Stochastic Learning Feedback Hybrid Automata For Dynamic Power Management In Embedded Systems

by

Teodora Erbes

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Dr. Pushkin Kachroo, Co-Chair
Dr. Sandeep Shukla, Co-Chair
Dr. A. Lynn Abbott

January 19, 2004
Blacksburg, Virginia

Keywords: Power Management, Dynamic, Feedback, Learning, Linear, Non-Linear, Reinforcement, Stochastic, Hybrid, Automata, Stationary, Non-Stationary

Stochastic Learning Feedback Hybrid Automata For Dynamic Power Management In Embedded Systems

Teodora Erbes

(ABSTRACT)

Dynamic Power Management (DPM) refers to the strategies employed at system level to reduce energy expenditure (i.e. to prolong battery life) in embedded systems. The trade-off involved in DPM techniques is between the reductions of energy consumption and latency incurred by the jobs to be executed by the system. Such trade-offs need to be decided at runtime making DPM an on-line problem. In this context, the contributions of this thesis are two-fold. Firstly, we formulate the DPM problem as a hybrid automaton control problem. We model a timed hybrid automaton to mathematically analyze various opportunities in optimizing energy in a given system model. Secondly, stochastic control is added to the automata model, whose control strategy is learnt dynamically using stochastic learning automata (SLA). Several linear and non-linear feedback algorithms are incorporated in the final Stochastic Learning Hybrid Automata (SLHA) model. Simulation-based experiments show the expediency of the feedback systems in stationary environments. Further experiments are conducted using real trace data to compare stochastic learning strategies to the outcomes of several former predictive algorithms. These reveal that SLHA attains better trade-offs than the other studied methods under certain trace data. Advanced characterization of trace sequences, which allows a better performance of SLHA, is a subject of further study.

Acknowledgements

I wish to thank Dr. Pushkin Kachroo for his enthusiasm for my scholastic work and for his constructive guidance in this research. I also wish to thank Dr. Sandeep Shukla for his support and motivation and for his supervision of this project. I extend my recognition to Dr. A. Lynn Abbott for being on my advisory committee and for his guidance and support as my Teaching Assistant supervising professor. Finally, I am thankful to Dr. Sandy Irani for her assistance on the DPM simulator.

I wish to express my gratitude to my parents Aleksandra and Milan and to my brother Andreja for their support throughout these years. I extend my recognition to Nana, Baba Mara, Ija, Deda and Deda Duško for their love and care throughout the years.

I finally wish to thank all my friends here at Virginia Tech, without whom it would have been difficult to carry on a Master's Degree in Blacksburg! A special regard to Yann, Camille, Carlos, Akin, Marcela, Marissa, the "Frenchies" and the Fermatians.

This work was partly funded by NSF Grant CCR-0237947.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Overview

The increasing usage of portable, mobile, and hand-held electronic equipment has led energy efficiency to become an increasingly significant consideration in system design. Portable systems have limited energy supply, thus reducing power dissipation directly results in an extension of battery life, and consequently represents an increase in the autonomy of the device. Furthermore, power management plays an important role in conventional systems, such as servers and workstations, which have excessive consumption in energy. The escalating integration of micro-electronics to embedded systems has brought on ever rising cost implications and hence an escalating concern in power management. It can be seen from the expanding research literature that Dynamic Power Management is growing in importance, and industry efforts such as Microsoft's OnNow [7] and ACPI [8] projects illustrate a true concern in the field.

1.2. Motivation

Low power design can be accomplished at different stages of the design process. Low power design at the micro-architectural, circuit and device levels has been thoroughly researched. System level power management allows the administration of numerous system components and takes advantage of the applications characteristics to handle the optimization. System devices, such as disk drives, microphones and modems, are built with multiple power states, which are accessible for management by the operating system through industry standard APIs, such as ACPI [8], and API [9]. The power management techniques thus need to be implemented at the OS level to benefit of these power modes. Dynamic Power Management refers to strategies for reducing system level power dissipation by switching system components to lower power modes when idle, and reviving them to the active state to service incoming requests. Namely, the operating system manages the states of the devices to meet functionality requirements with the least energy expenditure.

1.3. Previous Research

Dynamic Power Management has widely been researched for deriving techniques of device administration that yield the most reduction in energy consumption with the least amount of runtime computational effort. Among the various methods, some are heuristic shutdown policies [10], prediction based shutdown policies [11] [12], multiple voltage scaling [13], and stochastic modeling based policy optimization [14] [15]. Most approaches are formulated intuitively and are then experimentally assessed to be efficient. The purpose of this research is to establish a formal methodology for systematically conceiving strategies for optimization with a theoretical foundation.

1.3.1. Background

Dynamic power management is an online problem since an algorithm that administers power management must operate with no knowledge of the future. Indeed, an embedded system only has access to the data received to date, and has no information on the complete input sequence or on its characteristics. As a result, deterministic or stochastic predictions about the future are needed.

In our model, the input to the system is the length of a request for service and inter-arrival times between requests. At the reception of a process request, the system must immediately power up to the active state to service the request. If the system is busy at the time of the request, the latter is queued and will be serviced as soon as the previous requests on queue are finished being processed. The requests are processed on a first-come-first-serve basis. While the system is idle – no requests need to be serviced – the system may choose to power down some or all of its devices as a means of lowering its energy expenditure. Consequently, the problem of power management is translated into deciding whether to switch the system devices to lower power modes while the system is idle to reduce energy dissipation while maintaining functionality requirements.

1.3.2. Different Dynamic Power Management Schemes

One technique for transitioning between modes is based on a sequence of thresholds on the idle period, each of which indicates a switch to a lower power mode. Earlier research on prediction-based dynamic power management can be classified in two categories: adaptive [11] [12] [16] [17] [18] [19] and non-adaptive. An example of non-adaptive DPM can be found in the Microsoft Windows power management facility where the user provides a threshold time after which the devices will be turned off.

However, this method is usually inflexible and does not dynamically adapt the threshold with changing application requirements.

Unlike non-adaptive frameworks, where the thresholds are pre-defined and never modified, the adaptive strategies use the knowledge about previous inputs to predict future idle periods and adjust the thresholds accordingly. In that manner, the decisions of the algorithm are adapted more closely to the actual input experienced by the system.

Most adaptive strategies base their prediction on a sequence of previous idle period lengths, and express their prediction of the next idle period with a single value. These single-value prediction schemes, like [19], transition to the power state optimal for the idle time predicted, and usually switch to the lowest power mode if the idle period extends beyond a pre-defined threshold after the predicted length. However, a problem arises when two different idle lengths are predicted to be equally likely. Indeed, the transition can be made only according to one predicted value and a penalty is endured in the case that the idle period was of the length of the other prediction.

The probability-based strategies [14] [15] [20] [21] handle this uncertainty in the prediction by discovering a probability distribution for the idle periods from the input sequence. The algorithms then base their decisions according to the characteristics of the prediction, which allows for a larger flexibility in the estimates.

Two different research approaches arise from this probabilistic strategy. A first category of algorithms assumes a certain density function for the input and sets the thresholds accordingly. A second technique attempts to learn the probability distribution online and adapts the model parameters dynamically. This second approach, however, may be computationally expensive if trying to thoroughly learn the probability distribution from the entire elapsed input set.

1.4. Approach and Contributions of this work

To start this research, we developed a mathematical model of an embedded system. From a DPM viewpoint, an embedded system is an association of discrete states, the different power modes, and continuous dynamics, the power consumed while turned on. Consequently, we model the embedded system with a hybrid automaton, which is as well composed of discrete states, the states of the automaton, and continuous dynamics, differential equations that govern the continuous variables in each state.

Further, a study of different properties of the automaton was conducted to determine the behavior of the model given different types of input environments. Results are given on the consumption of the automaton in the case of incorrect input assumptions, and conclusions are drawn vis-à-vis of the behavior to undertake given an assumed idle-period length.

Given, from the results of the preceding analyses, that power management is severely handicapped when incorrect prediction of the idle periods is assumed, control theory is subsequently added to the model to guide the automaton through power modes while the system is idle.

In the following chapters, we add stochastic control to the mathematical model to attempt to probabilistically predict the lengths of the future idle periods. Several feedback algorithms are incorporated in the final Stochastic Learning Hybrid Automata (SLHA) model, attempting to teach the automaton the characteristics of the idle periods and hence the correct behavior during idle time.

Experiments are finally conducted on the developed SLHA model to compare its results to the outcomes of several former DPM Strategies.

1.5. Plan of this Thesis

In Chapter 2, the developed Hybrid Automata mathematical model is presented, and Chapter 3 presents the analyses of the behavior of the mathematical model given input characteristics.

In Chapter 4, we present a Stochastic Learning Feedback Hybrid Automata model for DPM and describe the simulator developed to theoretically test the stochastic model. Subsequently, several learning algorithms are presented in Chapter 5. Finally, Chapter 6 presents simulation results of the SLHA model given stationary environments and real-trace inputs.

In Chapter 7, we present several former DPM strategies that have been described in literature, and we compare them to the SLHA model in Chapter 8, where we assess the results of DPM given certain inputs.

Chapter 2

Mathematical Model

In this chapter, a timed hybrid automaton is developed to mathematically model an embedded system. From a DPM viewpoint, an embedded system is an association of discrete states, the different power modes, and continuous dynamics, the power consumed while turned on. Consequently, we model the embedded system with a hybrid automaton, which is as well composed of discrete states, the automata states, and continuous dynamics, differential equations that govern the continuous variables in each state.

First, a characterization is given of all the elements of the hybrid automata model, inspired by the work on Hybrid Dynamical Systems [2]. The functioning of the automaton is then described as related to the Dynamic Power Management problem.

No specific control theory was formulated in this initial stage of the study. External control management was assumed for the purpose of developing and describing the internal behavior of the mathematical model. Analysis of the control synthesis is detailed in the subsequent chapters of this thesis.

2.1. Hybrid Systems: Definition

Hybrid systems are typically described by continuous and discrete dynamics. These dynamics interact in the hybrid automata model, inducing changes in the system in response to both the continuous parameters, described by differential equations, and instantaneous events, characterizing the discrete states.

2.1.1. Components of a Hybrid Automaton

The hybrid automata model consists of continuous dynamics, generally expressed by differential equations or difference equations in time, and discrete control steps, in the form of finite automata. Evidently, these dynamics can be described by more general representations, such as partial or stochastic differential equations, pushdown automata or Turing machines, at the cost of the complexity of the involved decision problems.

The following gives a detailed description of the components of a hybrid system.

a) Continuous-Time State-Space Model

Continuous-time state-space systems are described by a set of state variables x and a set of external variables w, associated by a combined set of differential and algebraic equations of the form

$$F(x, \dot{x}, w) = 0$$

- $x \in X$, an n-dimensional state-space X , usually $\mathbb{R}^n$,
- $w \in \mathbb{R}^q$,
- $\dot{x} = dx/dt$

Solutions to $F(x, \dot{x}, w) = 0$ are all the sufficiently smooth time functions $x(t)$ and $w(t)$ such that $F(x(t), \dot{x}(t), w(t)) = 0$ for almost all times $t \in \mathbb{R}$, the continuous-time axis. It is reasonably required that $x(t)$ be continuous piecewise-differentiable and $w(t)$ be piecewise continuous. $F(x, \dot{x}, w) = 0$ must then be satisfied for all $t \in \mathbb{R}$, except at the points of non-differentiability of $x(t)$ and at the discontinuities of $w(t)$.

b) Finite Automata Model

Finite automata are defined as a triplet (L, A, E) where

- L , state space, is a finite set,
- A , alphabet, is a finite set containing symbols,
- $E \subset L \times A \times L$ is the transition rule containing transitions named edges or events.

A trajectory, or path, defines a sequence $(l_0, a_0, l_1, a_1, ..., l_{n-1}, a_{n-1}, l_n)$, with $(l_i, a_i, l_{i+1}) \in E$ for $i = 1, 2, \ldots, n-1$.

A finite automaton is illustrated by a graph whose vertices, the states, are elements of L . The edges, representing transitions between the states, are specified by arrows, elements of E , and are labeled by the symbols of A . Figure 2.1 gives an example of such a graph.

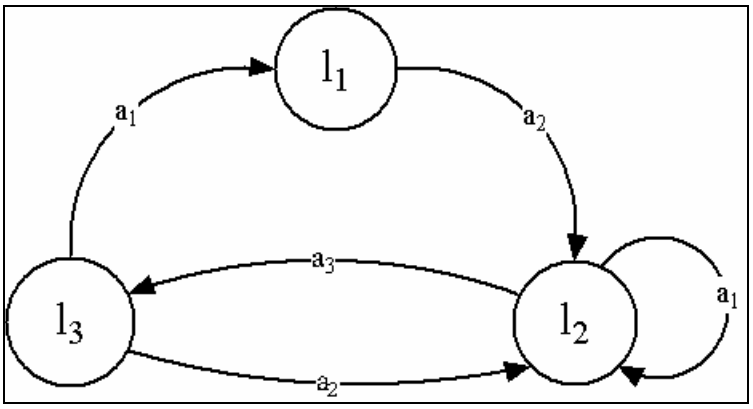


Figure 2.1: Finite Automaton

Deterministic input-output automata, as illustrated in Figure 2.2, are finite automata that have two symbols associated with every transition: an input symbol $i \in A$ and an output symbol $o \in A$. Such automata are represented by a quintuple (L, I, A, E, F), where $I \subset L$ is a set of initial states, and $F \subset L$ is a set of final states. It is required that for any state of the system at most one transition launched from that state be associated with every input symbol.

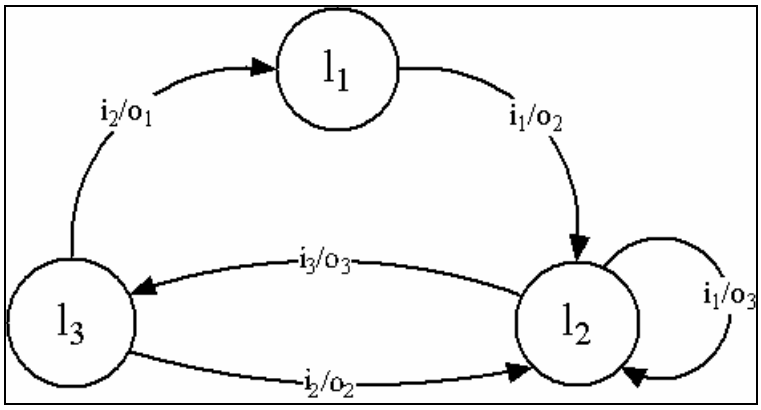


Figure 2.2: Finite Input-Output Automaton

The following equations can be used to specify such automata:

- $l^{\#} = \upsilon(l, i)$
- $o = \eta(l, i)$

At the time of an event i, these equations respectively determine the next state $l^{\#}$ and output o of the automaton, according to its current state.

A path $(l_0, a_0, l_1, a_1, ..., l_{n-1}, a_{n-1}, l_n)$, where $l_0 \in I$ and $l_n \in F$, is called a successful path.
Unlike for continuous-time systems, the notion of solution to a finite automaton is completely defined: it is a collection of the sequences of inputs of all successful paths.

c) Analogies between the two models

It may be observed that the continuous-time state-space model and finite automata present analogies. These are summarized in table 2.1.

Table 2.1. Analogies between Continuous-Time State-Space Model and Finite Automata

Continuous-Time State-Space Model	Finite Automaton Model
State Space X	State Space L
Space W	Symbol Alphabet A
Set of Equations $F(x, \dot{x}, w) = 0$	Set of Transition Rules E
Solutions to the Set of Equations	Successful Paths
$\frac{d(\ )}{dt}$ differentiation operator	$(\ )^{\#}$ next-state operator

2.1.2. Hybrid Automata Model

In this section, a hybrid automaton is defined and description is given of its properties and behavior.

a) Definition of Hybrid Automata

A definition of a hybrid automaton can be formulated by merging the previously stated definitions of continuous-time state-space models and finite automata.

Hybrid automata are described by a septuple $(L, X, A, W, E, Inv, Act)$ as follows:

- L is a finite set of discrete states or locations, illustrated by vertices on the graph.
- X is an n-dimensional continuous state space : $X \subset \mathbb{R}^n$,
  - $x \in X$ is the continuous state variable.
- A, the alphabet, is a finite set of symbols. The latter operate as labels of the edges of the graph.
- $W = \mathbb{R}^q$ is the continuous communication space,
  - $w \in W$ is the continuous external variable.
- E is the finite set of edges, named transitions or events. An edge is expressed as a quintuple $(l, a, Guard_{ll'}, Jump_{ll'}, l')$, where $(l, l') \in L$, $a \in A$, $Guard_{ll'} \subset X$, and $Jump_{ll'} \subset X \times X$.
  - Jumping from l to l' is allowed only when $x \in Guard_{ll'}$, and under the condition that $(x, x') \in Jump_{ll'}$ when x changes to x' during the transition.
- Inv maps the locations $l \in L$ to a subset of X : $Inv(l) \subset X$, $\forall l \in L$.
  - $Inv(l)$ is the location invariant of l, such that if $x \in Inv(l)$ when the system is in l.
- $Act(l)$ assigns $\forall l \in L$ a set of differential and algebraic equations $F_l(x, \dot{x}, w) = 0$, whose solutions are called the activities of l.

In summary, the state of a hybrid automaton is expressed by a discrete component $l \in L$ and a continuous variable $x \in X$. Similarly, the external factors consist of a discrete part

$a \in A$, and of a continuous parameter $w \in \mathbb{R}^q$. Finally, the dynamics are also articulated by discrete transitions from state to state, and continuous evolutions in the locations. Figure 2.3 illustrates an example of a hybrid automata model.

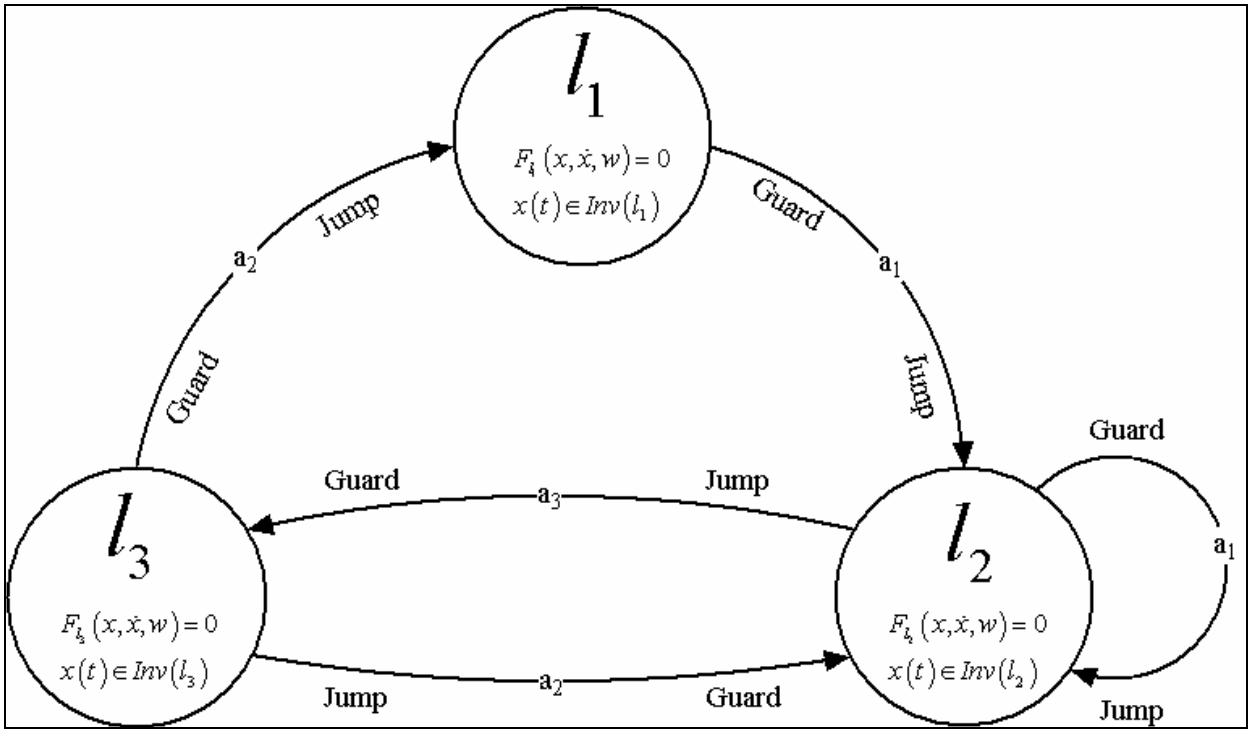


Figure 2.3: Hybrid Automaton

b) Solutions to Hybrid Automata

Similarly to finite automata solutions, the solutions to a hybrid automaton are trajectories, infinite sequences of continuous trajectories.

A continuous trajectory (l, δ, x, w) originating from location l is expressed by

- a time $\delta \geq 0$, the length of the continuous trajectory,
- a piecewise-continuous function $w : [0, \delta] \to W$,
- a continuous piecewise-differentiable function $x : [0, \delta] \to X$,

satisfying

- $x(t) \in Inv(l) \;\; \forall t \in (0, \delta)$,
- $F_l\left(x(t), \dot{x}(t), w(t)\right) = 0 \;\; \forall t \in (0, \delta)$ except at the discontinuities of w.

Consequently, a solution to the hybrid automaton is a succession

$$(l_0, \delta_0, x_0, w_0) \xrightarrow{a_0} (l_1, \delta_1, x_1, w_1) \xrightarrow{a_1} (l_2, \delta_2, x_2, w_2) \xrightarrow{a_2} \cdots$$

where at every event time

$$t_i = \sum_{i=0}^{\infty} \delta_i \;\; \forall i \in [0, \infty),$$

a discrete symbol a_i is associated with the i^{th} discrete transition, and the discrete transitions satisfy the following conditions:

$$x_i(t_i) \in Guard_{l_j l_{j+1}},\ \left(x_i(t_i), x_{i+1}(t_{i+1})\right) \in Jump_{l_i l_{i+1}}.$$

c) Properties of Hybrid Dynamics

Beginning in the initial state, the continuous state variable changes following the continuous dynamics specified for the state, while remaining in the location invariant. At an event at time $t \in \mathbb{R}$, an instantaneous discrete transition occurs, provided that the guard condition for the transition is fulfilled. A jump may also take place in the continuous dynamics. After the discrete change of location, the continuous variable evolves following the dynamics of the new location. This process occurs indefinitely, as long as the conditions for staying in a state (location invariant), or transitioning to a new state (guard, jump) are satisfied.

An event can occur in two ways. "Externally" induced events are stimulated by the discrete external variables (symbols), producing controlled transitioning. Alternatively, events can be "internally" induced, provoked by the non-fulfillment of the location invariants, or by the attainment of the guard conditions. Location invariants therefore

provide “enforcing conditions”, as opposed to guards that provide “enabling conditions”.

d) Analysis of Solutions

Deadlock is the most verified property in automata models. It is detected when at a given time $t \in \mathbb{R}$ no transition is possible from the current state of the system. This is a generally detrimental situation since the system is caught in one location and no more evolution is possible. However, if the last state of the system belongs to the set of final states, deadlock signifies that the path taken is a successful path, and hence identifies a solution to the automaton.

Similarly, a system may incur livelock, which denotes that it is indefinitely transitioning between multiple states. This may occur when the continuous trajectories are of zero time. In such cases, the location invariants are usually not satisfied while the guard conditions to the previous locations are valid.

2.2. Timed Hybrid Automata Model for DPM

In this research, a timed hybrid automaton has been studied to model a system with multiple power-down modes. The discrete states of the hybrid automata are used to model the power modes of the system, while the continuous dynamics account for the power consumed in each mode.

Following a detailed analysis of the different components of the model, the operation of the studied timed hybrid automaton is expressed in detail.

