W-window we sample W times from the L2–HMM. Note that the L2–HMM for long-term state i has its own parameters, different from those of the L2–HMM for a different long-term state j.

- A *mixture of multivariate Bernoulli distributions (L2–MMB)*. This mixture has M components, and each component has $W + 1$ parameters: a mixture proportion and a vector $\mathbf{p} = (p_1, \ldots, p_W)$ of Bernoulli parameters. Thus, this is not a sequential model: to emit a W-window we pick a component at random (according to their proportions) and then we sample from its W-dimensional Bernoulli the W-window at once.

We report experimental results with both models below. In both models, using a sufficiently large number of short-term states Q_2 or mixture components M allows us to model arbitrarily complex distributions of W-dimensional binary windows; for example, more or less bursty sequences. Importantly, note that if we modeled $p(\mathbf{x}|q)$ as a single Bernoulli (i.e., $M = 1$), then bits within the window would be independent from each other, leading to unrealistically oscillating behavior. The average PRR of a long-term state i is the mean of its emission distribution $p(\mathbf{x}|q = i)$.

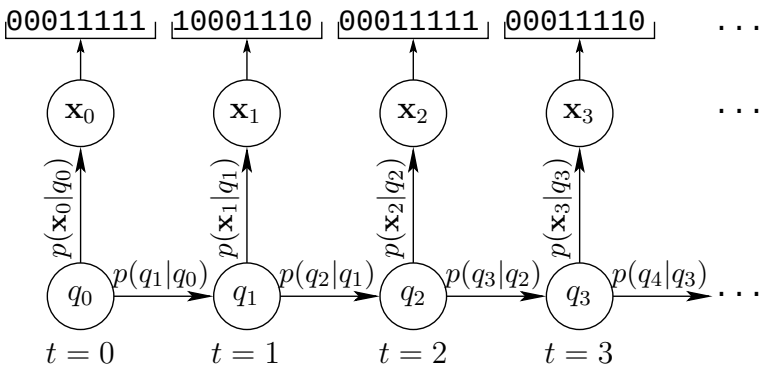


Figure 2.5: Graphical model of a HMM which emits binary strings $\mathbf{x}_t$ of length 8. In the M&M model, $p(\mathbf{q}_t|q_{t-1})$ is modeled by the transition matrix of the HMM, and $p(\mathbf{x}_t|q_t)$ is modeled using a MMB emission distribution for the HMM.

Next, we explain how to simulate a binary trace from our model (sampling), and how to estimate good model parameters from measured data (learning).

2.3.1 Sampling

In order to generate a trace of length L bits from the model, we sample as follows:

1. Generate a long-term state sequence of length L/W using the transition probabilities of the L1–HMM.

2. For each long-term state q of this sequence, we sample a W-window $\mathbf{x}$ from its $p(\mathbf{x}|q)$ (i.e., the corresponding L2–HMM or L2–MMB).

The trace is the concatenation of the L/W windows.

2.3.2 Learning

In a machine learning approach, estimating the parameters of a probabilistic model is usually done by maximizing the log-likelihood of a given data set (one or more observed traces) over all the model parameters (transition probabilities for all short- and long-term states, mixing proportions, Bernoulli parameters). For our multilevel HMM this can be quite complex and time-consuming, so we follow a simpler learning algorithm that is slightly suboptimal but faster and relatively robust to local optima, by first estimating the L1–HMM transition probabilities, and then estimating the L2–HMM transition probabilities or the L2–MMB parameters. The training process is as follows:

1. Training the L1–HMM transition probabilities. The binary input trace is transformed into a sequence of PRRs (in $[0, 1]$) computed over a window size W. We define a continuous HMM with Q_1 states and beta emission distributions and use the EM algorithm to estimate by maximum likelihood its beta parameters (which we then discard) and its transition probabilities, given the sequence of PRRs. We obtain good initial values for the beta parameters by running k–means on the PRR sequence.

2. Clustering the W-windows. After learning the parameters of the L1–HMM, we used the Viterbi algorithm [27] to obtain the most likely state sequence for each input trace, and grouped into the same cluster all windows assigned

to the same state. Practically speaking, this tends to group windows associated with similar PRR values.

3. For each long-term state, we trained its L2–HMM or L2–MMB model only on its corresponding cluster:

   - L2–HMM: we used again the EM algorithm for HMMs (now having univariate Bernoulli emission distributions), resulting in the L2–HMM transition probabilities and the Bernoulli parameters for each of the Q_2 short-term states.
   - L2–MMB: we used an EM algorithm for MMBs as described in [12], resulting in the proportion and Bernoulli W-dimensional vector for each of the M mixture components.

2.3.3 Multi-level Model Rationale

Instead of a multi-level approach like the M&M model, it is possible to model packet traces using just a L1–model having continuous emission distributions, which represent PRR computed over a window W, to capture the long term dynamics and the short term dynamics using the PRR value from the L1 emission distribution as the Bernoulli parameter for each of the W values in the window. This model is equivalent to the M&M model, wherein (i) for the L2–HMM, Bernoulli emission probability values for both states is c, and (ii) for the L2–MMB, $M = 1$ and the vector of Bernoulli parameters of length W is $\mathbf{p} = c \times (1, \ldots, 1)$ where c is the PRR outputted by the emission distribution of the L1–model. The model would only capture the long-term PRR dynamics. However, in a pure L1–model, we can see that the short-term correlations are not captured correctly because each value in the W-length window is now indepen-

dent and hence, uncorrelated to its neighbors. When training, the duration of our longest data traces was 12 hours. The level of long-term dynamics for traces up to 12 hours can be accounted for using the M&M model. For longer duration traces, of the order of days, months, year, the level of long-term dynamics might be greater than that of the 12 hour traces. For modeling such traces, a 2-level model may not suffice. However, the current modeling approach can be extended to longer time-scales using a N-level hierarchy.

2.3.4 HMM-MMB: M&M Reformulation

In Section 2.3.2, we solved the issue of parameter estimation for the M&M model in a faster but sub-optimal way by separately computing the transition matrix for the L1–HMM and the L2–HMM/L2–MMB output distribution for each state of the L1–HMM. Here, we reformulate the M&M model as a HMM with an MMB output distribution for each state, where the parameters are estimated jointly. Henceforth, we will refer to this as the HMM–MMB M&M model. The parameters of the HMM–MMB are the transition probability matrix $p(q_t = j|q_{t-1} = i)$, the MMB emission distribution $p(\mathbf{x}|q = i)$ for each state and the initial state probability $p(q_0 = i)$. We jointly estimates these parameters using the Expectation-Maximization (EM) algorithm we derived for HMM–MMBs. We optimize the following $\mathbf{Q}$ function given the HMM–MMB model with its $Q^2 + QM(W+1) + Q$ free parameters (denoted λ)[1]:

$$\mathbf{Q}(\lambda, \lambda') = \sum_{q \in Q} \sum_{m \in M} \log(P(\mathbf{X}, q|\lambda)) P(\mathbf{X}, q|\lambda') \tag{2.1}$$

where λ' are our estimates of the parameters in the previous iteration and where Q is the space of all state sequences of length T. $P(\mathbf{X}, q|\lambda)$ is the complete-data

[1]Notations followed are similar to procedure described in [8].

likelihood function given by:

$$P(\mathbf{X}, q|\lambda) = p(q_0)\prod_{t=1}^{T} p(q_t|q_{t-1})p(\mathbf{x}_t|q_t) \tag{2.2}$$

and, $p(\mathbf{x}_t|q_t)$ is the MMB emission distribution for state q_t:

$$p(\mathbf{x}_t|q_t) = \sum_{m=1}^{M} \pi_{q_t m} \prod_{w=1}^{W} p_{q_t m w}^{x_{tw}} (1 - p_{q_t m w})^{1-x_{tw}}$$

The Q function then becomes:

$$\begin{aligned}\mathbf{Q}(\lambda, \lambda') = &\sum_{q\in Q}\sum_{m\in M} \log(p(q_0))P(\mathbf{X}, q|\lambda') \\ &+ \sum_{q\in Q}\sum_{m\in M} \log(p(q_t|q_{t-1}))P(\mathbf{X}, q|\lambda') \\ &+ \sum_{q\in Q}\sum_{m\in M} \log(p(\mathbf{x}_t|q_t))P(\mathbf{X}, q|\lambda')\end{aligned} \tag{2.3}$$

Solving the $\mathbf{Q}$ function by setting the derivatives w.r.t. each parameter to zero, we get the following expressions:

$$p(q_0 = q) = \frac{P(\mathbf{X}, q_0 = q|\lambda')}{P(\mathbf{X}|\lambda')} \tag{2.4}$$

$$p(q_t = j|q_{t-1} = i) = \frac{\sum_{t=1}^{T} P(\mathbf{X}, q_{t-1} = i, q_t = j|\lambda')}{\sum_{t=1}^{T} P(\mathbf{X}, q_{t-1} = i|\lambda'}) \tag{2.5}$$

$$\pi_{il} = \frac{\sum_{t=1}^{T} P(q_t = i, m_{q_t t} = l|\mathbf{X}, \lambda')}{\sum_{t=1}^{T}\sum_{l=1}^{M} P(q_t = i, m_{q_t t} = l|\mathbf{X}, \lambda')} \tag{2.6}$$

$$\mathbf{p}_{il} = \frac{\sum_{t=1}^{T} \mathbf{x}_t P(q_t = i, m_{q_t t} = l|\mathbf{X}, \lambda')}{\sum_{t=1}^{T} P(q_t = i, m_{q_t t} = l|\mathbf{X}, \lambda')} \tag{2.7}$$

In each step of the EM algorithm, we first compute the complete data likelihood using the parameter estimates in the current step. Using this value, we

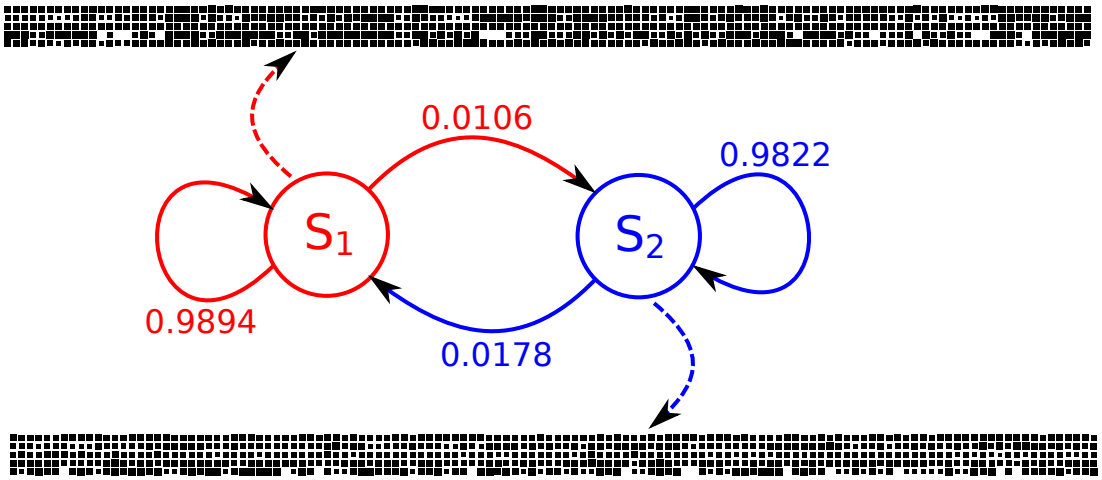


Figure 2.6: Graphical representation of the M&M model. The transition probabilities are overlaid on the arrows showing the state transitions. The MMB emission distribution for each state is represented by a matrix of shaded squares, wherein the degree of shading indicates if the output is 1 (black square) or 0 (white square).

compute the updated parameter estimates by maximizing the **Q** function. This continues till we reach a local optima. In the end, the most likely sequence of state values corresponding to an observed sequence can be obtained using the Viterbi algorithm [27]. A good way to initialize the transition matrix and MMB parameters for the HMM–MMB is to use the final L1–HMM and L2–MMB parameters using the procedure described in Section 2.3.2. From this initialization, the EM algorithm converges to one of several local optima for the problem.

2.3.5 Graphical Representation of the M&M model

Figure 2.6 shows a graphical representation of the M&M model for Q=2, M=5 and W=128. The transition probability matrix is shown using a transition diagram for the HMM. The MMB emission distribution of each state is represented using

a format similar to Hinton diagrams [36]. Each state output is represented by a $M \times W$ matrix of shaded squares, wherein the shaded area inside each square is proportional to the Bernoulli parameters for the MMB distribution. If $p = 1$, then the square appears black and if $p = 0$ then the square appears white. The diagram is meant to provide a visual representation for the 1/0 output of the M&M model. Longer runs of 1s would be outputted by a model if we see sequences of consecutive black square and longer runs of 0s would be outputted for sequences of consecutive white squares.

2.4 Evaluation of the M&M model

To evaluate the performance of our approach, we trained models for links with different reception rates from the experimental data traces (training set, length = 230400). As the problem is unsupervised (there is no ground truth to compare with) and the generated sequences can have any length, we do not compare the likelihood value that the models produce for a trace. Instead, we compare on the basis of statistics computed on the traces versus a different set of unseen data traces (testing set) having similar PRR characteristics. For each link, we proceeded as follows: (1) We learned the model parameters given the (training set) data traces and different combinations of model sizes i.e., Q ($\in \{2, 4, 6\}$), M ($\in \{2, 4, 5, 10, 15\}$) and W ($\in \{8, 16, 24, 32, 48, 64, 96, 128, 160, 192\}$). (2) For each model, we sampled a sequence as long as computationally possible (to reduce the variability in our statistics). (3) From this sequence, we computed the following statistics and compared them with the same statistics computed for the testing set (different from the training set):

1. PRR, to assess the long-term behavior of a link.

2. Distributions of run lengths of 1's, $r_1(n)$, and 0's, $r_0(n)$, for $n = 1, 2, \ldots$ This assesses both the global and local behavior. The run length (RL) distribution estimate is defined on a range independent of the data, namely $[1, \infty)$. Different RL distributions can be compared (e.g. with the Lp distance) and have statistics defined on them (e.g. variance). Each new bit changes the RL distribution in a localized way: it adds 1 to the appropriate run length. The information about long bursts is seen by looking at the tail of the RL distribution, and can be enhanced by having each run of length L count as L, instead of 1. We term this *weighted run length (WRL)* distribution. It is similar to the RL distribution except that it enhances the longer runs. In figures 2.9 and 2.12, we plot the WRL distributions to emphasize the occurrence of long runs of 1's and 0's.

3. The conditional packet delivery function (CPDF) [57] $C(n)$, defined as the conditional probability of observing a 1 after n consecutive 1's or 0's. This assesses the global and local behavior. The CPDF estimate is defined only on a range $[0, R]$ where R is the length of the longest run, which depends on the data sequence. It is not defined beyond R because no such run is observed. In fact, even in that range, $C(n)$ is highly sensitive to the sequence, particularly for the larger n. CPDFs are sensitive to the appearance of a single burst which adds an area of probability approximately equal to 1 around $n = L/2$ where L is the burst length. This happens no matter how long the trace is and no matter how often such bursts occur, as long as they occur at least once. Each new bit (1/0) in the sequence changes a possibly large part of the CPDF (up to the whole of it). Thus, CPDFs are good for detecting a burst of 1/0's but not suitable for determining the frequency of 1/0's. It is difficult to compare CPDFs from different datasets as the

length of the largest burst will vary from sequence to sequence. While one can eliminate all d values having less than a minimum number of runs, this loses information by truncating the tail.

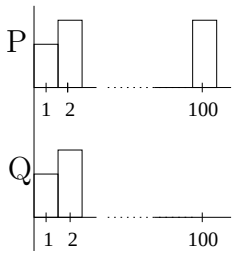
4. Allan deviation [4, 2] (AD) is computed as the square-root of one-half of the average differences between squares of successive samples over a given sampling period. The formula for AD of a sequence of samples x_i is:

$$AD = \sqrt{\frac{1}{2(n-1)} \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2}$$

The AD plot summarizes the difference between successive samples of a quantity at different time scales. In our case, the samples are PRRs computed over different window sizes. According to [2], the AD will be high for window sizes near the "characteristic burst length". In our study, we utilized the AD plots to determine the combination of Q, M and W that shows the most similarity between the simulated traces and the testing traces.

2.4.1 Comparing RL and CPDF Distributions

To compare differences in the distributions of the run lengths and CPDFs of the testing and simulated traces, we can compute the average L_1-norm between them. However, when computing the average L_1-norm, the difference in the two distributions is weighted equally for the common cases i.e., short runs/bursts of 1/0s and for the rare cases i.e., very long runs of 1/0s. The absence of rare cases in the simulated traces does not significantly affect the L_1-norm between the two distributions, thereby potentially misrepresenting the performance of a modeling approach. The inability of a modeling approach to account for the rare cases is a serious shortcoming for simulation users as they will not be able to adjust the behavior of algorithms/protocols for such cases which will eventually result



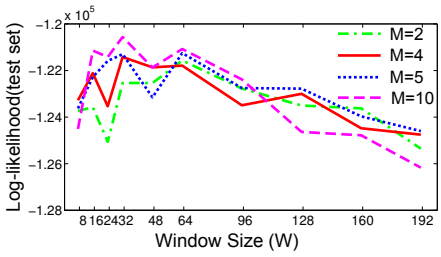
$$D(P,Q) = |P(1) - Q(1)| \ + \ |P(2) - Q(2)| \ + \ |P(100) - Q(2)| \ + \ |100 - 2|/1000$$

$$D(Q,P) = |Q(1) - P(1)| \ + \ |Q(2) - P(2)|$$

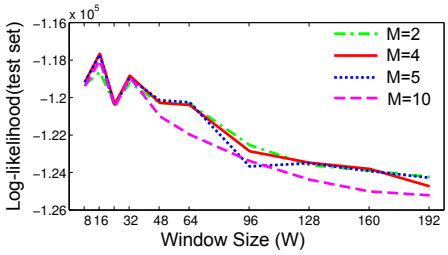
Figure 2.7: Computing the distance between two distributions P and Q. In this illustration, P is defined at 1, 2 and 100, and Q is defined at 1 and 2 only.

in failure under real world conditions. On the other hand, the L_1-norm would exaggerate the difference between traces from the same model when the length of the long runs/bursts varies slightly. To highlight the effect of the absence of rare cases and that of minor differences between rare cases from the same model, we designed a new metric called the Nearest Neighbor Distance. Although this is not the only way to emphasize the importance of rare events [55, 85], it worked well in our case.

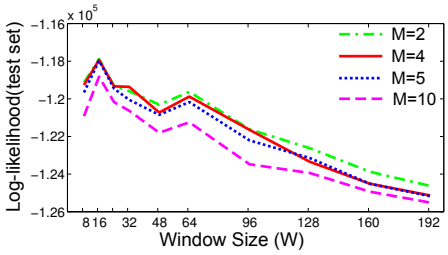
Nearest Neighbor Distance (NND): Let P and Q be two functions, each defined on a (possibly different) subset of the natural numbers. In our case, P and Q are the RL or CPDF distributions from the empirical and simulated traces, and we consider the RL distribution to be defined only where its value is positive. We define a non-symmetric distance $D(P,Q)$ as the sum over all the existing entries i of P of the following: $|P(i) - Q(i)|$ if $Q(i)$ is defined, and $|P(i) - Q(j)| \ + \ \alpha|j - i|$ if $Q(i)$ is not defined, where j is the closest entry to i for which $Q(j)$ is defined. That is, $D(P,Q)$ behaves like the L_1 distance where both P and Q are defined, and like a penalized L_1 distance to the closest entry where Q is defined, otherwise. We chose $\alpha = 1/1000$ empirically. We tried several values that could highlight differences in traces with regards to missing runs of 1's and 0's. With $\alpha = 1/1000$, we observed that we did not severely penalize



(a) $Q = 2$



(b) $Q = 4$



(c) $Q = 6$

Figure 2.8: Variation in Log-likelihood of the testing set as a function of Q, M and W using the M&M model for an empirical trace.

cases where the difference in run lengths was not significant < 50. On the other hand, if there was an absence of long runs of 1's or 0's, then our choice of α was sufficient to capture it. Figure 2.7 shows a sample calculation of $D(P, Q)$ and $D(Q, P)$. NND is then computed as $(D(P, Q) + D(Q, P))/2$, which is now symmetric.

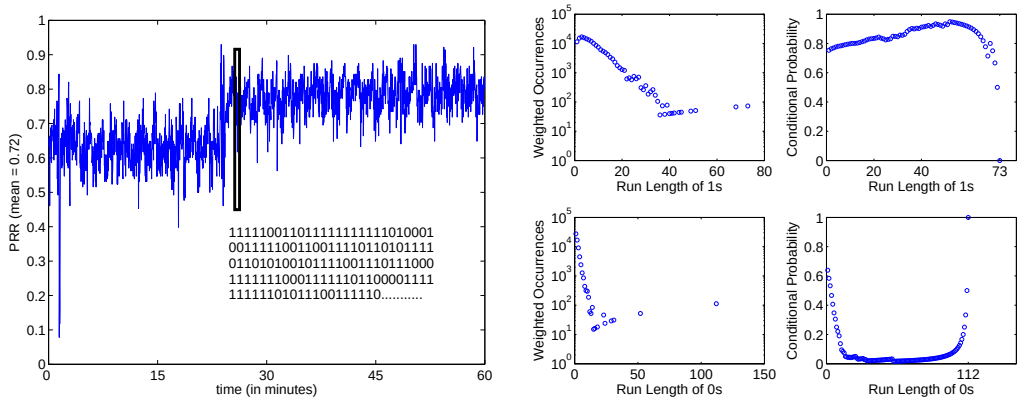
2.4.2 Model Selection - Tradeoff between Q, W and M

Before settling on a model size, we would like to analyze the tradeoffs associated with it. Using many parameters (higher Q, W and M) yields a powerful model but is more prone to over-fitting and local optima. In addition, such models are computationally costly. On the other hand, using few parameters may not yield a powerful enough model. Generally, a model with fewer parameters is preferred because (i) a model should make as few assumptions as possible (Occam's razor) and (ii) the data required for training the model to a given degree of accuracy grows exponentially with the number of variables. From the simulation users point of view, the model should be able to generalize to traces having similar long and short term dynamics, i.e., the simulated traces should have small NND values and the AD plots of the simulated and testing traces should be qualitatively similar. Also, the simulated traces should converge to the model PRR quickly (i.e. shorter trace length). Let us suppose that for training a model, we are using a trace of length L with a certain Q, W and M. Here, the ratio $\frac{L}{QMW}$ determines approximately the number of observed sequences that can be attributed to each of the MMB components in each state of the HMM. Due to the high dimensionality of the problem, it is necessary for each MMB component to have adequate number of training sequences. Therefore, as a rule of thumb, $\frac{L}{QMW} \geq 100$ would ensure that condition to be satisfied, i.e. each MMB component in each state of the HMM will have ≥ 100 training sequences (of window size W).

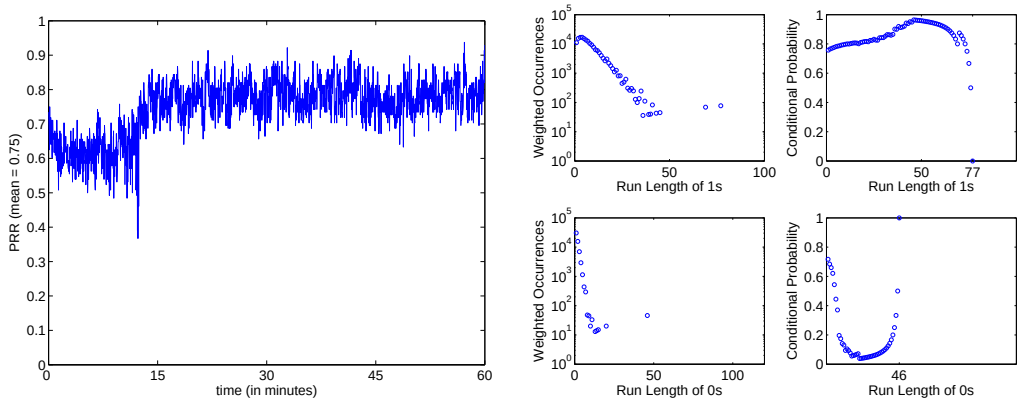
More parameters leads to higher log-likelihood on the training set but makes the model more prone to over-fitting. Therefore, we select a model based on its performance on some unseen data which was not used for training the model. When the EM algorithm for HMM with MMB emission distribution converges to a solution, the model having the highest log-likelihood on the testing trace is

selected. In general, the log-likelihood of a model computed for the test datasets is expected to exhibit a maximum value for a certain combination of Q, M and W, and decrease monotonically on either side of the maximum for all other combinations. Figure 2.8 shows the variation in the log-likelihood on the testing trace as a function of Q, W and M. The figure indicates that model selection on just the basis of log-likelihood will lead to models with small W $(= 8, 16, 24, 32)$.

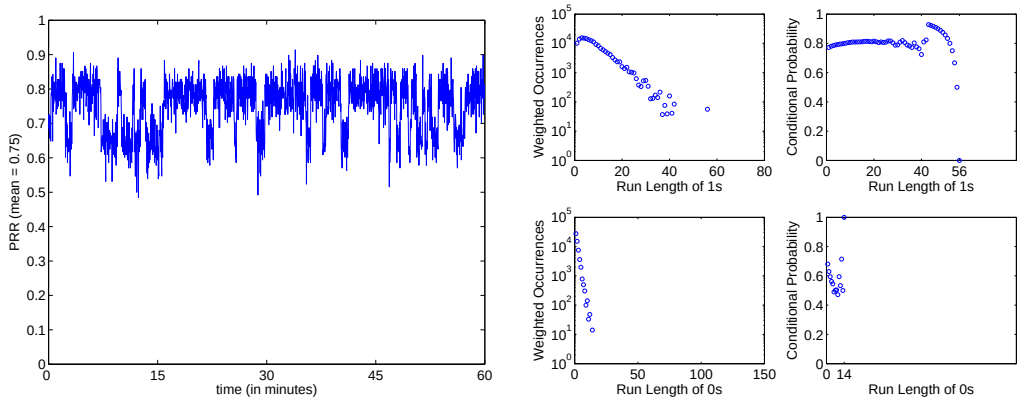
The role of W in the M&M model is to split the responsibilities between the transition probability matrix and the emission distribution. In principle, moving the modeling responsibility entirely to the transition matrix (by making $W = 1$ and having high Q) or to the emission distribution (by making W very large) could work by having a very large number of parameters. In practice it would not, because it would require a far larger training set and the model would be plagued with local optima of bad quality. Essentially, the short and long-term description is a divide and conquer strategy, and could be applied in general with a hierarchy of levels. Besides, long-term transitions can happen no faster than every W bits, which puts an upper limit (although very large in our traces) on W. Figure 2.9 plots the effect of variation in window size W on the quality of the packet loss model for a given training trace ($Q = 6$). From Figure 2.9(c), we see that at very small values of $W = 8$, the transitions between the long-term dynamics of the link are not captured accurately in the transition matrix of the underlying HMM-based model. As window size increases, models with higher values of W ($W = 64$) show similar variation in long term dynamics as the original link. Also, from Figure 2.9(c), we see that for small W $(= 8)$, the model is unable to account for the longer runs of 1's and 0's as seen in the original link. In contrast, the model for $W = 64$ has longest run of 77 1's (original link has 73 1's) and 46 0's (original link has 120 0's).



(a) Original PRR=72%



(c) M&M PRR = 75%, Q=6, M=5, W=64



(c) M&M PRR = 75%, Q=6, M=20, W=8

Figure 2.9: Average PRR over time from (a) experimental 1-hour data trace, (b) M&M simulated trace (W = 128) and (c) M&M simulated trace (W = 8), respectively.

Q	M	W	NND		LL
			RL	CPDF	
4	4	128	0.861	19.63	-123589
6	10	24	0.864	23.12	-120252
6	5	24	0.887	23.60	-119537
6	4	24	0.906	24.51	-119417
6	4	48	0.939	24.96	-120806
6	2	24	0.939	24.32	-119398
4	5	128	0.943	19.92	-123610
6	5	48	0.943	23.93	-120929
6	4	128	0.989	20.52	-123422
6	**5**	**64**	**0.993**	**22.17**	**-120129**

Table 2.3: NND and log-likelihood (LL) values for the testing trace using the M&M model for trace in Figure 2.9. The values in bold indicates the model with balanced performance in terms of NND and LL. The first model performs best in terms of NND but worse in terms of LL.

Table 2.3 ranks models based on the NND computed between the simulated and the testing trace. A lower NND is an indicator of how well the M&M model is able to simulate bursts of 1/0s. We observe that models with longer W ($>= 64$) perform better in terms of modeling long term dynamics and run length distributions of 1's and 0's compared to ones with smaller W (< 64) (from Figure 2.9). Thus, we need to select a model that strikes a balance between log-likelihood, NND values and long-term dynamics performance. The values in bold indicates the model size (Q, M, W) that provides the best balance in terms of the log-likelihood, NND computed w.r.t. the testing trace and long-term dynamics. We also plot the Allan deviation (see Figure 2.10) of three links: (i) a trace which was generated assuming independent packet loss pattern, (ii) the simulated trace and (iii) the testing trace. The point in the plot where the Allan deviation of the simulated and testing traces deviates from that of the independent trace indicates the characteristic burst length of the simulated and testing trace. From Figure 2.10, the Allan deviation plots of the testing trace indicates that packet reception is independent for time intervals less than 1 second but show bursty packet reception for intervals greater than 1 second.

In the measurement study of link burstiness, Srinivasan et al. [91] have observed that waiting for 500ms breaks the packet loss correlation. While studying the self-similarity property of links, Rusak et al. [86] have observed that links start displaying self-similarity (correlations) after 640ms. In the design of STLE [7, 3] set the threshold for identifying an intermediate link to 3 packets when the inter-packet interval (IPI) was set to 250ms. This design choice has the underlying assumption that intermediate links show stable short-term behavior over a period of >750ms. During the evaluation of Roofnet, Aguayo et al. [2] observed that bursty links show correlation out to at least 1 second. These empirical studies support the case for longer window sizes. Having longer window sizes for

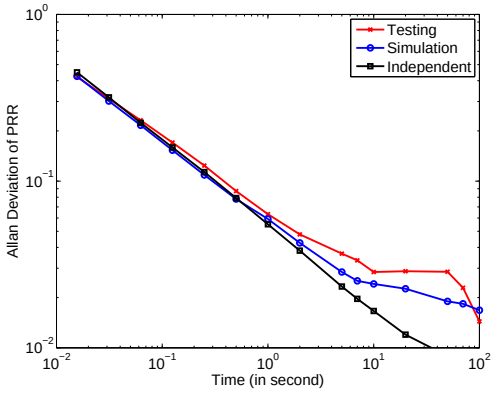


Figure 2.10: Variation in Allan Deviation of PRR for (i) a trace with independent packet loss pattern, (ii) the simulated trace with $Q = 6$, $M = 5$ and $W = 64$, and (iii) the testing trace.

the M&M models would help capture the correlations that exist in real world wireless links. While this would negatively affect model performance in terms of log-likelihood (see Figure 2.8 and Table 2.3), from a simulation users point of view M&M model's with longer window sizes would capture the correlations that exist in real world wireless links. Also, we saw in Figure 2.9(b) that M&M model with $W = 64$ was able to capture long and short term behavior better than models with smaller W $(= 8)$. In addition, the 1 second interval from the Allan deviation plot (refer Figure 2.10) coincides with window size $W = 64$.