2.2.1. Components of the DPM Model

Figure 2.4 illustrates the timed hybrid automata model developed for the study of DPM.

a) States of the Automaton

There are $n+1$ main states in the model, each representing a power mode of the system. State S_0, the Active state, is the initial state of the system. The system needs to be in this main state to process requests. The states labeled $S_i \quad \forall i \in [\![1, n]\!]$ represent the lower power modes of the system. The states are ordered from highest power consumption to lowest power consumption, such that the lowest power mode of the system is represented by the state with the highest index: S_n. Three constants are associated with each state:

- P_i, the Power Consumption, is the power consumed while in state S_i,
- E_i, the Start-Up Energy, is the energy required to power-up from S_i to S_0,
- t_i, the Start-Up Time, is the time that is takes the system to activate.

The following classifications are implied:

$\forall i, j$ where $j > i$

- $P_i > P_j$,
- $E_i < E_j$,
- $t_i < t_j$,

such that the states with lower power consumptions have higher start-up energies and times.

In addition, the model includes states labeled S_{i0} and S_{ir}, $\forall i \in [\![1,n]\!]$. These states represent intermediate states for transitioning from a lower power state S_i to the active state S_0. The power consumed in these states is $k_i = E_i / t_i$.

States S_{ir} are transitory states for powering-up from S_i to S_0, when no requests are waiting to be processed.

Similarly, states S_{ir} are intermediate states for transitioning from S_i to S_0, but in the presence of requests waiting to be processed.

b) Variables of the System

There are two categories of internal variables used in the model. The dynamically updated variables e, the total energy expedited by the system, and t, the temporary clock, are governed by the continuous dynamics specific to each state. t_L is a variable that accounts for the total latency incurred by requests in the system. It is updated at each input arrival when the system is not ready to process the request immediately. t_R is a semi-permanent cumulative request length. It accounts for the time during which the system needs to remain in the active state to process each batch of requests between two idle periods. The latter two variables are automatically updated at each input arrival.

Furthermore, the external parameter u is the control variable of the model. It manages the sequence of states that the system follows during an idle period.

Finally, r is an external symbol representing a request arrival and evaluating to the length of the process request.

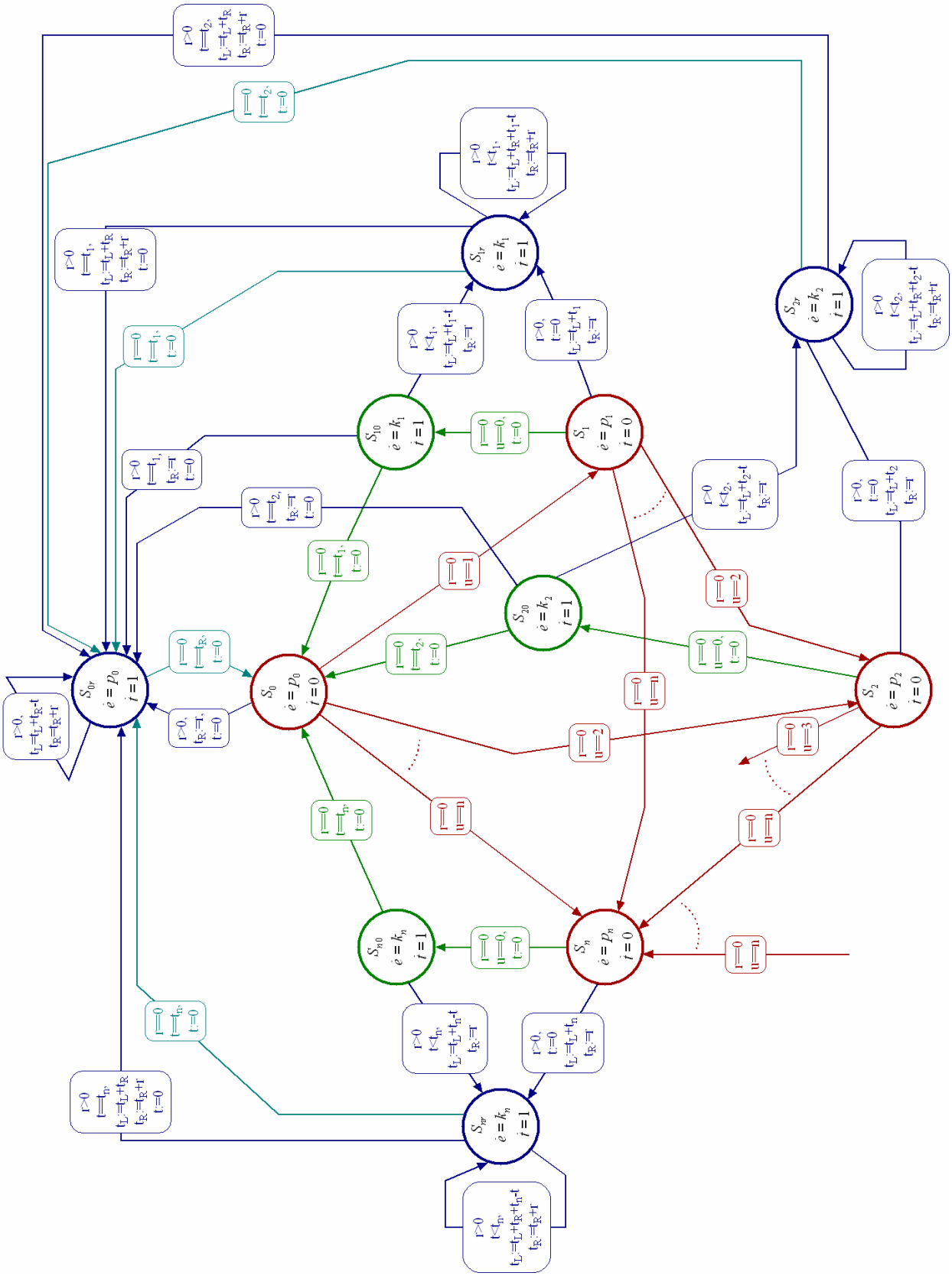


Figure 2.4: Timed Hybrid Automaton Model

c) Further Properties

It is assumed in this DPM problem that transitioning to any lower power state is allowed from any state of the system. Powering-up is however only allowed towards the active state. In other words, there is no transition allowed from a low-power state to another low-power state, of higher power consumption. This constraint is added to the DPM problem to simplify the control theory which will be used to manage the system during idle periods.

Moreover, the transition to a lower-power state is assumed to be immediate and energy-less. This may be considered because, by analyzing different devices with multiple power-down modes, it can be observed that there is usually negligible energy and latency involved in the powering-down of devices, but a significant amount of both energy and time may be required to revive the given device to the active state, depending on the level of "deepness" of the particular power mode. It can therefore be argued that if non-negligible energy and/or time costs occur during the transitioning to a lower power mode, these can easily be included in the power-up constraints if the former are additive. Namely, it is necessary that the energy and time costs for powering down from state A to state B, added to the consumption for switching down from state B to state C, be equal to the consumption for transitioning down from state A directly to state C.

2.2.2. Formal Description of the Timed Hybrid Automaton

As stated earlier, a hybrid automaton is described by a septuple $(L, X, A, W, E, Inv, Act)$.

Following is a description of the model studied:

- $L = \{S_i\}_{i=0}^{n} \cup \{S_{i0}\}_{i=1}^{n} \cup \{S_{ir}\}_{i=0}^{n}$
- $X = \{e, t\}$
- $A = \{r\}_{r=0}^{\infty}$
- $W = \varnothing$
- $E = \{E_{ll'} | l, l' \in L\}$. $E_{ll'} : (l, r, Guard_{ll'}, Jump_{ll'}, l')$
  - $Guard_{ll'}$ are the test conditions $r > 0$ and $r == 0$,
  - $Jump_{ll'}$ are the assignments such as $t_R := r$, $t := 0$, or $t_L := t_L + t_R$.
- $Inv(l) = [0, t_i] \ \forall l = S_{ir} \mid S_{i0} \ \forall i \in [\![1, n]\!]$, $Inv(S_{0r}) = [0, t_R]$
- $Act(S_i) : \begin{Bmatrix} \dot{e} = p_i \\ \dot{t} = 0 \end{Bmatrix} \ \forall i \in [\![0, n]\!]$,

  $Act(l) : \begin{Bmatrix} \dot{e} = k_i \\ \dot{t} = 1 \end{Bmatrix} \ \forall l = S_{ir} \mid S_{i0}$

  $\forall i \in [\![1, n]\!]$, $Act(S_{0r}) : \begin{Bmatrix} \dot{e} = p_0 \\ \dot{t} = 1 \end{Bmatrix}$

2.2.3. Operation of the DPM Hybrid Automata Model

The system is originally in S_0, the initial state. While there are no requests to be processed, the sequence of states is governed by the control variable u. According to the value of u, the model will remain in the current state, switch to a lower power mode, or power-up to the active state. If the model is ordered to transition to the active state when in state S_i, it will remain in state S_{i0} for t_i time before becoming active.

At the arrival of a request, the system will immediately transition from the current state S_i to state S_{ir}. If the current state was the active state, the system will begin processing the request immediately. Otherwise, it will remain in state S_{ir} for t_i time before

becoming active. The global counter for the total latency endured by the system t_L will reflect the waiting time of the request before it is processed. If the system was already in an intermediate state S_{i0} when the request arrived, it will transition to state S_{ir}, but will only remain there for time $(t_i - t)$, the time required to activate from state S_i less the time already spent in the intermediate state S_{i0}.

In the case of an additional input arriving before the previous request was finished being processed, t_R is updated to account for the length of the new request added to the remaining length of time required to service the previous inputs. Again, t_L will reflect the latency endured by the waiting request.

2.2.4. Control Variable u

As mentioned earlier, the control variable u governs the sequence of states through which the system will transition during idle periods. Depending on the value of u, the model will remain in its current state or shift to a different power mode. The next objective of this research is to devise a method for choosing a value for u at every instant of time.

For the purpose of the mathematical model, the control variable is assumed to be handled externally, and focus is given to the analysis of the internal behavior of the hybrid automaton. In a following chapter, a solution is proposed for the management of the control variable, which uses probabilities for deciding on the switching of states.

Chapter 3

Analysis of the Mathematical Model

In this chapter, mathematical properties are derived from the study of the developed Timed Hybrid Automata model. The behavior of the automata is examined as presented different input distributions, and the consequences of incorrect predictions of the idle period length are analyzed. Conclusions are finally given on the approaches to take for power management.

3.1. Different Procedures for Activating

Two different power-up strategies can be followed by a system governed by power management: wake-up "On Demand" and "Preemptive" activation. These directly affect the cost of switching to a lower power state during an idle period. Indeed, for a DPM problem, consumption encompasses both the energy expenditure in the power mode, which is the energy required to activate, and the latency incurred by an arrived request that waits to be processed.

The following analysis assumes that the length of the idle period is known in advance.

3.1.1. Wake-Up "On Demand"

The first power-up approach, wake-up "On Demand", is to switch to a lower power mode S_i at the start of the idle period, and remain there until receipt of a request. The system must then immediately switch to the active state S_0, transitioning through the intermediate "waiting" state S_{i0}. Figure 3.1 illustrates the time diagram of this method.

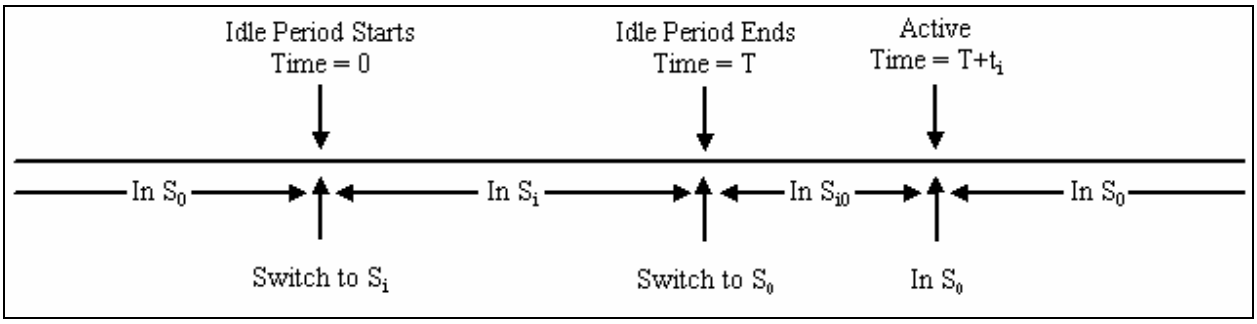


Figure 3.1: Time Diagram for "On Demand" Approach

The cost incurred for the entire idle period is:

$$C_{iW} = E \cdot \left(p_i \cdot T + e_i \right) + L \cdot t_i \,,\ \forall i \in [\![0, n]\!]$$

where

- E and L correspond to the weights of energy and latency respectively,
- p_i, e_i, and t_i are the power, energy and time parameters of S_i,
- T is the length of the idle period.

This cost corresponds to the power consumed in state S_i during T time, added the energy required to power up the system to the active state. The cost also encompasses

the latency incurred by the system, which is the time during which the process is waiting to be serviced because the system is not active.

Constants E and L are weights introduced in the equation of cost to adjust for the difference in units between the cost of energy, expressed in Joules, and the cost of latency, expressed in milli-seconds. To include these weights in the equation provokes the value of cost to be unitless.

3.1.2. "Preemptive" Wake-Up

Like for the first method, in this second power-up approach, "Preemptive" activation, the system powers down to S_i when idle. However, it remains there for a lesser period of time, powering-up to S_0 in order to be active at the arrival of the request. Figure 3.2 illustrates the timeline for this approach.

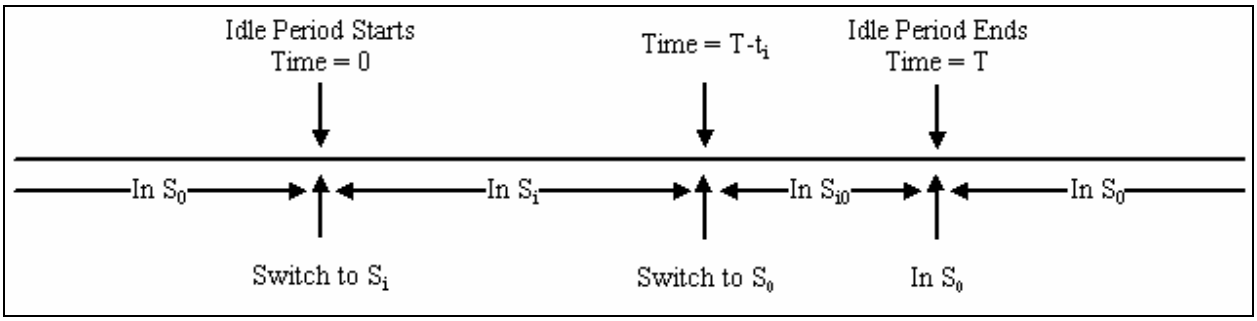


Figure 3.2: Time Diagram for "Preemptive" Approach

The cost experienced by the system for the idle period is:

$$C_{iA} = E \cdot \left(p_i \cdot \left(T - t_i \right) + e_i \right), \ \forall \left\{ i \in [\![0, n]\!] \middle| T - t_i \geq 0 \right\}$$

$$\text{if } T - t_i < 0 \text{ , } C_{iA} = E \cdot e_i - L \cdot \left(T - t_i \right)$$

where the parameters are as defined in the previous section.

Two different calculations of the cost are possible with this approach.

If the idle period is longer than the time that it takes to activate the system from the lower-power state, the cost corresponds to the power consumption in the low-power state for the time during which the system is in that state, added the energy required to activate. There is no latency cost in this situation because the system becomes active at the time of the request.

However, if the idle period is shorter than the time that it takes to activate the system from the lower-power state, the cost then corresponds to the energy required to activate added the latency incurred by the request waiting to be processed. In this case, it is assumed that the system immediately activates after reaching the low-power state. There is therefore no power consumed in the low-power state.

3.1.3. Cost of Waiting vs. Cost of Activating Ahead

By comparing the costs of the system in each case, it can easily be shown that the cost of activating ahead, "Preemptive" wake-up, is always smaller than the cost of waiting, "On Demand" wake-up:

$$\forall T,\ C_{iA} \leq C_{iW}$$

If $T - t_i \geq 0$:

$$C_{iA} = E \cdot p_i \cdot T - E \cdot p_i \cdot t_i + E \cdot e_i,\ \forall i \in [\![0, n]\!]$$

$$C_{iW} = E \cdot p_i \cdot T + E \cdot e_i + L \cdot t_i,\ \forall i \in [\![0, n]\!]$$

$$C_{iW} - C_{iA} = (E \cdot p_i + L) \cdot t_i \geq 0$$

If $T - t_i < 0$,

$$C_{iA} = E \cdot e_i - L \cdot T + L \cdot t_i,$$

$$C_{iW} = E \cdot p_i \cdot T + E \cdot e_i + L \cdot t_i$$

$$C_{iW} - C_{iA} = \left(E \cdot p_i + L\right) \cdot T \geq 0$$

Since all the terms in both expressions are positive, $C_{iW} - C_{iA} \geq 0$. It is therefore always more feasible to jump ahead to S_0 in order to be active at the time of the request.

For the purpose of future experiments and comparisons of the Hybrid Automata model to former DPM strategies presented in the literature, both "Preemptive" wake-up and wake-up "On Demand" will be studied in this research.

3.2. Choice of the Optimal State given the Idle Period Length

In this section, study is conducted on choosing the optimal state given the length of the idle period.

3.2.1. Cost of Switching to S_i vs. Cost of Switching to S_j

Figure 3.3 illustrates the automaton for the following study. In this section, we analyze the relative cost of switching to two different power modes given an idle period length. We will first examine the costs for the "Preemptive" wake-up method, followed by the similar study for the "On Demand" case.

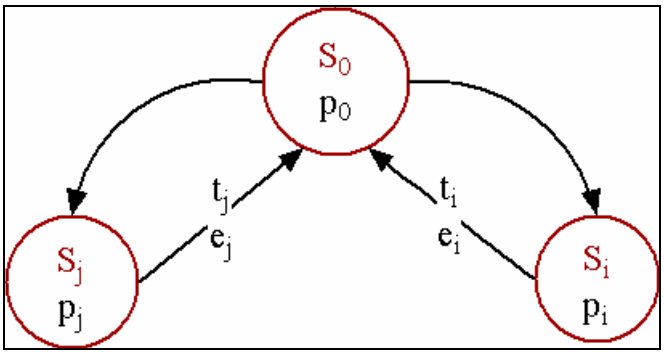


Figure 3.3: 3-State Automaton

a) "Preemptive" wake-up

With "Preemptive" wake-up, following are C_i and C_j, the costs of switching to S_i and S_j respectively for a given idle period of length T.

$$C_i = E \cdot \left(p_i \cdot (T - t_i) + e_i \right), \ \forall \left\{ i \in [\![0, n-1]\!] \middle| T - t_i \geq 0 \right\},$$

$$C_i = E \cdot e_i - L \cdot (T - t_i) \ \ \forall \left\{ i \in [\![0, n-1]\!] \middle| T - t_i < 0 \right\}$$

$$C_j = E \cdot \left(p_j \cdot (T - t_j) + e_j \right), \ \forall \left\{ j \in [\![1, n]\!] \middle| j > i, T - t_j \geq 0 \right\},$$

$$C_j = E \cdot e_j - L \cdot (T - t_j) \ \ \forall \left\{ j \in [\![1, n]\!] \middle| j > i, T - t_j < 0 \right\}$$

When comparing the two costs in every case, the following results encourage the system to switch to S_i for the minimization of consumption, if the given conditions on the length of the idle period are satisfied. It is otherwise preferable to switch to S_j for the given idle period.

If $T - t_i \geq 0$ and $T - t_j \geq 0$

$$C_i < C_j \quad \Leftrightarrow \quad E \cdot p_i \cdot T - E \cdot p_i \cdot t_i + E \cdot e_i < E \cdot p_j \cdot T - E \cdot p_j \cdot t_j + E \cdot e_j$$

$$\Leftrightarrow \quad E \cdot (p_i - p_j) \cdot T - E \cdot (p_i \cdot t_i - p_j \cdot t_j) + E \cdot (e_i - e_j) < 0$$

$$\Leftrightarrow \qquad 0 \le T < \frac{\left(p_i \cdot t_i - p_j \cdot t_j\right) - \left(e_i - e_j\right)}{p_i - p_j}$$

If $T - t_i \ge 0$ and $T - t_j < 0$

$$C_i < C_j \qquad \Leftrightarrow \qquad E \cdot p_i \cdot T - E \cdot p_i \cdot t_i + E \cdot e_i < E \cdot e_j - L \cdot T + L \cdot t_j$$

$$\Leftrightarrow \qquad T < \frac{L \cdot t_j + E \cdot p_i \cdot t_i - E \cdot \left(e_i - e_j\right)}{L + E \cdot p_i}$$

If $T - t_i < 0$ and $T - t_j < 0$

$$C_i < C_j \qquad \Leftrightarrow \qquad E \cdot e_i - L \cdot T + L \cdot t_i < E \cdot e_j - L \cdot T + L \cdot t_j$$

$$\Leftrightarrow \qquad \frac{e_j - e_i}{t_j - t_i} > -\frac{L}{E} \text{ is always true}$$

$$\Leftrightarrow \qquad \forall T \ge 0$$

Consequently, if the idle period is longer than the start-up time of state S_i, the cost of switching to S_i is lower than the cost of switching to S_j if and only if the corresponding conditions on T are satisfied. If, however, the idle period is shorter than the start-up times of both states, it is then always more feasible to switch to S_i. In other words, in a multi-state system, decision should be made to remain active for the idle period T, in this case.

b) "On Demand" wake-up

With wake-up "On Demand", C_i and C_j, the costs of switching to S_i and S_j respectively for a given idle period of length T are:

$$C_i = E \cdot \left(p_i \cdot T + e_i\right) + L \cdot t_i, \ \forall i \in [\![0, n-1]\!]$$

$$C_j = E \cdot \left(p_j \cdot T + e_j\right) + L \cdot t_j, \ \forall j \in [\![1, n]\!] \,|\, j > i$$

Again, when comparing the two costs in every case, the following results encourage the system to switch to S_i for the minimization of consumption, if the given conditions on the length of the idle period are satisfied. It is otherwise preferable to switch to S_j for the given idle period.

$$
\begin{aligned}
C_i < C_j \quad &\Leftrightarrow \quad E \cdot p_i \cdot T + E \cdot e_i + L \cdot t_i < E \cdot p_j \cdot T + E \cdot e_j + L \cdot t_j \\
&\Leftrightarrow \quad E \cdot \left(p_i - p_j \right) \cdot T - E \cdot \left(e_j - e_i \right) - L \cdot \left(t_j - t_i \right) < 0 \\
&\Leftrightarrow \quad 0 \le T < \frac{E \cdot \left(e_j - e_i \right) + L \cdot \left(t_j - t_i \right)}{E \cdot \left(p_i - p_j \right)}
\end{aligned}
$$

Consequently, if the idle period satisfies the given constraints, the cost of switching to S_i is lower than the cost of switching to S_j.

3.2.2. Cost of Switching to S_i followed by S_j vs. Cost of Switching Immediately to S_j

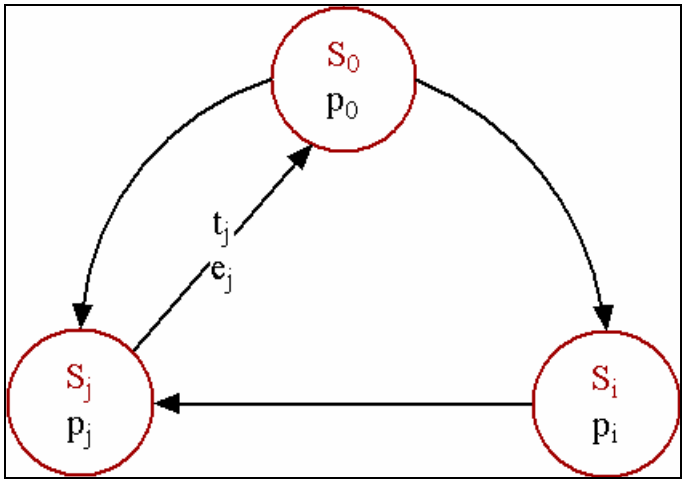


Figure 3.4: 3-State Automaton

In this section, analysis is made on the relative cost of switching through two low-power states, versus the cost of switching to the second state directly, during an idle period of known length. No assumption is made on the optimal state for the given idle period.