These results show that for the M&M model to be of practical use, we need longer window sizes. Hence, we settle on $W = 64$ as choice for window size. For number of states, we select $Q = 6$ and $M = 5$ for mixture components for each state. Our M&M models with the chosen size do not simultaneously perform best in terms of NND and LL but balance the tradeoff associated with it. Note that for this particular model size, we used traces that contain at a minimum 230400 packet with an IPI=16ms. There is no formula for selecting the model size as

links may display a different burst pattern and dynamics that might require a different choice of model size. As a reference for simulation users, we offer the following guidelines while choosing a desired model size:

1. For a given training wireless link, choose W by plotting the Allan deviation plot. Select a W such that the packets inside the W-length window cover a time interval greater than the burst length.

2. Select Q greater than or equal to the number of long term states, one observes in the wireless trace.

3. For M, choose a value of 5 or less. Having more than 5 components increases drastically the number of parameters in the model and may lead to overfitting.

4. In addition to the points 1–3, make sure that the $\frac{L}{QMW} \geq 100$ rule is satisfied as it ensures that each Bernoulli prototype gets enough training data (i.e. ≥ 100). As a caveat, note that the rule assumes a uniform distribution of training data for each prototype which might not be true always.

Q	W=32	W=64	W=96	W=128	W=160	W=192
2	1.0101	1.0105	1.0078	1.0064	1.0077	1.0059
4	1.0051	1.0016	1.0086	1.0118	***1.0123***	1.0095
6	1.0048	1.0115	1.0102	1.0122	1.0061	1.0037

Table 2.4: Table shows the variation in λ as we vary Q and W for an empirical trace.

2.4.3 Model Convergence

Another point of interest is the rate of convergence of traces generated from the model parameters, or simply put, the variance between the original and generated traces. Ideally, we want the average packet reception rate of the simulated traces to be equal to that of the original trace. However, in reality, the simulated traces show some variance due to the stochastic nature of the model. The rate of convergence of the generated traces can be inferred from the model parameters, making certain models more suitable than others.

Using the Perron-Frobenius theorem [68], for the transition matrix of an irreducible finite Markov chain, the eigenvector corresponding to the leading eigenvalue is the unique stationary distribution for that Markov chain. Hence, in the M&M model for a packet reception trace, using singular value decomposition, we can calculate the leading eigenvalue and the corresponding eigenvector that gives the stationary distribution of the HMM. The ratio, λ, between the first two eigenvalues gives the rate of convergence of traces generated from the model. The closer λ is to 1, the slower is the convergence. Table 2.4 shows the values of λ as a function of Q and W. The numbers in bold-italics indicate the best value of λ (faster convergence) ($Q = 4, W = 160$).

In addition, given the stationary distribution, we can compute the average PRR of the HMM–MMB model as follows:

$$PRR = \sum_{q=1}^{Q} \nu_q \mu_q$$

where, ν_q is the value in the eigenvector giving the stationary distribution of the HMM corresponding to the q^{th} state and μ_q is the average PRR. From the MMB

emission distribution,

$$\mu_q = \sum_{m=1}^{M} \pi_{qm} \frac{\sum_{w=1}^{W} p_{qmw}}{W}$$

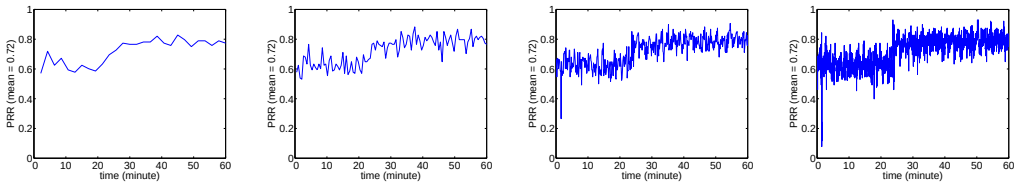
where, π_{qm} is the mixture proportion for the m^{th} mixture component in the q^{th} state and p_{qmw} is Bernoulli prototype in the w^{th} position.

2.4.4 Sensitivity Analysis: Dependence on IPI during Data Collection

During data collection for building the M&M model, we sent fixed size packets at a frequency of $64Hz$ or 64 packets per second (pps) in our experiments. In contrast, earlier studies [87] have collected the same data at a lower frequency ($4Hz$). To analyze the dependence of frequency of sending packets during the data collection phase on the quality of our model, we reduced the amount of data used for creating the model from the original $64Hz$ down to $1Hz$. Figure 2.11 shows the variation in reception rates for the same link modeled using different amounts of data. From Figures 2.11(a), 2.11(b), 2.11(c) and 2.11(d), we see that as the frequency increases, the greater amount of data used for creating the model helps the simulated trace follow the behavior of the original trace (see Figures 2.9(a)) very accurately at long and short time scales. As data collection frequency decreases, the simulation traces from the model get smoothed out, resulting in loss of detail in the short-term correlations. Hence, we advocate collecting data at high frequency when possible.

2.4.5 M&M Simulation Results

Table 2.5 shows comparisons between the test traces and the simulated traces from the M&M model ($Q = 6$, $M = 5$ and $W = 64$). The average difference be-



(a) $Freq. = 1Hz$ (b) $Freq. = 4Hz$ (c) $Freq. = 16Hz$ (d) $Freq. = 64Hz$

Figure 2.11: Dependence on frequency of sending packets during data collection.

tween the PRR of the simulated and the test link PRR is less than 1.9% whereas the average standard deviation in the PRR of the simulated M&M links is 0.017. The worst case difference in PRR is 6.6%. Table 2.5 also shows the difference between the run length and CPDF distributions in terms of the average L_1-norm and the NND. The minimum difference between the CPDFs in terms of the average L_1-norm and the NND is 0.11 and 3.1, respectively, and the maximum difference is 0.44 and 201, respectively. For the run lengths, the minimum difference in terms of the average L_1-norm and the NND is 0.004 and 0.4, and the maximum difference is 0.44 and 3.2, respectively. The maximum difference between the distributions occurred when the M&M model is not able to simulate the longer runs/bursts as seen in the testing trace and is captured by the NND computations.

When sampling state sequences from the M&M model, there is a possibility that the overall proportion of time the link stays is a particular state is different from the testing trace. Our goal was to be able to simulate traces which matched performance of the testing trace with similar long and short-term dynamics as observed in the training data. While simulating traces, we sampled state sequences from the model 20 times to observe the long term behavior. For computing other statistics, we sampled traces 100 times. However, we restricted our choice to traces that had a PRR within 5% of the training trace PRR. Therefore, in Table

V the standard deviation in PRR of simulated traces in all cases is less than 5%. While constraining the variability of the traces, it allows for a comparison to the testing trace on more even terms (i.e. state sequences (long term behavior) are similar but not same).

2.4.6 TOSSIM M&M Simulator

In order to make the M&M model accessible to WSN simulation users, we have incorporated it in the TOSSIM simulator for TinyOS 2.0. We have created a library of M&M models with intermediate PRRs ranging from 0% to 100% (using the model adaptation approach described in Section 3.5.1). Users have to compile their TinyOS program using "make micaz sim" command, which creates the TOSSIM related files. To test the working of their programs using radio communication simulated with the M&M model, users need to specify the M&M model file of required PRR on a link by link basis. The simulator generates PRR traces using these pre-computed models and utilizes the values (1/0) in the trace to make a decision regarding the link quality. In addition, the simulator can re-execute PRR traces generated in prior experiments or user supplied traces to allow for deeper analysis of program execution. Users can fine-tune the behavior of their algorithms to account for variation in link quality being simulated using the M&M model. The files required for the M&M simulator are available at [41].

2.5 Performance Comparisons with TOSSIM communication model

We conducted a statistical comparison between empirical data traces (testing set), simulation traces from the M&M model, and traces from TOSSIM, the

Test	M&M					TOSSIM				
Trace	PRR	Avg. L_1-norm		NND			Avg. L_1-norm		NND	
PRR	Mean±StdDev	CPDFs	RL	CPDFs	RL	PRR	CPDFs	RL	CPDFs	RL
46.9	47.5±2.0	0.12	0.007	40.1	2.1	41.7	0.698	0.029	40.6	2.8
52.0	51.2±0.2	0.44	0.052	10.2	2.2	0.2	0.990	1.026	4238	207
61.4	61.8±3.1	0.31	0.004	3.1	0.4	11.5	0.680	0.201	12.5	1.9
62.1	63.2±1.0	0.30	0.004	6.8	0.6	14.6	0.746	0.193	12.5	1.9
67.5	74.1±0.8	0.39	0.041	10.5	1.9	0.1	0.902	0.523	6965	181
70.6	74.0±2.0	0.24	0.008	201	2.1	22.5	0.859	0.180	261	5.2
72.3	74.8±2.4	0.37	0.057	101	3.2	11.6	0.854	0.204	111	4.1
72.8	75.0±2.9	0.21	0.021	27.3	1.7	27.0	0.844	0.135	159	5.2
88.6	89.0±1.7	0.11	0.030	4.7	1.9	0.1	0.979	1.001	80	227
90.6	91.2±1.1	0.26	0.049	30.7	1.6	6.5	0.941	0.210	40.6	8.3

Table 2.5: Comparison between empirical traces (testing set) and simulation traces using the M&M model and TOSSIM.

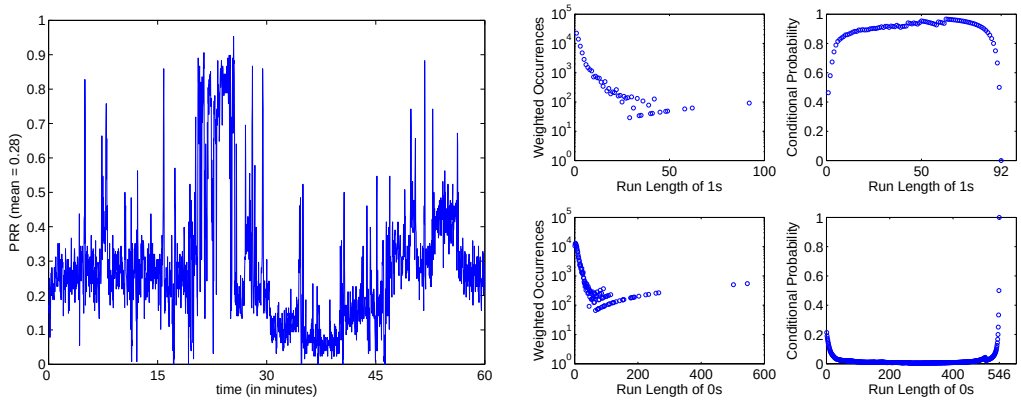
TinyOS simulation environment. TOSSIM requires a link gain model wherein a unidirectional link between a source and destination is associated with a gain value i.e., the received signal strength between the source and destination. For simulating traces in TOSSIM, for each of the empirical traces (testing set), we computed the median RSSI value of the received packets in the traces. We used this as the gain for the link gain model for the TOSSIM links.

TOSSIM utilizes a communication model called Closest-fit Pattern Matching (CPM) [57]. In order to utilize CPM, users must first collect a high-frequency noise trace from a deployed WSN that will be used to bootstrap the noise model. As mentioned in Section 2.2.1, we used the *RssiSample* application to collect these traces from the same environment where we collected our packet reception traces. To compare the performance of TOSSIM with the proposed M&M model, we bootstrapped the TOSSIM noise model using traces collected from the SE and MoteLab testbeds. The Signal-to-Noise Ratio (SNR) is computed using noise values generated by the CPM model. Using this SNR value, the corresponding

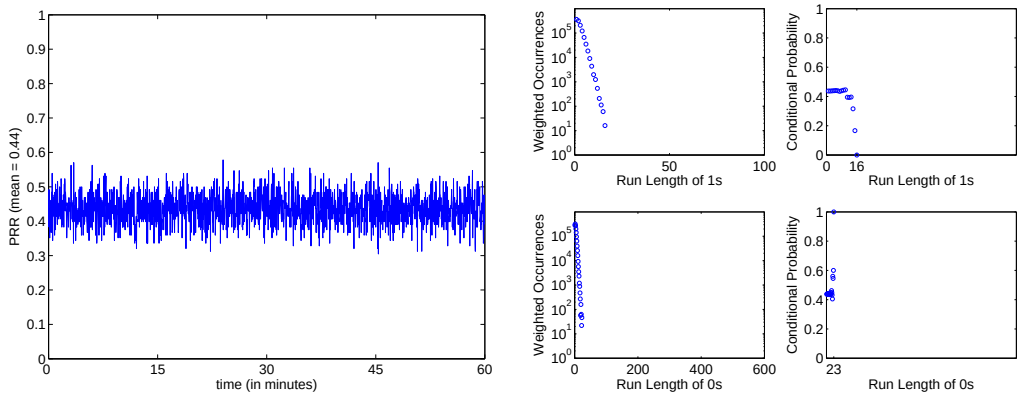
PRR value is determined using a SNR-PRR curve [58, 102]. The packet reception status (success/fail) for a packet is decided by sampling once from a Bernoulli distribution with $p = PRR$. In essence, the CPM model captures short term correlations as the noise dynamics are modeled using data collected over interval of $\approx 3 - 4$ minutes (1ms sampling interval). Note that during data collection, the noise samples were collected immediately after or very close to the time we collected the trace data.

Figure 2.12 shows the variation in PRR of a particular link and the simulated traces generated using TOSSIM and the M&M model trained on the same link. The goal of Figure 2.12 is to qualitatively contrast the link quality variation in simulation traces from TOSSIM and the M&M models with respect to an original link manifesting complex link dynamics. It is clear from Figure 2.12(b) that TOSSIM is unable to capture the long term variations in PRR that are better modeled by the M&M model (see Figure 2.12(c)). Furthermore, the average PRR of the M&M link (27%) is closer to the original link PRR (28.47%) than the TOSSIM link PRR (49.49%).

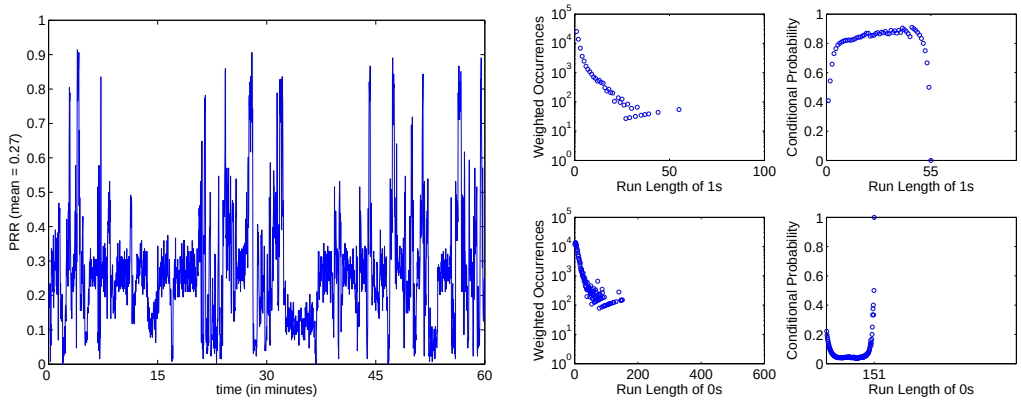
Figures 2.12(a), 2.12(b) and 2.12(c) show the weighted RL and CPDF distribution of 1's and 0's for the original link, TOSSIM simulation trace and M&M simulation trace, respectively. We observe that the M&M model has longest run of 55 1's (original link has 92 1's) and 151 0's (original link has 546 0's). On the other hand, TOSSIM generated trace with 16 1's and 23 0's. It is clear that TOSSIM is not able to simulate the longer runs of 1's and 0's. This is also reflected in the NND computed for the TOSSIM and M&M traces. The NND for the RL distribution of the TOSSIM and M&M traces is 4.07 and 1.8, respectively. The NND for the CPDF distribution of the TOSSIM and M&M traces is 82.2 and 40.16, respectively. These values indicate that quantitatively the M&M



(a) Original PRR=28%



(b) TOSSIM PRR = 49%



(c) M&M PRR = 27%, Q=6, M=5, W=64

Figure 2.12: Average PRR over time from (a) experimental 1-hour data trace, (b) TOSSIM simulated trace and (c) simulated trace using the M&M model, respectively.

traces are closer to the original traces than the TOSSIM traces.

Table 2.5 shows the summary of the comparison between the empirical traces (testing set) and traces generated using TOSSIM and the M&M model. The first point to notice is that there are significant differences in PRR between the actual link and TOSSIM model with a minimum difference of 5% and a maximum of 88%). In contrast, the M&M model has a maximum and minimum difference in PRR of 6.6% and 0.4%, respectively. The maximum NND for the run length distribution of the TOSSIM and M&M traces is 226.9 and 3.2, respectively. The maximum NND for the CPDF distribution of the TOSSIM and M&M traces is 6965 and 201, respectively. These values indicate that quantitatively the M&M traces are closer to the (unseen) testing traces than the TOSSIM traces. The combined knowledge of the difference in PRRs and the average L_1-norm and NND values for the distributions of run lengths and CPDFs indicate that TOSSIM does not do an adequate job of modeling the link variations.

We believe the poor performance of TOSSIM can be explained by the inadequate characterization by the path loss model and the noise model being able to account only for short-term correlations. Currently, TOSSIM uses the gain of the link and the noise value computed by CPM to decide whether the packet is received or dropped. However, the generic constants of the path loss model are not the same for all environments. This may lead TOSSIM to make significant errors while computing the PRR of a packet at the receiver. In addition, it has been shown that the RSSI values recorded by nodes have errors due to calibration issues [16]. These may introduce inaccuracies in computing the SNR which further propagate to PRR calculations. While it is possible to eliminate this error by calibrating all receiver nodes, it introduces an additional non-trivial overhead. Raman et al. [79] have shown that intermediate links occur due to the unpre-

dictable behavior at the RSSI threshold. This indicates that any model such as the path loss model which uses SNR values that are dependent upon RSSI would inadequately characterize the link behavior. This would not affect packet loss models such as ours that only use knowledge of packet receptions and losses to model the link behavior. In Table 2.5, there are cases where the TOSSIM traces differs significantly in PRR from the test trace. Also, in such cases the values of NND are extremely large. This can be attributed to the unpredictable link behavior at the RSSI threshold, resulting in the SNR–PRR curve in TOSSIM outputting a PRR that is very different from the one observed in the wireless trace. In other cases, lower NND for M&M model traces indicates better performance in terms of run length distributions. Here, our choice of α does not exaggerate the slightly worse performance of the simulation traces when there minor differences in the run length distributions.

2.6 Discussion

2.6.1 Relevance to Other Analytical Models

The Gilbert-Elliott model [30, 22] is a particular case of the M&M model where we have a single-bit window ($W = 1$) and $Q = 2$ states; and each state has a single-component MMB ($M = 1$). Its only tunable parameters are the transition probabilities and the Bernoulli parameters (total 4 parameters). The generality of our model allows us to model and learn from data, not just bursts, but far more complex behaviors. The Markov-Based Trace Analysis (MTA) [47] is an extension of the Gilbert-Elliott model wherein one state corresponds to the "error free" state of the channel and the other state is comprised of a discrete time Markov chain of order 6 to model the "lossy" state of the channel. This was further

extended to account for variability in wireless links by using a hierarchical model with multiple states [48], where each state is comprised of a 2-state MTA-based model. Salamatian et al. [88] used Hidden Markov Models with Bernoulli emission distributions for modeling packet reception traces from Internet communication channels. Their model is a particular case of the M&M model with single-bit window ($W = 1$) and $Q \leq 4$ states; and each state has a single-component MMB ($M = 1$).

2.6.2 User Control

The M&M model is a purely data-driven approach. It is possible to combine this with a non-adaptive approach so that the user may have manual control on specific characteristics (such as the amount of overall burstiness or fading rate) while still generating realistic traces. In fact one example of this is our combination of models of Section 3.5.1, where the within-model parameters are trained and the mixing proportions or the Bernoulli parameters can be chosen by the user. For example: consider a MMB having $W = 6$ and $M = 2$. π_i's indicate mixture proportions and p_i's indicate Bernoulli parameters for the mixture components. In this mixture, if the user wants the model to output increased runs of 1's of length 3 and runs of 0's of length 2, then the goal can be achieved by changing the mixture parameters as shown below:

	Before		After
π_i :	$(p_1, ..., p_6)$		$\pi_i^{'}$: $(p_1^{'}, ..., p_6^{'})$
.6 :	$(.4, .7, .6, .7, .8, .5)$	$\rightarrow$	.6 : $(.4, .2, .9, .9, .9, .2)$
.4 :	$(.4, .3, .3, .2, .2, .6)$	$\rightarrow$	.4 : $(.4, .3, .9, .1, .1, .9)$

Similarly, it is possible using a simple heuristic to find Bernoulli parameter values above/below a certain threshold (0.6 and 0.2 in the example) equal to the length of the required bursts and adjust them and their neighboring parameter values to ensure bursts of required lengths.

2.6.3 Model Limitations

A low power wireless link with say two long-term states having the same average PRR but completely different short term dynamics, independent behavior in one state and long bursts of 1/0s in the other, can be modeled adequately using the M&M model with proper initialization. However, modeling the transition matrix can be an issue if the transitions between the long term states occurs just once in the entire trace. This can be attributed to the lack of data and longer traces which could capture more transitions between long term states can help alleviate the problem. In addition, the M&M model does not perform well when modeling dynamics wherein the PRR changes monotonically with time, as the output distribution of the M&M model will converge to the average PRR, resulting in an inadequate model of the short-term dynamics.

2.7 Summary

In this chapter, a novel multilevel approach involving Hidden Markov Models (HMMs) and Mixtures of Multivariate Bernoullis (MMBs) for modeling the long and short time scale behavior of links in wireless sensor networks, that is, the binary sequence or trace of packet receptions (1s) and losses (0s) in the link, was presented. A HMM modeled the long-term evolution of the trace (level 1) as transitions among a set of unobserved, level-1 states. These states typically corre-

spond to a roughly constant packet reception rate (as determined by the data) and might correspond to different regimes of the link. Within each level-1 state, the short-term evolution of the trace (level 2) is modeled by an MMB that captures the faster, but not random, variations of the sequence of packet receptions and losses. The synthetic traces generated from the model were characterized in terms of several statistical measures: moments (mean and variance) of the distribution of packet reception traces, run length distributions of packet receptions and of packet losses, and conditional packet delivery functions (CPDFs). To compare run length and CPDF distributions, a new metric called the Nearest-Neighbor Distance was proposed. In addition, a full implementation of the M&M model for the TOSSIM simulator is provided. The M&M model significantly outperforms existing models such as CPM in TOSSIM that only account for short-term behavior.

In many settings, the benefits from a pure data-driven approach, such as the M&M model, are not that large because the generalizability of simulating from traces is a big limitation. For example, one would like to model the characteristics of a real environment in a simulated network without having to first deploy a network to measure its properties or by collecting significantly smaller data traces than the one used to train the model in a different environment. This problem can be solved by using model adaptation techniques and will be addressed in the next part of this dissertation.

CHAPTER 3

Adapting Data-Driven Communication Models

The M&M model aimed to alleviate wireless link modeling problems by capturing variations in bit patterns of wireless links over long (minutes, hours) and short (milliseconds) time scales. However, high quality data-driven models, such as the M&M model, necessitate the deployment of testbeds to collect extensive data traces that are essential to estimate good values for the numerous model parameters. But one could argue that it would take the same effort to deploy nodes and run actual experiments, instead of spending time setting up a testbed for gathering data for simulation models. This results in a circular argument where to improve results in real world experiments, one is advocating the use of simulation, but for accurate simulation, it is required to collect real world data.

In addition, one drawback of data-driven models with many parameters is the need for sufficiently large training sets to achieve reliable estimates. This means that for each link of a sensor network that we want to model, the network developer must first record data for enough time (hours or days). This prevents quick setup of a new link and is costly in resources (e.g. battery life), particularly for sensors in hard-to-reach locations (such as climate-sensing networks in Greenland). In these situations, it makes sense to re-use an existing model (which we call reference) that has been trained with extensive data and adapt it to the new situation (the target distribution) given a far smaller data trace from the new target link than would be necessary to train a new model from scratch. This is

the adaptation setting that we pursue in this study.

Our goal is to show that by gathering little data locally in a site similar to the target deployment, simulation users can construct link models that can generalize to links with similar PRR at the target deployment. For example, using a unique well-trained reference model of a wireless link from a local testbed, one can construct link models by gathering small amounts of traces in a park such as Muir Woods in California. This can allow for realistic simulations of at least some aspects of wireless communication at the deployment site in the Amazon rain forests in Brazil without going there.

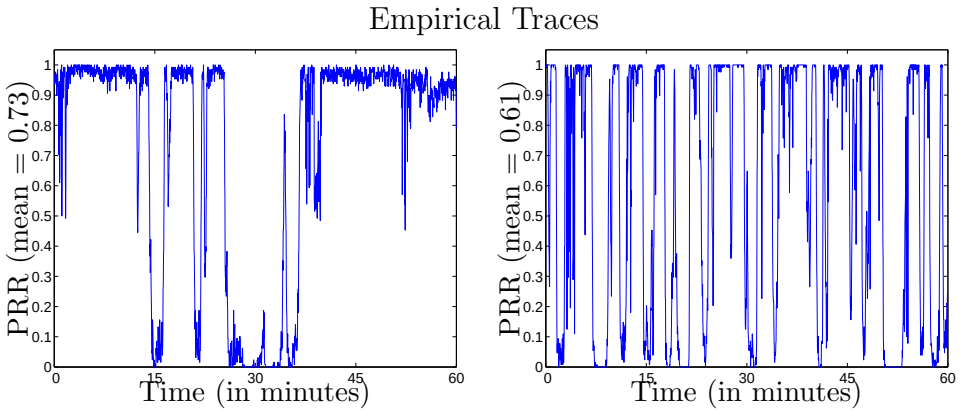
3.1 Which Parameters can be Adapted

The parameters of the M&M model are the transition matrix, which account for the long-term dynamics and the MMB emission distribution for each state, which captures the short-term correlations in packet reception. In this section, we take a look at which of these parameters are be adapted using short data traces.

3.1.1 Transition Matrix

In the M&M model, the long term variations in the behavior of wireless links are modeled using the transition matrix of the HMM–MMB. Figure 3.1 shows the PRR variation of two empirical wireless link traces over a period of 1 hour and their corresponding transition matrices. The diagonal elements in the transition matrix indicate the duration for which the link stays in a regime of fixed PRR. In this example, we can discern that the regimes of PRR are almost identical for these two links, one state having PRR$\approx$1 and the other having PRR$\approx$0. From the values of the transition matrices for these empirical traces, we can note

that there is very little difference between the diagonal elements. This small difference in values of the transition matrix drastically alters the look of the long term dynamics as seen in Figure 3.1. This means that in order to accurately estimate the transition matrix, it is required to utilize the entire data trace. In other words, with adaptation data recorded over a few minutes, it is not possible to infer about the long time scale behavior of any link.



Transition Matrix

$$\begin{bmatrix} 0.9658 & 0.0341 \\ 0.0171 & 0.9828 \end{bmatrix} \qquad \begin{bmatrix} 0.9462 & 0.0537 \\ 0.0687 & 0.9312 \end{bmatrix}$$

(a) Trace PRR = .73 (b) Trace PRR = .61

Figure 3.1: Difference between empirical traces having similar short term dynamics but different long term dynamics.

3.1.2 MMB Distribution per State

In the M&M model, the MMB output distribution for each state captures the correlations between successive packet receptions and losses. In order for model adaptation to work, there should exist some correlation between the parameters of an MMB distribution from one link to another (reference and target model).

We started by analyzing the MMB parameters from the M&M models created using packet data traces of links from the Indoor testbed. Our approach was as follows: Suppose X_i and Y_j represent the MMB output distributions of the i^{th} and j^{th} state from two separate links (X and Y). Both X_i and Y_j have dimension $M \times W$. We try to find an $\mathbf{a}$ such that:

$$\begin{aligned} {Y_j}_{M\times W} &= a_{M\times M} {X_i}_{M\times W} + \epsilon_{M\times W} \\ &\text{or} \\ {Y'_j}_{W\times M} &= a_{W\times W} {X'_i}_{W\times M} + \epsilon_{W\times M} \end{aligned} \tag{3.1}$$

where, $\mathbf{a}$ is diagonal and has dimensions $M \times M$ (or $W \times W$) and ϵ is as small as possible. In effect, when $\mathbf{a}$ is $M \times M$, each element of $\mathbf{a}$ is the (linear) transformation that is shared between the W Bernoulli parameters within each component of the MMB. Thus, the W parameters within each component are tied together (tying components). Alternately, when $\mathbf{a}$ is $W \times W$, each element of $\mathbf{a}$ is the (linear) transformation that is shared between the M Bernoulli parameters along each dimension of the MMB. Thus, the M parameters within each dimension are tied together (tying dimensions). The interpretation of $\mathbf{a}$ is that the values of Bernoulli parameters of the adapted model are derived by increasing, decreasing or keeping constant the values of the Bernoulli parameters along each component or dimension of the reference model. Hence, we need to estimate either M or W parameters for adapting the reference model to the target model.

We performed some preliminary analysis using some intermediate PRR links (SE testbed dataset, see Section 2.2.1). The MMB model for one of the links was designated as X and the remaining MMB models for other links were designated as Y's. The MMB parameters for each Y's component is increased or decreased uniformly using scaling factor a_{ii} till the PRR of the corresponding component in X is reached. The prototype values of each MMB component are not allowed to exceed 1 or fall below 0. Our preliminary analysis (Table 3.1) showed that

ϵ was in the order of 0.08 when using $\mathbf{a}$ of dimension $M \times M$, and 0.12 when using $\mathbf{a}$ of dimension $W \times W$. This implies that there exists a transformation, non-linear in this case, that could be utilized to adapt the parameters of the reference model to closely match those of the target trace. Therefore, we can apply model adaptation techniques to estimate these transformation parameters. Since, ϵ is smaller when using $\mathbf{a}_{M \times M}$, we will attempt model adaptation by tying components together.

Parms	Links									
M	.14	.08	.13	.05	.06	.14	.04	.13	.08	.05
W	.13	.07	.20	.10	.11	.18	.19	.14	.08	.15

Table 3.1: Residual values when attempting to find correlations between MMB distributions.

3.2 Background and Related Work

In machine learning, work on transfer and multitask learning has considered the problem of learning models such as classifiers in the presence of different domains [94]. In our adaptation setting, we do not know at training time the target distribution we need to model. Our work is most closely related to adaptation methods in speech processing [98], where given a Gaussian-mixture-based HMM trained for a reference speaker we want to adapt it to a new speaker given as little new speech as possible. Maximum-a-posteriori methods apply Bayes' rule using as prior the trained model [29] and converge to the true maximum-likelihood estimate with infinite data, but do poorly with little data because only a few parameters are updated. Thus, most work is based on tying together groups of parameters (means, covariances) and using linear transformations of the parameters or features [60, 18, 59, 76]. This does not converge to the maximum-likelihood

estimate but updates all parameters and significantly improves the model with little data. As we show later, linear transformations are not suitable with MMBs because (unlike the means of a Gaussian) the prototypes are constrained to be in $[0, 1]$.

Other work considers a space where each point represents a model, and constrains the target model to be in a manifold or cluster set spanned by existing trained models [52, 28]. However, this requires sufficiently many trained models, which may not be available in practice.