Figure 3.4 illustrates the automaton for this analysis.

a) "Preemptive" wake-up

From this section on, only the most common case, where $T - t_i \geq 0 \quad \forall i \geq 0$, will be considered when examining the "Preemptive" wake-up method. It can easily be proven that the results obtained also hold for the other cases identified earlier.

As previously defined, C_i and C_j are the costs of switching to S_i and S_j respectively for a given idle period of length T. C_{ij} is the cost of switching and remaining in S_i for $0 < T_i \leq T$ time , and then transitioning to S_j for the remainder of the idle period.

$T_j = T - T_i - t_j$ where $j > i$, is the time during which the system will remain in state S_j before activating. It is equal to the idle period length less the time during which it will remain in state S_i, and less the start-up time of state S_j.

Two cases arise for this study: it is possible for the system to remain in S_j before activating, or to immediately activate at the reach of S_j, therefore enduring no power consumption in this second state.

$\forall i \in [\![0, n-1]\!],\ \forall j \in [\![1, n]\!] \,|\, j > i$,

If $T - t_i \geq 0$ and $T_j \geq 0$,

$$C_i = E \cdot \left(p_i \cdot (T - t_i) + e_i \right),$$

$$C_j = E \cdot \left(p_j \cdot (T - t_j) + e_j \right) = E \cdot p_j \cdot T - E \cdot p_j \cdot t_j + E \cdot e_j$$

$$C_{ij} = E \cdot \left(p_i \cdot T_i \right) + E \cdot \left(p_j \cdot T_j + e_j \right) = E \cdot p_i \cdot T_i + E \cdot p_j \cdot T - E \cdot p_j \cdot T_i - E \cdot p_j \cdot t_j + E \cdot e_j$$

$$\Leftrightarrow C_{ij} = C_j + E \cdot T_i \left(p_i - p_j \right)$$

$$\Rightarrow C_{ij} > C_j \text{ since } p_i > p_j$$

If $T - t_i \geq 0$ and $T_j < 0$

$$C_i = E \cdot \left(p_i \cdot (T - t_i) + e_i \right),$$

$$C_j = E \cdot e_j - L \cdot \left(T - t_j \right) = E \cdot e_j - L \cdot T + L \cdot t_j,$$

$$C_{ij} = E \cdot \left(p_i \cdot T_i \right) + E \cdot e_j - L \cdot T_j = E \cdot p_i \cdot T_i + E \cdot e_j - L \cdot T + L \cdot T_i + L \cdot t_j$$

$$\Leftrightarrow C_{ij} = C_j + T_i \left(E \cdot p_i + L \right)$$

$$\Rightarrow C_{ij} > C_j \text{ since } p_i > 0$$

It is therefore always less costly to switch directly to a low-power state, than to transition through another power state beforehand.

b) "On Demand" wake-up

As in the previous section, C_i and C_j are the costs of switching to S_i and S_j respectively for a given idle period of length T. C_{ij} is the cost of switching and remaining in S_i for $0 < T_i \leq T$ time , and then transitioning to S_j for the remainder of the idle period.

$T_j = T - T_i$ where $j > i$, is the time during which the system will remain in state S_j before activating. It is equal to the idle period length less the time during which it will remain in state S_i.

$\forall i \in [\![0, n-1]\!],\ \forall j \in [\![1, n]\!] \,|\, j > i$,

$$C_i = E \cdot (p_i \cdot T + e_i) + L \cdot t_i,$$
$$C_j = E \cdot (p_j \cdot T + e_j) + L \cdot t_j,$$
$$C_{ij} = E \cdot (p_i \cdot T_i + p_j \cdot T_j + e_j) + L \cdot t_j = E \cdot p_i \cdot T_i + E \cdot p_j \cdot T_j + E \cdot e_j + L \cdot t_j$$
$$\Leftrightarrow C_{ij} = C_j + E \cdot p_i \cdot T_i$$
$$\Rightarrow C_{ij} > C_j$$

As with the "Preemptive" strategy, it is always less costly to switch directly to a low-power mode, than to transition through another power state beforehand.

3.2.3. Cost of Switching through a Sequence of Power States

This analysis is a generalization of the previous study. We will show that the cost of transitioning through a sequence of states is always greater than the cost of switching directly to the lowest power mode in the sequence:

$C_{ij\ldots n} > C_n$ where $i < j < \cdots < n$

Figure 3.5 illustrates the automaton for the following analysis.

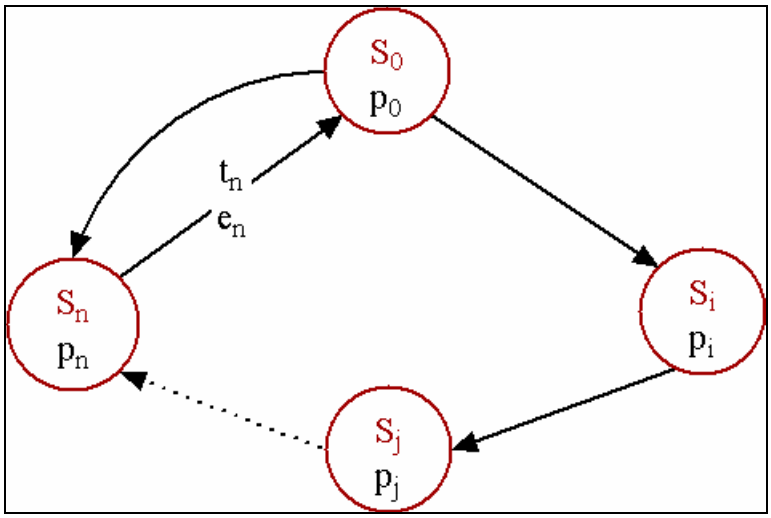


Figure 3.5: n-State Automaton

a) "Preemptive" wake-up

We consider in this analysis that T_k is the time spent in each state S_k $\forall k \in [\![i, n-1]\!]$ such that $0 < \sum_{k=i}^{n-1} T_k \leq T$, and that $T_n = T - t_n - \sum_{k=i}^{n-1} T_k$ is the time spent in the last state S_n in the sequence. Moreover, C_n is the cost of switching directly to S_n, and $C_{ij\ldots n}$ is the cost of switching through a sequence of power modes until reaching S_n.

Two cases arise again in this study: it is possible for the system to remain in S_n before activating, or to immediately activate at the reach of S_n.

If $T - t_k \geq 0$ $\forall k \in [\![0, n-1]\!], k < n$, $T_n > 0$

$$C_n = E \cdot \left(p_n \cdot (T - t_n) + e_n \right),$$

$$C_{ij\ldots n} = \left[\sum_{k=i}^{n-1} E \cdot \left(p_k \cdot T_k \right) \right] + E \cdot \left(p_n \cdot T_n + e_n \right)$$

$$\Rightarrow C_{ij\ldots n} - C_n = \left[\sum_{k=i}^{n-1} E \cdot \left(p_k \cdot T_k \right) \right] + E \cdot \left(p_n \cdot \left(T_n - T + t_n \right) \right)$$

$$\Leftrightarrow C_{ij\ldots n} - C_n = \left[\sum_{k=i}^{n-1} E \cdot \left(p_k \cdot T_k\right)\right] - E \cdot p_n \cdot \sum_{k=i}^{n-1} T_k$$

$$\Leftrightarrow C_{ij\ldots n} - C_n = E \cdot \left[\sum_{k=i}^{n-1} \left(p_k \cdot T_k\right) - \sum_{k=i}^{n-1} \left(p_n \cdot T_k\right)\right] > 0$$

because $\forall k < n,\ p_n < p_k$

If $T - t_k \geq 0 \quad \forall k \in [\![0, n-1]\!], k < n,\ T_n \leq 0$

$$C_n = E \cdot e_n - L \cdot \left(T - t_n\right) = E \cdot e_n - L \cdot T + L \cdot t_n,$$

$$C_{ij\ldots n} = \left[\sum_{k=i}^{n-1} E \cdot \left(p_k \cdot T_k\right)\right] + E \cdot e_n - L \cdot T_n = \left[\sum_{k=i}^{n-1} E \cdot \left(p_k \cdot T_k\right)\right] + E \cdot e_n - L \cdot T + L \cdot \sum_{k=i}^{n-1} T_k + L \cdot t_n$$

$$\Rightarrow C_{ij\ldots n} - C_n = \left[\sum_{k=i}^{n-1} E \cdot \left(p_k \cdot T_k\right)\right] + L \cdot \sum_{k=i}^{n-1} T_k$$

$$\Leftrightarrow C_{ij\ldots n} - C_n = \sum_{k=i}^{n-1} \left[T_k \cdot \left(E \cdot p_k + L\right)\right] > 0$$

because $\forall k < n,\ p_k > 0$

Hence $C_{ij\ldots n} > C_n \quad \forall i < j < \cdots < n$

It is proven from these equations that it is always more costly to transition through a sequence of power states than to immediately switch to the lowest power state in the sequence.

b) "On Demand" wake-up

Likewise, we consider in the "On Demand" analysis that T_k is the time spent in each state $S_k \quad \forall k \in [\![i, n-1]\!]$ such that $0 < \sum_{k=i}^{n-1} T_k \leq T$, and that $T_n = T - \sum_{k=i}^{n-1} T_k$ is the time spent in

the last state S_n in the sequence. Moreover, C_n is the cost of switching directly to S_n, and $C_{ij\ldots n}$ is the cost of switching through a sequence of power modes until reaching S_n.

$\forall k \in [\![0, n-1]\!], k < n$

$$C_n = E \cdot (p_n \cdot T + e_n) + L \cdot t_n,$$

$$C_{ij\ldots n} = \left[\sum_{k=i}^{n-1} E \cdot (p_k \cdot T_k) \right] + E \cdot (p_n \cdot T_n + e_n) + L \cdot t_n$$

$$\Rightarrow C_{ij\ldots n} - C_n = \left[\sum_{k=i}^{n-1} E \cdot (p_k \cdot T_k) \right] + E \cdot (p_n \cdot T_n + e_n) + L \cdot t_n - E \cdot (p_n \cdot T + e_n) - L \cdot t_n$$

$$\Leftrightarrow C_{ij\ldots n} - C_n = E \cdot \left[\sum_{k=i}^{n} (p_k \cdot T_k) \right] - E \cdot p_n \cdot T$$

$$\Leftrightarrow C_{ij\ldots n} - C_n = E \cdot \left[\sum_{k=i}^{n} (p_k \cdot T_k) - \sum_{k=i}^{n} (p_n \cdot T_k) \right]$$

$$\Leftrightarrow C_{ij\ldots n} - C_n = E \cdot \left[\sum_{k=i}^{n} (p_k - p_n) \cdot T_k \right] > 0$$

because $\forall k < n$, $p_n < p_k$

Hence $C_{ij\ldots n} > C_n \quad \forall i < j < \cdots < n$

As in the "Preemptive" case, it is proven from these equations that with wake-up "On Demand", it is always more costly to transition through a sequence of power states then to switch to the lowest power state in the sequence.

3.3. Cost of Activating Early or Late

To this point, it was assumed that decisions were made with accurate knowledge of the idle period length. The following analysis is concerned with the cost endured in the event that the predicted idle period is incorrect. Two cases rise from this study: the system fails to be in the active state at the time of the request: "Activate Late", or the system becomes active before the next request arrives: "Activate Early".

Only the study of the "Preemptive" wake-up method is conducted in this section, because the "On Demand" wake-up method is not concerned with predicting the length of the idle period: it activates at the arrival of the request.

3.3.1. Cost of Activating Late

The cost of activating late has already been discussed, since it corresponds to the "On Demand" wake-up method. This situation occurs when the predicted length of the idle period is longer than the actual idle period length. In this case, the system therefore remains in state S_i for the entire idle period T, and activates at the arrival of the request. The total cost includes latency in addition to energy expenditure:

$$C_{iL} = E \cdot \left(p_i \cdot T + e_i \right) + L \cdot t_i$$

3.3.2. Cost of Activating Early

This represents the case where the actual idle period is T, but the predicted idle length is shorter. Assuming the idle period to be of length $T_i + t_i < T$, the cost of activating early corresponds to the cost of remaining in state S_i for T_i time, plus to the cost of activating,

and the cost of staying in S_0 for the remainder of the idle period: $T - T_i - t_i$ time. The cost expenditure in this case is:

$$C_{iE} = E \cdot \left[p_i \cdot T_i + e_i + p_0 \cdot \left(T - T_i - t_i \right) \right]$$

Figure 3.6 illustrates the timeline in this case.

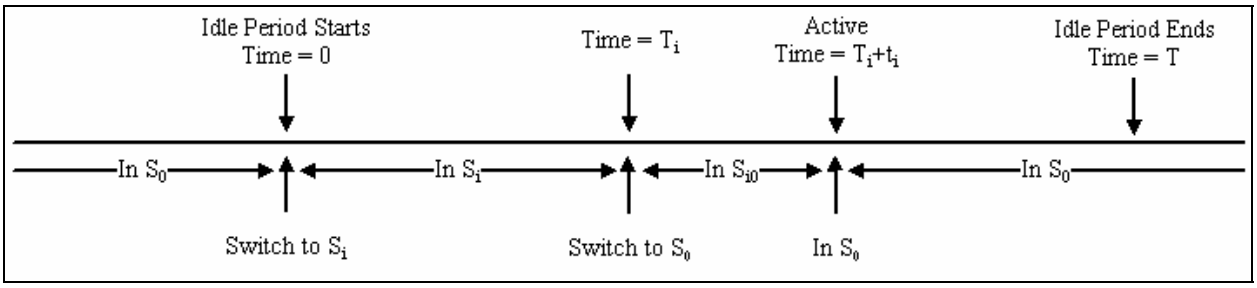


Figure 3.6: Time Diagram when "Activating Early"

3.3.3. Cost of Transitioning back to a Lower Power Mode if Activated Early

This situation considers is an extension of the "Activate Early" case. The system assumes an idle period of length $T_i + t_i < T$, and activates from state S_i at time T_i to be in S_0 at the arrival of the next request. However, once in the active state, the system learns the actual length of the idle period: $T = T_i + t_i + t_L$. Subsequently, decision needs to be made to stay in the active state until the end of the idle period (case 3.3.2.), or transition to a lower power mode S_j for $t_l - t_j$ time. This decision is equivalent to determining the optimal for an idle period of length t_l (section 3.1.2).

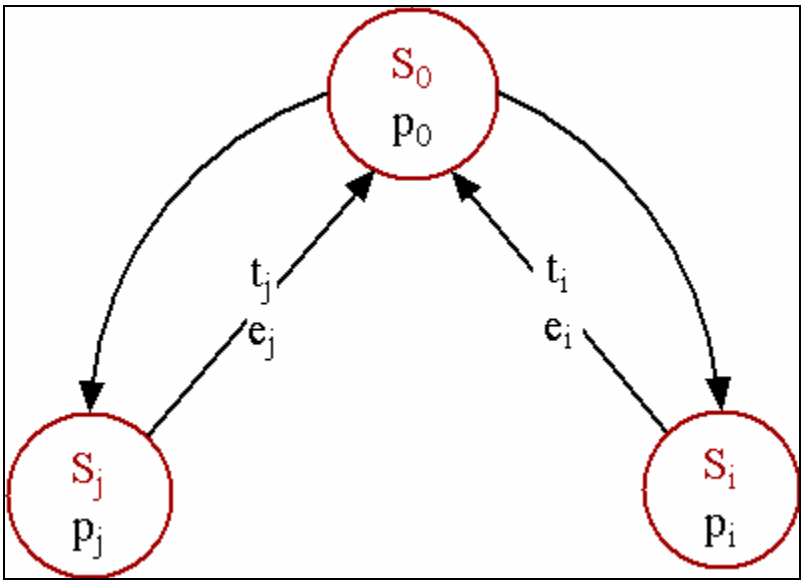


Figure 3.7: 3-State Automaton

Figure 3.7 illustrates the automaton in this case, and Figure 3.8 shows the corresponding time diagram.

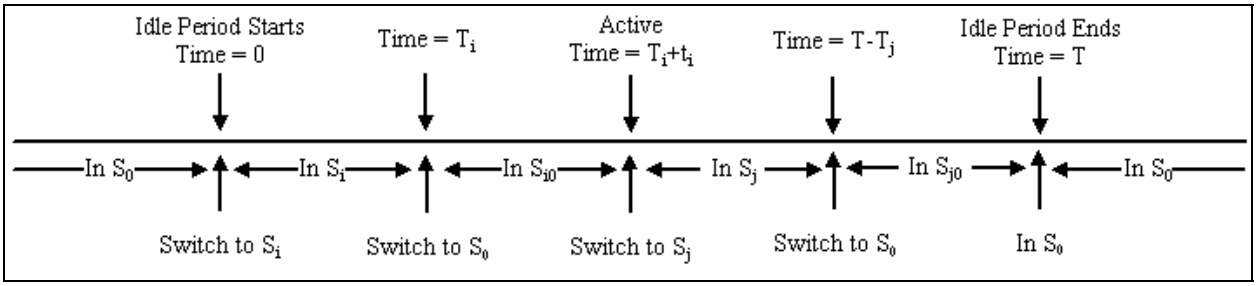


Figure 3.8: Time Diagram for Activating and Powering back Down

In the second case, where the systems transitions back to a lower power mode after activation, the total cost for the total idle period is equal to the cost of staying in state S_i, added the cost of activating from S_i, added the cost of staying in S_j, added the cost of activating from S_j.

$$C_{i0j} = E \cdot \left[p_i \cdot T_i + e_i + p_j \cdot \left(t_l - t_j \right) + e_j \right]$$

3.3.4. Switching to Another State if the Idle Period is Different than Predicted

This analysis is a generalization of the previous situations. It is assumed that the predicted idle period T_i is different than the actual idle period $T = T_i + t_d$, where $t_d \in \mathbb{R}^*$. Evidently, if $t_d > 0$ the actual idle period is longer than predicted, otherwise, it is shorter. It is also assumed that the actual idle period length is learned at time t, $0 < t < T$.

The case where $t = T_i + t_i$ has been discussed in section 3.3.3. The system learns in this case the actual length of the idle period as soon as it becomes active. As expressed earlier, the cost in this situation is:

$$C_{i0j} = E \cdot \left[p_i \cdot T_i + e_i + p_j \cdot \left(t_l - t_j \right) + e_j \right]$$

Similarly, if $t > T_i + t_i$, determination of the optimal state can be made using the results of section 3.1.2. for an idle period of length $T - t$. This situation represents the case when the system becomes active before learning the actual length of the idle period. It therefore endures power consumption in the active state. When it learns the actual idle period length, it may choose to transition to state S_j, optimal for the remainder of the idle period $T - t$.

The time diagram for this case is given in Figure 3.9.

Correspondingly, the cost endured is:

$$C_{i0j} = E \cdot \left[p_i \cdot T_i + e_i + p_0 \cdot \left(t - T_i - t_i \right) + p_j \cdot \left(T - t - t_j \right) + e_j \right]$$

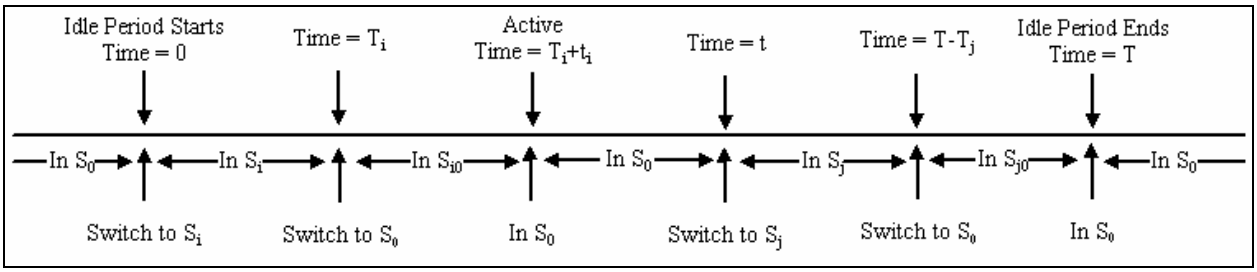


Figure 3.9: Time Diagram for Activating, Waiting, and Powering back Down

However, if $t < T_i$, the system learns the actual period length before activating. The automaton for this case is given in Figure 3.4. Decision can therefore be made to stay in the current state, or to transition to S_j where $j > i$ for the remainder of the idle period. Again, this is equivalent to determining the optimal state for an idle period of length $T - t$.

Figure 3.10 illustrates the timeline for the situation when the system decides to transition to a lower power mode.

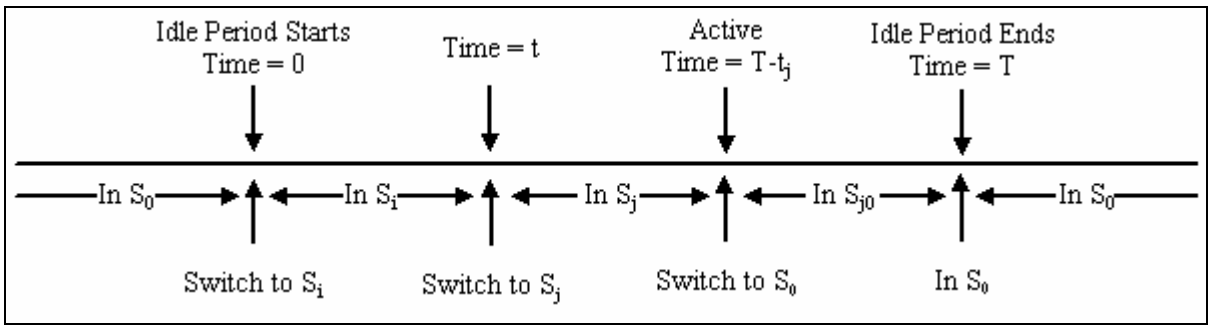


Figure 3.10: Time Diagram for Staying in Low Power Modes

The cost in this situation amounts to:

$$C_{ij} = E \cdot \left[p_i \cdot t + p_j \cdot \left(T - t - t_j \right) + e_j \right] \text{ if } T - t - t_j \geq 0,$$

$$C_{ij} = E \cdot \left[p_i \cdot t + e_j \right] - L \cdot \left(T - t - t_j \right) \text{ if } T - t - t_j < 0$$

3.4. Conclusion

From the studies of the Timed Hybrid Automata model, several observations can be concluded. If the length of the future idle period is known, it is more cost efficient to lead the system to be active at the arrival of the request: "Preemptive" wake-up strategy. Moreover, if the length of the future idle period is known, the optimal state can be determined, given the parameters of the states of the system. In particular, if the idle period is shorter than the start-up times of all the power modes, it is more advantageous to remain active during the idle period. Furthermore, it is always less advantageous to transition through a sequence of power modes during idle time, than to switch directly to the last state in the sequence.

Finally, in the last section of this chapter, costs are given in the cases when the predicted idle period is incorrect. In these situations, the system incurs more cost than it would have if it had chosen the idle state for the correct idle period length, as observed in the prior analyses.

It is therefore important to devise a strategy for correctly predicting the lengths of the future idle periods, for the minimization of cost, as for the DPM problem. In the next chapter of this thesis, a solution is proposed for the management of the Hybrid Automata model, which probabilistically decides on the future idle-period lengths and hence guides the system to the correct optimal state given its predictions.