3.3 Mixture of Multivariate Bernoulli Distributions

Mixtures of multivariate Bernoulli distributions (MMB) are widely used to model high-dimensional binary data in terms of a few latent classes, from bacterial taxonomy to binary images [25, 12]. Given a data vector $\mathbf{x} \in \{0, 1\}^W$ with W binary variables, its density is

$$p(\mathbf{x}) = \sum_{m=1}^{M} \pi_m p(\mathbf{x}|m) \quad p(\mathbf{x}|m) = \prod_{w=1}^{W} p_{mw}^{x_w} (1 - p_{mw})^{1-x_w}$$

where there are M components and the parameters are the mixing proportions π_m (which are positive and sum to one) and the prototypes $\mathbf{p}_m \in [0, 1]^W$. Thus, variables within a component are independent, but not across components. With enough components, an MMB can represent complex high-dimensional distributions.

Given a training set, an MMB is usually trained with an EM algorithm. The E step computes the posterior probability of each component given a data vector. The M step estimates the parameters of each component: its mixing proportion is proportional to the total posterior probability of the component, and its prototype is the average of the whole data wrt the posterior probabilities.

The EM algorithm needs initial values for the parameters and can converge to local optima.

In the context of adaptation, we will call *retraining* the process of estimating an MMB using this EM algorithm given the adaptation data, and initializing the parameters to those of the reference MMB. Retraining with little data leads to estimates that overtrain and generalize poorly to future data. For example, in applications like those we consider (binary images or windows), the space dimensionality W is large (hundreds), so if little adaptation data is available, some of the dimensions in the data may consist mostly (or only) of 0s or 1s. The correspoding p_{mw} value will clamp to (close to) 0 or 1 and will then rarely generate a 1 or a 0, respectively, during sampling, so the simulated traces will not be representative of the data.

3.4 Adapting the MMB

We now assume we have an MMB model (the *reference* model) that has been trained on an extensive dataset (say, from a given wireless link in a network), that is, we have the values of its parameters (mixing proportions and prototypes). We are given an *adaptation dataset*, sampled from an unknown *target* distribution, containing a small number N of binary W-dimensional vectors $\{\mathbf{x}_n\}_{n=1}^N$, and we want to learn a new MMB model, with parameters $\{\widetilde{\pi}_m, \widetilde{\mathbf{p}}_m\}_{m=1}^M$, for the target distribution. Our algorithm is based on the idea of tying the MMB parameters together through a transformation of the reference parameters. The transformation itself has very few parameters, so they can be learned from the small adaptation dataset, but their effect is propagated to all the MMB parameters through the transformation. Specifically, we obtain each new W-dimensional prototype $\widetilde{\mathbf{p}_m}$ as a *nonlinear transformation* $\mathbf{f}(\mathbf{p}_m, \boldsymbol{\theta}_m)$ of the reference prototype $\mathbf{p}_m$, indepen-

dently for each component, using a number of parameters $\boldsymbol{\theta}_m$ much less than W. The transformation is nonlinear because the prototypes must be in $[0, 1]$. With a linear transformation with shared parameters, the total amount of change in $\mathbf{p}_m$ is limited, because reference values p_{mw} close to either 0 or 1 would immediately reach 0 or 1 (saturate) and prevent the remaining, less extreme values from adapting. This is a major difference with existing adaptation work on Gaussian mixtures where the Gaussian means are unconstrained and linear transformations suffice. We apply a sigmoid transformation with parameters $a_m, b_m \in \mathbb{R}$ elementwise to each entry in $\mathbf{p}_m$:

$$\widetilde{p}_{mw} = \sigma(p_{mw}; a_m, b_m) = \frac{1}{1 + e^{-(a_m p_{mw} + b_m)}},\ w = 1, \ldots, W.$$

This allows large changes to all p_{mw} even if some are close to the boundaries. (In the nongeneric case where all p_{mw} values are identical within one component, there is an infinite number of (a_m, b_m) values that can map it to a given output, and our algorithm will find one of those.) As for the mixing proportions, since there is only one per component, we consider them as free during adaptation (subject to adding to 1).

Thus, our algorithm needs to maximize the likelihood of the adaptation data over a total of $3M - 1$ free parameters (mixing proportions $\widetilde{\pi}_1, \ldots, \widetilde{\pi}_{M-1}$ and sigmoid parameters $a_1, b_1, \ldots, a_M, b_M$), which with our high-dimensional data is far less than $(W + 1)M - 1$ parameters (proportions and prototypes) for the unconstrained MMB.

Like the EM algorithm for MMBs, our generalized EM (GEM) algorithm can converge to local optima. Since the point of adaptation is that the reference model should be relatively close to the target one, the initial values for $\{\widetilde{\pi}_m, a_m, b_m\}$ should be such that the MMB they represent is as close as possible to that of the reference model. This is achieved by setting $\widetilde{\pi}_m = \pi_m$ and $a_m = 5.47$, $b_m = -2.79$;

the latter correspond to the sigmoid that is closest to the identity transformation.

3.4.1 A Generalized EM algorithm for Adaptation

Our objective function is the log-likelihood of the adaptation data given the constrained MMB model with $3M - 1$ free parameters:

$$L\left(\{\widetilde{\pi}_m, a_m, b_m\}_{m=1}^M\right) = \sum_{n=1}^{N} \log \sum_{m=1}^{M} \widetilde{\pi}_m p(\mathbf{x}_n; a_m, b_m)$$

where $p(\mathbf{x}_n; a_m, b_m)$ is a multivariate Bernoulli with prototype $\widetilde{\mathbf{p}}_m = \sigma(\mathbf{p}; a_m, b_m)$. We provide a generalized expectation-maximization (EM) to maximize it [66]. Unlike in the EM algorithm to train an MMB, the use of a nonlinear transformation makes the M step now not solvable in closed form for $\{a_m, b_m\}$. Instead, we need to solve it iteratively; we have found the BFGS algorithm (a quasi-Newton algorithm with superlinear convergence; [73]) effective. Since this increases but (if we exit BFGS early) need not maximize the likelihood within the M step, our EM algorithm is generalized, and the theorems for convergence of GEM apply [66]. The E step is analogous to that of the EM algorithm for MMBs. In the equations below, note $\widetilde{p}_{mw}^{\tau} = \sigma(p_{mw}; a_m^{\tau}, b_m^{\tau})$.

E step This computes $r_{mn}^{\tau} = p(m|\mathbf{x}_n; \widetilde{\pi}_m^{\tau}, a_m^{\tau}, b_m^{\tau})$, the posterior probability of component m given data point $\mathbf{x}_n$ under the current iteration's (τ) parameters:

$$r_{mn}^{\tau} = \frac{\widetilde{\pi}_m \prod_{w=1}^{W} (\widetilde{p}_{mw}^{\tau})^{x_{nw}} (1 - \widetilde{p}_{mw}^{\tau})^{1-x_{nw}}}{\sum_{m'=1}^{M} \widetilde{\pi}_{m'} \prod_{w=1}^{W} (\widetilde{p}_{m'w}^{\tau})^{x_{nw}} (1 - \widetilde{p}_{m'w}^{\tau})^{1-x_{nw}}}.$$

M step This results from increasing or maximizing Q, the expected (wrt the

r_{mn}^{τ}) complete-data log-likelihood, over $\widetilde{\pi}_m, a_m, b_m$:

$$Q(\{\widetilde{\pi}_m, a_m, b_m\}_{m=1}^{M}; \{\widetilde{\pi}_m^{\tau}, a_m^{\tau}, b_m^{\tau}\}_{m=1}^{M}) = \sum_{n=1}^{N} \sum_{z_n=1}^{M} r_{mn}^{\tau} \log \left(p(z_n; \widetilde{\pi}_{z_n}) p(\mathbf{x}_n | z_n; a_{z_n}, b_{z_n}) \right)$$

where we call $1 \leq z_n \leq M$ the (unknown) index of the mixture component that generated data point $\mathbf{x}_n$. We obtain a closed-form solution for the mixing proportions:

$$\widetilde{\pi}_m^{\tau+1} = \frac{1}{N} \sum_{n=1}^{N} r_{mn}^{\tau}$$

but the expression for the gradient of Q wrt $\{a_m, b_m\}$

$$\frac{\partial Q}{\partial a_m} = \sum_{n=1}^{N} r_{mn}^{\tau} \sum_{w=1}^{W} p_{mw}(x_{nw} - \widetilde{p}_{mw})$$
$$\frac{\partial Q}{\partial b_m} = \sum_{n=1}^{N} r_{mn}^{\tau} \sum_{w=1}^{W} (x_{nw} - \widetilde{p}_{mw})$$

when equated to zero cannot be solved in closed form for $\{a_m^{\tau+1}, b_m^{\tau+1}\}$, so we iterate over $\{a_m, b_m\}$ using BFGS.

3.4.2 Computational Complexity

Our algorithm consists of an outer loop of GEM iterations, and an inner loop of BFGS iterations for the M step. Our experiments show how to set the exit tolerance and maximum number of inner-loop iterations. Computing the BFGS search direction is $\mathcal{O}(M^2)$ (a matrix-vector product of order M), which is negligible wrt computing the E step and gradient of Q, both of which cost $\mathcal{O}(MNW)$. Thus the algorithm runtime is $\mathcal{O}(NMW)$ times the total number of inner-loop iterations.

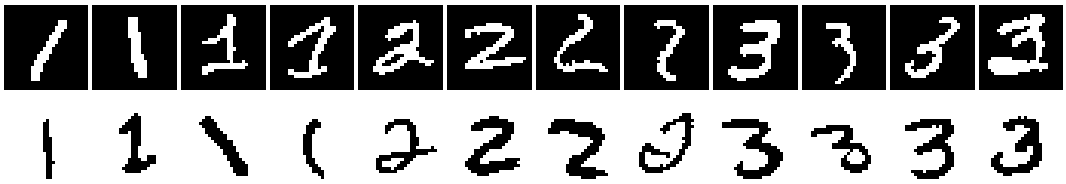


Figure 3.2: MNIST: sample training vectors $\mathbf{x}_n$ in the reference (top row) and target (bottom row) datasets.

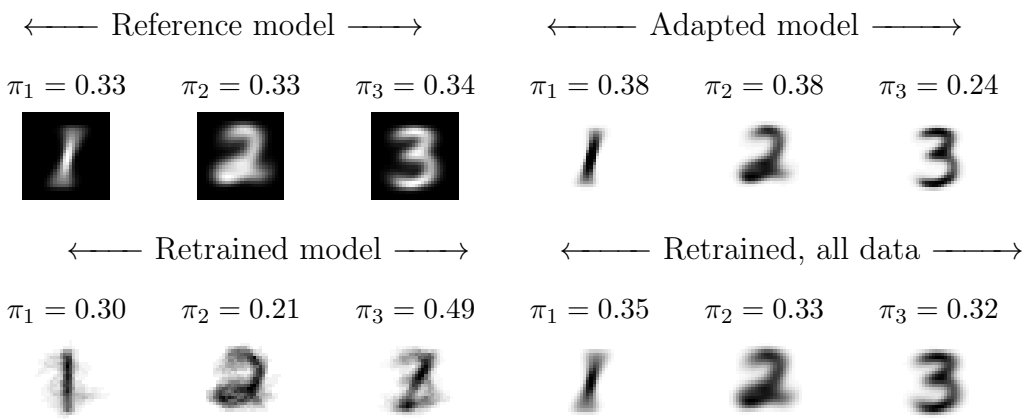


Figure 3.3: MNIST: MMB parameters for the reference model, adaptation (with $N = 100$ adaptation points), retraining (with $N = 100$) and retraining (with $N = 18\,000$).

3.4.3 Illustrative Experiment: MNIST handwritten digits

The MNIST dataset, commonly used in machine learning, contains grayscale images of 28×28 pixels ($W = 784$ dimensions) of handwritten digits from many writers. We illustrate our adaptation algorithm by having a reference model trained on a large subset of MNIST, and then adapting to a small subset where the pixel intensities have been inverted (see Figure 3.2). This represents a situation where the target distribution is very different from the reference distribution, but many of its characteristics (e.g. digit class, slant, thickness) do not change. These

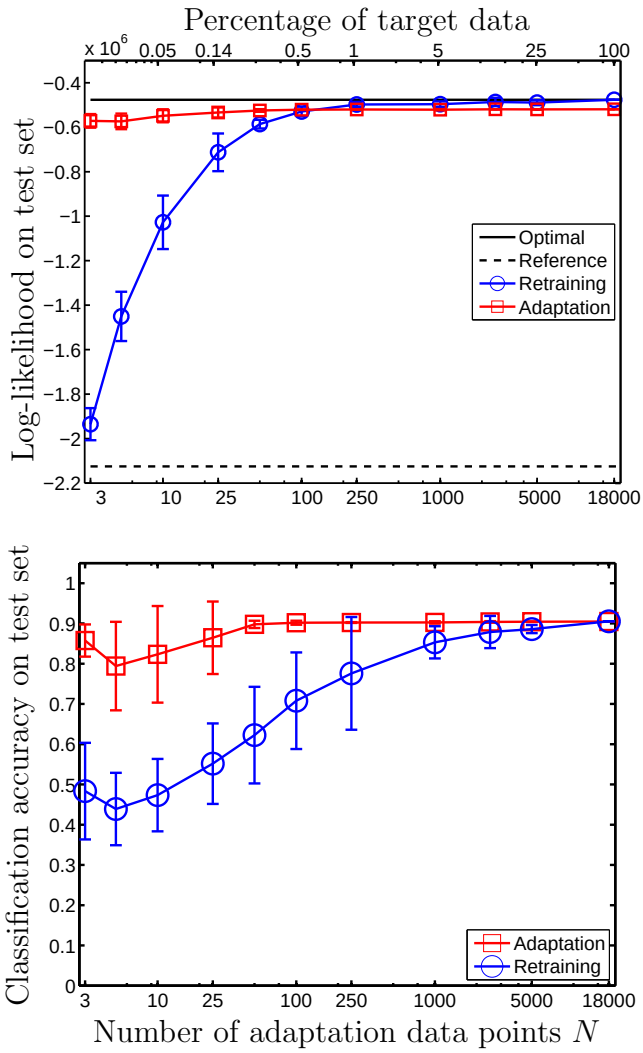


Figure 3.4: MNIST: results of the retraining (red) and adaptation (blue) algorithms, the reference model (dashed black) and retraining with all data (solid black) on the log-likelihood (top) and classification accuracy (bottom) on test sets, as a function of the adaptation set size N. Errorbars over 50 random subsets of adaptation points.

characteristics have been learnt by the reference model using a large training set, and the goal of the adaptation is to preserve those but change the intensity to match the new, inverted one.

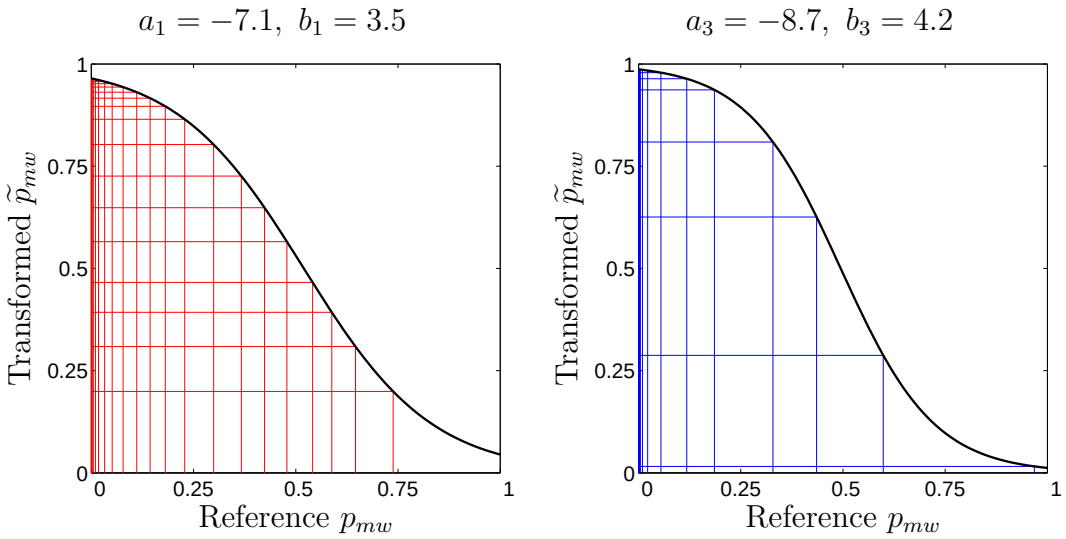


Figure 3.5: MNIST: estimated sigmoids for two of the components. The vertical and horizontal lines indicate pairs $(p_{mw}, \widetilde{p}_{mw})$ (not all W pairs are shown, to avoid clutter). Note how both essentially invert the input.

We used data from the digits '1', '2' and '3' only. We randomly split the 10 000 digits per class into a training set (3 000), a target dataset (6 000) and a test dataset (1 000). Although MNIST provides the digit label, none of our experiments use it, so all models are learnt in a purely unsupervised way. We converted the grayscales from $[0, 1]$ to binary using a threshold of 0.5, and inverted the images from the target and test sets. Our reference model (Figure 3.3) had $M = 3$ components (for a total of 2 354 parameters) and was learned on the training set with the EM algorithm. For adaptation (8 parameters) and retraining (2 354 parameters), we used randomly selected subsets of size N from the target set, where N varied from 3 to 18 000 (the whole target set); note that subsets did not necessarily contain the same number of '1', '2' or '3'. The experiments were repeated over 50 subsets each to obtain errorbars.

Figure 3.3 shows the learned parameters for a particular subset of $N = 100$

adaptation vectors. The parameters resulting from adaptation resemble very much the ones resulting from retraining with extensive data, which in turn resemble the reference ones but inverted (also shown by the inverted sigmoids in Figure 3.5). The prototypes look like smooth, average shapes representative of the entire populations of '1', '2' and '3'. Even though we adapt only 8 free parameters, all 2 354 parameters (prototypes and proportions) undergo large changes. However, retraining with $N = 100$ vectors gives poor results, with each component learning slant and other traits that are idiosyncratic to the particular adaptation set used. Figure 3.4(top) shows the log-likelihood on the test set as a function of N. The adaptation algorithm achieves results close to retraining with all data ("optimal" line) for very small N and reliably so (note the tight errorbars). As N increases, the adaptation performance stagnates without reaching the optimal baseline, a necessary consequence of learning very few free parameters. Retraining needs $N > 100$ vectors to equal adaptation and performs poorly and unreliably with small N. The classification accuracy (Figure 3.4, bottom), where we assigned a test image $\mathbf{x}$ to the cluster with largest posterior probability $p(m|\mathbf{x})$ under the MMB, shows similar results.

3.5 Model Adaptation for Wireless Link Modeling

Radio propagation models [80] are site-specific because the propagation mechanisms are affected by the characteristics of the surrounding environment. In [89], path loss readings were collected at 634 locations to characterize propagation in different buildings. Previously proposed bit error and packet loss approaches include curve-fitting of the error and error-free traces using exponential and Pareto distributions [72]. Jiao et al. [38] proposed a gap model utilizing time-inhomogeneous Markov chain to fit distributions of error and error-free burst

lengths. Kopke et al. [49] used chaotic maps to characterize bit error models using the complementary cumulative distribution function of the run lengths of 1's and 0's to estimate model parameters. The k-state threshold approach [10] modeled frame errors by characterizing model states with respect to the size of error-free bursts. The common theme in [38, 49, 10] is to compute statistics using the distribution of the lengths of error and error-free traces. This results in their shortcoming that the entire packet trace needs to be available for computing the model parameters. Therefore, to simulate communication realistically, a simulation user has to collect either radio propagation data or bit-error/packet traces. In this case, it is only fair to impose strict data requirements, in a qualitative and quantitative sense, on the modeling approach. Qualitatively, the data collection should not burden the user, i.e., least effort from the user (in terms of deploying nodes) and quantitatively, the modeling approach should be required to create realistic models by using this little training data.

3.5.1 Model Adaptation without Adaptation Data

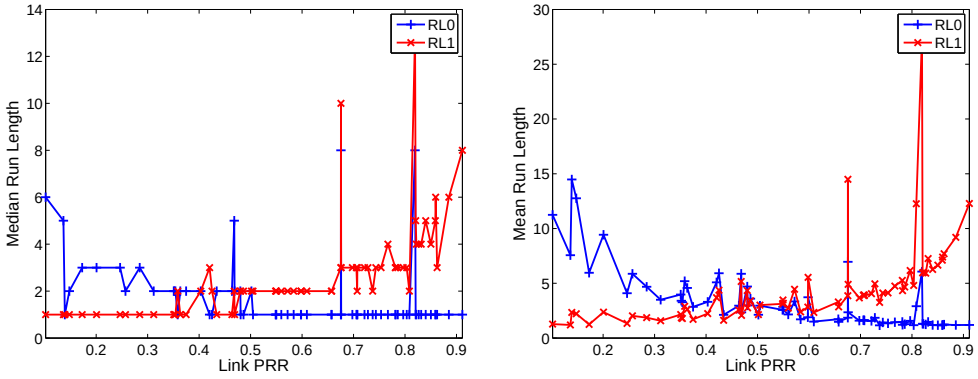
Typically, simulation users require a wide variety of intermediate-quality link PRR models. To create an adapted model for a link, we first need to gather packet reception traces as input to the model. This necessitates data collection which may or may not be possible for simulation users. In such cases, it would be useful to have some pre-existing models for such links or be able to modify the parameters of existing link models to get models for new links with different PRRs. In this section, we propose a method for creating new intermediate link PRR models from pre-existing M&M models, only using knowledge about the target link PRR.

3.5.1.1 Approach

For packet reception traces of good and bad quality links, we are more likely to have very long runs of 1's and 0's interspersed with a few 0's and 1's, respectively. Here the MMB parameters for the M&M model would be close to 1 for good-quality links and close to 0 for bad-quality links. Studies [91, 90] have shown that intermediate-quality links show bursty behavior, i.e., empirical links show complex patterns of 1's and 0's. The M&M model can capture these complex patterns of 1's and 0's as observed by it performance in terms of run length distributions. Before going into the detail of creating new intermediate link PRR models, we would like to give some intuition regarding the patterns of 1/0s seen in our dataset of real-world 802.15.4 links. In Figure 3.6, we plot the mean and median run lengths of 1's and 0's of intermediate-quality links as a function of link PRR. We observe that as link PRR increases, the mean and median run lengths of 1's increases and that of 0's decreases. This observation that increase in the length of 1's and a corresponding decrease in 0's as link PRR increases forms the basis of our approach to create models for links for which we lack input data to create the corresponding M&M model.

3.5.1.2 Methodology

For enabling simulation of links in WSNs, we create a library of K M&M models $p_1(\mathbf{X}), \ldots, p_K(\mathbf{X})$, where $\mathbf{X}$ represents a binary sequence, and each p_k is the distribution for the kth M&M model, each estimated as described earlier for a link with a different average reception rate ρ_k. We propose the following approach wherein the emission distribution parameters of existing models are modified to come up with a model of the target average PRR. Our approach selects a reference model (say, the one with closest PRR) and changes its emission probability



(a) (b)

Figure 3.6: Variation of median and mean run lengths of 1's (RL1) and 0's (RL0) of empirical 802.15.4 links with PRRs ranging from 10% to 90%. Each data-point in the figure shows the median and mean run length for a specific empirical 802.15.4 link.

parameters to match the desired PRR. This is simply done by incrementing or decrementing all the p Bernoulli parameters of the reference model by a constant, so that the resulting average PRR equals the target.

To modify the MMB emission distribution of a particular state of a pre-existing M&M model to a target PRR, we have the following approach: Given a desired PRR $\rho \in (0,1)$, we allow each p_{mw} of the reference model to shift by an amount $\alpha \in \mathbb{R}$ with the goal that the model's mean equals ρ (or gets as close as possible to it) while satisfying the constraint that each $p_{mw} \in [0,1]$. The mean of an MMB with M components and mixing proportions and prototypes $\{\pi_m, \mathbf{p}_m\}_{m=1}^{M}$ is $\mathbf{p} = \sum_{m=1}^{M} \pi_m \mathbf{p}_m$. The MMB's PRR is the mean of an infinitely long binary sequence $b_1, b_2, \ldots$ sampled from the MMB:

$$\text{PRR} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} b_t = \lim_{N\to\infty} \frac{1}{NW} \sum_{n=1}^{N} \sum_{w=1}^{W} x_{nw} = \frac{1}{W} \sum_{w=1}^{W} p_w = \frac{1}{W} \mathbf{1}^T \mathbf{p} = \frac{1}{W} \sum_{m=1}^{M} \pi_m \mathbf{1}^T \mathbf{p}_m$$

where each $\mathbf{x}_n$ is a vector of W bits and $\mathbf{1}$ is a column vector of ones. Thus

the MMB's PRR is the mean across dimensions of the MMB's mean vector. The shifted model has $p'_{mw} = p_{mw} + \alpha$ for $m = 1\ldots, M,\ w = 1, \ldots, W$, or $\mathbf{p}'_m = \mathbf{p}_m + \alpha\mathbf{1}$ for $m = 1\ldots, M$, so its mean is $\mathbf{p}' = \sum_{m=1}^{M} \pi_m(\mathbf{p}_m + \alpha\mathbf{1}) = \mathbf{p} + \alpha\mathbf{1}$ and its PRR is $\mathrm{PRR}' = \mathrm{PRR} + \alpha$, which should best match ρ under the constraints $p'_{mw} \in [0, 1]$ for $m = 1\ldots, M,\ w = 1, \ldots, W$. We then have the following constrained optimization problem:

$$\min_{\alpha \in \mathbb{R}} (\rho - \mathrm{PRR} - \alpha)^2 \text{ s.t. } 0 \leq p_{mw} + \alpha \leq 1,\ m = 1\ldots, M,\ w = 1, \ldots, W.$$

This is a convex quadratic programming problem in one variable. The constraints simplify to $-p_{\min} \leq \alpha \leq 1 - p_{\max}$ with $p_{\max} = \max_{m,w} p_{mw}$ and $p_{\min} = \min_{m,w} p_{mw}$. The solution is then

$$\alpha = \begin{cases} \min(\rho - \mathrm{PRR}, 1 - p_{\max}), & \mathrm{PRR} \leq \rho \\ \max(\rho - \mathrm{PRR}, -p_{\min}), & \mathrm{PRR} > \rho. \end{cases}$$

This is a very restrictive adaptation approach because: (1) the magnitude of α is limited by how close the smallest or largest p value in any component is to 0 or 1, respectively; and (2) since the solution is typically at the boundary of the feasible set, one (or more) p values will be 0 or 1, so they will never generate a 1 or a 0, respectively.

3.5.2 Model Adaptation with Adaptation Data

In model adaptation, if it is possible to collect some adaptation data from the target link, there are methods which allow for construction of new models which capture the short-term characteristics of the target link and not just its PRR. In the rest of the chapter, we assume that it is possible to collect little adaptation data from target link for model adaptation purposes.

In the M&M model, one first clusters the training set into separate subsets

roughly corresponding to different PRRs, and each cluster corresponding to one state of the HMM. Then a different MMB is learned separately for each state. Thus, we focus here on adapting not the entire M&M model but on adapting a single MMB. We assume that the data corresponding to this particular MMB has been selected ahead of time.

3.5.2.1 Motivation

In the MNIST dataset, the difference between the reference and target MMB models was caused by across-the-board changes in pixel intensity for each component, which pushes up or down all the Bernoulli parameters. Suppose data traces are comprised of packets being transmitted at the rate of 64 packets/second, then by using a window $W = 128$ each component of the MMB distribution captures correlations between packets over a period of 2s. Using a single sigmoid transformation as in case of the MNIST dataset would imply the underlying assumption that the PRR either increases or decreases uniformly for each component. However, in case of wireless links, empirical studies have observed link correlation at smaller time-scales. Aguayo et al. [2] observed that bursty links show correlation out to at least 1 second. In the measurement study of link burstiness, Srinivasan et al. [91] have observed that waiting for 500ms breaks the packet loss correlation. While studying the self-similarity property of links, Rusak et al. [86] have observed that links start displaying self-similarity (correlations) after 640ms. In the design of STLE [7, 3] set the threshold for identifying an intermediate link to 3 packets when the IPI was set to 250ms. This design choice has the underlying assumption that intermediate links show stable short-term behavior over a period of >750ms.

Therefore, for wireless link adaptation, a generalized model adaptation ap-

proach is required to capture the complex short-term correlations in wireless links that occur over smaller time intervals.

3.5.2.2 Methodology

To capture the complex correlations at sub-second time scales, we propose tying (K) consecutive groups of reference MMB parameters within each reference MMB component. Each group is of length W/K and uses a different sigmoid transformation. Having multiple transformations acting on smaller groups of reference MMB parameters allows for independent variations between these groups. The added flexibility with more parameters hopes to facilitate better modeling of short-term correlations. The adapted model parameters ($\widetilde{p}_{mw}$) can be expressed as:

$$\widetilde{p}_{mw} = \sigma(p_{mw}; a_{mi}, b_{mi}) = \frac{1}{1 + e^{-(a_{mi}p_{mw} + b_{mi})}}$$
$$w = (i-1)W/K + 1, \ldots, iW/K, \ i = 1, \ldots, K.$$

For each component, we have a vector $\mathbf{a}_m = a_{m1}, \ldots, a_{mK}$ and $\mathbf{b}_m = b_{m1}, \ldots, b_{mK}$ of sigmoid transformation parameters. Our approach establishes a continuum between retraining (all parameters are free, $K = W$, like in the M&M model) and the adaptation approach from Section 3.4 (all prototype dimensions are transformed with the same sigmoid, just 2 free parameters, $K = 1$). The mixture proportions, one per component, are free during adaptation subject to the constraint that they sum to 1. Our approach with multiple sigmoids has $M(2K+1)$ free parameters (mixing proportions and sigmoid parameters). This is still less than the $M(W+1)$ parameters that are to be estimated using retraining because $K \ll W$. We estimate these parameters using a generalized EM algorithm as shown in Section 3.5.2.4. Similar local transformation has been used in [77, 26]

for adapting tongue shape models, as opposed to global transformations in [76].

3.5.2.3 Approach Illustration

The flexibility of our approach is easily seen with the help of an illustration as follows: consider MMB distributions with $W = 4$ and $M = 1$. Suppose the reference model ($\mathbf{p}_R$) parameters are:

$$\mathbf{p}_R = [.1\ .9\ .1\ .9]$$

and the adaptation data for the target link comes from a target distribution $\mathbf{p}_T$ as follows:

$$\mathbf{p}_T = [.9\ .1\ .1\ .9]$$

Then, by adapting the MMB using one sigmoid transformation per component ($a_m = 0, b_m = 0$) , we reach a bad local optima for the adapted MMB model ($\mathbf{P}_{K=1}$):

$$\widetilde{\mathbf{p}}_{K=1} = [.5\ .5\ .5\ .5]$$

Now, by applying one sigmoid ($a_{m1} = -6, b_{m1} = 3$) to the first two Bernoulli parameters and a second sigmoid ($a_{m1} = 6, b_{m1} = -3$) to the remaining parameters within $\mathbf{p}_R$, we obtain a new adapted model ($\widetilde{\mathbf{p}}_{K=2}$):

$$\widetilde{\mathbf{p}}_{K=2} = [0.92\ 0.08\ 0.08\ 0.92]$$

We can clearly see that $\widetilde{\mathbf{p}}_{K=2}$ is a better estimate of $\mathbf{p}_T$ compared to $\widetilde{\mathbf{p}}_{K=1}$, demonstrating the greater flexibility when using multiple sigmoid transformations to the reference MMB parameters.

3.5.2.4 GEM algorithm for adaptation with multiple sigmoids

The new objective function is the log-likelihood of the adaptation data given the constrained MMB model with $M(2K+1)-1$ free parameters:

$$L\left(\{\widetilde{\pi}_m, \mathbf{a}_m, \mathbf{b}_m\}_{m=1}^{M}\right) = \sum_{n=1}^{N} \log \sum_{m=1}^{M} \widetilde{\pi}_m p(\mathbf{x}_n; \mathbf{a}_m, \mathbf{b}_m) \tag{3.2}$$

where $p(\mathbf{x}_n; \mathbf{a}_m, \mathbf{b}_m)$ is a multivariate Bernoulli with $\widetilde{\mathbf{p}}_m = \sigma(\mathbf{p}; \mathbf{a}_m, \mathbf{b}_m)$ ($\mathbf{a}_m = a_{m1}, \ldots, a_{mK}$ and $\mathbf{b}_m = b_{m1}, \ldots, b_{mK}$). Similar to Section 3.4.1, we provide a generalized expectation-maximization (EM) algorithm to estimate parameters.

E step: Computes $r_{mn}^{\tau} = p(m|\mathbf{x}_n; \widetilde{\pi}_m^{\tau}, \mathbf{a}_m^{\tau}, \mathbf{b}_m^{\tau})$, the posterior probability of component m given data point $\mathbf{x}_n$ and the parameter estimates in iteration τ:

$$r_{mn}^{\tau} = \frac{\widetilde{\pi}_m \prod_{w=1}^{W} (\widetilde{p}_{mw}^{\tau})^{x_{nw}} (1-\widetilde{p}_{mw}^{\tau})^{1-x_{nw}}}{\sum_{m'=1}^{M} \widetilde{\pi}_{m'} \prod_{w=1}^{W} (\widetilde{p}_{m'w}^{\tau})^{x_{nw}} (1-\widetilde{p}_{m'w}^{\tau})^{1-x_{nw}}}. \tag{3.3}$$

M step: Maximize an auxiliary function, Q defined as follows:

$$\begin{aligned} Q(\{\widetilde{\pi}_m, \mathbf{a}_m, \mathbf{b}_m\}_{m=1}^{M}; \{\widetilde{\pi}_m^{\tau}, \mathbf{a}_m^{\tau}, \mathbf{b}_m^{\tau}\}_{m=1}^{M}) = \\ \sum_{n=1}^{N} \sum_{z_n=1}^{M} r_{mn}^{\tau} \log \left(p(z_n; \widetilde{\pi}_{z_n}) p(\mathbf{x}_n | z_n; \mathbf{a}_{z_n}, \mathbf{b}_{z_n})\right) \\ - \lambda(\sigma_a^2 + \sigma_b^2) \end{aligned} \tag{3.4}$$

The first term is the expected complete-data log-likelihood, over π_m and $\{\mathbf{a}_{mi}, \mathbf{b}_{mi}\}_{i=1}^{K}$ for each component. $1 \leq z_n \leq M$ is the (unknown) index of the mixture component that generated data point $\mathbf{x}_n$. The second term involving λ is for regularization. It penalizes Q to prevent $\mathbf{a}_m$ and $\mathbf{b}_m$ within each component from approaching very high values. Choice of λ will be evaluated in Section 3.6.4. The

mixing proportions are updated as below:

$$\widetilde{\pi}_m^{\tau+1} = \frac{1}{N}\sum_{n=1}^{N} r_{mn}^{\tau} \tag{3.5}$$

The gradient of Q w.r.t. a_{mi}, b_{mi} ($i \in \{1, \ldots, K\}$) is:

$$\begin{aligned}
\frac{\partial Q}{\partial a_{mi}} &= \sum_{n=1}^{N} r_{mn}^{\tau} \sum_{w=(i-1)\frac{W}{K}+1}^{\frac{iW}{K}} p_{mw}(x_{nw} - \widetilde{p}_{mw}) + \frac{2\lambda(a_{mi} - E(\mathbf{a}_m))}{K} \\
\frac{\partial Q}{\partial b_{mi}} &= \sum_{n=1}^{N} r_{mn}^{\tau} \sum_{w=(i-1)\frac{W}{K}+1}^{\frac{iW}{K}} (x_{nw} - \widetilde{p}_{mw}) + \frac{2\lambda(b_{mi} - E(\mathbf{b}_m))}{K}
\end{aligned} \tag{3.6}$$

where we solve for $a_{mi}^{\tau+1}, b_{mi}^{\tau+1}$ using BFGS.

Alternate Initialization We initialize the parameters of the multiple sigmoids as follows:

(1) Separate the adaptation data into M clusters using k-means clustering.

(2) Assign the reference MMB p-values to nearest-neighbor k-means centroids in a 1-to-1 correspondence.

(3) For each component separately, find the sigmoid that best maps the $\frac{iW}{K}, \ldots, \frac{iW}{K}$ reference MMB parameters to corresponding centroids (K sigmoids per component) by solving a nonlinear least squares problem.

For the corresponding retrained model, we use the k-means centroids as initialization.

3.6 Results

3.6.1 Data Collection

We collected comprehensive packet reception traces using 802.15.4 complaint CC2420 radios in different environments, transmission power levels and inter-

ference conditions. Table 3.2 lists the details of our data collection. In addition, we used an 802.11 dataset submitted to the CRAWDAD repository [45] to test our approach under different packet sizes and transmission power that varies significantly from 802.15.4 radios. In all cases, we have single transmitter and multiple receivers logging the packet reception traces.

Testbed Testbed	Radio Radio	Interference Interference	Num. of Traces	Duration	CC2420 Tx power
Indoor	802.15.4/Ch. 26	N	10	1 hour	-11dBm
Motelab	802.15.4/Ch. 26	N	26	1 hour	0dBm
Outdoor	802.15.4/Ch. 26	N	36	1 hour	0dBm
Motelab	802.15.4/Ch. 11	Y	10	1 hour	0dBm
Indriya [19]	802.15.4/Ch. 11	Y	27	1 hour	0dBm
NIIT	802.11b	Y	2	4–5 hours	-

Table 3.2: Summary of different intermediate-quality ($10\% < PRR < 90\%$) experimental data traces used for constructing adapted models. The 802.15,.4 traces are comprised of 230,400 packets, whereas the 802.11 trace is comprised of 960,000 packets in all.

3.6.2 Methodology

We estimate the parameters of the transition matrix ($Q = 2$) for all links in Table 3.2 by following the training procedure described for the M&M model. Next, for each target link, we convert the binary input sequence into vectors of length $W = 128$. We term these binary sequences as *adaptation vectors* or *adaptation data*. The sequences are separated into Q clusters, wherein each cluster has sequences with similar PRR values, using the Viterbi algorithm [27].

Each cluster corresponds to a L1 state of the transition matrix. The adaptation data for each state are divided into training set (used for estimating the adapted and retrained model) and validation sets (for evaluating the model LL) with a 70-30 split.

Reference model: We train a unique reference MMB model using the entire adaptation data from an L1 state of one of the links from the Indoor dataset. We use this unique *single* link reference model as the starting point for constructing our adapted models for *all* target links.

Target link models:All other experimental data traces from Indoor, MoteLab, Indriya, Outdoor and NIIT test-beds are treated as target links. We use different subsets of adaptation data from the training set, to create MMB models using the adaptation and retraining approaches corresponding to each L1 state of the target link.

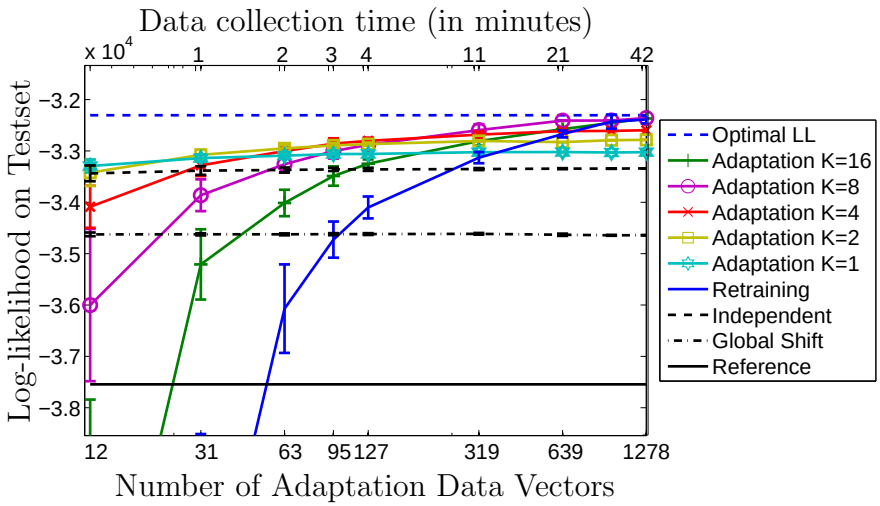
In addition, we also compare with these other models:

Independent model: It is constructed for each state of the training set link with $p = PRR(AdaptData)$.

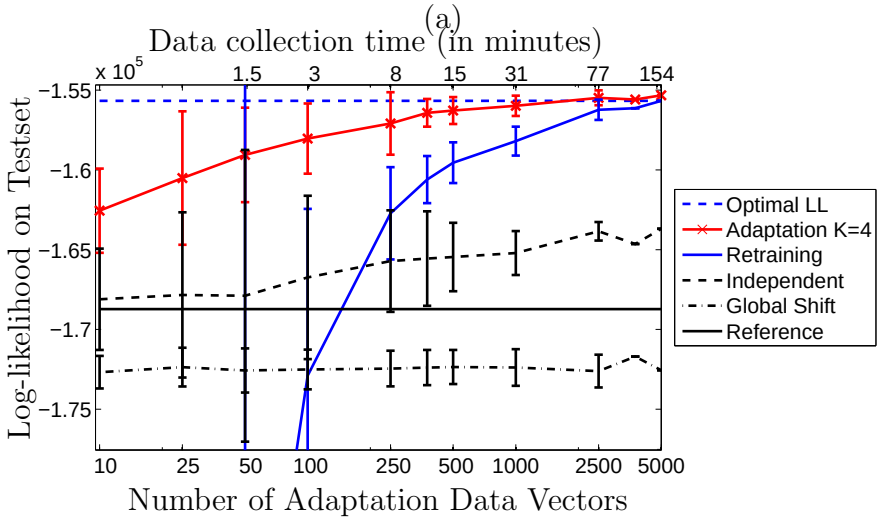
Global Shift Model: It is created by modifying the Bernoulli parameters of the reference MMB (all parameters are tied together) to match the PRR of the adaptation data [42]. The problem with this approach is the same as with the linear transformation. Once any of the p-values reach 1/0, it prevents less extreme values from adapting.

For each of the Q L1 states in the target links, we compare testset LL of the adapted, retrained, independent and global shift MMB models in Section 3.6.4. We use the L1 transition matrix along-with the different MMB models to simulate traces to compare PRR, run length distributions and NND in Section 3.6.4. In Section 3.6.6, the same simulation traces will be used for comparing perfor-

mance, with respect to experimental traces, in collection tree protocol (CTP) [31] simulations.



(a)



(b)

Figure 3.7: Log-likelihood (LL) when adapting a reference MMB from a 802.15.4 reference link (packet = 28 bytes, Tx power = -11dBm) from Indoor testbed using adaptation data from (a) 802.15.4 target link from Motelab (packet = 28 bytes, tx power = 0dBm) and (b) 802.11 target link from a NIIT (1000 byte payload, with interference). In most cases, errorbars were computed over 50 random subsets of adaptation data.

3.6.3 Performance Metrics

Log-likelihood (LL), quantifies the ability of the method to generalize beyond the training set. Our objective function when estimating the parameters of the adapted model is the LL. It is a direct measure of the performance of our model adaptation approach.

PRR, behavior of a link over long time-scales.

Run length distributions of 1's and 0's, quantify the bursty behavior of intermediate links.

Nearest Neighbor Distance (NND) was proposed for highlighting the effect of the absence of rare cases (long runs of 1/0's) and that of minor differences between common cases from the different models.

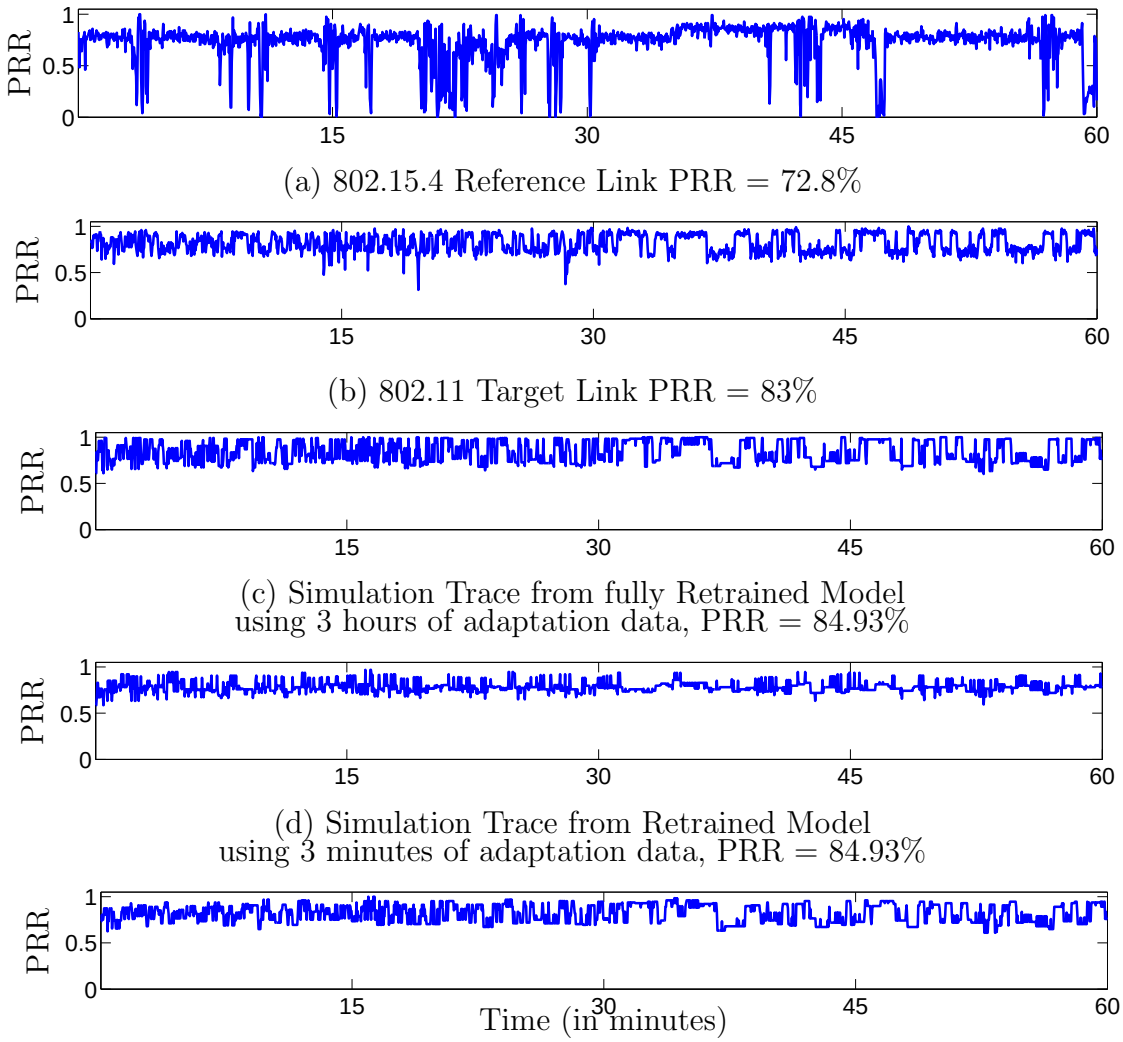
PRR, run length and NND are indirect measures as we compute them from simulation traces generated by our models. .

3.6.4 Model Comparisons

Figure 3.7 shows the variation in LL of the retrained and the adapted MMB models, for target links from two different testbeds, packet sizes and transmission power, as a function of the amount of adaptation data used for estimating the model parameters. Note that this figure is representative of behavior seen across all our target links when we vary K and the amount of adaptation data. The LL of the adapted models is significantly better than that of the corresponding retrained model when the number of adaptation vectors is small. As we increase the amount of adaptation data used for training, the LL of the adapted model approaches the optimal LL. The "optimal LL" is the LL of the fully retrained model constructed using 100% (all available) adaptation data. The retrained

model converges to the optimal as more adaptation data is used, but it needs much more data to achieve a comparable LL to the adapted model; and its performance is more variable (large error bars). For small amounts of adaptation data, the retrained model parameters converge to a local optima and, hence, it performs worse on the test set than even the reference model. Given the same amount of adaptation data, the adapted model parameters generalize better than the corresponding retrained model.

In Figure 3.7(a), we adapt the reference model to construct an MMB model for 802.15.4 target link from MoteLab. We observe that adaptation with $K = 1$ stagnates because the transformation is very restrictive since it is tying $W = 128$ parameters. Consequently, it is not flexible enough to model the variations observed in intermediate quality target link data. With $K > 1$, the model has more parameters, and hence greater flexibility to model. One has to be careful about adding many more transformations (i.e. tying fewer parameters) which can cause over-fitting. At the same time the transformations should be capable of changing small groups of reference MMB parameters to model the short term correlations that occur at sub-second time scales. $K = 1$ affects all Bernoulli parameters within an interval of 2s ($W/K = 128$). Higher K's reduce this interval by tying smaller groups of parameters. In [91, 86], it has been shown that these short-term correlations change after 500-640ms. This means that we need to tie groups of parameters that affect link quality over similar time intervals. Any $K \geq 4$ satisfies this requirement. However, with too many sigmoids (e.g. $K = 8, 16$), the adaptation approach performs worse on the test set with less adaptation data ($< 10\%$). Typically, adaptation approaches are attractive when they do not require extensive data collection. So, $K > 4$ are not feasible as they require more adaptation data to come close to optimal LL. Therefore, from the LL results and other factors such as data collection and short-term correlation



(a) 802.15.4 Reference Link PRR = 72.8%

(b) 802.11 Target Link PRR = 83%

(c) Simulation Trace from fully Retrained Model
using 3 hours of adaptation data, PRR = 84.93%

(d) Simulation Trace from Retrained Model
using 3 minutes of adaptation data, PRR = 84.93%

(e) Simulation Trace from Adaptation Model
using 3 minutes of adaptation data, PRR = 84.93%

Figure 3.8: PRR variation of reference 802.15.4 link, target 802.11 link, and simulation traces from retrained and adapted models.

interval requirement [91, 86], we determined the optimal K to be 4. For the MMB models shown in Figure 3.7(a), with $K = 4$, and ≥ 45 adaptation data vectors, the adapted MMB model does better, in terms of LL, than the one with $K = 1$. For simulating traces and comparing models in terms of PRR, run length distribution and NND, we use these values ($K = 4$ and 45 adaptation data vectors) for constructing adapted MMB models for each L1 state for all target 802.15.4 links. So, adapted models require 90 adaptation data vectors or 3 minutes of target link data to model it with $Q = 2$ L1 states. Figure 3.7(a) also shows the LL of the independent, reference and global shift MMB distribution. It is worse (to varying degree) as compared to adaptation.

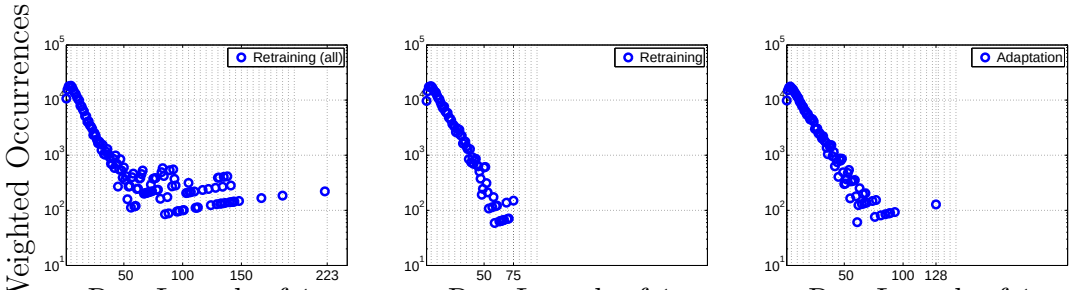
In Figure 3.7(b), we adapt the reference model to construct an MMB model for 802.11b target link from NIIT testbed. Using greater than 50 (0.1%) adaptation vectors (and $< 10\%$ data), the LL of the adapted model with $K = 4$ is greater than the corresponding retrained, global shift, reference and independent models. We choose the adapted models trained with 100 adaptation vectors (corresponds to $\approx$3 minutes of data from entire link) to compare with the different retrained models in Figure 3.8. Figure 3.8 demonstrates that qualitatively simulation traces from the adapted model using 3 minutes of available data match closely to the target trace and simulated traces from a fully retrained model.

Table 3.3 shows NND and the maximum run lengths of 0's and 1's for simulated traces from the adapted, retrained, global shift and independent models, and the target link in Figure 3.8. Since the link has high PRR, it does not have extremely long runs of 0's. Hence, all models perform similarly, when modeling run length distributions of 0's. For run lengths of 1's (see Figure 3.9), the model created using the adaptation technique and 2% data has longest run of 129$\pm$21 1's. This is close to the model created using retraining with 100% data, which

has longest run of 218±38 1's. Thus, the adapted model run length distribution of 1's is closer to the fully retrained model. This is also reflected in the lower values of the NND for the adapted model.

Metric	Target	Fully Retrained (3min)	Adapted (3min)	Retrained (3min)	Global Shift (3min)	Independent
PRR	0.8301	0.8169	0.8220	0.8223	0.7950	0.8302
NND	-	0.2256	0.5479	0.7424	0.9856	1.0793
RL0	12	9±1	9±1	8±0	8±0	6±0
RL1	376	218±38	129±21	74±8	66±7	64±10

Table 3.3: Statistics of target trace and simulated trace from other models. Traces from adapted MMB models are able to capture the long runs of 1 like the original M&M (retrained with all data) model.



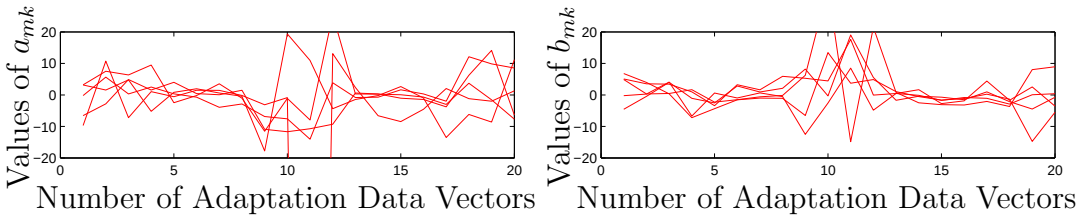
(a) Retraining (M&M Model, all data)

(b) Retraining 3min

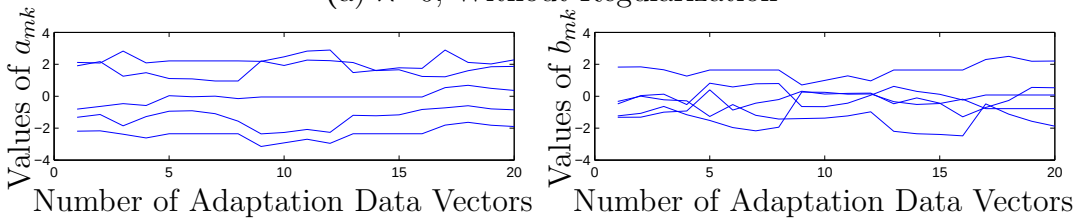
(c) Adaptation 3min

Figure 3.9: Run Length distribution of 1's from different models for target link from Figure 3.8 and Table 3.3.

Figure 3.10 summarizes the effect of the regularization term. When regularization term is absent ($\lambda = 0$), a_{mk}, b_{mk} can take high positive/negative values for adapted MMB models constructed with ≥ 45 adaptation vectors (see Figure 3.10(a)). This implies that the sigmoids are saturating so as to make all the



(a) λ=0, Without Regularization



(b) λ=100, With Regularization

Figure 3.10: Variation in values of a_{mk} and b_mk ($M = 5$ and $K = 4$,i.e., 20 a_{mk} and 20 b_{mk}) for a representative adapted model constructed with 45 adaptation vectors. Sequences of a_{mk} and b_mk for multiple adaptation datasets are overlapped in the figure. Without regularization ($\lambda = 0$), the values of a_{mk}, b_{mk} change quite drastically. Note that in (a) the y-axes in truncated between -20 and 20. With regularization ($\lambda = 100$), the values vary much less.

adapted model p_{mw}'s close to 1/0. This results in overfitting. In contrast, when using our proposed model adaptation technique with all available (100%) adaptation data, a_{mk}, b_{mk} do not take high positive/negative values. Figure 3.10(b) shows that by using $\lambda = 100$, we can prevent a_{mk}, b_{mk} from taking high values by imposing a strong penalty on the model LL.

This demonstrates that the proposed model adaptation procedure can perform similar to a fully retrained model, given a well-trained reference model and little adaptation data.

3.6.5 Model Generalization

In this section, we show that adapted link models constructed using adaptation data from target link in one environment generalize well to target links in a different but similar environment. We grouped our target links into two sets. In the first set, we compare adapted models of Motelab (environment 1) links with models (retrained, adapted and independent) of Indriya (environment 2) links, with both sets of target links from 802.15.4-ch. 11 i.e. prone to interference from 802.11 access points. In the second set, we compare adapted models of Indoor (environment 1) links with models of Motelab (environment 2) links, with both sets of target links from 802.15.4-ch. 26 i.e. unaffected by interference but having different transmission power levels. In both sets, we only compare links which differ in PRR by 1%. We computed the model LL of adapted model (5% data) from environment 1 (A^1) w.r.t. the test set of link from environment 2. The competing models for A^1 are the adapted model A^2, retrained model R^2, independent model I^2 and the model with optimal LL, namely the fully retrained model O^2.

Figure 3.11(a) shows the relative difference in LL w.r.t. optimal LL (O^2) of adapted models of wireless links from environment 1 (A^1) to other models (R^2, I^2, A^2) of link having similar PRR ($\|PRR_1 - PRR_2\| = 1\%$) in environment 2, across all links (with interference: MoteLab–Indriya, and without interference: Indoor–Motelab). We can see that A^1 models performs close to A^2 (in terms of LL), which was estimated using adaptation data of link from environment 2. Other models fare worse in comparison. Figure 3.11(b) shows the distribution of links when comparing links for the two sets. We do well across links with different PRRs showing that our adapted model generalize well (lower relative difference w.r.t. optimal) across a wide variety of link PRRs. Our results imply

that wireless model behavior is quite close at different places with similar environmental conditions. This observation is analogous to using the same constant values accounting for path loss

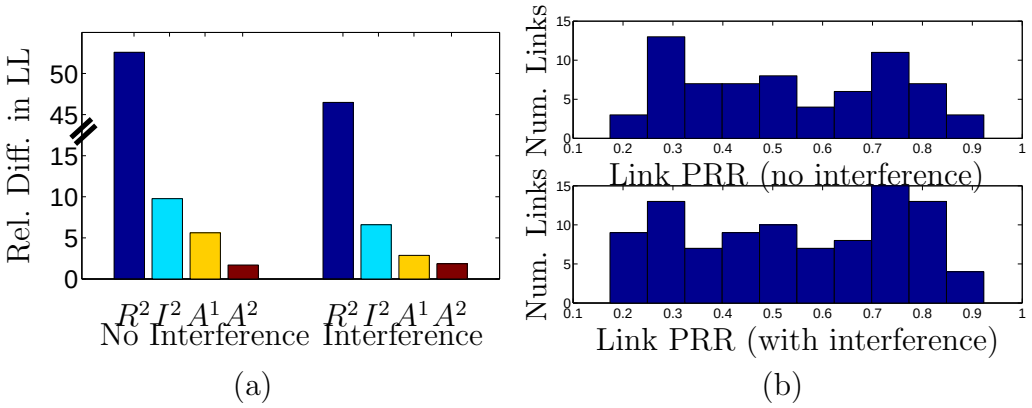


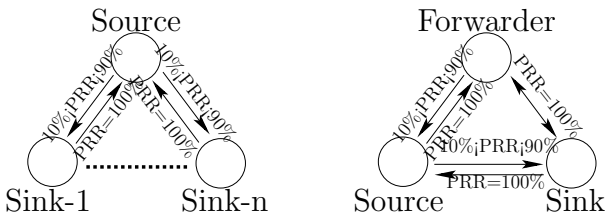
Figure 3.11: (a) Overall average relative difference in log-likelihood (LL) w.r.t. optimal LL (O^2) of adapted models of wireless links from environment 1 (A^1) to other models (R^2, I^2, A^2) of link having similar PRR in environment 2. (Note that y-axes is broken between 15% and 45%.) (b) Link PRR distribution when generalizing adapted models from environment 1 to environment 2 under different interference conditions.

in certain types of environments in RF studies [80]. Going back our anecdote at the beginning of Chapter 3, where a researcher from UC-Berkeley has an opportunity to go to the Amazon rain forest to conduct habitat monitoring experiments using sensor networks. Then from the above result, the researcher can simulate similar conditions to the Amazon rain forest by gathering a little data using battery-operated sensors in on-board memory locally at Muir Woods and our proposed model adaptation approach. In this way, the researcher does not need to collect traces in the Amazon rain forest (deployment site) to get a good idea of link dynamics.

3.6.6 Model Evaluation on Higher Level Protocols

Till this point, we have evaluated the behavior of adapted models, constructed using our proposed approach, using metrics such as model LL, PRR behavior and run length distributions. In this section, we evaluate the performance of the adapted models with respect to CTP [31], a well-known collection protocol in wireless sensor networks. CTP has been shown to provide high packet delivery rates and stable routing topologies. This can be attributed to its use of mostly good quality links (PRR>90%) for routing packets. One shortcoming of CTP is that it is unable to realize short-term changes in link quality, i.e., it ignores intermediate-quality links with bursty behavior. For example: if there are periods of PRR>90% in an otherwise intermediate quality link that has potential for lower delivery cost, then CTP is unable to take advantage of such a link as its link estimation metric (ETX) updates slowly. To overcome this, approaches such as STLE [3] has been proposed. STLE enables CTP to route packets over long-range bursty links by temporarily changing the parent node over to the intermediate link. As long as the intermediate link stays in a period of high PRR (>90%), CTP uses this temporary parent with lower path cost. CTP switches back to the original parent after the first packet loss over this intermediate link. In our evaluation, we use CTP with STLE enabled as it allows for swift route changes to take advantage of bursty, intermediate links and better represent network behavior under link dynamics.

We implemented CTP with STLE in a MATLAB framework for ease of simulation using experimental data traces collected under different real conditions. We experimented with a one-hop topology with multiple sinks and a simple multi-hop topology, as shown in Figure 3.12. Note that in the case of a one-hop topology, we can gather experimental traces (ground truth) from a single transmitter at high



(a) Single-hop topology (b) Multi-hop topology

Figure 3.12: Topologies used for simulating CTP with STLE ($n \in \{2, 3\}$)).

data rate and multiple receivers concurrently. However, in the case of multihop, we cannot evaluate the performance of our approach with live experiments in real-world conditions because it is not possible record high frequency data traces needed for ground truth when there are multiple transmitters in the same collision domain. Our multihop ground truth experimental data was obtained from links in the network (in the same collision domain) close in time, but not concurrently. Each experiment was conducted as follows: (1) Select a single or multihop topology. (2) Select bursty, intermediate quality experimental traces as links between source and sinks according to the choice of topology. These link traces are from data in Table 3.2 and represent the ground truth in terms of experimental performance. (3) For a given inter-packet interval ($IPI \in \{15.625ms, 250ms\}$), the source node transmits data packets towards the sinks. (4) Using traces simulated from the adapted, retrained, independent and fully retrained models (O), repeat step 3. (5) Measure retransmissions and packet delivery rate in each case.