Chapter 4

Stochastic Learning Feedback Hybrid Automata for DPM

As it has been observed in the earlier chapters of this thesis, control needs to be incorporated in the hybrid automata model in order to facilitate the power conservation as for the DPM problem. In chapter 2, when presenting the hybrid automata mathematical model proposed for DPM, the control variable was assumed to be deterministic, managed by an exterior system.

In this chapter, the control theory is made probabilistic, by formulating probabilities of switching states. Consequently, stochastic control is incorporated to the hybrid automaton, and learning feedback is added to the mathematical model. Such a system attempts to learn the length of the future idle period probabilistically and, accordingly, decides on its behavior during idle time.

This work was partly inspired by the study of learning automata [3].

4.1. Stochastic Automaton: Definition

4.1.1. The Environment

From a learning automaton perspective, the concept of environment designates the collection of all the external conditions that have an effect on the system.

The environment is described by a triplet $\{\alpha, c, \beta\}$ as follows:

- $\alpha = \{\alpha_i\}_{i=1}^{r}$ is a finite set of inputs,
- $\beta = \{\beta_1, \beta_2\}$ is a binary output set, usually $\{0,1\}$,
- $c = \{c_i\}_{i=1}^{r}$ is a set of penalty probabilities.

Each element c_i is associated with the corresponding input action α_i. Usually, the output set β is chosen such that $\beta_1 = 0$ and $\beta_2 = 1$.

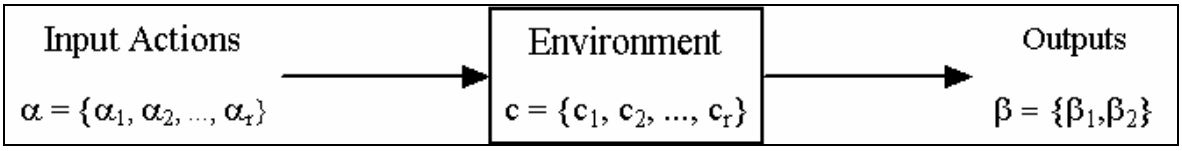


Figure 4.1: The Environment

As shown in Figure 4.1, the input to the environment at time $t = n \quad \forall n \in [\![0, \infty)$ is an action $\alpha(n) = \alpha_i$ where $i \in [\![1, r]\!]$. According to the input action, the environment generates an output $\beta(n)$ belonging to the set of permitted outputs. The inputs and the outputs of the environment are probabilistically associated by the penalty probabilities c_i.

If $\beta(n)=1$ is set to correspond to a failure and $\beta(n)=0$ to represent a successful response from the environment, then c_i characterizes the probability that action α_i will generate an unfavorable output from the environment.

$$c_i = P\left(\beta(n)=1 \middle| \alpha(n)=\alpha_i\right) \quad \forall i \in [\![1,r]\!]$$

4.1.2. The Automaton

Similarly to the definition given chapter 2, the automaton can be defined by a quintuple $\{\Phi, \alpha, \beta, F(.,.), H(.,.)\}$ where

- Φ is the set of internal states,
- α is the set of outputs,
- β is the set of input actions,
- $F : \Phi \times \beta \to \Phi$ is the transition function,
- $H : \Phi \times \beta \to \alpha$ is the output function.

If Φ, α, β are finite sets, the automaton is categorized as a finite automaton. Figure 4.2 illustrates a finite automaton.

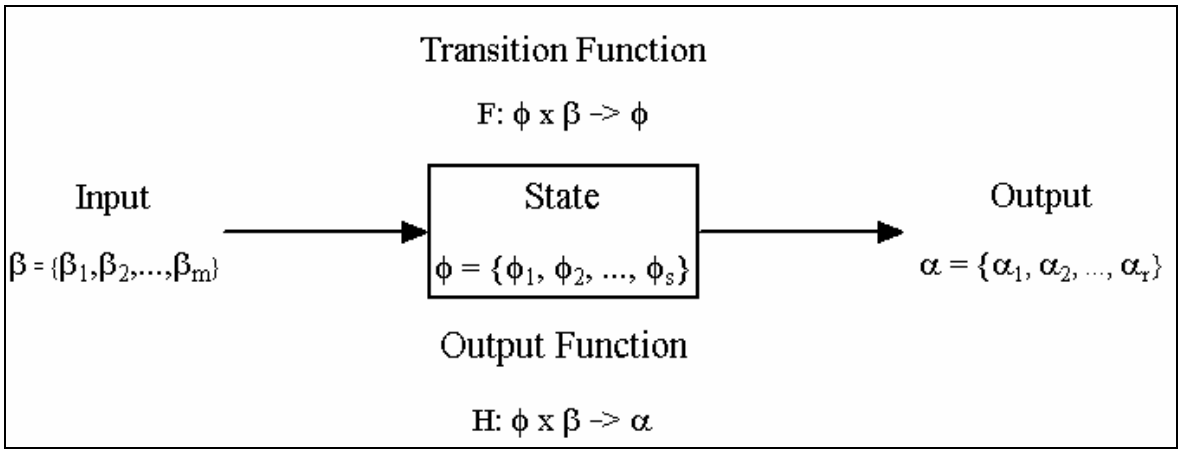


Figure 4.2: The Finite Automaton

At time $t = n \quad \forall n \in [\![0, \infty)$, a finite automaton is described by the following:

- the current state $\Phi(n) \in \{\Phi_i\}_{i=1}^{s}$,
- the current input $\beta(n) \in \{\beta_i\}_{i=1}^{m}$,
- the next state $\Phi(n+1) = F(\Phi(n), \beta(n))$,
- and the current output $\alpha(n) = H(\Phi(n), \beta(n))$, $\alpha(n) \in \{\alpha_i\}_{i=1}^{r}$.

State-output automata are systems where the current output only responds to the current state, and does not depend on the current input. The output function is then defined by $G : \Phi \to \alpha$. For a state-output automaton, the current output is hence governed by $\alpha(n) = G(\Phi(n))$, $\alpha(n) \in \{\alpha_i\}_{i=1}^{r}$.

a) Deterministic Automaton

For an automaton to be deterministic, mappings F and H both need to be deterministic. Graphs or matrices are used to represent the latter for every input.

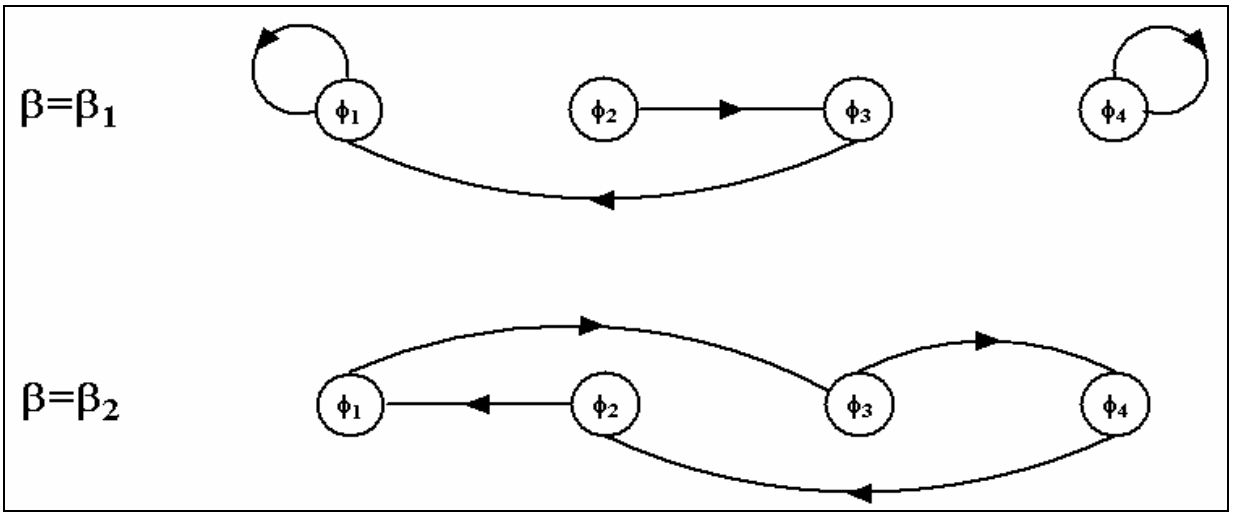


Figure 4.3: Deterministic Transition Graph

As an example, Figures 4.3 and 4.4 respectively illustrate the transition graphs and output graphs of a system with input symbols $\{\beta_1, \beta_2\}$, output symbols $\{\alpha_1, \alpha_2, \alpha_3\}$, and states $\{\Phi_1, \Phi_2, \Phi_3, \Phi_4\}$.

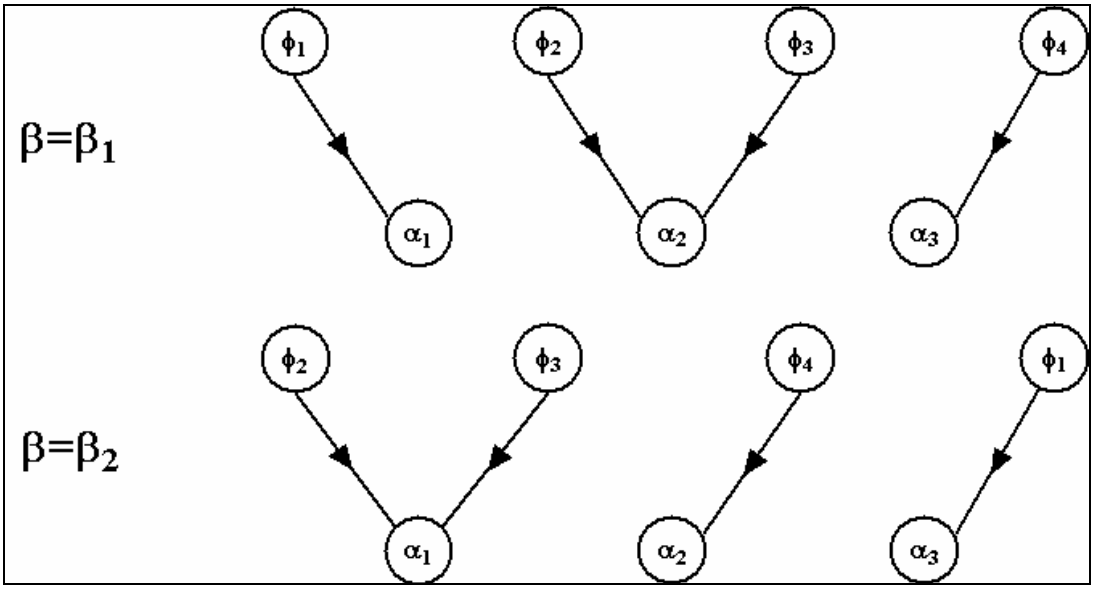


Figure 4.4: Deterministic Output Graph

Similarly, the following matrices can be used to describe the operation of the previous system:

- Transition function

$$F(\beta_1) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{cccc} \Phi_1 & \Phi_2 & \Phi_3 & \Phi_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad F(\beta_2) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{cccc} \Phi_1 & \Phi_2 & \Phi_3 & \Phi_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

such that $f_{ij}^{\beta_k} = \begin{cases} 1 & \text{if} \quad \Phi_i \to \Phi_j \quad \text{for} \quad \beta = \beta_k \\ 0 & \text{otherwise} \end{cases}$, $\forall i, j \in [1, s]$, $\forall k \in [1, m]$,

- Output function

$$H(\beta_1) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{ccc} \alpha_1 & \alpha_2 & \alpha_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \qquad H(\beta_2) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{ccc} \alpha_1 & \alpha_2 & \alpha_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$

such that $h_{ij}^{\beta_k} = \begin{cases} 1 & if \quad H(\Phi_i, \beta_k) = \alpha_j \\ 0 & otherwise \end{cases}$, $\forall i \in [1, s]$, $\forall j \in [1, r]$, $\forall k \in [1, m]$.

b) Stochastic Automaton

A stochastic automaton has at least one stochastic mapping. Given current state and input combinations, the next state and/or current output of the system are random. Hence, the stochastic transition function F, and the stochastic output function H respectively indicate the probabilities of switching states and the probabilities of output actions. Like for deterministic automata, stochastic automata can be represented by graphs or matrices for every possible input.

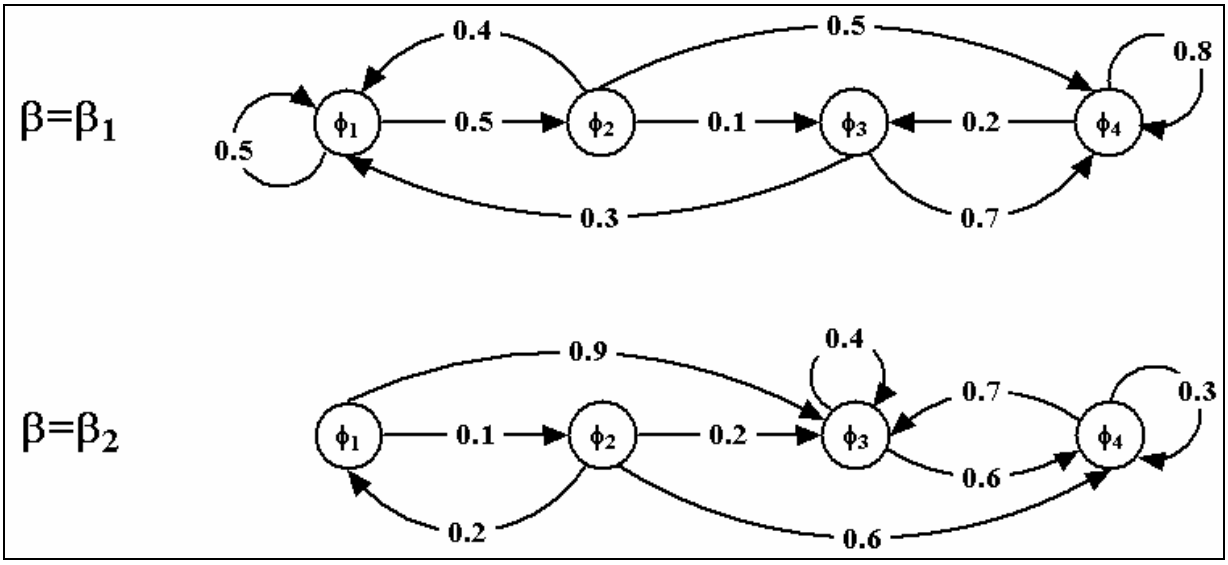


Figure 4.5: Stochastic Transition Graph

As an example, Figures 4.5 and 4.6 respectively illustrate the transition graphs and output graphs of a system with input symbols $\{\beta_1, \beta_2\}$, output symbols $\{\alpha_1, \alpha_2, \alpha_3\}$, and states $\{\Phi_1, \Phi_2, \Phi_3, \Phi_4\}$.

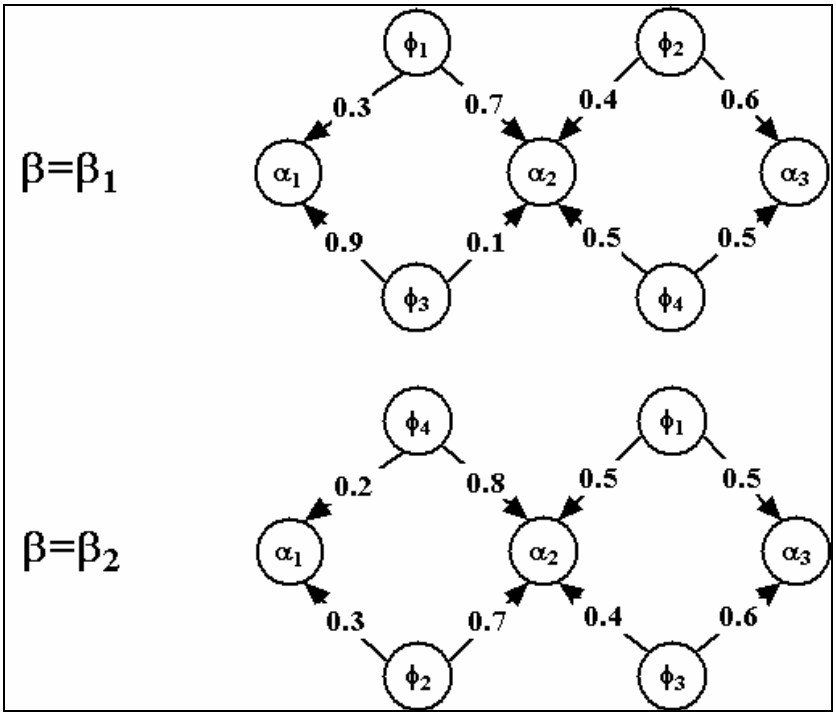


Figure 4.6: Stochastic Output Graph

Similarly, the following matrices can be used to describe the operation of a stochastic system:

- Transition function

$$F(\beta_1) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{cccc} \Phi_1 & \Phi_2 & \Phi_3 & \Phi_4 \end{array} \\ \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.4 & 0 & 0.1 & 0.5 \\ 0.3 & 0 & 0 & 0.7 \\ 0 & 0 & 0.2 & 0.8 \end{bmatrix} \end{array} \qquad F(\beta_2) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{cccc} \Phi_1 & \Phi_2 & \Phi_3 & \Phi_4 \end{array} \\ \begin{bmatrix} 0 & 0.1 & 0.9 & 0 \\ 0.2 & 0 & 0.2 & 0.6 \\ 0 & 0 & 0.4 & 0.6 \\ 0 & 0 & 0.7 & 0.3 \end{bmatrix} \end{array}$$

such that $f_{ij}^{\beta_k} = P\left(\Phi(n+1) = \Phi_j \,|\, \Phi(n) = \Phi_i, \beta(n) = \beta_k\right)$, $\forall i, j \in [1, s]$, $\forall k \in [1, m]$,

- Output function

$$H(\beta_1) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{ccc} \alpha_1 & \alpha_2 & \alpha_3 \end{array} \\ \begin{bmatrix} 0.3 & 0.7 & 0 \\ 0 & 0.4 & 0.6 \\ 0.9 & 0.1 & 0 \\ 0 & 0.5 & 0.5 \end{bmatrix} \end{array} \qquad H(\beta_2) = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{array} \begin{array}{c} \begin{array}{ccc} \alpha_1 & \alpha_2 & \alpha_3 \end{array} \\ \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.3 & 0.7 & 0 \\ 0 & 0.4 & 0.6 \\ 0.2 & 0.8 & 0 \end{bmatrix} \end{array}$$

such that $h_{ij}^{\beta_k} = P\left(\alpha(n) = \beta_j \middle| \Phi(n) = \Phi_i, \beta(n) = \beta_k\right),\ \forall i \in [1,s],\ \forall j \in [1,r],\ \forall k \in [1,m]$.

The following conditions must hold:

- $\sum_{j=1}^{s} f_{ij}^{\beta_k} = 1,\ \forall i \in [1,s],\ \forall k \in [1,m]$,
- $\sum_{j=1}^{r} h_{ij}^{\beta_k} = 1,\ \forall i \in [1,s],\ \forall k \in [1,m]$.

c) Stochastic Automata with Deterministic Output Mapping

A stochastic automaton with stochastic transition and stochastic output mappings can easily be converted into an automaton with deterministic output mapping by transforming the states of the system. New states $\hat{\Phi}_{ij}$ can be set as combinations of the old states of the automaton Φ_i and possible inputs in each state α_j, such that $\hat{\Phi}_{ij} = (\Phi_i, \alpha_j)$. The resulting automaton will contain sr states and its output function will be deterministic.

The transition and output probabilities are then set as follows:

- $\hat{f}_{(ij)(kl)}^{\beta_u} = P\left(\Phi(n+1) = \Phi_k, \alpha(n+1) = \alpha_l \middle| \Phi(n) = \Phi_i, \alpha(n) = \alpha_j, \beta(n) = \beta_u\right)$

  $\Leftrightarrow\ \hat{f}_{(ij)(kl)}^{\beta_u} = g_{kl} f_{ik}^{\beta_u}$

- $g_{(ij)(l)} = P\left(\alpha(n+1) = \alpha_l \left| \hat{\Phi}(n+1) = \hat{\Phi}_{ij} \right.\right)$

  $\Leftrightarrow g_{(ij)(l)} = \begin{cases} 1 & if\ j = l \\ 0 & if\ j \neq l \end{cases}$

4.1.3. Complete Stochastic Learning System

A complete system, as illustrated in Figure 4.7, is composed of an automaton and a random environment.

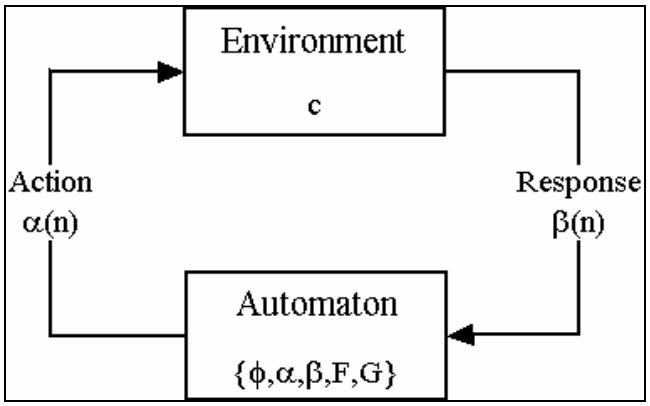


Figure 4.7: Complete Automaton/Environment System

4.1.4. State Probabilities and Action Probabilities

a) State Probabilities

At every time instant $n \geq 0$, the total state probability vector $\pi(n)$ presents the probability of the automaton being in a certain state. The state probability vector can as well be used to convey the functioning of the system.

$\pi(n) = \left[\pi_i(n)\right]_{i=1}^{s}$

Such that

$$\pi_i(0) = P\left(\Phi(0) = \Phi_i\right),$$

$$\pi_i(n) = P\left(\Phi(n) = \Phi_i \,\middle|\, \beta(0), ..., \beta(n-1)\right).$$

Hence,

$$\pi(n) = \left[\prod_{i=1}^{n} F^{\mathrm{T}}\left(\beta(n-i)\right)\right]\pi(0)$$

b) Action Probabilities

Similarly, the total action probability vector $p(n)$ can be used to describe the operation of the system.

$$p(n) = \left[p_i(n)\right]_{i=1}^{r},$$

such that

$$p_i(n) = P\left(\alpha(n) = \alpha_i \,\middle|\, \beta(0), ..., \beta(n-1)\right).$$

It can be shown that

$$p(n) = G^{\mathrm{T}}\pi(n).$$

4.2. Variable Structure Automaton

Variable structure automata are learning automata whose state and action probabilities are frequently recomputed using reinforcement techniques. These probabilities are usually updated at every new input arrival using linear or non-linear reward-penalty or reward-inaction methods.

Such automata can be expressed by a quintuple $\{\Phi, \alpha, \beta, A, G\}$ where $\{\Phi, \alpha, \beta, G\}$ correspond to the definition stated in the previous section and A represents the reinforcement method.

Algorithms for updating action probabilities are predominant. It is consequently considered that only one action is associated to every state, leading to $G = I$ and $r = s$. Automata can hence be represented by a triple $\{\alpha, \beta, A\}$.

Given a set of permitted actions in every state of a system, automaton learning is the concept of educating the system to choose the optimal action to execute at every stage.

4.2.1. Reinforcement Schemes

An updating algorithm is typically expressed by

$$p(n+1) = T\left[p(n), \alpha(n), \beta(n) \right],$$

suggesting that at every time instant $n \geq 0$, the action probability at the next time instant is mapped by T to the current action probability, output action, and input, which is the output of the stationary random environment.

Reward-penalty is the most popular updating scheme. If the application of an action α_i results in a success, the algorithm amplifies the action probability p_i and reduces the others. On the contrary, if action α_i results in a failure, the algorithm will decrease the action probability p_i and increase all the others.

As a result, in a system with r possible actions, and a binary response from the environment $\beta = \{0,1\}$, the updating scheme at time $t = n \ \ \forall n \geq 0$ after the application of action $\alpha(n) = \alpha_i$ is as follows:

If $\beta(n) = 0$ (Success),

- $p_i(n+1) = p_i(n) + \sum_{\substack{j=1 \\ j \neq i}}^{r} g_j(p(n))$
- $p_j(n+1) = p_j(n) - g_j(p(n)) \ \ \forall j \neq i$

If $\beta(n) = 1$ (Failure),

- $p_i(n+1) = p_i(n) - \sum_{\substack{j=1 \\ j \neq i}}^{r} h_j(p(n))$
- $p_j(n+1) = p_j(n) + h_j(p(n)) \ \ \forall j \neq i$

with the conditions that

- g_j and h_j are continuous non-negative functions
- $0 < g_j(p) < p_j$
- $0 < \sum_{\substack{j=1 \\ j \neq i}}^{r} \left[p_j + h_j(p) \right] < 1 \ \ \forall i \in [1, r]$,

Consequently, $p(n)$ conserves its probability characteristics since

$$\sum_{j=1}^{r} p_j(n) = 1 \text{ and } \forall j, \ p_j(n+1) \in (0,1) \text{ if } p_j(n) \in (0,1).$$

In addition, the strict inequalities in the previous equations guarantee that $p(n+1) \neq p(n)$.

4.3. Stochastic Learning Hybrid Automata Model for DPM

4.3.1. The Stochastic Learning Hybrid Automata Model

In this thesis, a stochastic learning hybrid automaton (SLHA) was researched to model the power management system. The hybrid model described in chapter 2 was adapted such that the control variable u was customized to be represented by switching probabilities in the SLHA model. Figure 4.8 illustrates the SLHA model as developed in UPPAAL model checker.

As described in chapter 2, states $S_i \ \forall i \in [\![0,n]\!]$ represent the main internal states of the system, states $S_{i0} \ \forall i \in [\![1,n]\!]$ illustrate transient states for powering up to the active state, and states $S_{ir} \ \forall i \in [\![1,n]\!]$ symbolize temporary states for powering up to the active state and running received requests. Internal variables e and t_L account for the energy expenditure and latency of the system at every instant. Variables t_R and t respectively hold temporary values of the remaining request length to process and time to spend in the transient states before powering up. The input constant r designates the processing length of the incoming request. Finally, p_{ij} represent the action probabilities, and label the transitions between two states.

4.3.2. Switching Probabilities as Control Variables

As mentioned earlier, the control variable u briefly discussed in chapter 2 is represented by switching probabilities in the SLHA model.

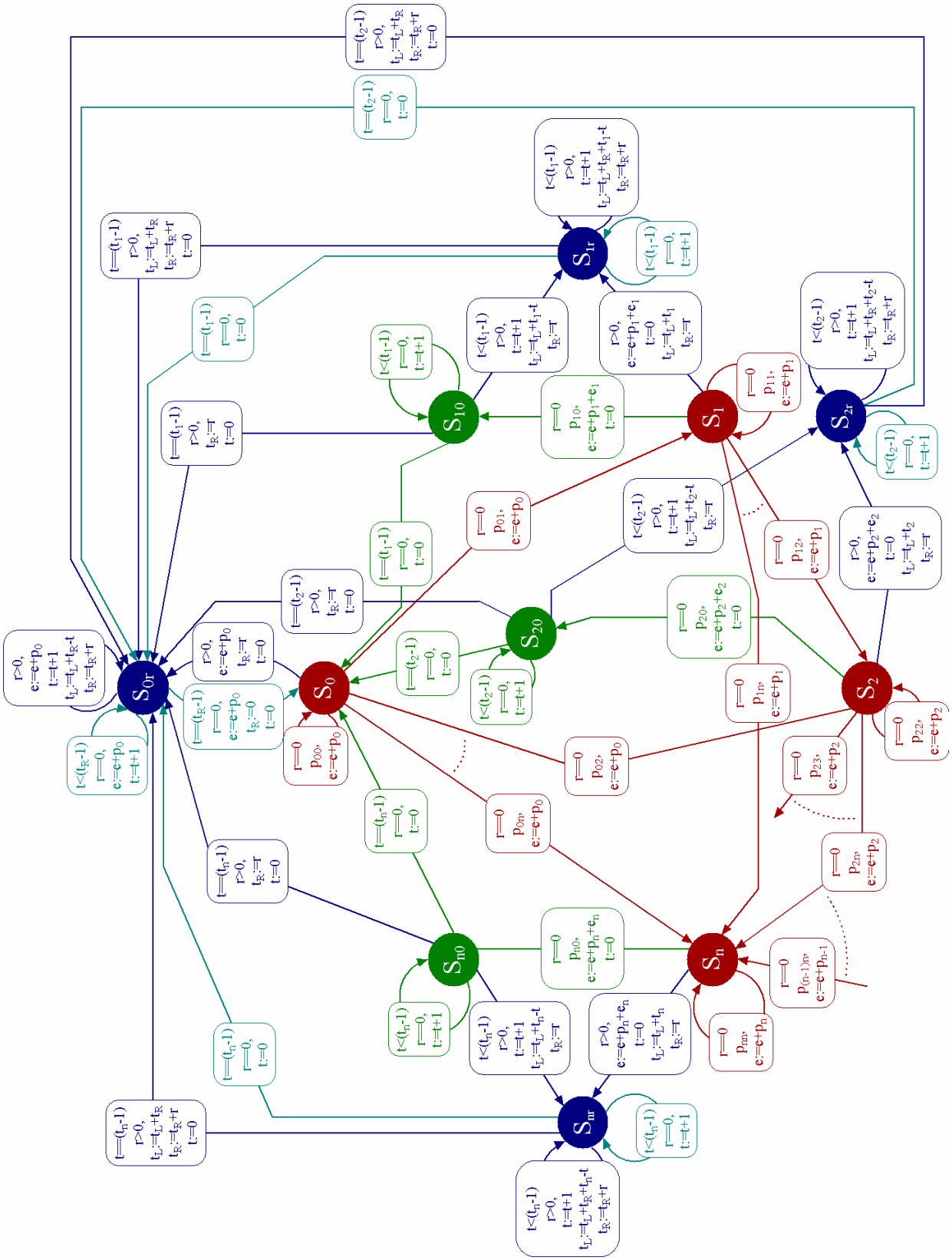


Figure 4.8: Stochastic Learning Hybrid Automata Model

Every allowed main-state transition $S_i - S_j \quad \forall i, j \in [\![0, n]\!], j \neq i$ is labeled by a probability p_{ij} that represents the probability of switching from state S_i to state S_j. These probabilities hold the following property:

$\forall i \in [\![1, n]\!], \sum_{j=1}^{n} p_{ij} = 1$ given that transition $S_i - S_j$ is allowed.

The values of the probabilities are regularly updated using the feedback algorithm selected for the system.

4.4. SLHA Simulation Software

Until now, theoretical analysis was made on the behavior of the Hybrid Automata model for DPM. In order to examine the suitability of the mathematical model for Dynamic Power Management, the system must be tested in real-time. The first step, before building the system in hardware and testing it in real-life situations, is to develop a simulator to test, in software, the behavior of the model presented with real-time data.

Software [6] was consequently developed in C++ to simulate the SLHA model for Power Management. The simulator follows the guidelines for DPM, as detailed earlier in this work. Its behavior is controlled stochastically by switching probabilities, whose learning is performed by selected reinforcement schemes.

The input to the program is a sequence of requests for service, and the purpose of the simulator is to stochastically manage the system, while attempting to reduce the energy consumption, and latency incurred by the processes.

On a command console, the executable file is run with six command line arguments: input filename, configuration filename, E, L, T and M. The software then outputs to the screen the total time and consumption details of the specified system for the tested input file. It also saves an output file with details of its operation.

4.4.1. Input File Specifications

a) Command Line Parameters

Parameters E and L represent the weights of energy and latency respectively in the calculation of consumption. It is assumed that $consumption = E \cdot energy + L \cdot latency$. E and L may hold any positive values.

Parameter T corresponds to the value of the time increment steps, in milliseconds, at which to determine the next state of the automaton. It may hold any positive value.

Finally, M determines the activation strategy of the automata: using "preemptive" wake-up ($M = 0$) or power-up "on demand" ($M = 1$).

b) Configuration File

The first file, the configuration file, is used to specify the different parameters of the reinforcement schemes. It is formatted similarly to the following example:

```
//Config
A       0.5
B       0.5
D       1
O       1
Q
```

Constant $O \in [1,5]$ characterizes the learning method to simulate. Five different schemes have been implemented in the software, as listed below:

- $O = 1$: Linear Reward-Penalty,
- $O = 2$: Linear Reward-Inaction,
- $O = 3$: Non-Linear 1,
- $O = 4$: Non-Linear 2,
- $O = 5$: Hybrid H.

The mathematical procedures for the above algorithms are detailed in chapter 5.

Constants A, B, and D are factors in the learning algorithms implemented. The first two represent reward and penalty parameters, and the latter determines the degree of non-linearity of the second non-linear updating scheme.

As it can be implied from the example, lines beginning with comment characters "//" are ignored by the software. In addition, there is no specific order to present the parameters in the file. Finally, character Q expresses the end of the configuration file. The program will not read any following statements in the file.

c) Input File

The second command line argument, the input file, contains two different sections. First, the parameters of the states of the system are expressed in the following format.

```
S    0    1.9    0      0
S    1    0.9    0.56   40
S    2    0.2    1.575  1500
S    3    0      4.75   5000
```

The first character *S* specifies to the program that the parameters are defining states of the system. The second number indicates the power mode: 0 represents the active state, and the following states have state-ids that increase according to their level of inactiveness. Here, state 3 corresponds to the deepest sleep state. The following values correspond to "power consumption", "start-up energy" and "start-up time" respectively. These figures have been explained in chapter 2.

Next, the requests are listed specified by a character *R*, a request time in *ms*, and the time to process the request also in *ms* as in the following example:

```
R       9828        0
R       748809829   0.4
R       835         3.25
```

Finally, the end of the input file is specified by character *Q*. Similarly to the configuration file, the program will not consider any statement expressed after this ending symbol and will ignore the lines that begin with "//". Furthermore, the declaration of the states, as well as the inventory of the requests, does not need to be listed in any particular order. The software will sort them in ascending order when importing the values.

4.4.2. Operation of the simulator

After establishing the user-specified configurations, the software starts simulating the particular system, starting the clock at the time of the first request. It then executes according to the power-management specifications detailed in chapter 1.

During idle periods, the program stochastically determines in what state to remain at every "tick" of the clock, interval specified by *T*. This is performed by randomly selecting a number between 0 and 1, which determines the action to execute according

to the probability ranges of the possible transitions. Different simulations were run with $T = \{1,10,100\}\, ms$ clock increments.

When a request arrives, it immediately transitions to the active state to process the input. At that time, it also records the length of the idle period that it has just experienced and establishes the optimal state for that idle time. This is done by finding for every state the total consumption (energy and latency) of the system if it had remained in that state for the entire idle period, and then determining the state with the lowest expenditure. The chosen learning algorithm is then selected to update the switching probabilities to reflect the environment's response to the action performed. In other words, the probabilities of switching to the optimal state are rewarded and, according to the reinforcement scheme, the other probabilities may be penalized.

Finally, the simulator quits after processing the last request. It then creates an output file with the details of the simulation, and outputs the statistics to the command output.

Chapter 5

Feedback Algorithms

Several feedback stochastic learning algorithms are described in this section, and analysis is made on the theoretical convergence of the hybrid automaton given a stationary environment. This study will also be examined experimentally in the subsequent chapters of this thesis.

The following analyses were inspired by previous work on learning automata [3].

5.1. Introduction

The general reinforcement scheme on action probabilities was presented in chapter 4. In this chapter, specific linear and non-linear schemes are described, and analysis is made on the stability and convergence of the systems in stationary environments.

As described in chapter 4, for a system with r possible actions, and a binary response from the environment $\beta = \{0,1\}$, the updating scheme at time $t = n \quad \forall n \geq 0$ after the application of action $\alpha(n) = \alpha_i$ is as follows:

If $\beta(n) = 0$ (Success),

- $p_i(n+1) = p_i(n) + \sum_{\substack{j=1 \\ j \neq i}}^{r} g_j(p(n))$
- $p_j(n+1) = p_j(n) - g_j(p(n)) \quad \forall j \neq i$

If $\beta(n) = 1$ (Failure),

- $p_i(n+1) = p_i(n) - \sum_{\substack{j=1 \\ j \neq i}}^{r} h_j(p(n))$
- $p_j(n+1) = p_j(n) + h_j(p(n)) \quad \forall j \neq i$

5.2. Linear Learning Scheme

5.2.1. General Linear Reward-Penalty Scheme

For linear learning schemes, the following reinforcement functions are chosen:

$$g_j[p(n)] = a \cdot p_j(n),$$

$$h_j[p(n)] = \frac{b}{r-1} - b \cdot p_j(n)$$

where $0 < a < 1,\ 0 \leq b < 1$

Hence, the general linear reward-penalty scheme

If $\beta(n) = 0$ (Success),

- $p_i(n+1) = a + (1-a) \cdot p_i(n)$
- $p_j(n+1) = (1-a) \cdot p_j(n) \quad \forall j \neq i$

If $\beta(n) = 1$ (Failure),

- $p_i(n+1) = (1-b) \cdot p_i(n)$
- $p_j(n+1) = \frac{b}{r-1} + (1-b) \cdot p_j(n) \quad \forall j \neq i$

5.2.2. Symmetric Linear Reward-Penalty Scheme

The symmetric linear reward-penalty scheme is equivalent to the general linear reward-penalty scheme, with the particular condition that the reward and penalty parameters are equal: $a = b$. This clause engenders symmetric reward and penalty updates such that the learning for the probability p_i of action α_i in the case when the application of action α_i results in a success is identical to the learning engendered when the application of action α_j results in a failure.

The linear reward-penalty scheme is defined as follows:

If $\beta(n) = 0$ (Success),

- $p_i(n+1) = a + (1-a) \cdot p_i(n)$
- $p_j(n+1) = (1-a) \cdot p_j(n) \quad \forall j \neq i$

If $\beta(n) = 1$ (Failure),

- $p_i(n+1) = (1-a) \cdot p_i(n)$
- $p_j(n+1) = \frac{a}{r-1} + (1-a) \cdot p_j(n) \quad \forall j \neq i$

5.2.3. Linear Reward-Inaction Scheme

The linear reward-inaction scheme is a special case of the general linear reward-penalty scheme, with the stipulation that there is no learning penalty in the case of failure: $b = 0$.

The linear reward-inaction scheme is defined as follows:

If $\beta(n) = 0$ (Success),

- $p_i(n+1) = a + (1-a) \cdot p_i(n)$
- $p_j(n+1) = (1-a) \cdot p_j(n) \quad \forall j \neq i$

If $\beta(n) = 1$ (Failure),

- $p_i(n+1) = p_i(n)$
- $p_j(n+1) = p_j(n) \quad \forall j \neq i$

5.3. Non-linear Learning Schemes

5.3.1. Nonlinear Scheme 1

For this learning scheme, the following reinforcement functions are chosen:

$$g_j\left[p(n)\right] = \frac{a}{r-1}\cdot p_i(n)\cdot\left(1-p_i(n)\right),$$

$$h_j\left[p(n)\right] = \frac{b}{r-1}\cdot p_i(n)\cdot\left(1-p_i(n)\right)$$

where $0 < a \le 1,\ 0 < b \le 1$.

Hence, the nonlinear scheme 1:

If $\beta(n) = 0$ (Success),

- $p_i(n+1) = \left(1 + a - a\cdot p_i(n)\right)\cdot p_i(n)$
- $p_j(n+1) = p_j(n) - \frac{a}{r-1}\cdot p_i(n)\cdot\left(1-p_i(n)\right)\ \ \forall j \ne i$

If $\beta(n) = 1$ (Failure),

- $p_i(n+1) = \left(1 - b + b\cdot p_i(n)\right)\cdot p_i(n)$
- $p_j(n+1) = p_j(n) + \frac{b}{r-1}\cdot p_i(n)\cdot\left(1-p_i(n)\right)\ \ \forall j \ne i$

5.3.2. Nonlinear Scheme 2

For the nonlinear learning scheme 2, the following reinforcement functions are chosen:

$$g_j\left[p(n)\right] = p_j(n) - \phi\left[p_j(n)\right],$$

$$h_j\left[p(n)\right] = \frac{p_i(n) - \phi\left[p_i(n)\right]}{r-1},$$

where $0 \le \phi\left[p_j(n)\right] \le p_j(n)$.

ϕ is usually chosen as:

$$\phi(x) = ax^m \text{ where } 0 < a \le 1,\ m \in [\![2, \infty]\!].$$

Hence, the nonlinear scheme 2:

If $\beta(n) = 0$ (Success),

- $p_i(n+1) = 1 - \sum_{\substack{j=1 \\ j \ne i}}^{r} \phi\left[p_j(n)\right]$
- $p_j(n+1) = \phi\left[p_j(n)\right] \quad \forall j \ne i$

If $\beta(n) = 1$ (Failure),

- $p_i(n+1) = \phi\left[p_i(n)\right]$
- $p_j(n+1) = p_j(n) + \frac{p_i(n) - \phi\left[p_i(n)\right]}{r-1} \quad \forall j \ne i$

Or, equivalently:

If $\beta(n)=0$ (Success),

- $p_i(n+1)=1-\sum_{\substack{j=1 \\ j\neq i}}^{r} a\cdot p_j^m(n)$
- $p_j(n+1)=a\cdot p_j^m(n) \quad \forall j\neq i$

If $\beta(n)=1$ (Failure),

- $p_i(n+1)=a\cdot p_i^m(n)$
- $p_j(n+1)=p_j(n)+\frac{p_i(n)-a\cdot p_i^m(n)}{r-1} \quad \forall j\neq i$

5.3.3. Hybrid Scheme H

For the Hybrid H learning scheme, the following reinforcement functions are chosen:

$$g_j\left[p(n)\right]=a\cdot p_j(n),$$

$$h_j\left[p(n)\right]=\begin{cases} a\cdot p_j(n) & \text{if} \quad p_i(n)\in\left[\frac{a}{1+a},\frac{1}{1+a}\right] \\ 0 & \text{otherwise} \end{cases}$$

where $0<a<1$.

Thus, the Hybrid H scheme:

If $\beta(n)=0$ (Success),

- $p_i(n+1)=(1-a)\cdot p_i(n)+a$
- $p_j(n+1)=(1-a)\cdot p_j(n) \quad \forall j \neq i$

If $\beta(n)=1$ (Failure)

If $p_i(n) \in \left[\frac{a}{1+a}, \frac{1}{1+a}\right]$

- $p_i(n+1)=(1+a)\cdot p_i(n)-a$
- $p_j(n+1)=(1+a)\cdot p_j(n) \quad \forall j \neq i$

If $p_i(n) \notin \left[\frac{a}{1+a}, \frac{1}{1+a}\right]$

- $p_i(n+1)=p_i(n)$
- $p_j(n+1)=p_j(n) \quad \forall j \neq i$

5.4. Convergence in Stationary Environments

Analysis of expediency in stationary environments can be done by examining the conditional expectation of the update of the probability of action α_i given its current value. Assuming that the automaton runs in a stationary random environment, the conditional expected value of $p_i(n+1)$ given $p_i(n)$ for the general learning scheme can be calculated:

$$\begin{aligned} E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[p_i(n+1)\middle|\alpha(n)=\alpha_i,\beta(n)=0\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] \\ &+\left[p_i(n+1)\middle|\alpha(n)=\alpha_j,\beta(n)=1\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot c_j\right] \\ &+\left[p_i(n+1)\middle|\alpha(n)=\alpha_i,\beta(n)=1\right]\cdot\left[p_i(n)\cdot c_i\right] \\ &+\left[p_i(n+1)\middle|\alpha(n)=\alpha_j,\beta(n)=0\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot(1-c_j)\right] \end{aligned}$$

$$\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right] = \left[p_i(n)+\sum_{\substack{j=1\\ j\neq i}}^{r} g_j\left(p(n)\right)\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] + \left[p_i(n)+h_i\left(p(n)\right)\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot c_j\right] + \left[p_i(n)-\sum_{\substack{j=1\\ j\neq i}}^{r} h_j\left(p(n)\right)\right]\cdot\left[p_i(n)\cdot c_i\right] + \left[p_i(n)-g_i\left(p(n)\right)\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot(1-c_j)\right]$$

5.4.1. Study of Two-Action Automata

The conditional expectation of $p_i(n+1)$ given $p_i(n)$ for a two-action automaton is formulated as follows:

$$E\left[p_i(n+1)\middle|p_i(n)\right] = \left[p_i(n+1)\middle|\alpha(n)=\alpha_i,\beta(n)=0\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] + \left[p_i(n+1)\middle|\alpha(n)=\alpha_j,\beta(n)=1\right]\cdot\left[p_j(n)\cdot c_j\right] + \left[p_i(n+1)\middle|\alpha(n)=\alpha_i,\beta(n)=1\right]\cdot\left[p_i(n)\cdot c_i\right] + \left[p_i(n+1)\middle|\alpha(n)=\alpha_j,\beta(n)=0\right]\cdot\left[p_j(n)\cdot(1-c_j)\right]$$

$$\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right] = \left[p_i(n)+g_j\left(p(n)\right)\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] + \left[p_i(n)+h_i\left(p(n)\right)\right]\cdot\left[\left(1-p_i(n)\right)\cdot c_j\right] + \left[p_i(n)-h_j\left(p(n)\right)\right]\cdot\left[p_i(n)\cdot c_i\right] + \left[p_i(n)-g_i\left(p(n)\right)\right]\cdot\left[\left(1-p_i(n)\right)\cdot(1-c_j)\right]$$

a. General Linear Reward-Penalty Scheme

$$\begin{aligned} E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[a+(1-a)\,p_i(n)\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] \\ &+\left[b+(1-b)\,p_i(n)\right]\cdot\left[\left(1-p_i(n)\right)\cdot c_j\right] \\ &+\left[(1-b)\,p_i(n)\right]\cdot\left[p_i(n)\cdot c_i\right] \\ &+\left[(1-a)\,p_i(n)\right]\cdot\left[\left(1-p_i(n)\right)\cdot(1-c_j)\right] \end{aligned}$$

$$\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right] = b\cdot c_j+(a-b)\cdot\left(c_i-c_j\right)\cdot p_i^2(n)+\left[1-2b\cdot c_j+a\cdot\left(c_j-c_i\right)\right]\cdot p_i(n)$$

Let

$$\Delta p_i(n) \triangleq E\left[p_i(n+1)\middle|p_i(n)\right]-p_i(n)$$

$$\Leftrightarrow \Delta p_i(n) = b\cdot c_j+p_i^2(n)\cdot(a-b)\cdot\left(c_i-c_j\right)+p_i(n)\cdot\left[a\cdot\left(c_j-c_i\right)-2b\cdot c_j\right]$$

❖ **Symmetric Linear Reward-Penalty Scheme**

$$E\left[p_i(n+1)\middle|p_i(n)\right] = a\cdot c_j+\left[1-a\cdot\left(c_i+c_j\right)\right]\cdot p_i(n)$$

$$\Rightarrow E\left[E\left[p_i(n+1)\middle|p_i(n)\right]\right] = E\left[p_i(n+1)\right] = a\cdot c_j+\left[1-a\cdot\left(c_i+c_j\right)\right]\cdot E\left[p_i(n)\right]$$

This linear difference equation in $E\left[p_i(n)\right]$ yields the following solution:

$$E\left[p_i(n)\right] = \left[1-a\cdot\left(c_i+c_j\right)\right]^n\cdot p_i(0)+a\cdot c_j\cdot\frac{1-\left[1-a\cdot\left(c_i+c_j\right)\right]^n}{a\cdot\left(c_i+c_j\right)}$$

Hence,

$$\text{if } c_i\neq c_j,\ \lim_{n\to\infty}E\left[p_i(n)\right]=\frac{c_j}{c_i+c_j}$$

Consequently, the average penalty as the number of iterations reaches infinity is:

$$\begin{aligned}\lim_{n\to\infty} E\left[M(n)\right] &= c_i \cdot \lim_{n\to\infty} E\left[p_i(n)\right] + c_j \cdot \lim_{n\to\infty} E\left[p_j(n)\right] \\ &= 2\frac{c_i \cdot c_j}{c_i + c_j} < \frac{c_i + c_j}{2}\end{aligned}$$

$$\Leftrightarrow \lim_{n\to\infty} E\left[M(n)\right] < M_0$$

This inequality holds for any value of the penalty probabilities c_i, c_j and initial transition probabilities $p_i(0), p_j(0)$. The linear reward-penalty scheme is therefore expedient for all initial conditions and stationary environments where $c_i \neq c_j$.