In total, we conducted 25 single-hop experiments out of which 11 experiments had intermediate links sampled from 802.15.4 channel 26, which shows minimal external interference and 14 experiments with links from channel 11, which overlaps with 802.11 channel 1. Similarly, we conducted 17 multi-hop experiments (6: without interference and 11: with interference). Each experiment was repeated 5 times. Table 3.4 shows the relative difference (as a percentage) of each model's

IPI (in ms)	Inter-ference	Metric	O	A	R	I
SINGLE-HOP SIMULATIONS						
15	NO	DR	4.0	2.6	3.2	13.5
15	NO	RT	9.2	11.7	13.6	26.0
15	YES	DR	2.2	2.5	2.7	11.1
15	YES	RT	15.8	17.4	21.1	30.2
250	NO	DR	5.1	3.0	4.7	12.0
250	NO	RT	6.9	11.4	16.2	26.6
250	YES	DR	2.7	3.4	4.6	8.5
250	YES	RT	8.1	11.7	15.5	21.0
MULTI-HOP SIMULATIONS						
15	NO	RT	6.29	7.13	10.59	24.31
15	YES	RT	8.60	13.78	17.32	24.12
250	NO	RT	5.48	7.85	8.03	18.32
250	YES	RT	11.40	14.91	18.86	26.91

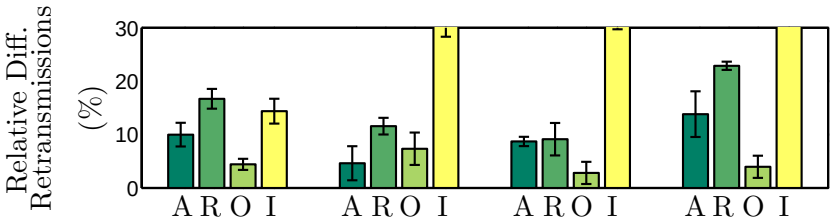
Table 3.4: Relative difference (as %) in overall delivery rate (DR) and retransmission (RT) of simulation model traces in single and multi-hop CTP-STLE simulations w.r.t. experimental intermediate-quality links (ground truth).

performance w.r.t. ground truth for delivery rate and retransmissions averaged across all experiments. Note that A = Adapted Model; R = corresponding Retrained Model; I = corresponding Independent Model and O = Fully Retrained Model estimated using 100% adaptation data.

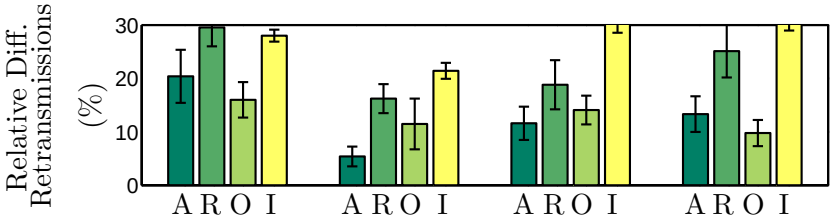
3.6.6.1 Retransmissions

Retransmissions are a direct measure of a node's energy consumption (lower is better). Averaged across all cases by topology, inter-packet intervals (IPI) and interference conditions, retransmission for $O < A < R < I$. Retransmissions were consistently lower for adapted models compared to retrained and independent models. This can be attributed to our adapted models generalizing to burst patterns similar to the performance of a fully retrained model. This result is significant from a simulation point-of-view as a network simulated with link models created using our adaptation approach mimics closely the behavior of a network simulated with link models constructed with at least 20 times more data. The retransmissions performance of is closer to fully retrained model and in turn to the experimental traces. This makes our adaptation approach appealing to users who want to introduce more realism in the communication aspect of their sensor network simulations by collecting a little adaptation data.

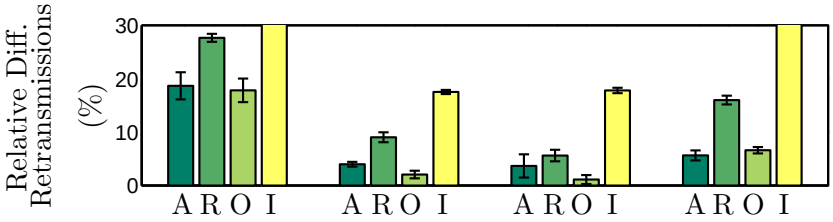
Figures 3.13 and 3.14 shows the relative difference (as %) of different model's performance w.r.t. experimental traces (ground truth) for particular cases. In certain cases, we observe that the fully retrained and adapted models have significantly higher retransmissions compared to the ground truth. This is explained as follows: the underlying links in these experiment showed bursty behavior resulting in more than 2 L1 states when the traces were manually inspected. However, our simulation models were restricted to 2 L1 states. Therefore, the parameters of the fully retrained MMB model averaged out the bursty behavior seen in the adaptation data vectors. For the adapted models, the small amounts of adaptation data did not contain some of the bursty behavior seen in the experimental traces and hence were unable to account for it in simulation. Despite such cases, in most experimental links in our data-sets we observed only 2 L1 states.



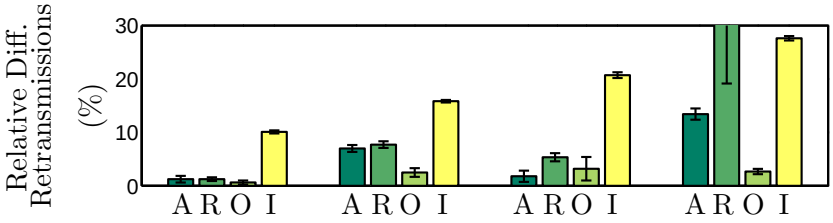
(a) Cases WITHOUT Interference IPI=250ms



(b) Cases WITH Interference IPI=250ms

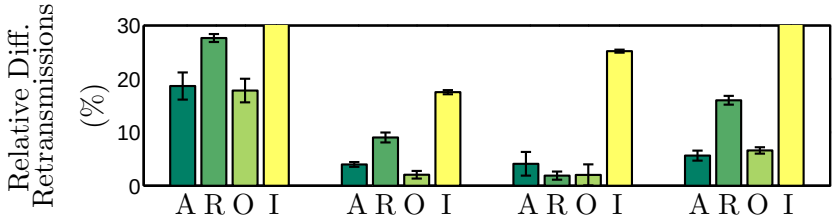


(c) Cases WITHOUT Interference IPI≈15ms

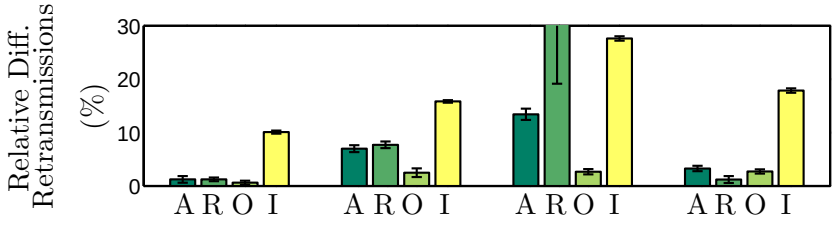


(d) Cases WITH Interference IPI≈15ms

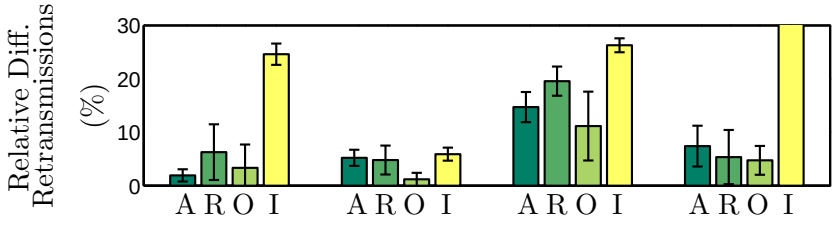
Figure 3.13: Single-hop CTP-STLE simulations. Relative difference (as %) in retransmission of simulation model traces w.r.t. experimental intermediate-quality links (ground truth). Note: A=Adapted Model; R=Retrained Model; O=Fully Retrained Model; I=Independent Model. Adapted model is similar to fully retrained model which has optimal performance in most cases. Errorbars computed over multiple simulation traces for each model.



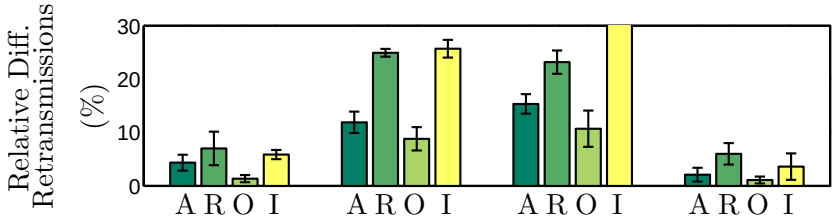
(a) Cases WITHOUT Interference IPI≈15ms



(b) Cases WITH Interference IPI≈15ms



(c) Cases WITHOUT Interference IPI=250ms



(d) Cases WITH Interference IPI=250ms

Figure 3.14: Comparison of relative difference in retransmissions for multi-hop topology CTP-STLE simulations. Legend is same as for Figure 3.13.

3.6.6.2 Delivery Rate

A, R and O models showed similar performance in terms of overall packet delivery rate, while independent model was last in all cases (see Table 3.4). This result was not surprising given the fact that independent link models cannot account for the bursty behavior as produced by better models (A,R,O) of intermediate links. A, R and O perform similarly as each model can account for the bursty behavior of intermediate-quality links, even though we have seen (Figure 3.7) that R models generalize poorly in terms of log-likelihood.

3.7 Discussion

3.7.1 How much data should a user collect

When collecting traces, one would like to know if the collected data is enough for a high-quality model. To determine this, we analyzed *all* the links from the 802.15.4 datasets (1 hour duration). We found the average time duration that a link stays in an L1 states is 254 seconds (approx. 4 minutes), and the average number of transitions between states was 41. Also, 90% of the time if a user collects data for 150 seconds (or 75 vectors in all), s/he will have at a minimum 5% adaptation data for at least a single MMB (L1–state). This way, one can collect data corresponding to each different L1 regimes in under 3 minutes. If s/he wants to gather data for more states, then the sampling time needs to be greater to collect the 5% data for multiple PPR regime. On a average, it takes 540 seconds to collect data for adaptation for multiple PRR regimes. Note that this is the time required for collecting data. It is not the amount of data vectors expressed as a function of time that our adaptation approach needs to construct a model (3 minutes).

3.7.2 How close does the reference model need to be to the target data

We have already established experimentally that adapting few parameters (each sigmoids' a and b) far outperforms retraining the entire model of a wireless link. This has two different causes: (1) Having few free parameters $\{a_m, b_m\}$ reduces the risk of overfitting with a small training set. (2) The reference model contains useful information about the target link that cannot be inferred from a small adaptation dataset. To control for the effect of factor (2) and determine how good the reference model is for the target, we repeated experiments where we adapted from a random reference model (having random $\mathbf{p}$ and π values). We find that adapting from a random model does consistently worse than from the reference link model, but, surprisingly, by not so large difference; and that this difference decreases as K increases. Upon further inspection, for the links used in this study, the p-values of the optimal model (trained using all the adaptation data) vary relatively little within each component (eg $0.25 \leq p_{mw} \leq 0.35$ for $w = 1, \ldots, W$ within a certain component m). A sigmoid that tries to best map some random input p-values to these target p-values will essentially become flat and output a nearly constant value (e.g. 0.3 in the case above), and yield a relatively good adapted model (since 0.3 can't be farther than 0.05 from any p value in that component). This also explains the fact that the random model tends to do relatively better as K increases, because then each group (adapted by a different sigmoid) can have a different "constant" p-value. Thus, we suspect that the random reference model will do very badly for links having p-vectors with wider oscillations, e.g. a p-vector that alternates between 0.1 and 0.9, corresponding to a trace that alternates quasi-periodically between 0 and 1. In summary, adapting from a reference that is not so close or even random may still yield better models

than retraining if we have little adaptation data, because the implicit regularization caused by sharing parameters reduces overfitting. However, adaptation works best with a reference model that captures useful information about the target that cannot be inferred from the adaptation data. The determination of how to pick the best reference model from a library of models could be done by analyzing the models in a multidimensional space, including statistical momentums like average, std. dev., skewness, kurtosis and even entropy and finding the correct "distance" metric.

3.7.3 Why does adaptation work under different conditions

The main factor that affects correlations between successive packets is the inter-packet interval, which is required to be fixed but not the same, for data from the reference and target link. In addition, the inter-packet interval needs to be relatively close between the reference model link and the target link. This is because, in wireless communication, the conditions of the channel are correlated at short time scales [80]. Therefore, the input data is arranged in the form of binary sequences of length W of packet reception data. This input format abstracts away the characteristics of the underlying data including the type of radio, transmission power, modulation scheme, environment and packet size, among others. Once the data is transformed into a binary sequence, the adaptation technique tries to estimate the parameters of the sigmoid transformation to each component of the reference MMB distribution such that it matches the distribution of the new dataset. Therefore, it is able to adapt using data from completely different scenarios.

3.7.4 Why retraining does badly with less data

The retrained models estimated using very little data ($< 10\%$) do significantly worse than the other models. The reason is that, with few training vectors, each component becomes responsible for an even smaller subset of vectors (sometimes even just one). Imagine for simplicity that the window size was W=1 and we have just M=1 component, so we have a single Bernoulli coin. If the few tosses we observe are all 0s (the argument for all 1s is analogous) then we set p=0. But then if in the test set we happen to observe a 1, the model would assign probability 0 to it (or $-\infty$ log-likelihood). With W=128 independent coins and only a few observed tosses for each, it becomes highly likely for one coin to observe only 0s (or only 1s). With few training vectors it is hard to retrain all the parameters and get a model that will generalize well to test data (even if initializing from a relatively good reference model).

3.7.5 Computation time

Computation-wise, adaptation with 5% data takes ≈6 seconds ($K = 4$), retraining with the same data takes ≈0.1 second and retraining using 100% adaptation data requires ≈12 seconds to complete (1.66GHz CPU, 2GB RAM). The slow speed of adaptation can be attributed to the BFGS part of the generalized EM algorithm. Retraining is fast due to the closed-form solutions for the M-step of the EM algorithm [12]. Practically, the adaptation times, though slower, are acceptable and allow for quick computation of a MMB model for each PRR regime of a link.

3.7.6 Strong points of proposed wireless link adaptation

Proposed model adaptation approach can model the short term correlations observed in inter-mediate quality wireless links using 3 minutes of data. Accounting for short term correlations is important for accurately simulating wireless link conditions for protocols such as STLE [3] and 4C [65]. Our technique is agnostic of the data trace characteristics such as packet size, inter packet interval, transmit power, radio type and environment. It is only dependent on the patterns of 1's and 0's. For construction of adapted models, our reference model was the same across all target links.

While real world wireless conditions change over time-scales of hours and days, our approach is useful for researchers wanting to simulate a subset of the wireless conditions at their deployment environment by collecting adaptation data at a similar nearby location. Our approach eliminates the need for deployment of testbeds, a time-consuming endeavor.

We envision our model adaptation approach being used in virtual testbeds [17]. Such testbeds combine physical wireless links with virtual (simulated) link models. In such cases, our model adaptation approach would quickly create models that capture the short-term correlations. Long time scale changes in the link can now be accounted by measuring the link again for a short period and constructing a new updated link model to be used for simulating traces in these virtual links.

3.7.7 Weaknesses of proposed wireless link adaptation

In this study, we utilized L1 transition matrices computed from the entire target link to compare adapted MMB models with fully retrained MMB model. We simulated traces from these models by following the M&M approach. In prac-

tice, it is difficult to infer about the long time scale behavior of any link with 3 minutes of data. Simulation users are advised to use transition matrices from existing library of links. Alternately, the transition matrix can be user-defined. In addition, since our adaptation approach creates a packet loss model, it is unable to account for communication errors because of reasons such as simultaneous transmission to same receiver. However, this problem is common to all packet loss modeling approaches that abstract away from link attributes such as signal strength.

3.8 Summary

In this chapter, the goal was to enable simulation users to compute high-quality short term wireless models without extensive data collection. The proposed approach quickly adapt a reference MMB model to a new distribution given a few samples from the latter. We nonlinearly and separately transform each prototype from the reference MMB using a small number of parameters and estimate these with a generalized EM algorithm. It was shown that by utilizing a total 3 minutes of data from a target link, one can adapt a reference MMB using multiple sigmoid transformations to model target link MMB distributions. This is an order of magnitude decrease in the amount of data required to train a high quality model to capture the short term correlations of a link. Also, the model adaptation procedure can adapt to target links under widely different conditions (environments, packet size, tx. power level and radios). Additionally, our adapted model perform close to the ground truth in protocol simulations.

Similar to the wireless link modeling scenario, there are other situations in the field of WSNs where model adaptation techniques are applicable. This is because, certain sensor network applications, such as occupancy modeling in indoor

environments, require large datasets (weeks, months) to estimate the underlying model parameter values and it is difficult to design a WSN that can balance the trade-offs associated with application performance and network lifetime. In the next chapter, I propose to address this issue on two fronts: (i) on the system side: by designing a novel WSN infrastructure using cameras to address the detection performance, and (ii) on the modeling side: by applying parameter tying-based, model adaptation techniques, similar to wireless link model adaptation, to solve the data requirement problem for occupancy models.

CHAPTER 4

Adapting Data-Driven Occupancy Modeling

In the building energy management domain, studies [11, 24, 71, 23] have shown that by regulating ventilation based on occupancy data, it is possible to significantly reduce HVAC energy usage, thereby increasing energy efficiency. The core ideas are to instrument the building extensively to gather occupancy data and incorporate it in models that can be used to actuate the HVAC systems in a manner that increases the energy efficiency. However, there are some obstacles facing the sensor network community when attempting to solve the building energy management problem. One obstacle is the need for sensor network infrastructure that can gather occupancy data traces. In the past, several approaches [93, 20, 71, 84, 1] have proposed to infer occupancy data. The other obstacle is the need for large training datasets collected over long periods of time to compute reliable estimates for the numerous occupancy model parameters. Once a model is created for one building, it is not possible to directly port it to a different building due to different floor-plans, room functions and the resulting occupancy patterns. Therefore, it implies that for modeling new buildings, it is required to record occupancy traces for extended periods of time (weeks, months) using extensive sensor network infrastructure comprising of sensor nodes, data logging units, power cables, etc.

In this study, I propose to address these issues by combining machine learning approaches such as model adaptation with a novel wireless camera-based occu-

pancy sensing infrastructure. Wireless camera sensor networks have to balance the conflicting challenges imposed by the detection performance, latency and lifetime requirements in surveillance applications. While previous studies for camera sensor networks have addressed these issues separately, they have not quantified the trade-offs between these requirements. In this study, we discuss the design and implementation of SCOPES, a distributed Smart Camera Object Position Estimation System that balances the trade-offs associated with camera sensor networks. To remedy the large data requirement problem for occupancy modeling, I propose using model adaptation techniques. Using model adaptation, it is possible to use a reference building occupancy model that has been trained with extensive data and adapt it to the new building given a far smaller occupancy data trace than would be necessary to train a new model from scratch. Such occupancy datasets can be recorded using few sensors with the data being stored in the on-board memory. Also, there is no need to install cabling to power these nodes as they can survive for a single day using their battery packs. Such an inter-disciplinary approach can enable applications such as building energy auditing to make use of occupancy data to maximize the energy savings of a new building using short occupancy data traces (1-2 days).

4.1 Background and Related Work

4.1.1 Occupancy Sensing

Object detection and tracking systems depend on the underlying hardware and software infrastructure for efficient performance. The software infrastructure comprises of image processing algorithms for feature extraction and interpretation from the incoming data (images/video). In [101], design of a feature extractor is

described for an image thresholding algorithm. It involves extracting the "expressive" and "discriminating" features using a modified Hebbian learning approach. In [63], a principal feature extraction approach is proposed for classifying a pixel as belonging to the background or foreground. Han et al. [34] describe a multiple hypothesis approach for pruning the possibilities of the number and trajectories of objects in the video stream. However, these techniques are computationally intensive and find use in surveillance and monitoring applications with permanent deployments and no dearth of processing and communications capabilities. Such techniques are not suitable for implementation in resource-constrained, untethered platforms like wireless sensor networks that have to maintain a fine balance between a long-life and operational performance.

The hardware infrastructure for object detection and tracking systems ranges all the way from binary sensors to sensors delivering high quality video over 802.11 networks. Some of the networks [33] demonstrated for use in tracking movements of objects comprise of hundreds of small, densely distributed wireless sensor nodes deployed in the field. They utilize collaboration between neighbors to detect and track a moving target, and alert other sensor nodes along the projected path of the target. In [53, 54], SensEye, a multi- tier camera sensor network is described for surveillance applications. The work aimed at showing that a multi-tier network can balance the conflicting goals of latency and energy-efficiency. However, the experiments conducted in some of these works [53, 54, 33] are not representative of real-world scenarios.

Using a binary sensor network, Aslam et al. [5] describe a framework for tracking a moving object by using a particle-filter algorithm. However, these sensors do not provide enough information to localize the object. In [32], VigilNet, a detection and classification system is described which is used for tracking the motion of

ferrous objects. It combines information from magnetometer and acoustic (microphone) sensors to infer the location and direction of travel of objects. Information is condensed by the group head node to avoid overwhelming the base station with data. In [35], a similar approach using MICA2 motes along with magnetometer sensors is shown for detecting objects capable of generating magnetic fields. In [83], Ren et al. propose an analytical framework for object detection in sensor networks and develop wave sensing scheduling protocols to maximize network lifetimes while achieving bounded worst-case object detection quality.

Using smart cameras we can overcome restrictions like magnetic fields for the detection of objects. In [93], a histogram based approach was proposed to filter moving objects within a specied size range. It was used to localize human-sized moving entities in the field of view of each camera node. In contrast, I present results from a real-life deployment of the system regarding quality of detection and tracking, information arrival latency, memory and power consumption in view of a position estimation application which provides maps with distribution of people in indoor environments at any point of time. Also, I addresses issues like maximizing the sensing period for camera sensors using local adaptive image processing algorithms.

4.1.2 Building Energy Modeling

Building energy simulation tools such as eQuest and EnergyPlus use predefined occupancy profiles for office buildings based on day-type and maximum room occupancy. Recent studies have optimized energy consumption by adjusting the HVAC based on occupancy data. These approaches involve maps of electrical loads in a room[84], measurement of electrical loads[71], extensive deployment of sensors such as CO_2[71], passive infra-red (PIR) sensors[20] and computer net-

work activity monitoring[71] for determining whether a room is occupied. Other approaches have utilized PIR and door sensors, which give a binary indication of occupancy to optimally ventilate rooms in office buildings [1]. In [24, 23], occupancy models were proposed which were estimated using data from a camera sensor network. In the aforementioned approaches, the studies assumed/required the presence of IP-based connectivity for a back-channel to transmit and store data, electronic door locks which could provide access logs, submetering to monitor the consumption within a building and dense camera sensor network deployment. Also, to enable their models to predict occupancy in advance to optimally control lighting and ventilation, the data streams need to be recorded over days, weeks and months to incorporate occupancy data to improve building conditioning by doing occupancy prediction. For example: 5 day dataset using a camera sensor network [24, 23], 3 month dataset from contact and PIR sensors, network activity, building energy usage [71] and 10 month dataset from multiple streams (room maps, appliance power meters, building sub-metering) [84]. Therefore, we can clearly see that this model adaptation can be applied to this application domain for fast construction of occupancy models.

4.2 SCOPES: Smart Camera Object Position Estimation System

In order to collect occupancy data traces, a sensing infrastructure that can give accurate estimates of occupancy is needed. Using camera sensor networks, we can infer position and direction information from a single sensor by identifying an object and tracking its position in successive images, respectively. This reduces the density of nodes required to gather data while improving the data fidelity. However, camera sensors are not without their caveats; namely, network lifetime,

latency and detection performance. The power consumption of wireless radios transmitting high bandwidth image/video data to a central base station would result in rapid decrease in the lifetime of the sensor mote, in addition to causing network congestion. In-node computations process the raw data into summarized meta-data, which reduces the amount of data to be transmitted. However, this introduces latency because of limitations in the computational capabilities of sensor nodes. This necessitates, to reduce latency, the usage of lightweight processing algorithms and, to avoid missed detections, the redistribution of sensing over multiple sensors. At the same time, the low complexity of the data processing techniques should not compromise the detection requirements of the application. In this section, I detail the design and implementation of SCOPES, a distributed Smart Camera Object Position Estimation System that is capable of counting people as they move along the public hallways of a building.

4.2.1 System Description

In this section, the SCOPES system is described, including hardware and software architecture, image processing algorithms and details of the experimental deployment.

4.2.1.1 Hardware

Our SCOPES implementation comprises of an Agilent Cyclops camera [78] interfaced with a Moteiv Tmote Sky [69] module via an intermediate custom adapter board. The Cyclops camera performs local detection and processing of the visual information and the Tmote module provides multi-hop communication capability. The Cyclops board and the Tmote share power from the Tmote's battery back.

Cyclops: The Cyclops consists of an Agilent ADCM-1700 imager module, an ATMEL ATmega128L micro-controller (MCU), a Xilinx XC2C256 CoolRunner CPLD, an external SRAM (0.5MB) and an external Flash (0.5MB). The MCU controls the Cyclops sensor, sets the image capture parameters for the imager and instructs the imager to capture a frame. It is capable of running simple image processing algorithms and is responsible for communication with the host Tmote module. The Cyclops uses the external SRAM to supplement the limited internal MCU memory (4KB). However, as the MCU can address a maximum of 64KB of memory, the external SRAM and Flash is divided into eight 64KB memory banks. The entire 512KB of SRAM can be utilized by switching between memory banks.

Tmote: The Moteiv Tmote Sky module is comprised of an ultra low power Texas Instruments MSP430 F1611 micro-controller featuring 10KB of RAM, 48KB of flash and a Chipcon CC2420 radio for wireless communications. Tmote Sky has two expansion connectors, a 10-pin and a 6-pin IDC header. The Cyclops is connected to the Tmote module through these expansion connectors. The Cyclops is powered by the voltage pins mirrored on the expansion connector. Communication between the Tmote and the Cyclops is carried over the I2C bus. Whenever the Cyclops wants to transfer information to the Tmote, it interrupts the operation of the Tmote using an external trigger pin connected to the User Interrupt pin on the expansion connector. The Tmote then posts a read request to the slave device (Cyclops), which responds by sending the data over the I2C bus.

Interface Board: The interface board has a 51-pin Hirose connector on one side and a 10-pin and 6-pin female connector on the other side for accommodating the Cyclops board and the Tmote Sky module, respectively. The interface board

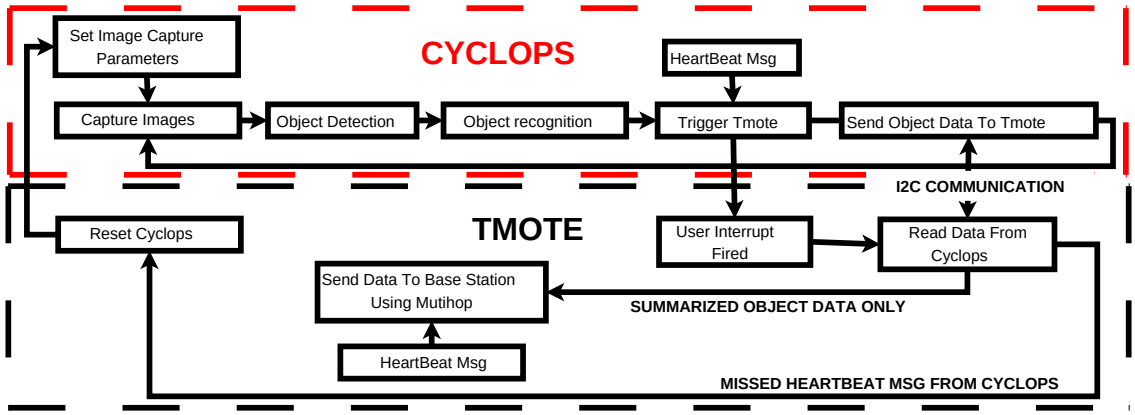


Figure 4.1: Software Block Diagram of SCOPES. The figure shows the different operations being executed on the Cyclops and Tmote modules. The arrows indicate the logical sequence of operations and interactions between the two devices.

also contains pull-up resistors for the clock and data lines for the I2C bus on the Tmote.

4.2.1.2 Software

The Cyclops and the Tmote run the event-based TinyOS operating system for wireless sensor network platforms. The Cyclops and the Tmote run the event-based TinyOS operating system for wireless sensor network platforms. TinyOS is written in nesC. It is characterized by a small memory footprint and direct-access to the underlying hardware. TinyOS has a synchronous execution model wherein a task will run to completion if not preempted. Due to this, long computations are split into smaller tasks that are posted sequentially. The TinyOS scheduler handles all the hardware resources and ensures fairness among the different modules. A block diagram of the software framework is shown in Figure 4.1.

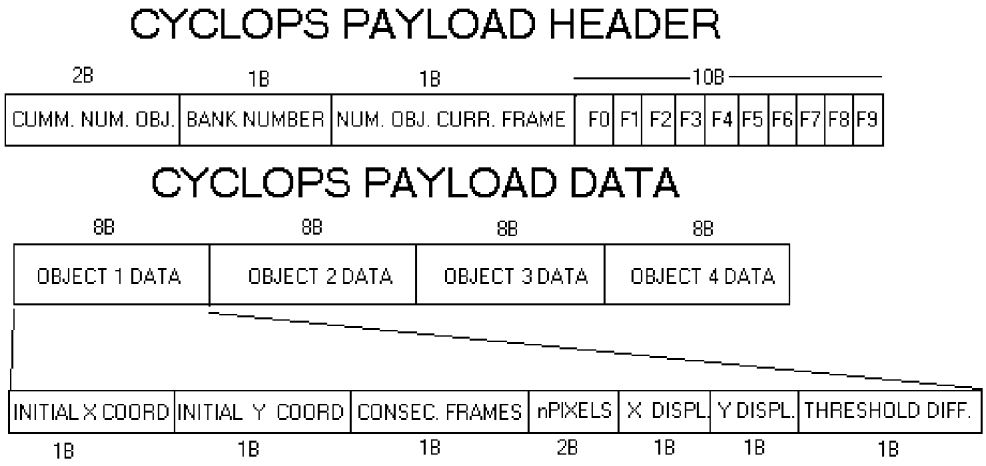


Figure 4.2: Cyclops Message Format. It is the payload of the TOS_Msg sent by the radio to the base station. (Note: 1. B stands for bytes 2. Figure does not include the other fields used for recording additional details like battery reading, routing tree information, time stamp, etc.)

Cyclops: The image processing (object detection and tracking) algorithms running on the Cyclops for SCOPES are implemented within the matrix operations libraries. The *abssub2thres* function is responsible for background subtraction and returns the cumulative absolute difference in pixel intensity for an image. The *group* function groups neighboring pixels which exceed a preset threshold into objects and stores information regarding their centroid, size and average threshold difference.

Whenever the Cyclops has anything significant to report, it triggers the User-Interrupt pin on the Tmote. It waits for the Tmote to post an I2C read request to which it responds by sending the payload comprising of centroid, size, threshold difference and direction information regarding detected objects. The structure of the cyclops payload (see Figure 4.2) is explained as follows: CUMulative NUMber of OBJects is the number of objects detected by the Cyclops till that time and

acts as an implicit sequence number, BANK NUMBER is the current memory bank number, NUMber of OBJects in CURRent FRAME is the number of objects regarding which data is transferred in the payload and F_0,....,F_9 contain the number of objects detected in each frame. The OBJECT DATA is comprised of the INITIAL X and Y COORDinates of the object (centroid), Number of CONSECutive FRAMES is the number of consecutive frames the object appeared in, nPIXELS is the object size in pixels, Total X DISPLacement and Total Y DISPLacement are the displacements of the object along the X and Y axes and THRESHOLD DIFFerence is the average absolute difference in threshold. This information is used by the centralized Density Estimation Algorithm (see Section 4.2.1.3). In addition to this, the Cyclops sends an "is active" status message to the Tmote every minute, over the I2C bus.

Tmote: The Tmote runs a derivative of the TinyOS operating system called Boomerang provided by Moteiv Inc., which provides extra function primitives like resource (I2C, radio) handlers, MultiHop communication (MultiHopLQI) and time synchronization (NetSyncM) native to the Tmote module. The I2C, SPI, UART and USART0 lines are common in the Tmote. In order to exclusively access any of these buses, a resource handle needs to be granted.

Whenever, the Cyclops activates the UserInterrupt pin on the Tmote, the interrupt handler posts a resource request for proceeding with an I2C read operation. When the I2C resource is granted, the Tmote requests data from the Cyclops using the appropriate slave device address. After the read operation is completed, the resource is released by the Tmote. To ensure that the Cyclops is functioning correctly, the Tmote expects an "is active" or "heart-beat" message from the Cyclops every minute. If the Cyclops misses four consecutive "heart-beats", then it is reset by the Tmote. At the same time, every four minutes, the Tmote

sends a "heart-beat" message to the base-station with a special payload to differentiate it from the object detection payload from the Cyclops. The TinyOS data packets sent over the radio are 76 bytes long ($TOSH\_DATA\_LENGTH = 76$).

4.2.1.3 Algorithms

Object detection is of great importance in surveillance and monitoring applications. It should be robust to noise, adaptive to gradual changes in background –such as illumination changes– and have low processing latency in order to capture significant events. The first step is background subtraction, followed by object detection and grouping, and finally, the direction inference part. Table 4.1 shows the time taken to execute some of these operations on the Cyclops.

Cyclops	**nFrames**				
Action	1	3	5	7	10
Image Capture (IC) Only	0.68s	1.04s	1.43s	1.85s	2.51s
IC + Background Subtraction (BS)	0.7s	1.15s	1.54s	2.01s	2.74s
IC + BS + Object Grouping	1.3s	2.76s	4.71s	6.49s	9.50s

Table 4.1: Average Time (in second) required for executing various image processing operations $nFrames$ images at a time for the Cyclops camera. (Note: Time information was recorded by executing each of the operations 100 times on two different Cyclops camera modules.)

Object Detection (OD): The goal of object detection is to determine the presence of an object in the image foreground, if any, and to update the background. In order to achieve low processing latency, a modified background subtraction algorithm is implemented for object detection. After background subtraction, we use two preset thresholds (OBJECT-THRES = 40 and SHADOW-THRES = 15)

```
count = 0
for (each pixel i in current image) do
    delta = | img(i) - bg(i) |
    if (delta ≥ OBJECT-THRESH)
        pixel(i) = OBJECT
        bg(i) = 0.99 * bg(i) + 0.01 * img(i)
        count++
    else if (delta ≥ SHADOW-THRESH)
        pixel(i) = SHADOW
        bg(i) = 0.95 * bg(i) + 0.05 * img(i)
    else
        pixel(i) = BG
        bg(i) = 0.85 * bg(i) + 0.15 * img(i)
```

Figure 4.3: Pseudo-code for background subtraction and update

to assign a label (OBJECT, SHADOW or BG) to each pixel (see Figure 4.3). Depending on the label assignment, the value of the corresponding background pixel is updated using an Exponentially Weighted Moving Average (EWMA). In the EWMA, we give less weight to the value of the current pixel if it is classified as an object or shadow, as these changes to the background are less likely to be of a long term nature. The weights are preset based on the image capture speed of the camera (see Table 4.1), the area covered by the camera 2.4m x 2.4m) and the speed of objects (approx. 1.2 m/s). In the current implementation, an object would need to be immobile for atleast 5s before being classified as a background pixel. This way, temporal background changes will be assimilated into the background over time. When the number of pixels labelled as OBJECT exceed a

preset threshold, an object detection is signalled. In the case of an object detection, pixels labelled SHADOW are relabelled as OBJECT if they have at least one neighbor that is an OBJECT pixel.

Object Recognition (OR): In this step, we try to group all pixels labelled as OBJECT by raster scanning the image starting from the top left corner. A pixel is considered to be part of an object if its left, top, top-left and top-right neighbors are labelled OBJECT. For example, in Figure 4.4, let x be the pixel in consideration and a, b, c and d are its top-left, top, top-right and left neighbors, respectively. If x, a, b, c and d are all labelled OBJECT, then x is considered to be part of an object. Assigning group id's is done using the rules explained in Table 4.2. Small groups in close proximity of each other are merged to account for fragmentation of a big object. For each object, information regarding the centroid (x and y coordinates), the number of pixels and the number of consecutive frames in which the object is detected, is maintained.

```
a  b  c
d  x
```

Figure 4.4: Grouping Pixels

Pixel b	**Pixel** d	**Action**
-	-	new group id
G_1	-	group G_1
-	G_1	group G_1
G_1	G_2	group G_1 Merge G_2 in G_1

Table 4.2: Basic algorithmic rules for group assignments for pixel x. (Note:− indicates group id is not assigned.)

Direction Inference: In SCOPES, the nodes were deployed in hallways to detect the transitions of people between different sections of a building floorplan

(see Section 4.2.2.3). This entailed determining movement of people in only two directions. In order to infer direction, objects in successive frames are matched according to their size (in pixels). Any object in the current frame that cannot be matched to another object in the previous frame is stored as a new object. Information for objects from previous frames that disappear or cannot be matched to objects in current frame are saved into a data structure. Information regarding the original position, displacement and number of consecutive frames is maintained for each object that appears across successive images in the current memory bank. After processing all the images in the current bank, an array of data structures containing information on a maximum of four objects is transferred to the base station via the Tmote.

Density Estimation Algorithm: On the base station, the packets coming from the various nodes are deconstructed. Messages are classified by source (node id) and time of occurrence. From the object data in the Cyclops payload, we can infer the direction of motion from the initial position and the displacement vector. Since, we have prior information about the deployment of the nodes, counting the transitions of objects enables the base station to compute the distribution of people in different sections of the building over time (assuming some initial distribution). Objects with no direction information are filtered out.

4.2.2 Performance Evaluation of SCOPES

In this section, we present results quantifying the trade-offs between data processing latency, memory usage, power consumption and detection performance. Table 4.3 shows the notations for the parameters used in the discussion sections.

Term	**E**xplanation
$nFrames$	number of images captured consecutively
$nBanks$	number of memory banks
$T_S^{nFrames}$	cyclops IC time for $nFrames$ images
T_S	total cyclops IC time (camera ON), depends upon $nBanks$
T_{OD}	(avg.) object detection time for $nBanks \times nFrames$ images
T_{OR}	(avg.) object recognition time per image
P	power consumption per node
DP	detection probability per node
DFP	detection failure prob. per node, $1 - DP$

Table 4.3: Notations used in SCOPES with the associated meanings.

4.2.2.1 Objective Functions

Our goal for SCOPES was to function as a surveillance system to monitor the occupancy of indoor environments such as office buildings. Some metrics and objective functions of interest are as follows:

Global/Local Density Estimate: How accurately can we estimate the occupancy of each section and of the total area covered?

Power Consumption: What is the system lifetime when powered by batteries?

Memory Usage: How much memory is required to achieve acceptable performance? How is the performance affected by memory size?

Detection Latency: How long does the system take to report data?

Detection Probability: How good is the estimate of the movement of people across different sections in the building?

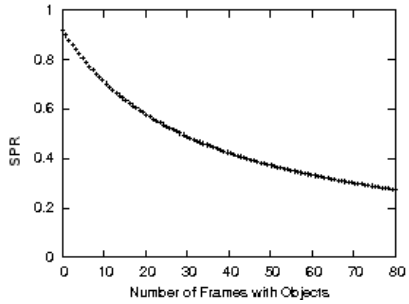
4.2.2.2 Simulation and Analysis

Since people passing under a camera can be regarded as a Poisson process, we model their inter-arrival times with an exponential distribution as follows:

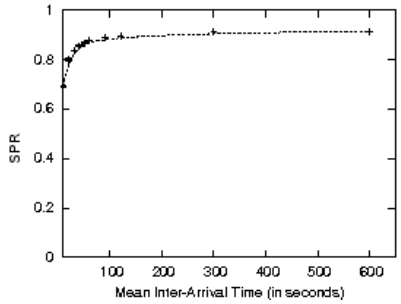
$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & , \quad x > 0 \\ 0 & , \quad x < 0 \end{cases}$$

where $\frac{1}{\lambda}$ is the mean inter-arrival time for an event. The number of frames in which an object appears is modeled by a uniform distribution (min=2;max=10) to account for variation in speed of people. We simulate the operation of a SCOPES node in the GNU R statistical computing package.

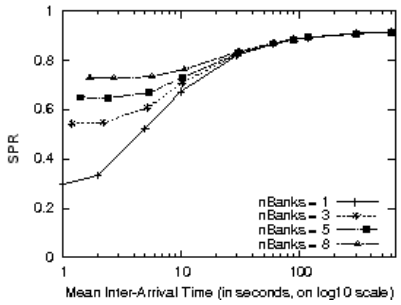
Sensing-Processing Ratio (SPR): In SCOPES, the camera is not operated in trigger-driven or schedule-driven modes discussed in previous studies [40]. Instead, each camera is either in active period, capturing and processing images, or in idle period, wait interval between successive active periods. *We refer to the ratio of time taken to capture images by the camera and the total active period i.e., the sum of the image capture and processing times, as sensing-processing ratio.* In our discussions, the sensing-processing ratio is the penalty incurred by the system as a result of the data processing latency. A high sensing-processing ratio is an indicator of lower data processing latency and vice-versa. There are two main reasons for operating the camera as described above; first, the camera hardware in current sensor networks does not allow concurrent image capture and processing of data and second, the object recognition algorithm introduces long latencies between successive image capture periods. Understanding sensing-processing ratio is important, since it affects many of our objective functions, including global/local position estimation, power consumption, detection probability and detection latency.



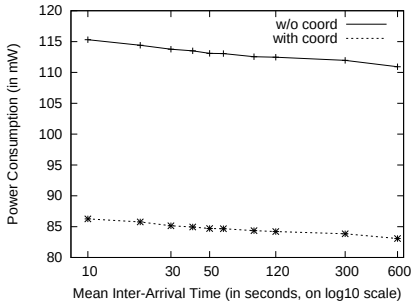
(a) SPR vs Objects Detected



(b) SPR vs Mean Inter-Arrival Time



(c) SPR vs Memory Usage



(d) Power Consumption vs Mean Inter-Arrival Time

Figure 4.5: (a) shows the variation in SPR of a node as a function of the number of image frames containing an object. (b) shows the SPR as a function of the mean inter-arrival time. (c) shows the change in SPR as a function of the mean inter-arrival times for different memory usages. (d) shows Power Consumption as function of the mean inter-arrival times.

In Figure 4.5(a), we observe the variation in SPR of a node with respect to the total number of images N in which an object (person) is detected. This relationship can be expressed as follows:

$$SPR = \frac{T_S}{T_S + T_{OD} + N \times T_{OR}} \tag{4.1}$$

Background subtraction needs to be performed for all the images resulting in a fixed cost T_{OD}. The object recognition function needs to be executed on only the

N frames in which an object is detected. As object recognition incurs the highest processing cost (T_{OR}, see Table 4.1), the SPR of a node is affected by the time taken for object recognition in each image. During the image processing latency period, an object passing beneath the camera will be missed as the Cyclops is not capable of simultaneously capturing and processing images. Thus, a low SPR will result in lower detection probability.

Figure 4.5(b) shows the variation in SPR with respect to the mean inter-arrival time $\frac{1}{\lambda}$ of an object detected by the camera with fixed amount of memory ($nBanks = 8$ and $nFrames = 10$, refer Section 4.2.1.1 for cause of $nBanks$). This relationship can be expressed as follows:

$$SPR = \frac{nT_S}{n(T_S + T_{OD}) + \sum_{i=1}^{n} \alpha_i \beta_i T_{OR}} \tag{4.2}$$

where n is the number of times we capture a set of 80 images, α_i is the number of times an object is detected and β_i is the average number of frames occupied by the object in the current (i^{th}) set of images. When the mean inter-arrival time is low, more objects are detected and the camera spends a longer time processing the image data, leading to a low SPR. Hence, longer data processing time leads to lower detection probability as the camera cannot capture images during that period.As the inter-arrival time increases, the SPR increases because the relative proportion of image processing time decreases.

In Figure 4.5(c), we observe the variation in SPR with respect to the mean inter-arrival time as a function of memory usage (varying $nBanks$). The amount of available memory dictates the space available to store images captured in time T_S ($=nBanks \times T_S^{nFrames}$). This relationship can be expressed as:

$$SPR = \frac{n(nBanksT_S^{nFrames})}{n(nBanksT_S^{nFrames} + T_{OD}) + \sum_{i=1}^{n} \alpha_i \beta_i T_{OR}} \tag{4.3}$$

As the mean inter-arrival times varies from low to high, the SPR increases with

memory usage since a node captures more images while spending a lower percentage of its time processing the image data. Hence, in general, more memory leads to a better SPR. For long mean inter-arrival times, the amount of memory does not have a significant effect on the SPR of a node.

Power Consumption (P): The lifetime of battery powered sensor nodes is directly affected by the power consumption of the system. The power consumed by the Cyclops and the Tmote Sky in different modes of operation is given in Table 4.4. The relationship between the power consumption and the different modes of operation of the node can be expressed as follows:

$$P = P_{cyclops}^{sensing} + P_{cyclops}^{proc} + P_{cyclops}^{sleep} + P_{tmote}^{RX} + P_{tmote}^{TX} \tag{4.4}$$

We analyse the power consumption under two different scenarios for node operation: (i) without coordination (multiple nodes sense the area at the same time) and (ii) with coordination (multiple nodes sense the area in non-overlapping intervals of time, see Section 4.2.3 for node coordination scheme details). Figure 4.5(d) shows the variation in power consumption for each of 3 nodes deployed to sense an area as a function of the mean inter-arrival time of an object.

For case (i), the Cyclops on each of the nodes is capturing and processing data all the time, i.e., there are no idle periods (Sleep mode). The power consumed by the Cyclops is slightly higher when it is processing image data stored in the external SRAM as compared to the power consumed in Image Capture mode (refer Table 4.4). This leads to higher power consumption when the inter-arrival time is low as the Cyclops spends a higher proportion of its active time processing data. For case (ii), with sensing coordination between the three nodes, the power consumed by each node is significantly lower than in the first case as the Cyclops has idle periods while waiting for its turn to sense the area.

Device Operation	**N**otation	**P**ower
Cyclops		
- Image Capture	$P^{sensing}_{cyclops}$	42mW
- Extended Memory Access	$P^{proc}_{cyclops}$	51.5mW
- Sleep	$P^{sleep}_{cyclops}$	0.7mW
Tmote		
- MCU + Radio RX	P^{RX}_{tmote}	65.4mW
- MCU + Radio TX (0dBm)	P^{RX}_{tmote}	58.3mW

Table 4.4: Power Consumption of the Cyclops and Tmote Sky Modules. (For more details, refer to [78] and the Tmote Sky data sheet.)

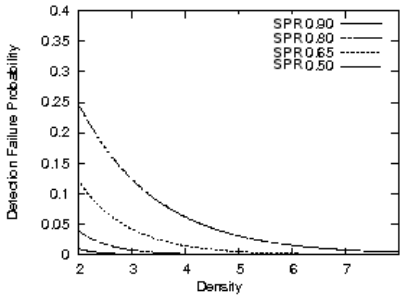


Figure 4.6: Detection Failure Probability as a function of density of nodes covering the same area and SPR.

Detection Failure Probability (DFP): Detection failures occur in the form of false negatives and false positives. However, we define Detection Failure Probability (DFP) as the probability that none of the camera nodes covering a section report the presence of a person passing under the camera. DFP quantifies the effect of *only false negatives* on our system. False negatives mainly depends on the SPR and to a lesser extent on the static thresholds in the object detection algorithm. Detection failures, due to static thresholds, are difficult to simulate as they might not provide an accurate representation of real-life conditions. We only analyse the relationship between DFP and multiple nodes n sensing an area at the same time (sensing without coordination) as a function of varying SPR. This relationship can be expressed as:

$$DFP = (1 - SPR)^n \tag{4.5}$$

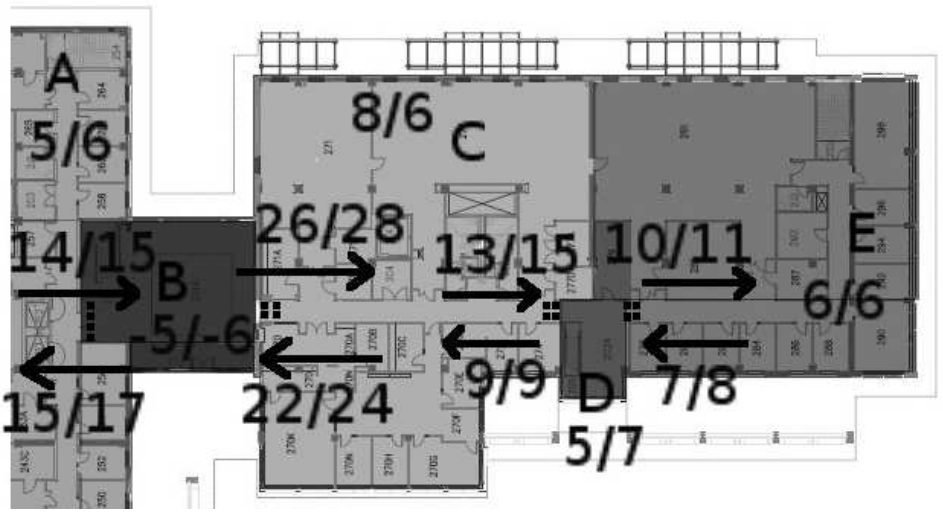
Figure 4.6 shows that the DFP decreases with higher SPR and number of nodes n. Since, the worst case processing time is bounded because of memory constraints, the SPR cannot fall below a certain number. Having coordination among the

nodes will improve detection failure probability by eliminating the chances of a missed detection due to SPR. However, there will still be some missed detections because of the deficiencies of the hardware and software in the underlying platform (See Section 4.2.3.3).

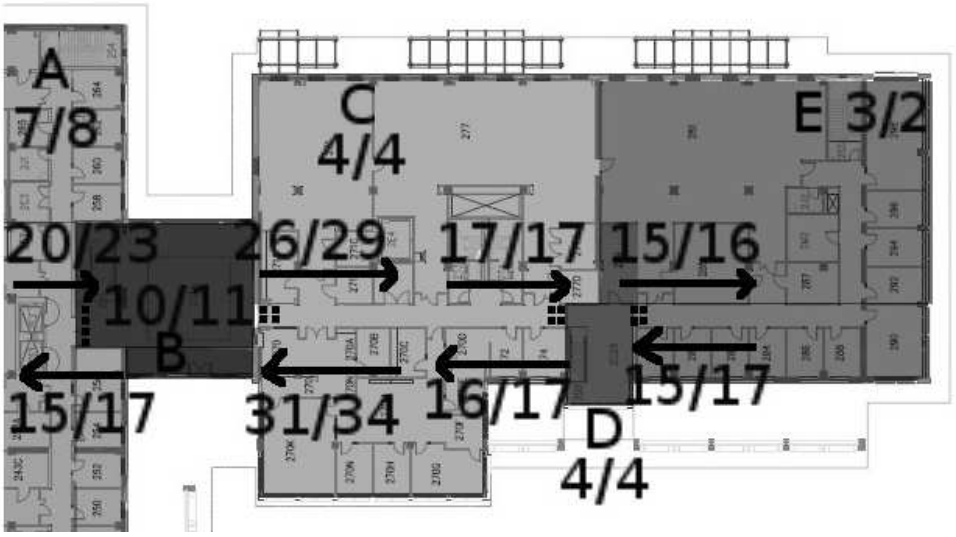
4.2.2.3 System Deployment

We deployed 16 nodes on the ceiling of the corridors in the short bar of the Science and Engineering building at the University of California - Merced. The deployment of nodes was done in this fashion to reduce privacy concerns of individuals by sensing in the common areas of the floorplan. The floorplan was separated into 5 sections by deploying the nodes in groups at transition points. *In each group, multiple nodes sense the same area at the same time, i.e., operating without coordination.* For collecting the ground truth, we installed two Panasonic KX-HCM280A network web cameras to record the movement of people. They are capable of capturing 10 frames per second (fps) at a resolution of 640×480 pixels. These images are timestamped using an NTP synchronized machine. The ground truth data is processed using haar cascades implemented in the OpenCV library [74] to provide a list of images in which a human being is detected. We manually corrected the OpenCV output for the false positives and false negatives in the processed ground truth. For computing detection probability and latency, we compare the manually processed and corrected ground truth data with the data collected from the SCOPES logs. The following is a list of experiments that we conducted for the performance evaluation:

Occupancy and Flow Estimation: 4 groups of nodes with 4 nodes in each group were deployed ($nBanks = 8$, $nFrames = 10$). The experiment was conducted twice for different two-hour periods of the day



(a) Morning Map



(b) Noon Map

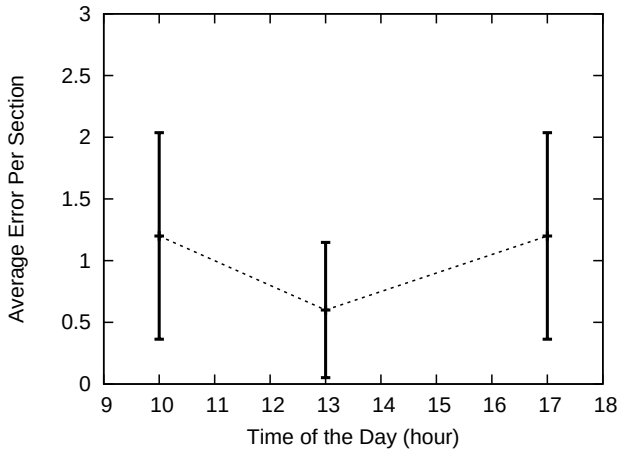
Figure 4.7: Occupancy and Transition Maps. The arrows indicate the direction of motion. The different sections of the floorplan are shaded and labelled with different alphabets. The numbers indicate counts from SCOPES (left) and from ground truth (right).

Memory Usage and Detection Latency: 3 groups of nodes with 4 nodes in each group were deployed for each value of $nBanks$ ($nFrames = 10$).
Detection Probability: experiment repeated thrice with 3, 4, 5 and 6 nodes deployed in each group ($nBanks = 8$, $nFrames = 10$).

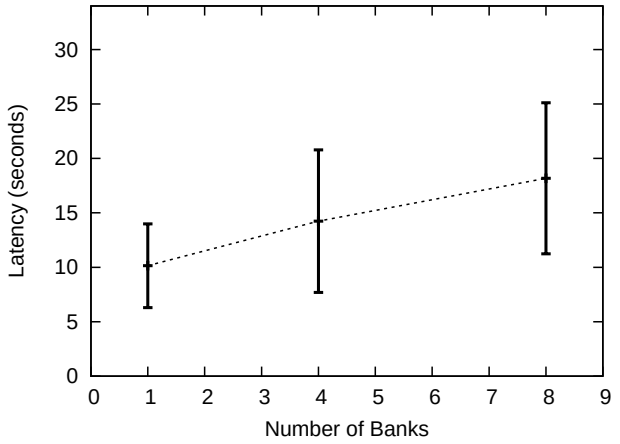
4.2.2.4 Experimental Results

Density Estimation: The first aspect we address is the evaluation of SCOPES for building density estimation maps of area occupancy by counting the transitions across the different sections. Figure 4.7 shows the occupancy and transition estimation maps created from data acquired from individual SCOPES experiments. From our results, we see that, we are able to track the movement of people over more than 73 sq. meters of the Engineering Building with a small, reasonable error. Figure 4.8(a) shows the average density estimation error for all the sections at different times of the day. Since, the error bars are overlapping with the mean values, we can say that there is no statistically significant change in average error which remains bounded (under 2) at different times of the day. The result shows that with suitable number of nodes and deployment location, embedded camera sensor networks such as SCOPES can provide adequate performance for density estimation purposes in real-world scenarios.

Detection Latency: Detection Latency is the time it takes for a node to report a person transitioning among different areas to the base station. In the Cyclops, the 512KB of available memory is partitioned into eight 64KB banks. Each node first captures $nBanks \times nFrames$ images and then it starts processing the image data in each bank. When the Cyclops is able to infer direction for an object



(a) Position Estimation Error vs Time of the Day



(b) Detection Latency vs Memory Usage

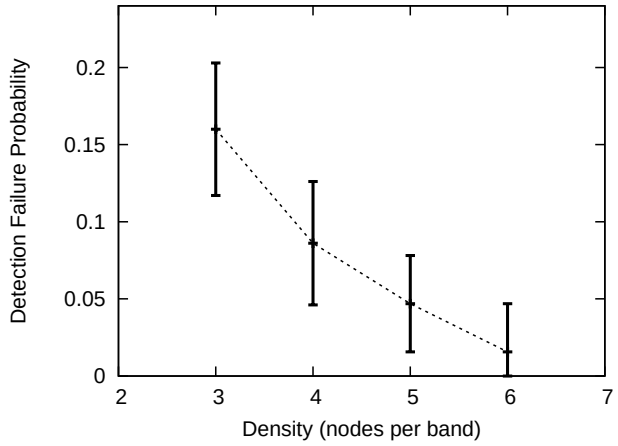
Figure 4.8: Fig. 4.8(a) shows the Average Position Estimation Error (number of people) per section for different times of the day. Fig. 4.8(b) shows the Detection Latency as a function of the number of memory banks. Errorbars are computed over multiple runs of the experiment.

from images in a certain bank, it will transfer the summarized object data to the Tmote for that bank. The Tmote routes the data to the base station via the radio. In our experiments, the base station was located 2 hops away from the farthest group of nodes. Figure 4.8(b) shows the variation in detection latency as a

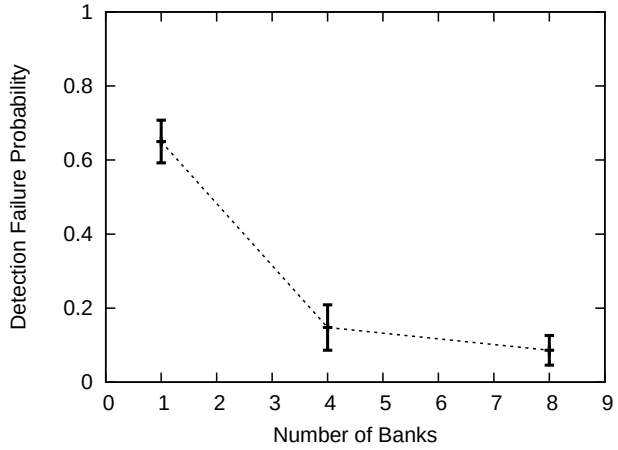
function of the number of memory banks used for storing images. In Figure 4.8(b), we observe that the detection latency is directly proportional to the amount of memory utilized for storing images. The detection latency is 10 seconds when $nBanks = 1$. It increases to 18 seconds when $nBanks = 8$. As the amount of available memory ($nBanks$) increases, the Cyclops can capture a higher number of images before processing the image data, resulting in longer detection latency. This shows for camera sensor networks that capture sequences of images before processing them, storing more image data can increase the detection latency, which could adversely affect the responsiveness of the surveillance system.

Detection Failure Probability (DFP): Figure 4.9(a) shows the variation in DFP i.e., false negatives as a function of the number of nodes used to cover an area. In general, deploying more nodes improves the DFP which agrees with our simulation results (see Figure 4.6). However, beyond 6 nodes we do not see an improvement in DFP because of the limited capabilities of our nodes. In Figure 4.9(b), we see the variation in DFP under memory constraints. In these experiments, we vary the number of memory banks used for storing image data. Figure 4.9(b) shows that DFP gets significantly reduced as we increase the number of memory banks. From Figures 4.9(b) and 4.5(c), we confirm that as the numbers of memory banks increases, SPR increases, leading to lower DFP.

Our experimental evaluation highlights and quantifies the trade-off between detection latency and detection performance as a function of memory usage and node density. For camera sensor networks like SCOPES, increasing the number of nodes would keep the detection latency low and improve detection performance at the expense of increased deployment cost. Also, by deploying sufficient number of nodes to cope with worst case SPR, we can enable lower detection latency and hence, a more responsive system.



(a) Detection Failure Probability vs Density of Nodes



(b) Detection Failure Probability vs Memory Usage

Figure 4.9: Fig. 4.9(a) shows Detection Failure Probability as a function of the density of nodes covering the same area. Fig. 4.9(b) shows the Detection Failure Probability as a function of the number of memory banks.

4.2.3 Improving SCOPES using Node Coordination

To ameliorate the effects of concurrent data processing latency periods for nodes working in a uncoordinated manner, we decided to implement a scheduling and coordinated sensing scheme. The goal of the scheme is to improve the perfor-

```
INITIALIZE()
    Change RF Power Level to RF2.
    Set Timer1 to fire after a random interval

TIMER1.FIRED()
    If no GROUP_ASSOC packet received,
        broadcast a GROUP_ASSOC message with group head = current node and set isCH = TRUE
        Set Timer2 to fire after a specific interval

TIMER2.FIRED()
    Broadcast a GROUP_INFORM_MEM message with
    information such as group head and other group members.

RECEIVEMSG.RECEIVE()
1   GROUP_ASSOC message received,
        isCh = FALSE
        send GROUP_INFORM_CH message to associate with a group head
2   GROUP_INFORM_CH message received,
        associate sender node id as part of group
        If Timer2 is not running, set Timer2 to fire after a specific interval
3   GROUP_INFORM_MEM message received,
        copy group data from packet (group head and other members information)
```

Figure 4.10: Grouping Algorithm Pseudo-Code

mance of the existing system by reducing the detection failure as well to decreasing the power consumption of the nodes (refer Section 4.2.2.2). The design requirements for our node coordination scheme are two-fold: (1) Nodes covering the same area or nodes in close proximity should provide near-continuous sensing coverage for the area, and (2) Nodes should provide near-continuous sensing coverage for an area even if some nodes stop functioning.