Moreover,

$$\text{if } c_i < c_j,\ \lim_{n\to\infty} E\left[p_i(n)\right] > \lim_{n\to\infty} E\left[p_j(n)\right]$$

The latter indicates that if the application of action α_i is less likely to induce an unfavorable response than the application of action α_j, then, with iterations, action α_i will tend to be chosen more frequently than action α_j.

❖ **Linear Reward-Inaction Scheme**

$$E\left[p_i(n+1)\middle| p_i(n)\right] = a \cdot \left(c_i - c_j\right) \cdot p_i^2(n) + \left[1 + a \cdot \left(c_j - c_i\right)\right] \cdot p_i(n)$$

$$\Leftrightarrow \Delta p_i(n) = a \cdot \left(c_j - c_i\right) \cdot \left(1 - p_i(n)\right) \cdot p_i(n)$$

Hence,

$$\Delta p_i(n)\begin{cases}\geq 0 & if \quad c_j > c_i \\ \leq 0 & if \quad c_j < c_i\end{cases}$$

and

$$\Delta p_i(n) = 0 \text{ if } p_i(n) = \{0,1\}$$

$$\Rightarrow E\left[p_i(n+1)\right]\begin{cases}> E\left[p_i(n)\right] & if \quad c_j > c_i \\ < E\left[p_i(n)\right] & if \quad c_j < c_i\end{cases} \text{ when } p_i(n) \in (0,1)$$

This implies that if the application of action α_i is less likely to induce an unfavorable response than the application of action α_j, then, with iterations, the probability of action α_i will increase monotonically. Action α_i will therefore tend to be chosen more frequently with time. This property holds for any value of the penalty probabilities c_i, c_j and initial transition probabilities $p_i(0), p_j(0) \in (0,1)$. The linear reward-inaction scheme is therefore absolutely expedient.

b. Nonlinear Scheme 1

$$\begin{aligned}E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[\left(1+a-a\cdot p_i(n)\right)\cdot p_i(n)\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] \\ &+\left[\left(1+b-b\cdot p_i(n)\right)\cdot p_i(n)\right]\cdot\left[\left(1-p_i(n)\right)\cdot c_j\right] \\ &+\left[\left(1-b+b\cdot p_i(n)\right)\cdot p_i(n)\right]\cdot\left[p_i(n)\cdot c_i\right] \\ &+\left[\left(1-a+a\cdot p_i(n)\right)\cdot p_i(n)\right]\cdot\left[\left(1-p_i(n)\right)\cdot\left(1-c_j\right)\right]\end{aligned}$$

$$\begin{aligned}\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[1-a+c_j\cdot(a+b)\right]\cdot p_i(n) \\ &+\left[3\cdot a-\left(c_i+2\cdot c_j\right)\cdot(a+b)\right]\cdot p_i^2(n) \\ &+\left[-2\cdot a+\left(c_i+c_j\right)\cdot(a+b)\right]\cdot p_i^3(n)\end{aligned}$$

$$\begin{aligned}\Rightarrow \Delta p_i(n) &= \left[-a + c_j \cdot (a+b)\right] \cdot p_i(n) \\ &+ \left[3 \cdot a - \left(c_i + 2 \cdot c_j\right) \cdot (a+b)\right] \cdot p_i^2(n) \\ &+ \left[-2 \cdot a + \left(c_i + c_j\right) \cdot (a+b)\right] \cdot p_i^3(n)\end{aligned}$$

Given that

$$\Delta p_i(n) + \Delta p_j(n) = 0 \quad \forall n,$$

If $c_i < \frac{a}{a+b} < c_j$ and $p_i(n) \in (0,1)$,

$$\Delta p_i(n) > 0 \text{ and } \Delta p_j(n) < 0.$$

Conversely, if $c_i > \frac{a}{a+b} > c_j$ and $p_i(n) \in (0,1)$,

$$\Delta p_i(n) < 0 \text{ and } \Delta p_j(n) > 0.$$

Moreover,

$$\Delta M(n) = \left(c_i - c_j\right) \cdot \Delta p_i(n) < 0.$$

Consequently, this scheme is absolutely expedient for the assumed restricted initial conditions and environments.

c. Nonlinear Scheme 2

$$\begin{aligned}E\left[p_i(n+1) \middle| p_i(n)\right] &= \left[1 - \phi\left[p_j(n)\right]\right] \cdot \left[p_i(n) \cdot (1 - c_i)\right] \\ &+ \left[1 - \phi\left[p_j(n)\right]\right] \cdot \left[\left(1 - p_i(n)\right) \cdot c_j\right] \\ &+ \phi\left[p_i(n)\right] \cdot \left[p_i(n) \cdot c_i\right] \\ &+ \phi\left[p_i(n)\right] \cdot \left[\left(1 - p_i(n)\right) \cdot \left(1 - c_j\right)\right]\end{aligned}$$

$$\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right]=\left[p_i(n)\cdot\left(1-c_i-c_j\right)+c_j\right]\cdot\left[1-\phi\left[p_i(n)\right]-\phi\left[p_j(n)\right]\right]+\phi\left[p_i(n)\right]$$

$$\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right]=\left[p_i(n)\cdot\left(1-c_i-c_j\right)+c_j\right]\cdot\left[1-a\cdot\left(p_i^m(n)+p_j^m(n)\right)\right]+a\cdot p_j^m(n)$$

Similarly to the previous scheme, the sign definiteness of $\Delta p_k(n)$ for the nonlinear scheme 2 is guaranteed provided the following rigorous conditions on the penalty probabilities:

$$c_k<\frac{1}{m} \text{ and } c_j>\frac{1}{m} \quad \forall j\neq k\,.$$

d. Hybrid Scheme H

$$\begin{aligned}E\left[p_i(n+1)\middle|p_i(n)\right] &=\left[(1-a)\,p_i(n)+a\right]\cdot\left[p_i(n)\cdot(1-c_i)\right]\\ &+\left[(1+a)\,p_i(n)\right]\cdot\left[(1-p_i(n))\cdot c_j\right]\\ &+\left[(1+a)\,p_i(n)-a\right]\cdot\left[p_i(n)\cdot c_i\right]\\ &+\left[(1-a)\,p_i(n)\right]\cdot\left[(1-p_i(n))\cdot(1-c_j)\right]\end{aligned}$$

$$\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right]=\left[1-2\cdot a\cdot\left(c_i-c_j\right)\right]\cdot p_i(n)-2\cdot a\cdot c_j\cdot p_i^2(n)$$

If $\frac{a}{1+a}\le p_i(n)\le\frac{1}{1+a}$, this scheme is equivalent to the linear reward-reward penalty algorithm. Otherwise, this scheme follows the principles of the linear reward-inaction updates. The updating scheme is therefore expedient or absolutely expedient, respectively to the reinforcement algorithm that it follows.

5.4.2. Multi-Action Automata

Equivalent analyses of convergence can be extended to the multi-action automata. The properties of the different feedback schemes persist in the multi-action case.

a. General Linear Reward-Penalty Scheme

$$\begin{aligned} E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[a+(1-a)\,p_i(n)\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] \\ &+\left[\frac{b}{r-1}+(1-b)\,p_i(n)\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot c_j\right] \\ &+\left[(1-b)\,p_i(n)\right]\cdot\left[p_i(n)\cdot c_i\right] \\ &+\left[(1-a)\,p_i(n)\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot(1-c_j)\right] \end{aligned}$$

$$\begin{aligned} \Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[(a-b)\cdot c_i\right]\cdot p_i^2 \\ &+\left[1-a\cdot c_i+(a-b)\cdot\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot c_j\right]\cdot p_i \\ &+\left[\frac{b}{r-1}\cdot\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot c_j\right] \end{aligned}$$

❖ **Symmetric Linear Reward-Penalty Scheme**

$$E\left[p_i(n+1)\middle|p_i(n)\right]=\frac{a}{r-1}\cdot\sum_{\substack{j=1\\ j\neq i}}^{r} p_j(n)\cdot c_j+\left[1-a\cdot c_i\right]\cdot p_i$$

❖ **Linear Reward-Inaction Scheme**

$$E\left[\, p_i(n+1) \middle| p_i(n) \right] = a \cdot c_i \cdot p_i^2 + \left[1 - a \cdot c_i + a \cdot \sum_{\substack{j=1 \\ j \neq i}}^{r} p_j(n) \cdot c_j \right] \cdot p_i$$

b. Nonlinear Scheme 1

$$\begin{aligned} E\left[\, p_i(n+1) \middle| p_i(n) \right] &= \left[\left(1 + a - a \cdot p_i(n)\right) \cdot p_i(n) \right] \cdot \left[p_i(n) \cdot (1 - c_i) \right] \\ &+ \left[\left(1 + b - b \cdot p_i(n)\right) \cdot p_i(n) \right] \cdot \left[\sum_{\substack{j=1 \\ j \neq i}}^{r} p_j(n) \cdot c_j \right] \\ &+ \left[\left(1 - b + b \cdot p_i(n)\right) \cdot p_i(n) \right] \cdot \left[p_i(n) \cdot c_i \right] \\ &+ \left[\left(1 - a + a \cdot p_i(n)\right) \cdot p_i(n) \right] \cdot \left[\sum_{\substack{j=1 \\ j \neq i}}^{r} p_j(n) \cdot (1 - c_j) \right] \end{aligned}$$

$$\begin{aligned} \Leftrightarrow E\left[\, p_i(n+1) \middle| p_i(n) \right] &= \left[1 - a + (a+b) \cdot \sum_{\substack{j=1 \\ j \neq i}}^{r} p_j(n) \cdot c_j \right] \cdot p_i(n) \\ &+ \left[3 \cdot a - (a+b) \cdot \left(c_i + \sum_{\substack{j=1 \\ j \neq i}}^{r} p_j(n) \cdot c_j \right) \right] \cdot p_i^2(n) \\ &- \left[2 \cdot a - c_i (a+b) \right] \cdot p_i^3(n) \end{aligned}$$

c. Nonlinear Scheme 2

$$
\begin{aligned}
E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[1-\sum_{\substack{j=1\\ j\neq i}}^{r}\phi\left[p_j(n)\right]\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] \\
&+\left[p_i(n)+\frac{p_j(n)-\phi\left[p_j(n)\right]}{r-1}\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r}p_j(n)\cdot c_j\right] \\
&+\phi\left[p_i(n)\right]\cdot\left[p_i(n)\cdot c_i+\sum_{\substack{j=1\\ j\neq i}}^{r}p_j(n)\cdot(1-c_j)\right]
\end{aligned}
$$

d. Hybrid Scheme H

$$
\begin{aligned}
E\left[p_i(n+1)\middle|p_i(n)\right] &= \left[(1-a)\,p_i(n)+a\right]\cdot\left[p_i(n)\cdot(1-c_i)\right] \\
&+\left[(1+a)\,p_i(n)\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r}p_j(n)\cdot c_j\right] \\
&+\left[(1+a)\,p_i(n)-a\right]\cdot\left[p_i(n)\cdot c_i\right] \\
&+\left[(1-a)\,p_i(n)\right]\cdot\left[\sum_{\substack{j=1\\ j\neq i}}^{r}p_j(n)\cdot(1-c_j)\right]
\end{aligned}
$$

$$
\Leftrightarrow E\left[p_i(n+1)\middle|p_i(n)\right]=2\cdot a\cdot c_i\cdot p_i^2(n)+\left[1-2\cdot a\cdot c_i\right]\cdot p_i(n)
$$

5.5. Expediency in Non-Stationary Environments

The penalty probabilities of a non-stationary environment vary with time. Assuming this variation to be deterministic, the definition of the penalty probability for action $\alpha(n)=\alpha_i$ is expressed as follows:

$$c_i(n) = P\left[\beta(n) = 1 \middle| \alpha(n) = \alpha_i\right]$$

$$\Leftrightarrow c_i(n) = E\left[\beta(n) \middle| \alpha(n) = \alpha_i\right].$$

The average penalty is:

$$M(n) = E\left[\beta(n) \middle| p(n)\right]$$

$$\Leftrightarrow M(n) = P\left[\beta(n) = 1 \middle| p(n)\right]$$

$$\Leftrightarrow M(n) = \sum_{i=1}^{r} P\left[\beta(n) = 1 \middle| p(n), \alpha(n) = \alpha_i\right] \cdot P\left[\alpha(n) = \alpha_i\right]$$

$$\Leftrightarrow M(n) = \sum_{i=1}^{r} c_i(n) \cdot p_i(n).$$

Hence, the average penalty of the pure chance automaton:

$$M_0(n) = \frac{1}{r} \cdot \sum_{i=1}^{r} c_i(n).$$

Consequently, a learning scheme is expedient in a non-stationary environment if

$\exists n_0$ such that $\forall n > n_0$

$$E\left[M(n)\right] - M_0(n) < 0.$$

In other words, the expected value of the average penalty has to become smaller than the average penalty of the pure chance automaton.

5.6. Conclusion

In stationary environments, all of the examined feedback schemes are expedient for given initial conditions on the action probabilities and conditions on the environments.

The examination of expediency in non-stationary environments will be performed in the subsequent chapter of this work, through simulations.

Chapter 6

Experimental Results with SLHA

As discussed in the earlier chapters, the objective of this project is to optimize the SLHA model in order to minimize the consumption (energy and latency) of the system for the DPM problem. This is achieved by teaching the automaton to always switch to the optimal state for the ongoing idle period. Reinforcement schemes are employed to dynamically teach the characteristics of the input distribution to the system and probabilistic control theory is hence chosen to guide the system through correct behavior.

To study the behavior of the SLHA system given various input distributions, simulations are performed with several combinations of the parameters of the mathematical model, and the results are analyzed based on the consumption of each system studied.

A SLHA model of a four-state mobile hard-drive from IBM [4] was employed for simulating a DPM. The time and energy specifications of the embedded system are shown in table 6.1, following the format detailed in Chapter 2.

Table 6.1. Power-Mode Characteristics for IBM HardDrive

State	Power Consumption (Watts)	Start-Up Energy (Joules)	Start-Up Time (ms)
Active S_0	1.9	0	0
Idle S_1	0.9	0.56	40
Stand-By S_2	0.2	1.575	1500
Sleep S_3	0	4.75	5000

6.1. Preliminary Analysis: Two-State Automata

Firstly, simulations were performed to determine the optimal parameters of the SLHA model to reach correct convergence in stationary environments. These were undertaken on simple two-state automata, solely containing an active state (S_0) and a sleep state (S_3), the latter corresponding to the lowest power mode of the modeled IBM system.

6.1.1. Test Input Files

Initially, two input files with constant idle period lengths were generated. The first file, containing a sequence of requests with permanent one millisecond inter-request time and zero request lengths, teaches the system to always stay in the active state during idle periods. Equivalently to the first, the second input file contains a sequence of requests with permanent two-thousand millisecond inter-request time and zero request lengths, consequently teaching the system to always switch to the sleep state during idle periods. The system was let to run for five-thousand requests, following these input

files that permanently encourage the application of one same action. The behavior of the automata in each case were analyzed and the optimal parameters A, B and *Degree* were determined for convergence with the different learning algorithms implemented.

Moreover, fixed bipolar distributions were employed for further analysis of the expediency of the reinforcement schemes. Following the same method, similar input files were generated, containing the idle period length for the first case in x% of the instances, and the length corresponding to the other case in the rest of the instances. Files were therefore generated holding {90%, 70%, 50%, 30%, 10%} one millisecond idle period lengths and {10%, 30%, 50%, 70%, 90%} two-thousand millisecond lengths respectively.

Appendix A illustrates the histograms of the input files tested.

6.1.2. Configuration Parameters

For each input file simulated, various configurations of the system were tested, corresponding to several possible parameter specifications for every learning algorithm. Following are the parameter specifications, as they have been discussed in Chapter 5.

- Linear Reward-Penalty: $0 < A < 1,\ 0 < B < 1,\ O = 1$
- Linear Reward-Inaction: $0 < A < 1,\ O = 2$
- Non-Linear 1: $0 < A \leq 1,\ 0 < B \leq 1,\ O = 3$
- Non-Linear 2: $0 < A \leq 1,\ D > 1,\ O = 4$
- Hybrid: $0 < A < 1,\ O = 5$

The configuration files used for the initial simulations are provided in Appendix B.

6.1.3. Preliminary Results

Each input pattern was simulated for every configuration file, and with $T = 1ms$ as the time increment. Two separate simulations were run for the "Preemptive" method and the "On Demand" method: $M = 0$ and $M = 1$ respectively. The described experiments were realized in three different categories: optimize energy and latency ($E = 1$, $L = 0.001$), optimize only energy ($E = 1$, $L = 0$), optimize only latency ($E = 0$, $L = 0.001$). A value of $L = 0.001$ was used in order to assimilate the cost of energy expenditure, in Joules, to the cost of latency, in seconds.

The energy, latency and consumption results for each configuration were summed over all the input files, and the outcomes were divided by the corresponding results of the optimal algorithm.

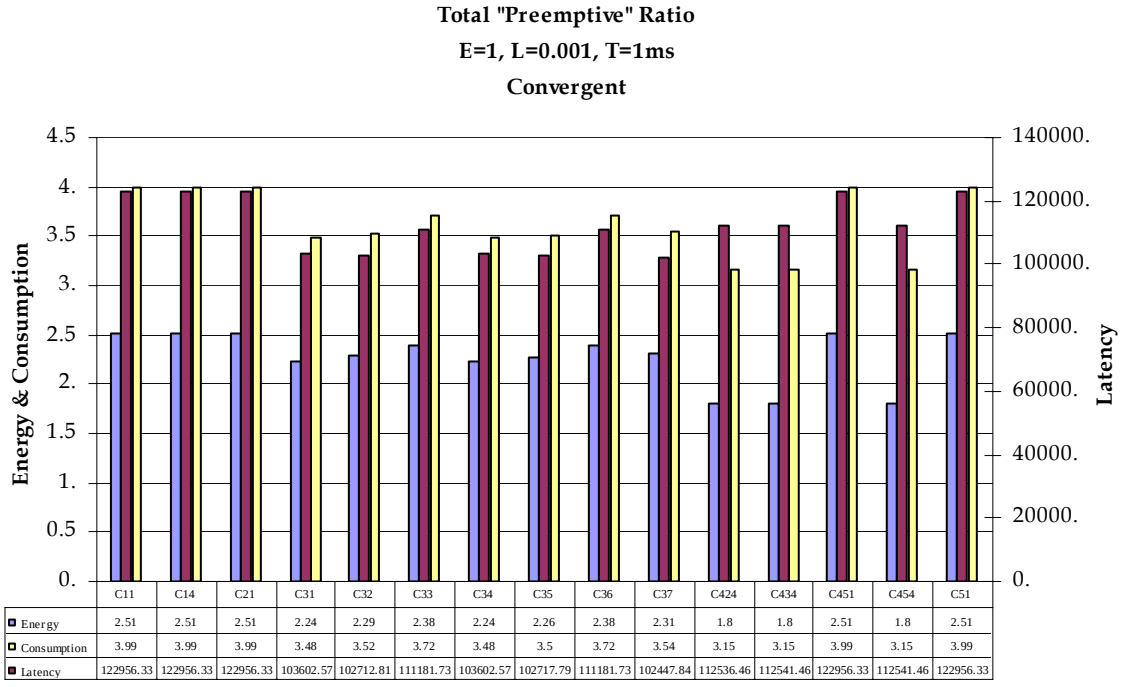
a) Optimization of Energy and Latency

Figures 6.1 through 6.4 hold the histograms of the total output ratios of these simulations.

There are two categories of output graphs presented for this simulation set. By analyzing the behavior of the different systems studied, it can be observed that only configurations $\{C_{11}, C_{14}, C_{21}, C_{31} - C_{37}, C_{424}, C_{434}, C_{451}, C_{454}, C_{51}\}$ bring the automata to correct convergence according to the tested input distributions (Figure 6.1 and Figure 6.3). Conversely, the other configurations either cause the systems to oscillate between states as the idle-period lengths vary, or converge to the incorrect optimal state (Figure 6.2 and Figure 6.4).

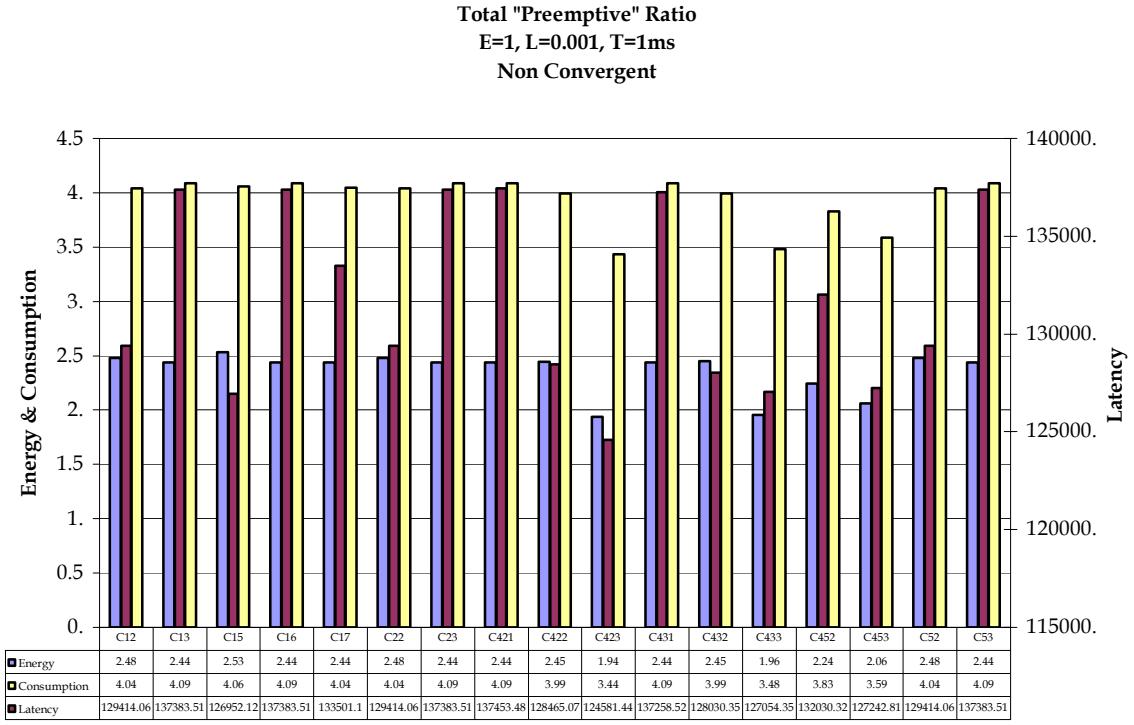
The observation of the output histograms exemplifies that the outputs of the converging systems yield lower latency and consumption costs than the non-converging systems, though the non-converging systems yield similar energy expenditures. This phenomenon may be explained by the short idle-time periods, which yield very close results in energy consumption between the two power modes.