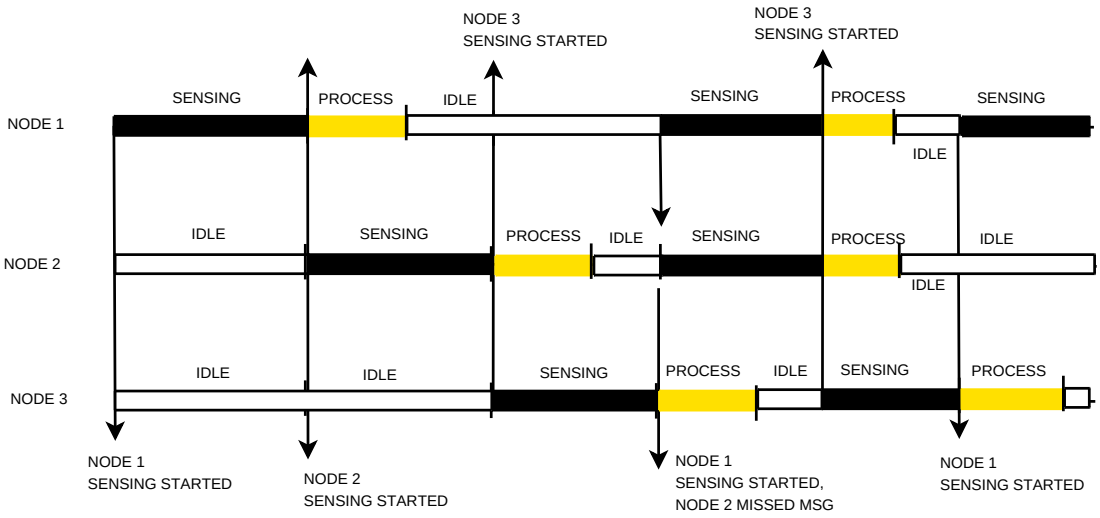


Figure 4.11: Illustration of the Group Coordination Algorithm.

4.2.3.1 Clustering Algorithm

The clustering algorithm (see Figure 4.10) executes periodically once every hour. At the beginning, the nodes change their RF power level to reduce the transmission distance ($RF2$). Each node then starts a one shot timer (Timer1) with a random interval up to a maximum of T1 seconds. After the Timer1 fires, a node sends a $GROUP\_ASSOC$ message declaring itself as the group head. It then starts Timer2 with an interval of T2 seconds. All nodes that are within close proximity of the group head respond by sending a $GROUP\_INFORM\_CH$ message. After Timer2 fires, the group head broadcasts a message containing information regarding the group head and the associated group members. Since, these radio messages do not propagate beyond a certain distance, we ensure that nodes in close proximity are part of the same group. This approach will work if the distance between groups of nodes is greater than the radio propagation distance for the set RF power level. For SCOPES, we empirically set T1=8s and T2=60s. The RF power level for group communication was set to $< -25dBm$.

4.2.3.2 Distributed Coordination & Scheduling

Once groups are formed, a node coordination scheme enables non-overlapping, continuous sensing coverage. Our notion of node coordination uses "soft-state" [37] to achieve continuous sensing coverage for a particular area. An illustration of the working on the scheme is shown in Figure 4.11. Here, each node sends an update message to its group members before it starts sensing. When the other nodes in the same group receive an update message, they advance their "start sensing" timers by one sensing period. When the sensing timer expires, another node sends an update message informing that it has started sensing the area. This way, we can achieve continuous sensing of an area, given sufficient deployment. As all the coordination messages are sent over the radio, we can never discount the possibility that an update message was missed by a particular node. By design, if an update message is lost, the system does not break down. In the event of a lost update message, nodes are expected to start sensing when their sensing timer expires. This also helps to provide continuous coverage in the event of node failures. When an update message is lost, multiple nodes will sense the area in that cycle but the schedule is resumed as soon as the new update messages are received in the following cycle. The current scheme adapts to node failures while ensuring that cameras are providing continuous sensing coverage all the time and non-overlapping coverage for a majority of the time.

Operation Mode	DFP/ False Negative %	False Positive %	Number of Objects
Without Coordination	20.0%	21%	85
With Coordination	15.3%	18.5%	103

Table 4.5: Comparison of detection performance with and without node coordination.

Mis-Detections	Reason	Occurrences	Percentage
False Negatives	Object at memory bank border	4/103	3.8%
	Object at limit of Sensing Range	8/103	7.7%
	Software Inadequacy	4/103	3.8%
False Positives	Objects in background	15/193	7.7%
	Over-counting due to split-objects	21/193	10.8%

Table 4.6: Counting the number of occurrences of false positives and false negatives along with their causes (NOTE: 1. Data is acquired from a single, two hour experiment with 3 nodes working together with coordination. 2. Under false negatives, 103 was the total number of people passing beneath the SCOPES nodes. 3. Under false positives, 193 is the total number of messages received at the base station).

4.2.3.3 Performance Evaluation

We performed experiments to evaluate the performance of the SCOPES when nodes work in coordination, sensing in non-overlapping intervals of time. The results are presented (refer Table 4.5) for a single, two hour experiment involving 3 nodes. We compare the performance of the SCOPES system in the presence and absence of coordination (see Section 4.2.2.4). Here, we see that DFP is reduced to 15% when nodes work with coordination as compared to 20% without any coordination. The DFP is also the false negative percentage for the system. Also, DFP shows significant improvement with node coordination. We have shown earlier (see Figure 4.5(d)) that we can significantly reduce the power consumption by using node coordination. Also, the relationship between detection performance, detection latency and memory usage (see Figures 4.8(b) and 4.9(b)) is independent of the presence or absence of coordination and hence, we expect these results

to show similar trends. To complete our evaluation of the system, we provide a quantitative analysis of the detection failures in SCOPES.

Analysis of Detection Failures: In Table 4.6, we enumerate the number of occurrences and their respective percentages along with the associated reason for misdetection, in the presence of node coordination.

False Negatives: We report a false negative when the ground truth indicates that there is an object in the foreground whereas our system reports none.

Objects missed due to software: Under-counting occurs when the object detection algorithm is unable to differentiate the object from the background. This happens because the colors of objects in the foreground do not contrast enough against the background to trigger object detection. Another scenario where under-counting could occur is when the object recognition algorithm (see Section 4.2.1.3) merges two objects that are in close proximity to each other.

Under-counting objects due to hardware: This occurs due to the following reasons: (a) SPR of nodes and (b) limitations of the sensing hardware. Detection failures due to SPR are avoided by using node coordination. However, the camera fails to detect an object due to loss of image data when the camera is switching memory banks. Since, the object is seen in only one frame in each bank, the algorithm would report no direction information as it does not combine information from successive banks. This problem could be resolved if memory was continuous and not split into banks. Objects that move close to the sensing range of the camera are missed because the camera is not able to cover the entire object from its point of view.

False Positives: False positives result mainly, due to the high sensitivity of the simple background subtraction algorithms used to detect the presence of objects

in the image foreground from the fixed thresholds. This might be due to over counting of objects in the foreground and camera hardware calibration.

Over-Counting Objects: Over-counting occurs because the object recognition algorithm might split one object into two objects. It also occurs when a foreign object becomes part of the background for a short time.

Camera Hardware: In SCOPES, when the camera starts capturing images, at times, the image at the start of the burst exhibits higher brightness as compared to all the rest due to calibration issues. This behavior of the imager results in false positives. However, the resulting message contains information about an object with disproportionately large number of pixels and no direction information. We neglect such objects when computing the false positives for our system.

4.2.4 Comparisons with previous work

In this section, we compare SCOPES with related work in the area of embedded camera sensor networks on issues like processing algorithms, latency, memory usage, detection probability and evaluation methods.

Kulkarni et al. [54] presented the design, implementation and evaluation of SensEye, a multi-tier camera sensor network for surveillance applications. The work aimed at showing that a multi-tier network can balance the conflicting goals of latency and energy-efficiency. In the evaluation experiments, circular objects were projected onto a wall with an area of $3m \times 1.65m$. Objects appeared at different location for a certain time duration with only one object present at a time. SensEye detected 42 out of 50 object appearances. It achieved 100% detection probability when objects are in view for 9 seconds which decreases to 52% when object time duration is 5 seconds. For moving objects, speeds were varied from 0.2m/s (all objects detected) to 0.6m/s (38% objects detected).

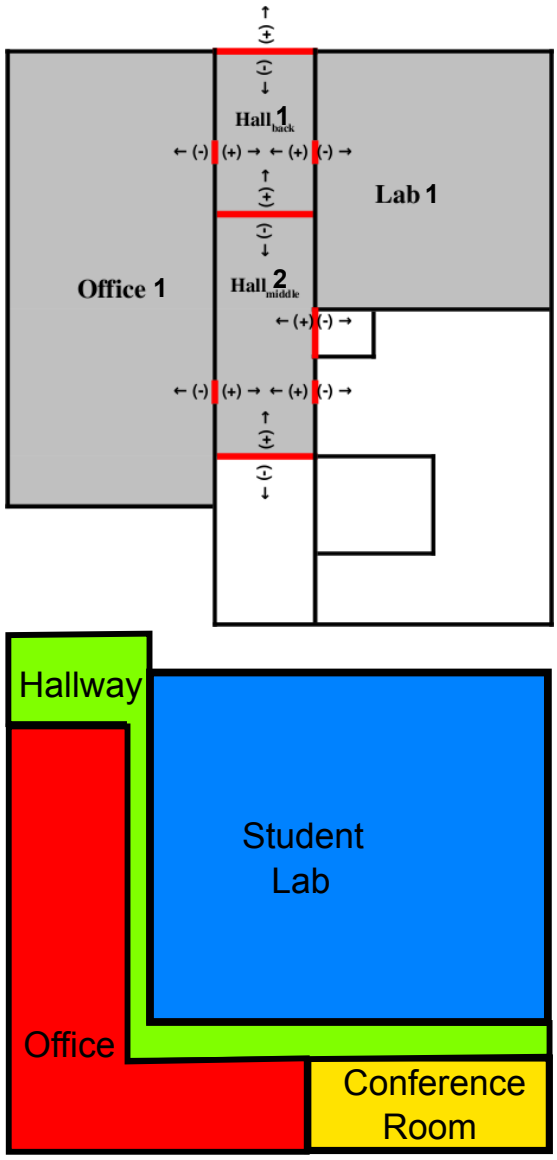
As seen in SensEye, a camera-based surveillance system fails when the speed of the object exceeds the capability of the system. From empirical data, it is said that humans move at speeds ranging from 1-1.5m/s [70]. The main difference between the evaluation of SensEye and SCOPES is that SCOPES was evaluated in uncontrolled real-life conditions where it had to account for variations in light conditions, shadows, occlusions, and the size and speed of people moving in the environment. SCOPES still has an average detection probability of 84% when we deploy 3 camera nodes to cover an area, which improves to 98% for 6 nodes. Based on the image capture speed of the camera, SCOPES will fail to capture information required to detect an object if the object moves at a speed greater than 8m/s (no image data collected) and will fail to infer the direction if the object moves faster than 4m/s (only one image frame collected).

Teixeira et al. [93] proposed a motion histogram approach to count people in indoor spaces. The hardware infrastructure comprised of Intel iMote2 sensor nodes with OmniVision OV7649 imagers. The iMote2 sensor platform operates at 104MHz and is capable of processing 8fps while consuming 322mW (Imote + Camera) of power. Six nodes were deployed with minimum overlap between areas covered by the cameras. Each camera has a field of view of 3m x 2m. The experiments consisted of five people moving inside a lab setting.The system has a detection rate of 89.5% when a single person is present inside the camera network. This drops to 82.48% when two people are present and 79.8% for three. Using the case study of the same camera node, Jung et al. [40] present lifetime models for trigger-driven and schedule-driven sensor networks. Their models predict the energy budgets under different application requirements. The results show the variation in the lifetime of the camera sensor network with respect to the detection probability and object inter-arrival rate. These results are compiled using data from the specifications of the underlying platform.

In comparison, in SCOPES, the Cyclops board operates at 4MHz and is capable of processing 1fps while consuming 115mW of power (Tmote + Cyclops). Inspite of the speed of the Cyclops platform (26 times slower than the iMote2) and the simple image processing algorithms, SCOPES is able to achieve detection probability of 84% with 3 cameras which increases to approx. 98% with 6 cameras covering the same area (see Figure 4.9). On a faster platform such as the iMote2, the SCOPES image processing algorithms would execute in roughly 26ms, eliminating the need for multiple cameras to provide coverage during the detection latency period of a camera while achieving comparable, if not superior, performance to the motion histogram approach. The performance evaluation of SCOPES highlights the point that by using computationally simple image processing techniques it is possible to achieve detection probabilities comparable to techniques like the motion histogram approach, inspite of differences in the computational capabilities of the underlying platforms. In SCOPES we analysed the power consumption for continuous sensing (without coordination) or interleaved sensing (with coordination) nodes, which differ from the operation models considered in [40]. In addition, in SCOPES, we also provide a detailed analysis and evaluation of the memory usage, detection latency and detection probability as a function of the system parameters.

4.3 Occupancy Model Adaptation

Figure 4.12 shows parts of two different building floorplans. The occupancy patterns for areas within these floorplans is shown in Figure 4.13. As the occupancy patterns are different over different floorplans, it is not possible to port occupancy models created for one floorplan to a completely different floorplan . Then the only way to create a new model is to deploy a sensing infrastructure



(a) (b)

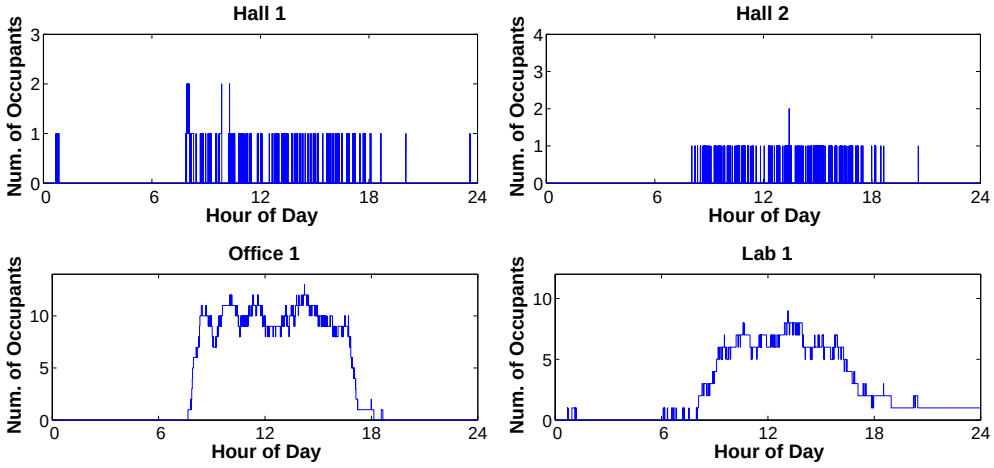
Figure 4.12: Building floorplans for reference model dataset (a) and adaptation model dataset (b), respectively.

to record the occupancy patterns. However, most of the current occupancy estimation approaches are ill-suited for existing buildings that lack infrastructure such as submetering systems and electronic locking systems among others. In

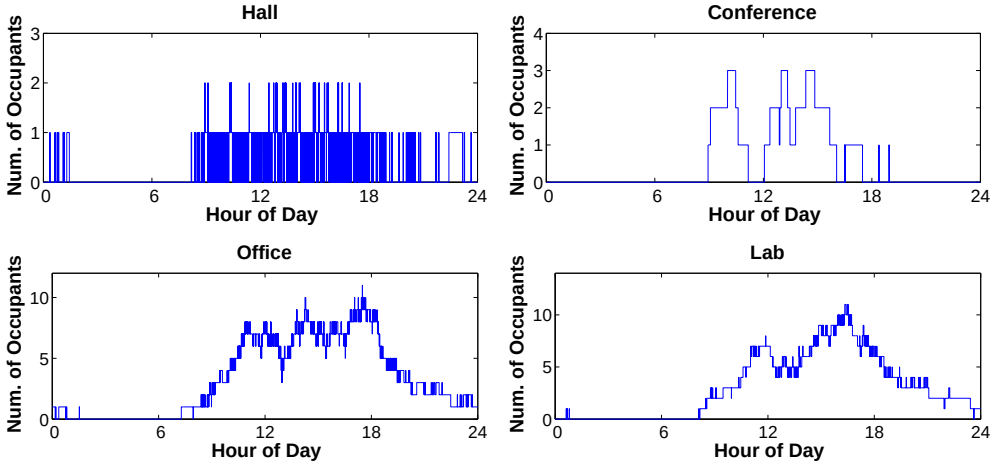
such situations, the importance of wireless sensor networks (WSNs) is realized as they allow for sensing without any pre-existing infrastructure. However, the limitations of a WSN, namely limited storage and limited battery life, are exposed when deployed to record occupancy data. This is because when there is no network access, the sensor nodes cannot transmit the data and hence, need to store it on-board. Passive infra-red (PIR) sensor have long lifetimes but give binary indication of occupancy. In Section 4.2, we discussed the design and implementation of SCOPES, a distributed camera sensor network for measuring occupancy. Using such a system, it is possible to collect only 1 or 2 days of occupancy data before the battery is depleted. This imposes a practical limit on the amount of data that can be recorded using a sensor network to train occupancy models. For applications such as building energy auditing where data collection is limited to 1-2 days [51], a sensor network infrastructure such as SCOPES is adequate. The biggest need for energy auditing applications is techniques/models that can estimate occupancy using 1 or 2 days of training data as input.

4.3.1 Proposed Approach

In general, model adaptation techniques adapt the parameters of a pre-existing model, called reference model, to match the distribution of the new data (adaptation data) from a small dataset. In this study, I present a technique to adapt the parameters of a previously proposed Gaussian model [24]. Our reference model is trained using an extensive 5-day occupancy dataset from building with floorplan as shown in Figure 4.12(a). The occupancy data that we record for the new building floorplan (see Figure 4.12(b)) is called adaptation data. The variation in room occupancies for the reference model dataset and adaptation dataset is shown in Figure 4.13. The reference model dataset rooms show occupancy pat-



(a) Reference Dataset Room Occupancy



(b) Adaptation Dataset Room Occupancy

Figure 4.13: Room occupancy averaged over the length of dataset (5-days) for every hour for the reference model (a) and adaptation (and retrained) model (b), respectively.

terns that are drastically different from the adaptation dataset rooms. The room occupancies for both datasets were recorded using SCOPES.

4.3.2 Reference Occupancy Model

In [24], Erickson et al. have proposed a multivariate Gaussian model for modeling occupancy data. A separate multivariate Gaussian distribution was trained for every hour of the day using occupancy data from several rooms of a building floorplan. A natural extension of the model is to use a Gaussian mixture model (GMM) in place of a single Gaussian. This gives the model additional flexibility to model the room occupancy patterns for every day of the week for each hour. We use a separate GMM for modeling occupancy for each hour. Let $X_h = (x_{h1}, \ldots, x_{hN})$ denote a series of N (=3600, one per second) occupancy data observations collected during hour h of a day. Each x_{hn} is a vector of dimension D, where D is the number of rooms. The probability density is given by:

$$p(\mathbf{x}_{hn}) = \sum_{m=1}^{M} \pi_{hm} p(\mathbf{x}_{hn}|m)$$

and

$$p(\mathbf{x}_{hn}|m) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_h|^{\frac{1}{2}}} \exp\left\{\frac{1}{2}(x_{hn} - \mu_{hm})^T \Sigma_h^{-1}(x_{hn} - \mu_{hm})\right\}$$

where there are M components and the parameters are the mixing proportions π_{hm} (which are positive and sum to one), the means μ_{hm} ($\mu_{hm} \in \mathbb{R}^D,\ \mu_{hm} \geq 0$) and the covariance matrix Σ_h. From this, the average occupancy vector for any given hour can be computed as follows:

$$\mu_h = \sum_{m=1}^{M} \pi_{hm} \mu_{hm}$$

The parameter estimation for a Gaussian mixture model is done with an EM algorithm [9]. Even though occupancy is expressed in terms of whole numbers,

a GMM is used as opposed to a discrete distribution because it allows for computation of conditional probabilities in an easy manner. In case of occupancy modeling, when little adaptation data is available, some rooms may be μ_{hm} will clamp to 0, indicating that the room remains unoccupied at all times, whereas the room may be heavily utilized on other days.

4.3.3 Adapting the Gaussian Mixture Model

Suppose we have a GMM (the *reference* model) corresponding to a building floorplan for every hour ($h \in \{1, \ldots, 24\}$) of the day. The reference model for each hour is trained on an extensive occupancy dataset resulting in good parameter estimates for the mean (μ_m), covariance Σ (same for all components) and mixture proportions π_m where $m \in \{1, \ldots, M\}$ (h is dropped from subscript to simplify notation). We are now given an adaptation occupancy dataset comprised of N D-dimensional vectors for a different building layout for which we need to estimate the adapted GMM parameters, $\{\widetilde{\pi}_m, \widetilde{\mu}_m\}_{m=1}^{M}$ and $\widetilde{\Sigma}$. Our adaptation algorithm is based on the idea of tying the GMM parameters, specifically the means, through a transformation of the reference GMM means for each hour. Adapting with a linear transformation would be simpler but it could allow the changes in some of the $\widetilde{\mu}_{md}$ to exceeed maximum room occupancy. Adapting with a sigmoid transformation avoids this problem. This strategy is similar to the approach in [43], where Bernoulli parameters must lie in $[0, 1]$. In our proposed approach, we tie the means of the reference model together using a scaled-sigmoid transformation as follows:

$$\widetilde{\mu}_{md} = \sigma(\mu_{md}; a_m, b_m) = \frac{O_m}{1 + e^{-(a_m \mu_{md}/O_m + b_m)}},\ d = 1, \ldots, D$$

where, O_m is the maximum room occupancy out of all rooms and is known a priori. The transformation has only two parameters a_m, b_m per component, which

enables us to estimate good values from the small adaptation occupancy dataset. As a_m, b_m are shared among the D-reference model means, their effect is propagated among all the component means. The mixture proportions are considered free during adaptation, subject to constraint that they sum to 1. The covariance for the mixture distribution is kept fixed ($\widetilde{\Sigma} = \Sigma$) during the adaptation. Thus, our algorithm needs to maximize the likelihood of the adaptation data over $3M-1$ parameters ($\widetilde{\pi}_1, \ldots, \widetilde{\pi}_{M-1}$ and $a_1, b_1, \ldots, a_M, b_M$). In our adaptation dataset, the number of rooms (or D) is 4, and therefore, by having to estimate one a_m,b_m for each component we do not have a significant saving in terms of number of estimated parameters. However, the savings will be significant as D is higher.

We solve for a_m, b_m for the adapted GMMs for each hour using a generalized EM algorithm. The objective function is the log-likelihood of the adaptation data given the constrained GMM with $3M-1$ free parameters:

$$L\left(\{\widetilde{\pi}_m, a_m, b_m\}_{m=1}^{M}\right) = \sum_{n=1}^{N} \log \sum_{m=1}^{M} \widetilde{\pi}_m p(\mathbf{x}_n; a_m, b_m)$$

where $p(\mathbf{x}_n; a_m, b_m)$ is a Gaussian mixture where $\widetilde{\mu}_{md} = \sigma(\mu_{md}; a_m, b_m)$. We maximize the objective function using a generalized EM (GEM) algorithm [66], which resembles the one in [43]. We initialize from the reference model, setting $\widetilde{\pi}_m = \pi_m$, and use an identity transformation (approximated using a sigmoid with $a_m = 6, b_m = -3$) to set $\widetilde{\mu}_{md} = \mu_{md}$. The GEM algorithm stops iterating between the E and M steps after converging to a local minima (estimates of a_m, b_m, π_m where ($m \in \{1, \ldots, M\}$).

4.3.4 Modeling Results

To evaluate the performance of our proposed approach, we compare the adapted models with the retrained models in terms of model log-likelihood. Out of the

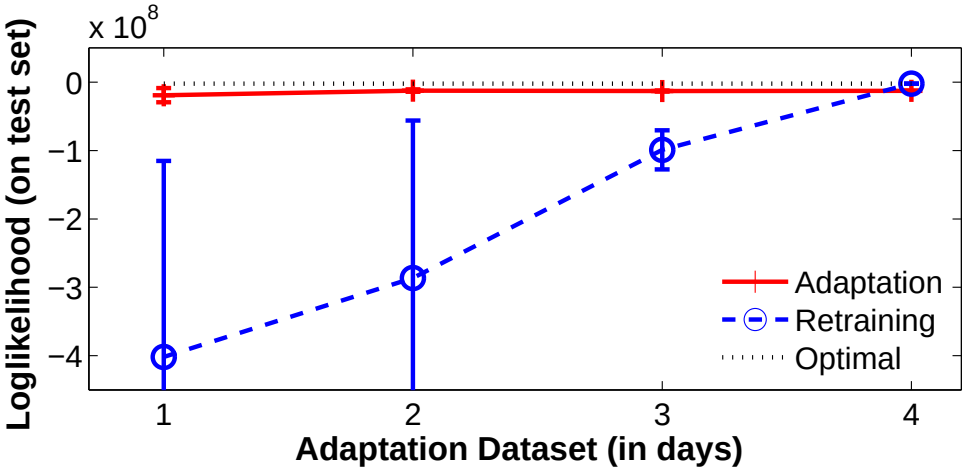


Figure 4.14: Log-likelihood of the different models as a function of the days in the adaptation dataset. Errorbars are computed over subsets of adaptation data. 5-day adaptation dataset, we used 4 days for training using the retraining and adaptation, and the 5^{th} day was used as the test set.

Figure 4.14 shows the variation in test set log-likelihood of the retrained and the adapted GMMs as a function of the amount of adaptation data. With just 1 day of data, the log-likelihood of the adapted model (GMM-A1) is already significantly better than the corresponding retrained model (GMM-R1). As we increase the adaptation data, the log-likelihood of the adapted model approaches the optimal log-likelihood (GMM-R4). The retrained model needs much more data to achieve a comparable log-likelihood to the adapted model; and its performance is much more variable depending on the specific adaptation vectors used (large error bars). Figure 4.15 displays the occupancy models for every hour for each room. We can observe that for rooms such as the Office and Lab that are heavily utilized, there is very little difference between the 3 models. However, for the conference room (Figure 4.15), we can that GMM-A1 is able to capture the occupancy behavior similar to GMM-R4 and also the average occupancy as

shown in Figure 4.13. This is in stark contrast to GMM-R1 that overfits to the adaptation data used for training the model, where the conference room was occupied only between 18:00-20:00 hours.

This clearly demonstrates that the adapted model generalizes to data in the unseen test-sets much better than the retrained model with significantly less training data.

4.3.5 Building Energy Simulation Results

Using the GMM-R4, GMM-R1 and GMM-A1 models, we construct an expected occupancy schedule for each hour using the average occupancy vector. For simulating the building energy consumption, we use these occupancy schedules to estimate ventilation and temperature schedules. We inputted these schedules into an EnergyPlus model of the building floorplan (total 32,000 sq.ft.) from which we have adaptation data for a Hall, Office, Lab and Conference room (approx. 12,000 sq.ft., see Figure 4.12). The building is compliant to applicable codes and has a single duct terminal-reheat HVAC system. To estimate the energy savings and conditioning effectiveness, we compared to a baseline strategy. The baseline strategy conditions rooms assuming maximum room occupancy between 7 a.m. and 10 p.m. and is off at other times. Also, we compare with OBSERVE [23]. It utilizes a Markov chain, in conjunction with a camera sensor network, to model the temporal changes in occupancy of a building. In OBSERVE, the camera sensor network sources the data to contrast and correct the transition probabilities of the Markov chain. OBSERVE has shown significant energy savings and close to optimal conditioning.

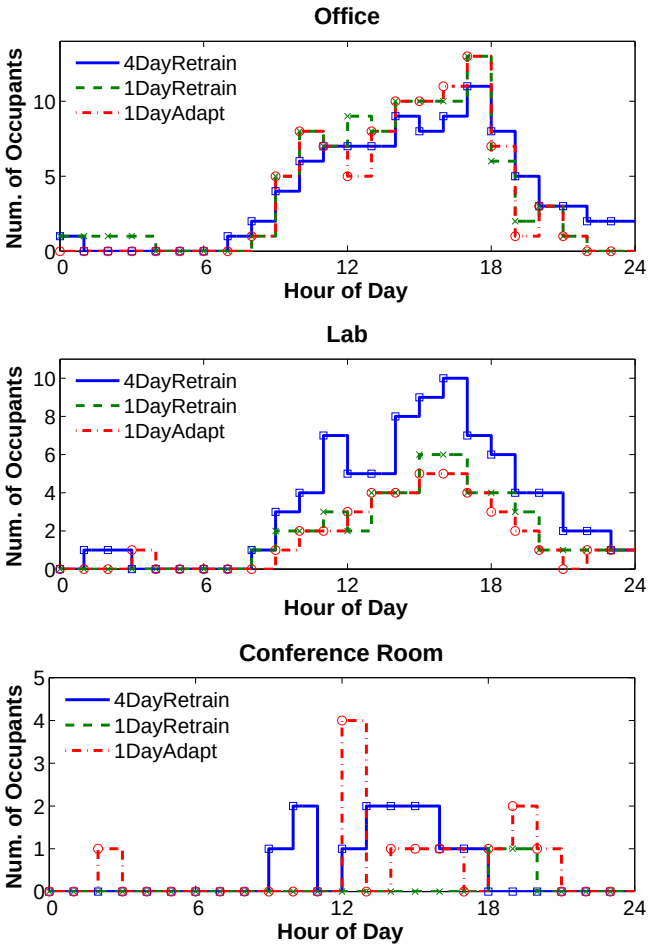


Figure 4.15: Occupancy models for Office, Lab and Conference rooms using retrained GMM trained with 4 days of data (GMM-R4), adapted GMM trained with 1 day of data (GMM-A1), and the corresponding retrained model GMM-R1.

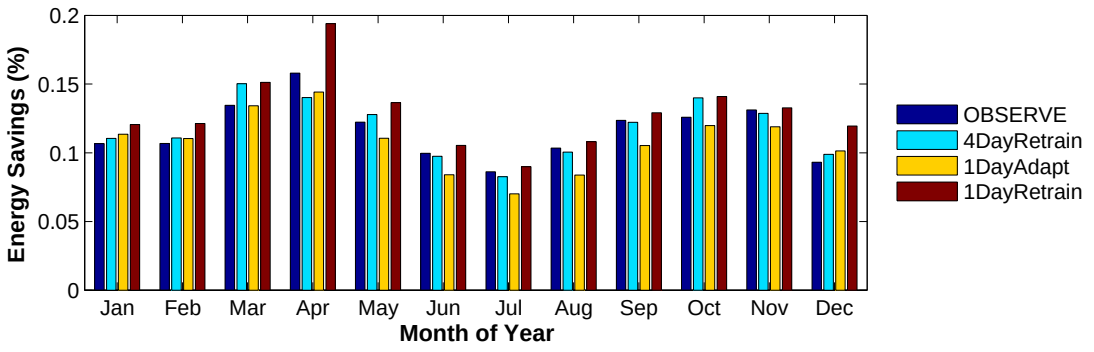


Figure 4.16: Estimated energy savings using different strategies. OBSERVE and GMM-R4 perform similarly. GMM-A1 is conservative whereas GMM-R1 overestimates savings

4.3.6 Energy Savings

Figure 4.16 shows the energy savings. Note, only the Hall, Office, Lab and Conference rooms (12,000 sq.ft.), for which adaptation data was available, were conditioned using GMM-R4, GMM-R1, GMM-A1 and OBSERVE. The other parts of the floorplan (20,000 sq.ft.) were conditioned using the baseline strategy. In our EnergyPlus simulations, energy consumption was computed for the major HVAC components; the fans, heating (gas), and cooling (gas). OBSERVE and GMM-R4 which are trained using 4 days of occupancy data show average annual energy savings of 11.2% and 11.4%, respectively. In comparison, our proposed approach GMM-A1, which used only 1 day of occupancy data, shows annual savings of 10.9%. GMM-R1 estimates annual savings of 12.9%. This is explained by the fact that the GMM-R1 model parameters (Figure 4.15) indicate that the conference room is mostly empty compared to GMM-R4 and GMM-A1. OBSERVE predicts occupancy using its Markov chain model and also corrects the ventilation and temperature using occupancy data updates from the sensor network. Therefore, the energy savings from OBSERVE are the most that one can

expect to achieve. Our proposed approach, GMM-A1 is conservative in its energy saving estimate, whereas GMM-R1 over-estimates savings due to overfitting of the model parameters.