Only the configurations which result in the convergence of the systems are further analyzed. The first category of converging systems corresponds to the configurations with linear or hybrid updating schemes, with a low reward parameter ($A = 0.1$). This is understandable because, by learning slower, such systems tend to converge slower but more accurately when they are presented with inputs distributions that support a globally optimal state. In contrast, when higher parameters are chosen, the systems are more likely to oscillate between states when they are taught with difficult input distributions. This can result in the non-convergence of the automata.



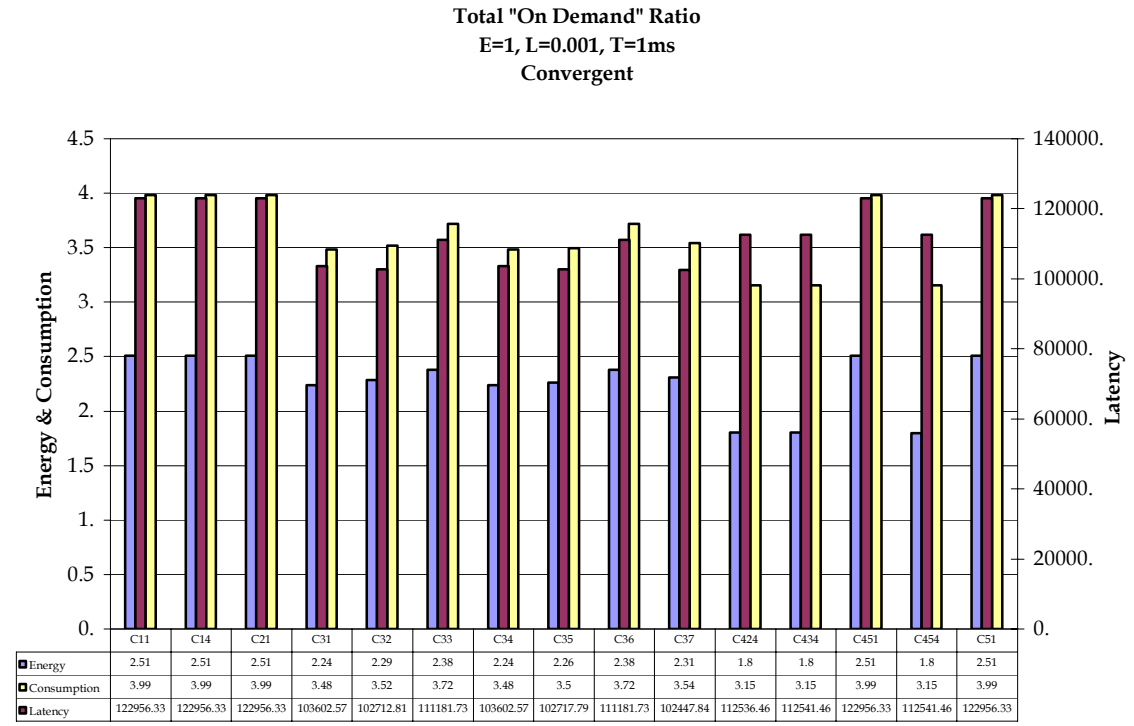
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	2.51	2.51	2.51	2.24	2.29	2.38	2.24	2.26	2.38	2.31	1.8	1.8	2.51	1.8	2.51
Consumption	3.99	3.99	3.99	3.48	3.52	3.72	3.48	3.5	3.72	3.54	3.15	3.15	3.99	3.15	3.99
Latency	122956.33	122956.33	122956.33	103602.57	102712.81	111181.73	103602.57	102717.79	111181.73	102447.84	112536.46	112541.46	122956.33	112541.46	122956.33

Figure 6.1: Total Output Ratio Histogram of Convergent Systems for "Preemptive" Method



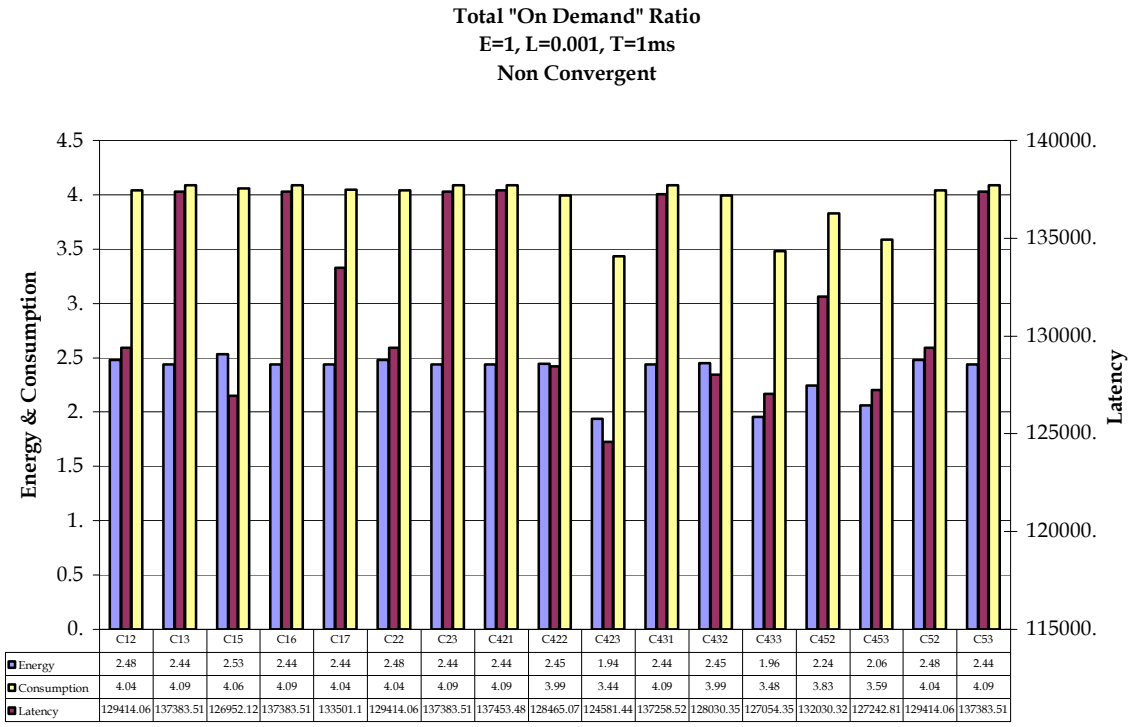
	C12	C13	C15	C16	C17	C22	C23	C421	C422	C423	C431	C432	C433	C452	C453	C52	C53
Energy	2.48	2.44	2.53	2.44	2.44	2.48	2.44	2.44	2.45	1.94	2.44	2.45	1.96	2.24	2.06	2.48	2.44
Consumption	4.04	4.09	4.06	4.09	4.04	4.04	4.09	4.09	3.99	3.44	4.09	3.99	3.48	3.83	3.59	4.04	4.09
Latency	129414.06	137383.51	126952.12	137383.51	133501.1	129414.06	137383.51	137453.48	128465.07	124581.44	137258.52	128030.35	127054.35	132030.32	127242.81	129414.06	137383.51

Figure 6.2: Total Output Ratio Histogram of Non-Convergent Systems for "Preemptive" Method



	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	2.51	2.51	2.51	2.24	2.29	2.38	2.24	2.26	2.38	2.31	1.8	1.8	2.51	1.8	2.51
Consumption	3.99	3.99	3.99	3.48	3.52	3.72	3.48	3.5	3.72	3.54	3.15	3.15	3.99	3.15	3.99
Latency	122956.33	122956.33	122956.33	103602.57	102712.81	111181.73	103602.57	102717.79	111181.73	102447.84	112536.46	112541.46	122956.33	112541.46	122956.33

Figure 6.3: Total Output Ratio Histogram of Convergent Systems for "On Demand" Method



	C12	C13	C15	C16	C17	C22	C23	C421	C422	C423	C431	C432	C433	C452	C453	C52	C53
Energy	2.48	2.44	2.53	2.44	2.44	2.48	2.44	2.44	2.45	1.94	2.44	2.45	1.96	2.24	2.06	2.48	2.44
Consumption	4.04	4.09	4.06	4.09	4.04	4.04	4.09	4.09	3.99	3.44	4.09	3.99	3.48	3.83	3.59	4.04	4.09
Latency	129414.06	137383.51	126952.12	137383.51	133501.1	129414.06	137383.51	137453.48	128465.07	124581.44	137258.52	128030.35	127054.35	132030.32	127242.81	129414.06	137383.51

Figure 6.4: Total Output Ratio Histogram of Non-Convergent Systems for "On Demand" Method

Moreover, the first non-linear learning scheme yields converging systems for all the tested configurations. This scheme hence demonstrates to be the most robust for the input distributions examined.

It may furthermore be observed that the systems with the second non-linear reinforcement scheme converge when using a high reward parameter $A=1$ for all degrees of non-linearity, and with a low reward parameter $A=0.1$ for high degrees of non-linearity, in this case $D=5$.

By further examining Figures 6.1 and 6.3, it can be observed that configurations $\{C_{31}, C_{32}, C_{34}, C_{35}, C_{37}\}$ yield much lower latency costs than the other configurations for both "Preemptive" and "On Demand" methods. Additionally, configurations $\{C_{424}, C_{434}, C_{454}\}$ present much lower energy and consumption expenditures for both

wake-up methods. The observed contrast between energy and latency is explained by the definition of consumption, which is a weighted sum of the two factors. Hence, a low consumption can equally be reached with high energy and low latency expenditures, or conversely with low energy and high latency costs.

Finally, when comparing the results between the two wake-up methods, it can be observed that "Preemptive" wake-up and wake-up "On Demand" yield equivalent results in all three cost measures.

b) Optimization of Energy

Figures 6.5 through 6.8 hold the histograms of the total output ratios of these simulations.

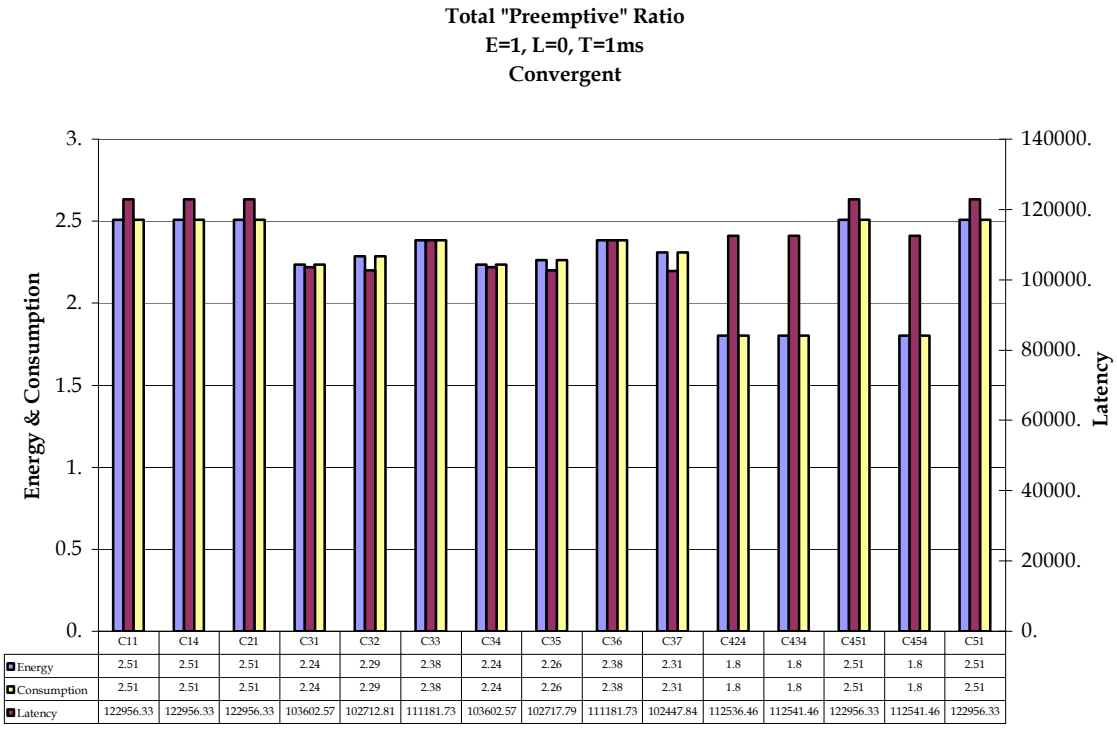
Like for the optimization of both energy and latency, there are two categories of output graphs presented for this simulation set. A similar behavior of the system can be observed in these tests: the same $\{C_{11}, C_{14}, C_{21}, C_{31} - C_{37}, C_{424}, C_{434}, C_{451}, C_{454}, C_{51}\}$ configurations bring the automata to correct convergence (Figure 6.5 and Figure 6.7), and the other configurations also either cause the systems to oscillate or to converge to the incorrect optimal states (Figure 6.6 and Figure 6.8).

Again, the observation of the output histograms exemplifies that the outputs of the converging systems yield lower latency costs than the non-converging systems. Correspondingly, the non-converging systems yield equivalent energy and consumption expenditures.

Like in the previous experiments, the configurations which result in convergence of the systems are further analyzed. From Figures 6.5 and 6.7, it can once more be observed

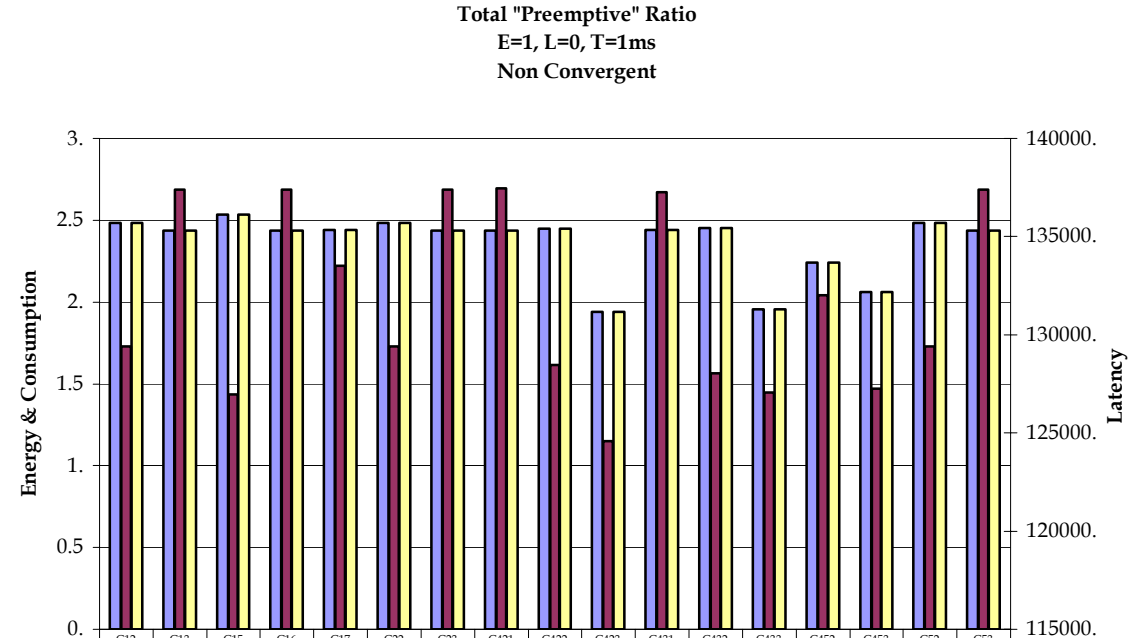
that configurations $\{C_{31}, C_{32}, C_{34}, C_{35}, C_{37}\}$ yield much lower latency costs than the other configurations for both "Preemptive" and "On Demand" methods. Similarly, configurations $\{C_{424}, C_{434}, C_{454}\}$ present much lower energy and consumption expenditures for both wake-up methods.

When comparing the results between the two wake-up methods, it can once more be observed that "Preemptive" wake-up and wake-up "On Demand" yield equivalent results in all three cost measures.



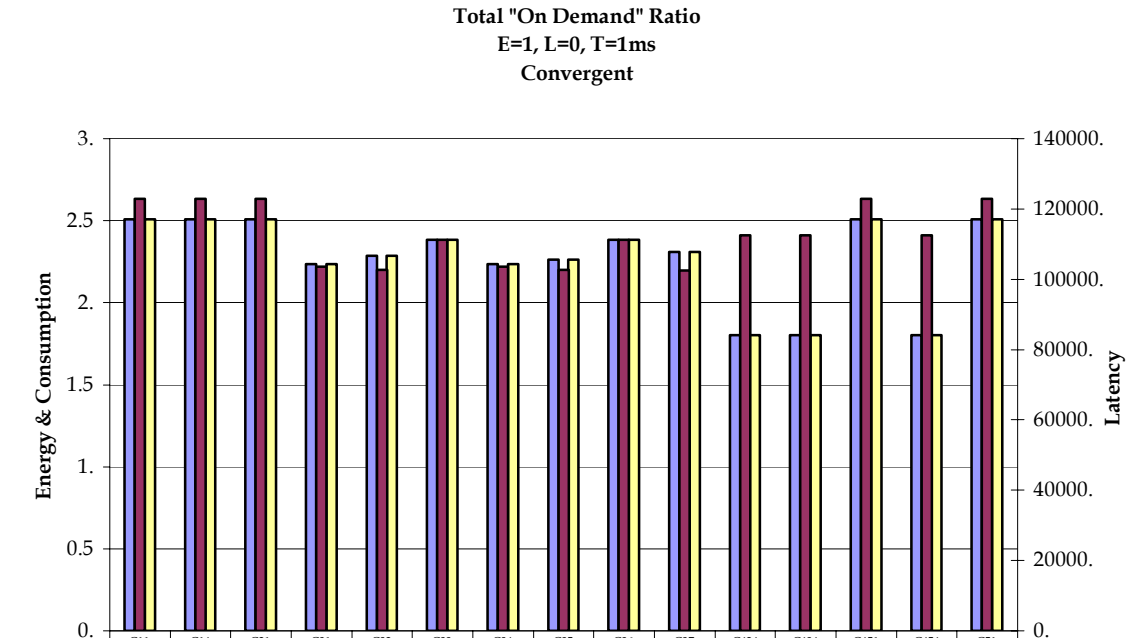
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	2.51	2.51	2.51	2.24	2.29	2.38	2.24	2.26	2.38	2.31	1.8	1.8	2.51	1.8	2.51
Consumption	2.51	2.51	2.51	2.24	2.29	2.38	2.24	2.26	2.38	2.31	1.8	1.8	2.51	1.8	2.51
Latency	122956.33	122956.33	122956.33	103602.57	102712.81	111181.73	103602.57	102717.79	111181.73	102447.84	112536.46	112541.46	122956.33	112541.46	122956.33

Figure 6.5: Total Output Ratio Histogram of Convergent Systems for "Preemptive" Method



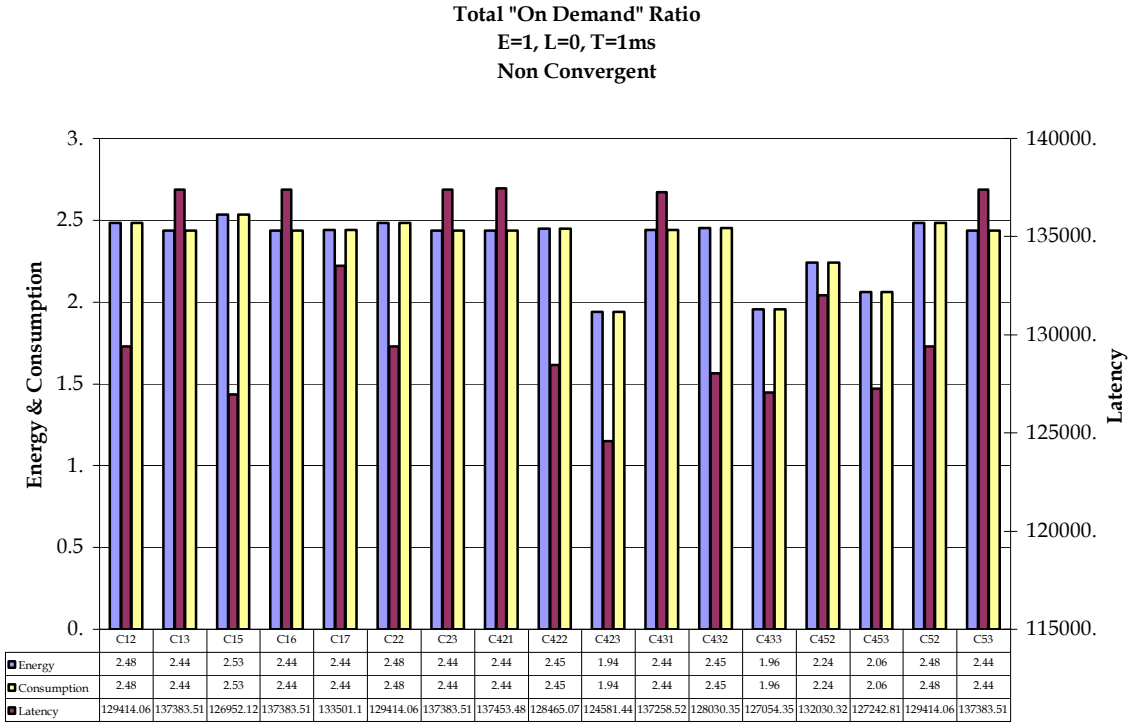
	C12	C13	C15	C16	C17	C22	C23	C421	C422	C423	C431	C432	C433	C452	C453	C52	C53
Energy	2.48	2.44	2.53	2.44	2.44	2.48	2.44	2.44	2.45	1.94	2.44	2.45	1.96	2.24	2.06	2.48	2.44
Consumption	2.48	2.44	2.53	2.44	2.44	2.48	2.44	2.44	2.45	1.94	2.44	2.45	1.96	2.24	2.06	2.48	2.44
Latency	129414.06	137383.51	126952.12	137383.51	133501.1	129414.06	137383.51	137453.48	128465.07	124581.44	137258.52	128030.35	127054.35	132030.32	127242.81	129414.06	137383.51

Figure 6.6: Total Output Ratio Histogram of Non-Convergent Systems for "Preemptive" Method



	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	2.51	2.51	2.51	2.24	2.29	2.38	2.24	2.26	2.38	2.31	1.8	1.8	2.51	1.8	2.51
Consumption	2.51	2.51	2.51	2.24	2.29	2.38	2.24	2.26	2.38	2.31	1.8	1.8	2.51	1.8	2.51
Latency	122956.33	122956.33	122956.33	103602.57	102712.81	111181.73	103602.57	102717.79	111181.73	102447.84	112536.46	112541.46	122956.33	112541.46	122956.33

Figure 6.7: Total Output Ratio Histogram of Convergent Systems for "On Demand" Method