4.3.7 Conditioning Effectiveness

There are two different conditioning criteria that need to be examined: thermal comfort (ASHRAE Std. 55) and outside air ventilation effectiveness (ASHRAE Std. 62.1). We compare the different models on the test day occupancy conditions for the winter and summer seasons.

4.3.7.1 Thermal Comfort

The optimal temperatures for the heating and cooling schedules used in the analysis are 78°F and 75°F respectively. Figure 4.17 shows the root mean squared error (RMSE) of the room temperature to the target temperature during occupied periods for the Conference Room and Office for the winter and summer seasons. The Baseline strategy is optimal between 7:00-22:00 hours as it conditions rooms between those times and is off at other times. OBSERVE has the best overall performance of the four models with lower RMSE for almost all hours for both the winter and summer. This is because OBSERVE has perfect visibility of the room occupancy based on data from the camera sensor network. In contrast, the models (GMM-R4, GMM-R1 and GMM-A1) have static temperature schedules created using the mean occupancy of each model at every hour.

For Office and Lab, all models do equally well. This is because, the adaptation data used for training the models is not significantly different from the test day. However, for the conference room, which is mostly unoccupied, except for certain hours on certain days, we can see the deviation from the Baseline. In summer, the

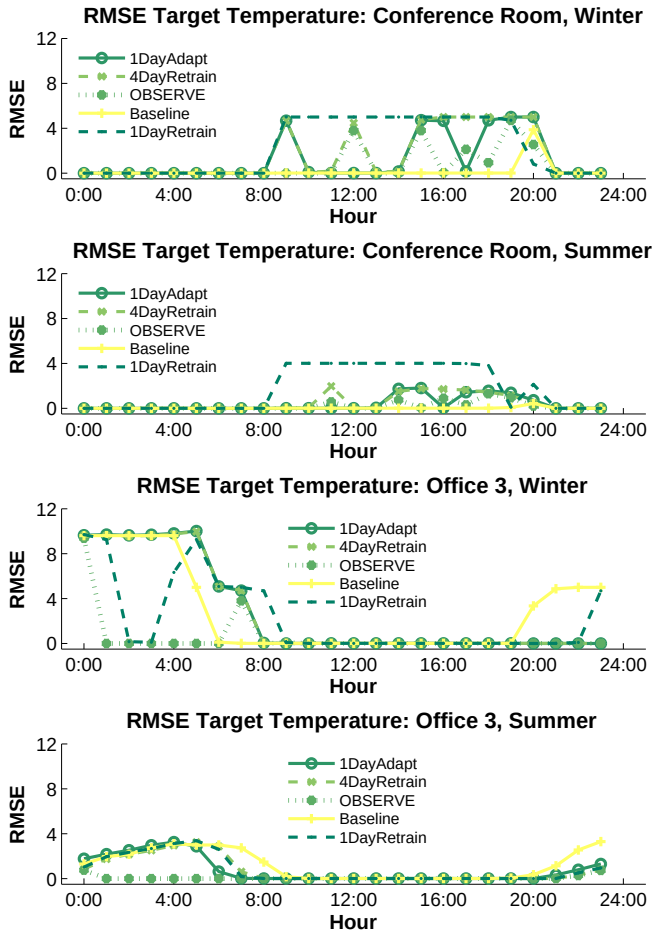


Figure 4.17: Comparison of the models with respect to temperature in the Conference room and Office 3 for the summer and winter seasons. The root mean square error of the target and measured temperature is examined for each hour.

RMSE of the Baseline, GMM-R4, OBSERVE and GMM-A1 models is below 0.5°F whereas the RSME for GMM-R1 is 1.8°F. In winter, the RMSE of the Baseline, GMM-R4, OBSERVE and GMM-A1 models is below 1.4°F whereas the RSME for GMM-R1 is 2.4°F. The GMM-R1 model conditions the conference room only starting from 18:00 hours till 20:00 hours, as the corresponding schedule indicates occupancy between those hours. This explains the worse behavior of GMM-R1.

The RMSE error is higher for the winter than the summer because the heating plant used for the simulation is less effective than the cooling plant.

4.3.7.2 Ventilation

Figure 4.18 shows a comparison of the ventilation rates. We computed the ventilation required for the area occupied by Hall, Lab, Office and Conference rooms. All ventilation strategies perform better than Baseline (maximum ventilation from 7am-10pm); the Baseline rate exceeds the required by at least 23%. OBSERVE has perfect visibility and therefore, ventilates the best, exceeding required by 2.8%. The static models over and under ventilate at times. GMM-R4, GMM-R1 and GMM-A1 all perform similarly, exceeding required by average of 8.3%. At times, when the models under-ventilate, it is by an average of 12 CFM (2 people) over a floorplan area of roughly 12,000 sq.ft.

4.3.8 Discussion

Our adapted model, GMM-A1 outperforms the corresponding retrained model, GMM-R1, trained with one day of adaptation data in terms on test-set log-likelihood. GMM-A1 gives a conservative estimate of energy savings when compared to GMM-R1, which over-estimates the energy savings. In terms on conditioning effectiveness, GMM-A1 matches GMM-R4 in being closer to target temperature compared to GMM-R1. This is because, GMM-R1 overfits to the adaptation data and estimates that the conference will only be occupied only between 18:00-20:00 hours for all days of the week, resulting in greater energy savings than all other models. In contrast, GMM-A1 generalizes beyond the adaptation data, estimating that the conference room will be occupied more times than is observed in the limited adaptation data. This behavior can be attributed to the use of the

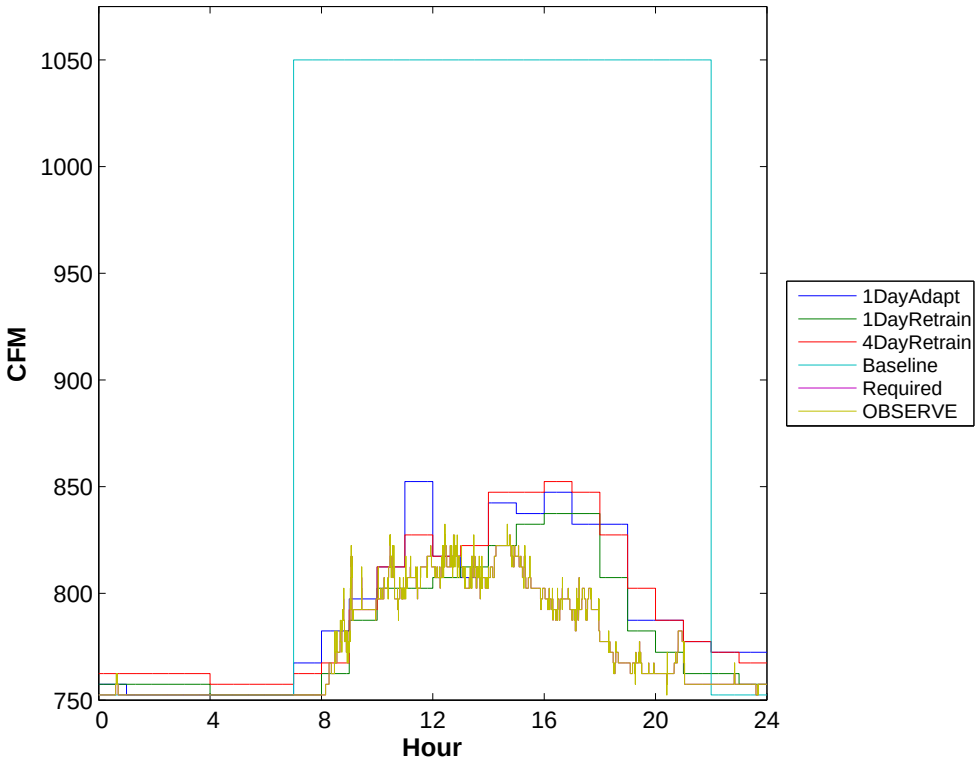


Figure 4.18: Comparison of the ventilation rates. The ventilation rate is the sum of Hall, Lab, Office and the Conference Room ventilation rates.

parameter-sharing based model adaptation approach. As the reference model parameters are tied together with the transformation, the resulting adapted model parameters cannot clamp to 0 when the room is unoccupied. This results in the adapted model parameters having non-zero mean occupancy for rooms that are unoccupied in the adaptation data, which is closer to reality.

Typically, model adaptation is effective if the reference model is close enough to the new adaptation occupancy data, such that the transformation parameters can then map to the distribution of the adaptation data. As an analogy, for a English word-recognition system, the reference model from one English language speaker can be adapted to a different (new) English speaker with just a few utterances. However, this does not work when using a English speaker reference model for French word recognition system, i.e. adaptation data is from a French speaker. Therefore, in our study, our reference and adaptation data were from two different floorplans but both were office buildings. Our results may be different if the adaptation data were from a building used exclusively for classrooms. For the purpose of energy auditing, it would be useful to have reference models for a few different types of buildings with fundamentally different usage patterns such as government offices, private offices, schools, research buildings, etc.

4.3.9 Summary

In this chapter, we showed that previous work had not addressed the problem associated with sensor network infrastructure in terms of application performance and network lifetime for occupancy modeling. This raised doubts regarding the sustainability of such systems. This dissertation addressed the problem in terms of system-design and modeling. It was shown in SCOPES that deployment of multiple sensor nodes working in coordination with each other eliminates some

of the problems associated with network lifetime, data processing latency and quality of detection performance. In addition, we analysed the detection failures of SCOPES by describing the causes of misdetections and quantifying their effects. Our system provides on par or better detection performance than other approaches that have computationally intensive algorithms and more capable hardware, with a slightly higher deployment cost. On the modeling side, it was shown that by collecting only 1 day of occupancy data and using the proposed model adaptation technique, we reduced the amount of data required for estimating good parameter values for the occupancy models to 1 day. The adapted model generalizes to occupancy conditions not recorded in the small training dataset. In building energy simulations, the adapted occupancy models performed on par with other models that require 4 times as much training data.

CHAPTER 5

Conclusion and Future Work

Without real data, it is not possible to create a realistic model of a complex phenomenon such as wireless communication. In this study, we presented a new data-driven, multi-level Markov model (M&M) to replicate more realistic short- and long-term dynamics in wireless simulations. The M&M model generalizes many existing wireless link models, can model complex correlations if sufficient parameters are used, and is straightforward to learn from data and to sample from. New M&M models can be created by mixing preexisting M&M models from a library. Based on extensive evaluation using long experimental data traces collected in multiple testbed environments, our study demonstrated that the model significantly outperforms other simulation tools available in the WSN community.

The study also addressed the data requirement concerns by proposing a novel parameter-tying based, model adaptation approach for constructing high-quality short term wireless link models. This study showed that by utilizing a total 3 minutes of data from a target link, we can adapt a reference MMB using multiple sigmoid transformations to model target link MMB distributions. This is an order of magnitude decrease in the amount of data required to train a high quality model to capture the short term correlations of a link. Also, we showed that the model adaptation procedure can adapt to target links under widely different conditions (environments, packet size, tx. power level and radios). Additionally,

we demonstrated that our adapted model performs close to the ground truth in protocol simulations.

In addition, data modeling issues were explored in a completely different application domain, namely building occupancy modeling. The study validated through analysis, simulation and extensive experimentation, the behavior of SCOPES, a camera sensor network that balanced the tradeoff between sensor network lifetime, data processing latency and quality of detection performance. Also, it presented evidence that WSNs can benefit from application of model adaptation techqniues by providing support for application such as energy auditing. The study addressed the issue of estimating occupancy models when it is possible to collect only 1 or 2 days of occupancy data. Using model adaptation, the amount of data required for estimating good parameter values for the occupancy models was reduced to 1 day. The adapted models generalized to occupancy conditions not recorded in the small training dataset, conservatively predicting 10.9% energy savings and conditioning effectiveness on par with other models that require 4 times as much training data.

There are multiple areas for future work. Regarding modeling, one can use for the emission distribution restricted Boltzmann machines, which are another powerful way of representing high-dimensional binary data. Moreover, the model can be extended to emit signal strength values, thus, modeling physical layer characteristics such as RSSI values of wireless traces. Model adaptation techniques that can transform transition matrices would help account for long term dynamics in wireless link models. In terms of system issues in WSNs, for the case of building occupancy modeling, the study has narrowed down the list of problems to being able to correct for detection errors. This can be addressed using approaches such as particle filters. Also, the impact of model adaptation techniques on address-

ing WSN performance issues has paved the way for the inclusion of powerful, well-studied machine learning techniques in the wireless sensor network domain. Refining and applying these techniques will advance wireless sensor networks from the research arena to being useful tools in everyday life.

APPENDIX A

Appendix

A.1 Hidden Markov Models

A HMM models an observed sequence of (continuous or discrete) vectors in terms of a sequence $q_0,\ q_1, \ldots$ of hidden (unobserved) random variables called states and a sequence $\mathbf{x}_0,\ \mathbf{x}_1, \ldots$ of observed random variables (see fig. 2.5). The HMM represents the probability of the observed sequence in terms of the *state transition probability* $p(q = j|q = i)$ (which assumes the Markov property and is independent of time) between every pair of state values, and the *emission probability* $p(\mathbf{x}|q = i)$ of outputting a vector $\mathbf{x}$ when in state i. The latter can be, for example, a Gaussian or beta (or mixture thereof) for continuous $\mathbf{x}$ and a Bernoulli, multinomial (or mixture thereof) or a simple probability table for discrete $\mathbf{x}$. Thus, the probability of observing $\mathbf{x}_0,\ \mathbf{x}_1, \ldots, \mathbf{x}_T$ is

$$p(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) = \sum_{q_0,\ldots,q_T} p(q_0) \prod_{t=1}^{T} p(\mathbf{x}_t|q_t) p(q_t|q_{t-1})$$

where the sum is over all possible state sequences. A HMM is then described by the dimension W of the observed vector $\mathbf{x}$, the number of state values Q, the $Q \times Q$ matrix of transition probabilities $a_{ij} = p(q = j|q = i)$, and the parameters of the emission distribution for each state value.

For simple emission distributions, the HMM parameters (transition probabilities and emission parameters) can be estimated given only a sequence of observed

vectors $\{\mathbf{x}_t\}$ by maximum likelihood using an expectation maximization (EM) algorithm [6], which iterates from initial parameter values. This is the training or learning problem, and it is possible to converge to a local optimum. The most likely sequence of state values corresponding to an observed sequence can be obtained using the Viterbi algorithm. This is the decoding problem. Sampling from a trained HMM given an initial state value simply requires sampling states from the transition probabilities and sampling an $\mathbf{x}$ for each state from its emission distribution.

A.2 Mixtures of Multivariate Bernoulli Distributions

A Bernoulli distribution for a binary random variable x assigns probability p to $x = 1$ and $1 - p$ to $x = 0$. A Bernoulli distribution in W binary variables is the product of W independent univariate Bernoulli distributions with parameter vector $\mathbf{p} = (p_1, \ldots, p_W)^T$:

$$p(\mathbf{x}) = \prod_{i=1}^{W} p_i^{x_i}(1 - p_i)^{1-x_i}.$$

A mixture distribution is constructed given M component distributions $p_1(\mathbf{x}), \ldots, p_M(\mathbf{x})$ and M component proportions $\pi_1, \ldots, \pi_M$ (with each $\pi_m \in (0, 1)$ and $\sum_{m=1}^{M} \pi_m = 1$):

$$p(\mathbf{x}) = \sum_{m=1}^{M} \pi_m p_m(\mathbf{x})$$

and, if $M > 1$, then the components of $\mathbf{x}$ are not, in general, independent from each other; in fact, we can model complex correlations this way. The parameters $\{\pi_m, \mathbf{p}_m\}_{m=1}^{M}$ of a mixture of multivariate Bernoulli distributions (MMB) can be estimated given a collection of N W-dimensional binary vectors using an EM algorithm [12], which iterates from initial parameter values and can converge to

a local optimum. Sampling from a MMB simply requires picking a component with probability proportional to its proportion, and then sampling the binary vector from its Bernoulli.

References

[1] Y. Agarwal, B. Balaji, S. Dutta, R. K. Gupta, and T. Weng. Duty-cycling buildings aggressively: The next frontier in HVAC control. In *IPSN'11: Proceedings of the 10th International Conference on Information Processing in Sensor Networks*, pages 246–257, Chicago, IL, USA, Apr. 2011.

[2] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *SIGCOMM Computer Communication Review*, 34(4):121–132, 2004.

[3] M. H. Alizai, O. Landsiedel, J. Á. B. Link, S. Götz, and K. Wehrle. Bursty traffic over bursty links. In *SenSys'09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, New York, NY, USA, 2009. ACM.

[4] D. W. Allan. http://www.allanstime.com/.

[5] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *SenSys'03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 150–161, New York, NY, USA, 2003. ACM.

[6] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[7] A. Becher, O. Landsiedel, G. Kunz, and K. Wehrle. Towards short-term wireless link quality estimation. In *HotEmNets'08: Proceedings of the 5th Workshop on Embedded Networked Sensors*, Charlottesville, Virginia, USA, June 2008. ACM.

[8] J. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models. Technical Report TR-97-021, ICSI, 1998.

[9] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[10] G. Boggia, P. Camarda, and A. D'Alconzo. Performance of Markov models for frame-level errors in IEEE 802.11 wireless LANs. *International Journal of Communication Systems*, 22(6):695–718, 2009.

[11] M. J. Brandemuehl and J. E. Braun. The impact of demand-controlled and economizer ventilation strategies on energy use in buildings. *ASHRAE Transactions*, 105(2), 1999.

[12] M. Á. Carreira-Perpiñán and S. Renals. Practical identifiability of finite mixtures of multivariate Bernoulli distributions. *Neural Computation*, 12(1):141–152, Jan. 2000.

[13] A. Cerpa, N. Busek, and D. Estrin. SCALE: A tool for simple connectivity assessment in lossy environments. Technical Report CENS Technical Report 0021, University of California, Los Angeles, Sept. 2003.

[14] A. Cerpa, J. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *IPSN'05: Fourth International Symposium on Information Processing in Sensor Networks*, pages 81–88, Los Angeles, CA, USA, July 2005. ACM/IEEE.

[15] A. Cerpa, J. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: Modeling and implications on multi-hop routing. In *MobiHoc'05: Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 414–425, Urbana-Champaign, Illinois, USA, Aug. 2005. Association for Computing Machinery (ACM).

[16] Y. Chen and A. Terzis. On the mechanisms and effects of calibrating RSSI measurements for 802.15.4 radios. In J. Silva, B. Krishnamachari, and F. Boavida, editors, *Wireless Sensor Networks*, volume 5970 of *Lecture Notes in Computer Science*, pages 256–271. Springer, 2010.

[17] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S. P. Fekete, A. Kröller, and T. Baumgartner. Flexible experimentation in wireless sensor networks. *Commun. ACM*, 55(1):82–90, Jan. 2012.

[18] V. V. Digalakis, D. Rtischev, and L. G. Neumeyer. Speaker adaptation using constrained estimation of Gaussian mixtures. *IEEE Trans. Speech and Audio Process.*, 3(5):357–366, Sept. 1995.

[19] M. Doddavenkatappa, M. C. Chan, and A. A. L. Indriya: A low-cost, 3D wireless sensor network testbed. In *TridentCom 2011*.

[20] R. H. Dodier, G. P. Henze, D. K. Tiller, and X. Guo. Building occupancy detection through sensor belief networks. *Energy and Buildings*, 38(9):1033–1043, 2006.

[21] D. M. Doolin and N. Sitar. Wireless sensors for wildfire monitoring. In *Proceedings of SPIE Symposium on Smart Structures and Materials/ NDE 2005*, 2005.

[22] E. O. Elliot. A model of the switched telephone network for data communications. *Bell Systems Technical Journal*, 4(1):89–109, 1965.

[23] V. L. Erickson, M. Á. Carreira-Perpiñán, and A. Cerpa. OBSERVE: Occupancy-based system for efficient reduction of HVAC energy. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011)*, pages 258–269, Chicago, IL, USA, Apr. 2011.

[24] V. L. Erickson, Y. Lin, A. Kamthe, R. Brahme, A. Cerpa, M. D. Sohn, , and S. Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *Proceedings of the 1st ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys 2009) in conjunction with ACM SenSys 2009*, Berkeley, CA, USA, Nov. 2009. ACM.

[25] B. S. Everitt and D. J. Hand. *Finite Mixture Distributions*. Chapman & Hall, 1981.

[26] M. Farhadloo and M. Á. Carreira-Perpiñán. Regularising an adaptation algorithm for tongue shape models. In *Proc. of the 37th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012.

[27] J. Forney, G.D. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, Mar. 1973.

[28] M. J. F. Gales. Cluster adaptive training of hidden Markov models. *IEEE Trans. Speech and Audio Process.*, 8(4):417–428, July 2000.

[29] J.-L. Gauvain and C.-H. Lee. Bayesian learning for hidden Markov model with Gaussian mixture state observation densities. *Speech Communication*, 11(2–3):205–213, June 1992.

[30] E. N. Gilbert. Capacity of a burst-noise channel. *Bell System Technical Journal*, 39(5):1253–1266, Sept. 1960.

[31] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *SenSys'09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14, New York, NY, USA, 2009. ACM.

[32] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *SenSys '05: Proceedings of the 3rd international conference on embedded networked sensor systems*, pages 205–217, New York, USA, 2005. ACM Press.

[33] R. Gupta and S. R. Das. Tracking moving targets in a smart sensor network. In *VTC '03: Proceedings of 57th IEEE Vehicular Technology Conference*, pages 3035–3039, 2003.

[34] M. Han, A. Sethi, W. Hua, and Y. Gong. A detection-based multiple object tracking method. In *ICIP '04: International Conference on Image Processing.*, 2004.

[35] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, pages 270–283, New York, USA, 2004. ACM Press.

[36] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1):185–234, 1989.

[37] P. Ji, Z. Ge, J. Kurose, and D. Towsley. A comparison of hard-state and soft-state signaling protocols. *IEEE/ACM Transactions on Networking*, 15(2):281–294, 2007.

[38] C. Jiao, L. Schwiebert, and B. Xu. On modeling the packet error statistics in bursty channels. *Local Computer Networks, Annual IEEE Conference on*, 2002.

[39] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with Zebranet. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 96–107, New York, USA, 2002. ACM Press.

[40] D. Jung, T. Teixeira, A. Barton-Sweeney, and A. Savvides. Model-based design exploration of wireless sensor node lifetimes. In *EWSN*, 2007.

[41] A. Kamthe. M&M simulator for TOSSIM. `http://www.andes.ucmerced.edu/software`.

[42] A. Kamthe, M. Á. Carreira-Perpiñán, and A. Cerpa. M&M: Multi-level Markov model for wireless link simulations. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009)*, Berkeley, CA, Nov. 2009. ACM.

[43] A. Kamthe, M. Á. Carreira-Perpiñán, and A. Cerpa. Adaptation of a mixture of multivariate Bernoulli distributions. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[44] A. Kashyap, S. Ganguly, and S. R. Das. Measurement-based approaches for accurate simulation of 802.11-based wireless networks. In *MSWiM '08: Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 54–59, New York, NY, USA, 2008. ACM.

[45] S. A. Khayam and Y. Cho. CRAWDAD trace set niit bit errors 802.11 (v. 2008-07-08), July 2008.

[46] S. A. Khayam and H. Radha. Markov-based modeling of wireless local area networks. In *MSWIM '03: Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 100–107, New York, USA, 2003.

[47] A. Konrad, B. Zhao, A. Joseph, and R. Ludwig. A Markov-based channel model algorithm for wireless networks. In *Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2001)*, 2001.

[48] A. Konrad, B. Y. Zhao, and A. D. Joseph. Determining model accuracy of network traces. *J. Comput. Syst. Sci.*, 72(7):1156–1171, 2006.

[49] A. Kopke, A. Willig, and H. Karl. Chaotic maps as parsimonious bit error models of wireless channels. volume 1, pages 513–523, Mar. 2003.

[50] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 78–82, New York, USA, 2004. ACM.

[51] M. Krarti. *Energy audit of building systems: An engineering approach*. CRC Press, 2011.

[52] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski. Rapid speaker adaptation in eigenvoice space. *IEEE Trans. Speech and Audio Process.*, 8(6):695–707, Nov. 2000.

[53] P. Kulkarni, D. Ganesan, and P. Shenoy. The case for multi–tier camera sensor networks. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 141–146, New York, NY, USA, 2005. ACM.

[54] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu. SensEye: a multi-tier camera sensor network. In *MULTIMEDIA 2005*, pages 229–238, New York, USA, 2005. ACM Press.

[55] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[56] B. Kusy, A. Ledeczi, and X. Koutsoukos. Tracking mobile nodes using RF doppler shifts. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys '07, pages 29–42, New York, NY, USA, 2007. ACM.

[57] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *IPSN'07: Sixth International Symposium on Information Processing in Sensor Networks*, pages 21–30, Cambridge, MA, USA, Apr. 2007. ACM/IEEE.

[58] H. Lee and P. Levis. http://tinyos.cvs.sourceforge.net/viewvc/tinyos/tinyos-2.x/tos/lib/tossim/cpmmodelC.nc.

[59] L. Lee and R. Rose. A frequency warping approach to speaker normalization. *IEEE Trans. Speech and Audio Process.*, 6(1):49–60, Jan. 1998.

[60] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9(2):171–185, Apr. 1995.

[61] V. Lenders and M. Martonosi. Repeatable and realistic experimentation in mobile wireless networks. *IEEE Transactions on Mobile Computing*, 8:1718–1728, 2009. recent article hence the low citation count.

[62] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire tinyOS applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, USA, 2003. ACM.

[63] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13:1459–1472, 2004.

[64] S. Lin, J. Zhang, G. Zhou, L. Gu, J. A. Stankovic, and T. He. ATPC: adaptive transmission power control for wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 223–236, New York, USA, 2006. ACM.

[65] T. Liu and A. Cerpa. Foresee (4C): Wireless link prediction using link features. In *IPSN'11: Proceedings of the 10th International Conference on Information Processing in Sensor Networks*, pages 294–305, Chicago, IL, USA, Apr. 2011.

[66] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, second edition, 2008.

[67] C. Metcalf. TOSSIM live: Towards a testbed in a thread. 2007.

[68] C. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Feb. 2001.

[69] Moteiv Corporation. http://www.moteiv.com/.

[70] NationMaster. http://www.nationmaster.com/encyclopedia/.

[71] G. R. Newsham and B. J. Birt. Building-level occupancy data to improve ARIMA-based electricity use forecasts. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 13–18, New York, NY, USA, 2010. ACM.

[72] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan. A trace-based approach for modeling wireless channel behavior. In *Winter Simulation Conference*, pages 597–604, 1996.

[73] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, Aug. 2006.

[74] OpenCV. http://opencvlibrary.sourceforge.net/.

[75] K. Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, 40(1):132–139, Jan. 2002.

[76] C. Qin and M. Á. Carreira-Perpiñán. Adaptation of a predictive model of tongue shapes. In *Interspeech'09*, 2009.

[77] C. Qin, M. Á. Carreira-Perpiñán, and M. Farhadloo. Adaptation of a tongue shape model by local feature transformations. In *INTERSPEECH*, 2010.

[78] M. Rahimi, D. Estrin, R. Baer, H. Uyeno, and J. Warrior. Cyclops, image sensing and interpretation in wireless networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 311–311, New York, NY, USA, 2004. ACM.

[79] B. Raman, K. Chebrolu, D. Gokhale, and S. Sen. On the feasibility of the link abstraction in wireless mesh networks. *ACM Trans. Netw.*, 17(2):528–541, Apr. 2009.

[80] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[81] D. Reddy and G. Riley. Measurement based physical layer modeling for wireless network simulations. *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, 0:46–53, 2007.

[82] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference in static wireless networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):51–62, 2006.

[83] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang. Design and analysis of sensing scheduling algorithms under partial coverage for object detection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems.*, 18:334–350, 2007.

[84] A. Rice, S. Hay, and D. Ryder-Cook. A limited-data model of building energy consumption. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 67–72, New York, NY, USA, 2010. ACM.

[85] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, Nov. 2000.

[86] T. Rusak and P. Levis. Burstiness and scaling in the structure of low-power wireless links. *SIGMOBILE Mob. Comput. Commun. Rev.*, 13:60–64, June 2009.

[87] T. Rusak and P. A. Levis. Investigating a physically-based signal power model for robust low power wireless link simulation. In *MSWiM '08: Proceedings of the 11th international symposium on Modeling, analysis and*

simulation of wireless and mobile systems, pages 37–46, New York, USA, 2008. ACM.

[88] K. Salamatian and S. Vaton. Hidden Markov modeling for network communication channels. *SIGMETRICS Perform. Eval. Rev.*, 29(1):92–101, 2001.

[89] S. Y. Seidel and T. S. Rappaport. 914 MHz path loss prediction models for indoor wireless communications in multifloored buildings. volume 40, pages 207–217, Feb. 1992.

[90] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *ACM Trans. Sen. Netw.*, 6, Mar. 2010.

[91] K. Srinivasan,, M. A. Kazandjieva,, S. Agarwal,, and P. Levis,. The β-factor: measuring wireless link burstiness. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 29–42, New York, USA, 2008. ACM.

[92] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Commun. ACM*, 47(6):34–40, 2004.

[93] T. Teixeira and A. Savvides. Lightweight people counting and localizing in indoor spaces using camera sensor nodes. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 36–43, Sept. 2007.

[94] S. Thrun and L. Y. Pratt, editors. *Learning To Learn*. Kluwer Academic Publishers Group, 1998.

[95] D. Towsley, M. Yajnik, S. B. Moon, and J. Kurose. Measurement and modeling of the temporal dependence in packet loss. In *Proc. IEEE Infocom'99*, New York, Mar. 1999.

[96] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 381–396, Berkeley, CA, USA, 2006. USENIX Association.

[97] G. Werner-Allen,, P. Swieskowski,, and M. Welsh,. Motelab: a wireless sensor network testbed. In *IPSN'05: Fourth International Symposium on Information Processing in Sensor Networks*, page 68, Piscataway, NJ, USA, 2005. IEEE Press.

[98] P. C. Woodland. Speaker adaptation for continuous density HMMs: A review. In *Adaptation Methods for Speech Recognition, ISCA Tutorial and Research Workshop (ITRW)*, 2001.

[99] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in ZebraNet. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 227–238, New York, USA, 2004. ACM Press.

[100] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys'03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13, New York, USA, 2003. ACM.

[101] Y.-J. Zheng and B. Bhanu. Adaptive object detection based on modified hebbian learning. In *Proceedings of the 13th International Conference on Pattern Recognition*, 1996.

[102] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *SECON 2004 - IEEE Sensor and Ad Hoc Communications and Networks*, pages 517–526. IEEE, 2004.