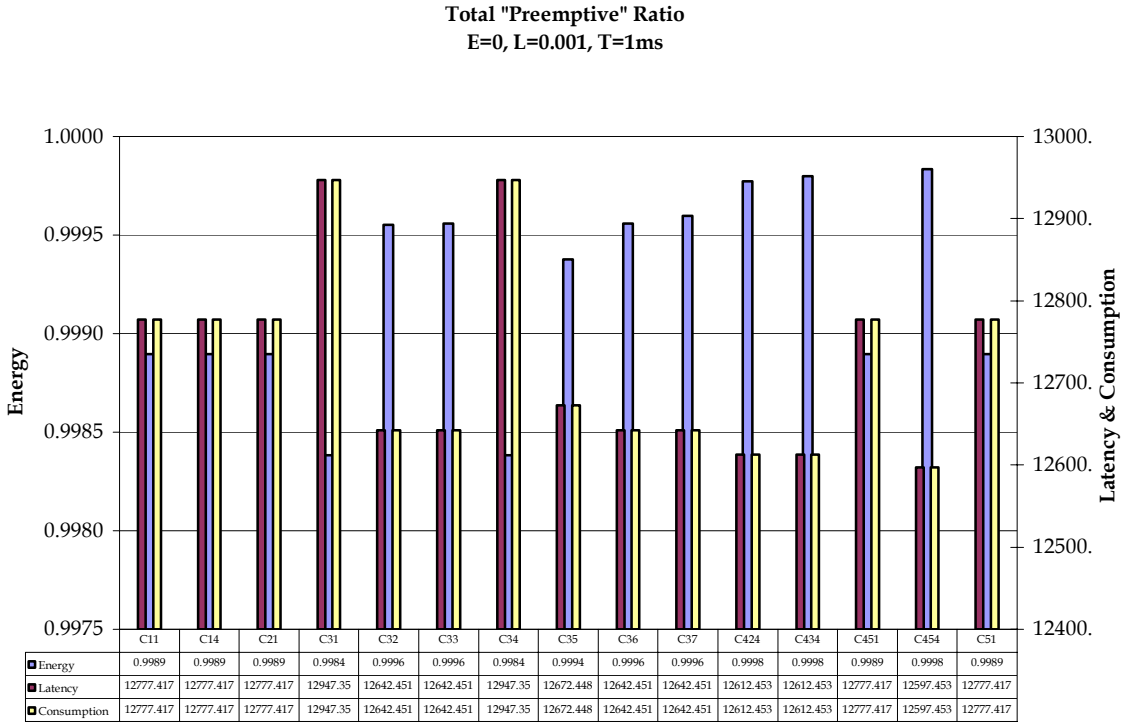
	C12	C13	C15	C16	C17	C22	C23	C421	C422	C423	C431	C432	C433	C452	C453	C52	C53
Energy	2.48	2.44	2.53	2.44	2.44	2.48	2.44	2.44	2.45	1.94	2.44	2.45	1.96	2.24	2.06	2.48	2.44
Consumption	2.48	2.44	2.53	2.44	2.44	2.48	2.44	2.44	2.45	1.94	2.44	2.45	1.96	2.24	2.06	2.48	2.44
Latency	129414.06	137383.51	126952.12	137383.51	133501.1	129414.06	137383.51	137453.48	128465.07	124581.44	137258.52	128030.35	127054.35	132030.32	127242.81	129414.06	137383.51

Figure 6.8: Total Output Ratio Histogram of Non-Convergent Systems for "On Demand" Method

c) Optimization of Latency

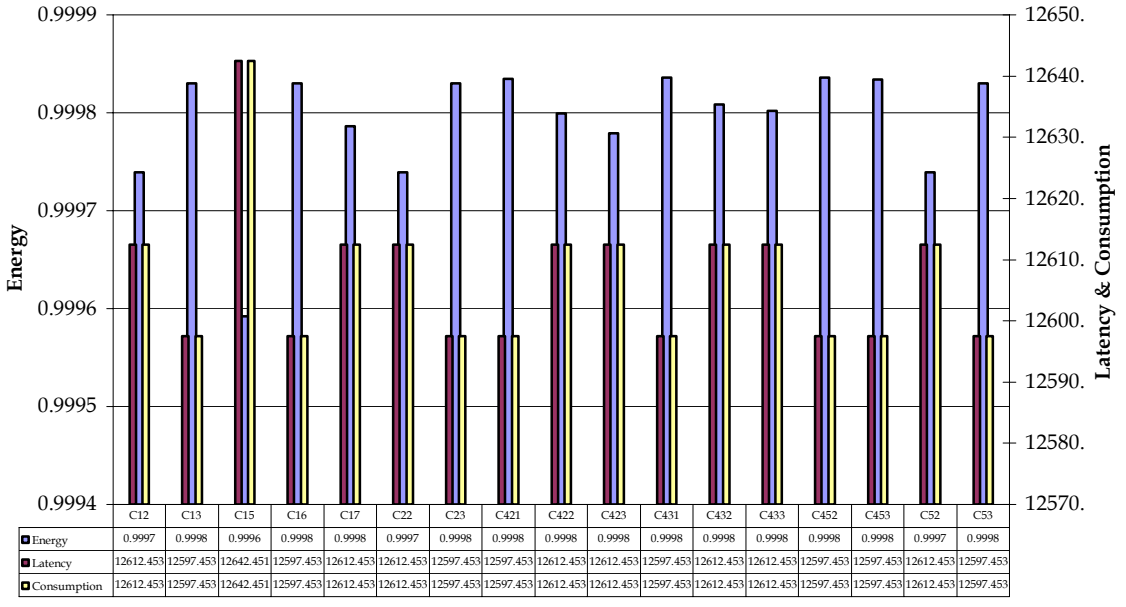
Figures 6.9 through 6.12 hold the histograms of the total output ratios of these simulations.

Unlike for the past two cases, none of the configurations in this simulation set bring the systems to converge to the correct optimal state. It can further be observed from the total output ratio figures that the energy and consumption expenditures are similar between the configurations. Configurations $\{C_{11}, C_{14}, C_{21}, C_{31} - C_{37}, C_{424}, C_{434}, C_{451}, C_{454}, C_{51}\}$, which lead the systems to converge in the previous two experiments, yield in this case slightly higher latency and consumption costs among all systems.



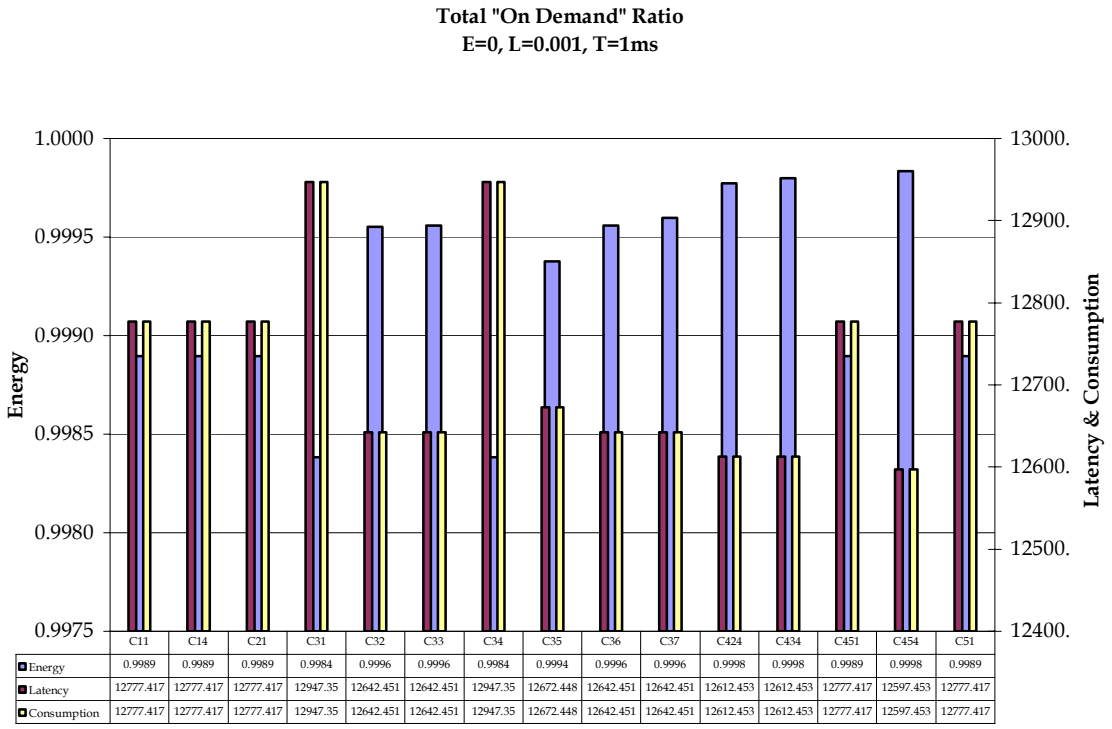
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.9989	0.9989	0.9989	0.9984	0.9996	0.9996	0.9984	0.9994	0.9996	0.9996	0.9998	0.9998	0.9989	0.9998	0.9989
Latency	12777.417	12777.417	12777.417	12947.35	12642.451	12642.451	12947.35	12672.448	12642.451	12642.451	12612.453	12612.453	12777.417	12597.453	12777.417
Consumption	12777.417	12777.417	12777.417	12947.35	12642.451	12642.451	12947.35	12672.448	12642.451	12642.451	12612.453	12612.453	12777.417	12597.453	12777.417

Figure 6.9: Total Output Ratio Histogram for "Preemptive" Method



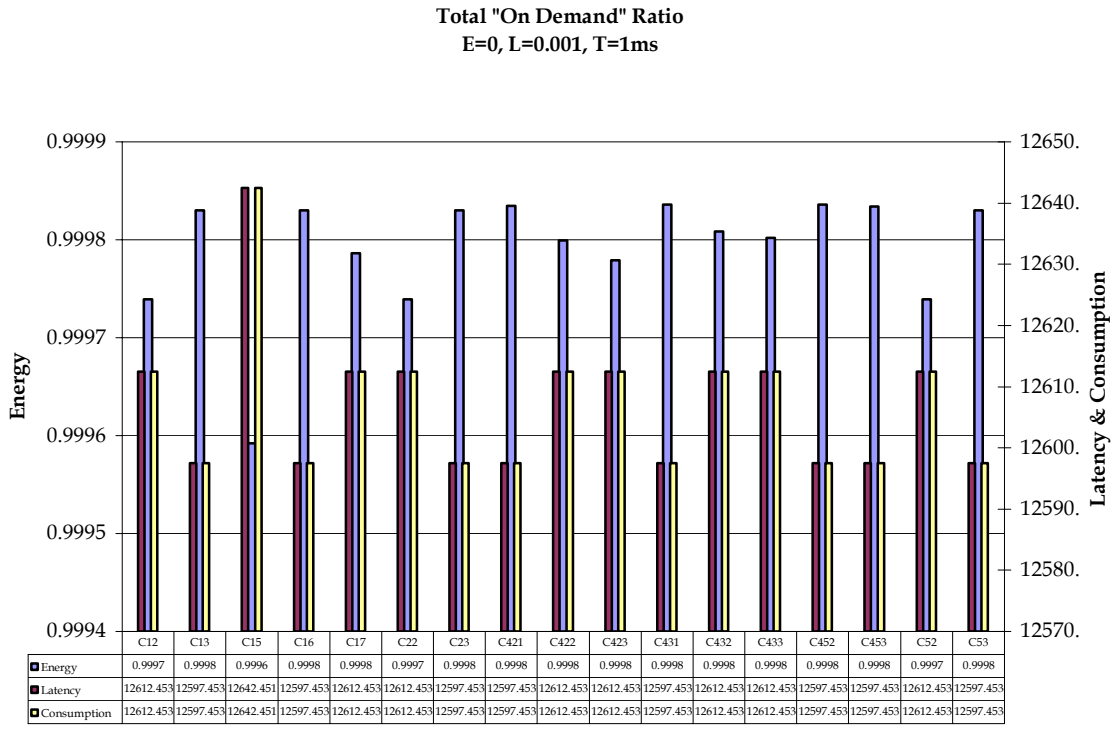
	C12	C13	C15	C16	C17	C22	C23	C421	C422	C423	C431	C432	C433	C452	C453	C52	C53
Energy	0.9997	0.9998	0.9996	0.9998	0.9998	0.9997	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9997	0.9998
Latency	12612.453	12597.453	12642.451	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12612.453	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12597.453
Consumption	12612.453	12597.453	12642.451	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12612.453	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12597.453

Figure 6.10: Total Output Ratio Histogram for "Preemptive" Method



	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.9989	0.9989	0.9989	0.9984	0.9996	0.9996	0.9984	0.9994	0.9996	0.9996	0.9998	0.9998	0.9989	0.9998	0.9989
Latency	12777.417	12777.417	12777.417	12947.35	12642.451	12642.451	12947.35	12672.448	12642.451	12642.451	12612.453	12612.453	12777.417	12597.453	12777.417
Consumption	12777.417	12777.417	12777.417	12947.35	12642.451	12642.451	12947.35	12672.448	12642.451	12642.451	12612.453	12612.453	12777.417	12597.453	12777.417

Figure 6.11: Total Output Ratio Histogram for "On Demand" Method



	C12	C13	C15	C16	C17	C22	C23	C421	C422	C423	C431	C432	C433	C452	C453	C52	C53
Energy	0.9997	0.9998	0.9996	0.9998	0.9998	0.9997	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9997	0.9998
Latency	12612.453	12597.453	12642.451	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12612.453	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12597.453
Consumption	12612.453	12597.453	12642.451	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12612.453	12597.453	12612.453	12612.453	12597.453	12597.453	12612.453	12597.453

Figure 6.12: Total Output Ratio Histogram for "On Demand" Method

Furthermore, there are no significant differences between the respective results of the two wake-up methods.

6.1.4. Preliminary Conclusions

To conclude the preliminary analyses, observation can be made that the systems tend to reach convergence more easily when configured with low reward updating parameters. Additionally, the systems that use the second non-linear learning scheme converge with a high reward parameter, for all degrees of non-linearity, and with a low reward parameter for high degrees of non-linearity. Finally, the first non-linear learning scheme is robust for the input distributions examined.

Furthermore, the two wake-up methods present equivalent energy results. In addition, the configurations yield equivalent results for systems that optimize energy and latency, and for systems that optimize only energy. However, when optimizing latency, no configuration brought the systems to convergence.

6.2. Real Trace Analysis: Four-State SLHA

6.2.1. Description of the Experiment

The tested input files were adapted from trace data obtained from the auspex file server archive [5]. The arrival times and lengths of requests for 0.4 million disk accesses were collected and separated into multiple trace files, corresponding to different hours of the day. The histograms of the idle-period lengths of each trace are illustrated in Appendix

C. As it can be noticed, the different input files have significantly different idle-period distributions, which will yield notably different behaviors of the systems.

Finally, the preferred configurations $\{C_{11}, C_{14}, C_{21}, C_{31} - C_{37}, C_{424}, C_{434}, C_{451}, C_{454}, C_{51}\}$, which yielded convergence in the first two preliminary simulation sets, were used for this part of the experimentation. Several sets of simulations were run for different values of the time increment $T = \{1,10,100\}\, ms$.

From the results of the preliminary experiments, simulations with the real traces were performed in two categories: optimization of energy and latency ($E = 1,\ L = 0.001$), and optimization of only energy ($E = 1,\ L = 0$). The optimization of only latency ($E = 0,\ L = 0.001$) was not tested due to the non-converging characteristics of the preliminary results.

6.2.2. Experimental Results

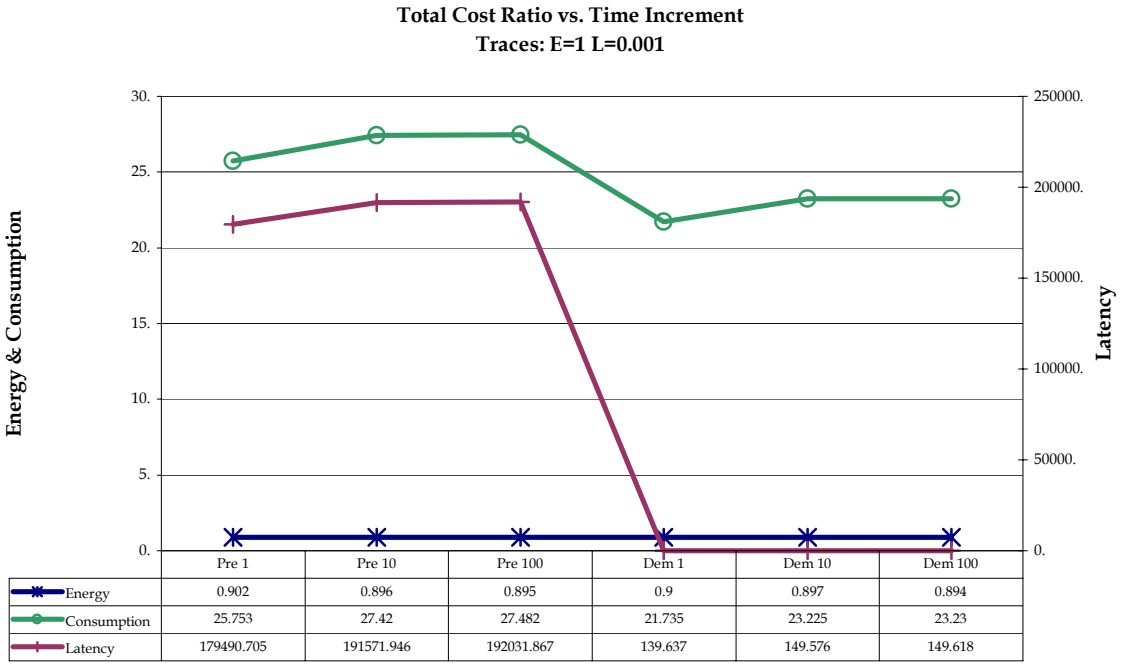
a) Optimization of Energy and Latency

Similarly to the preliminary results, the total output ratio histograms for these experiments for each value of T are illustrated in Appendix D.

From the files in Appendix D, it can be seen that configuration C_{454} yields low latency and consumption costs for both "Preemptive" and "On Demand" methods. Configurations $\{C_{32}, C_{33}, C_{36}, C_{37}\}$, however, yield lower energy expenditures while holding higher latency costs with the "Preemptive" wake-up. Moreover, configuration C_{36} yields the least energy expenditure with wake-up "On Demand". The opposite characteristics observed between energy and latency can be once more explained by the definition of consumption.

When comparing the results of the two wake-up methods, while energy expenditure is equivalent, it can be detected that the latency incurred by the systems is smaller with wake-up "On Demand". This is explained by the fact that in such wake-up, the optimal offline algorithm also provokes latency, such that the quotient of the two figures produces a smaller value than the corresponding ratio of the "Preemptive" method.

It can further be observed that the systems produce slightly different results according to the refreshing frequency T. This is related to the total idle-period lengths presented. If T is usually longer than the idle-periods, the system has a lower opportunity to switch states during the idle times, which makes it harder to reach optimality. On the other hand, if the time increment T is too low, the system will more favorably reach optimality but will however require more, possibly unnecessary, processing time and power.



	Pre 1	Pre 10	Pre 100	Dem 1	Dem 10	Dem 100
Energy	0.902	0.896	0.895	0.9	0.897	0.894
Consumption	25.753	27.42	27.482	21.735	23.225	23.23
Latency	179490.705	191571.946	192031.867	139.637	149.576	149.618

Figure 6.13: Total Cost Ratio vs. Time Increment

Figure 6.13 illustrates the total cost ratios as a function of the value of the time increment. From this graph, it can be observed that the "On Demand" wake-up method is more energy, latency and consumption conservative than the "Preemptive" method. Additionally, as the time increment T is increased, the systems present higher costs.

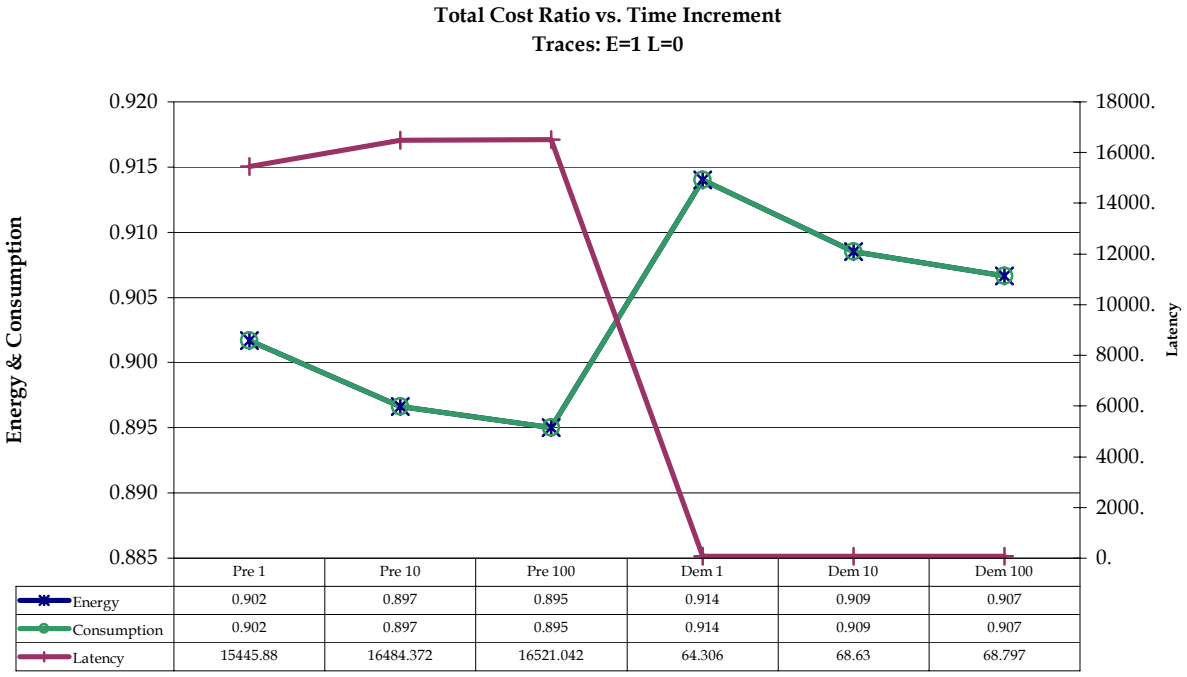
b) Optimization of Energy

Correspondingly to the previous simulations, the total output-ratio histograms for these experiments for each value of T are given in Appendix E.

From the files in Appendix E, it can be seen that configuration C_{454} yields low latency costs for both "Preemptive" and "On Demand" methods. Configurations $\{C_{33}, C_{36}, C_{37}\}$, however, yield lower energy and consumption expenditures while holding higher latency costs with the "Preemptive" wake-up. Moreover, configuration C_{36} yields the least energy and latency expenditure with wake-up "On Demand".

Similarly to the previous experiments, when comparing the results of the two wake-up methods, we can observe that, while the energy expenditure is equivalent, the latency incurred by the systems is smaller with wake-up "On Demand".

Figure 6.14 illustrates the total cost ratios as a function of the value of the time increment. From this graph, it can be observed that the "On Demand" wake-up method is more consumption conservative than the "Preemptive" method. Additionally, as the time increment T is increased, the systems present lower latency costs.



	Pre 1	Pre 10	Pre 100	Dem 1	Dem 10	Dem 100
Energy	0.902	0.897	0.895	0.914	0.909	0.907
Consumption	0.902	0.897	0.895	0.914	0.909	0.907
Latency	15445.88	16484.372	16521.042	64.306	68.63	68.797

Figure 6.14: Total Cost Ratio vs. Time Increment

6.2.3. Conclusions

From the real-trace simulations, it can be observed that "On Demand" wake-up tends to better minimize energy and latency expenditure than the "Preemptive" wake-up method.

Moreover, configurations $\{C_{33}, C_{36}, C_{37}\}$, corresponding to the first non-linear reinforcement scheme with high reward and penalty parameters, perform the best minimization of energy for the presented traces.

Furthermore, configuration C_{454}, corresponding to the second non-linear updating scheme with a high reward parameter and a high degree of nonlinearity, performs the best minimization of latency for the presented traces.

6.3. Conclusion

The results of the best SLHA models are given in Appendix F. The chosen SLHA systems presented the lowest results in either of the values measured: energy expenditure, latency incurred or consumption cost for each wake-up method.

When comparing the two methods for optimization, optimizing only energy yielded similar energy expenditures, but significantly lower latency costs than the method for the optimization of both energy and latency.

Chapter 7

Related DPM Strategies

This section of the research was inspired by the research detailed in [1]. Here, several former predictive algorithms are analyzed to be compared to the SLHA model for Dynamic Power Management.

7.1 Overview of the Models

Two groups of predictive strategies can be distinguished. The common trait of one group of methods is characterized by the series of thresholds used by the algorithms to determine the sequence of transitions of the models through lower power states. The algorithms of the second group, single-value prediction algorithms, express their prediction of the next idle period with a single value and transition to the power state optimal for the idle time predicted. The main difference between these two strategies is hence in the manner to determine the length of the next idle period.

7.1.1. Optimal Offline Algorithm (OPT)

The optimal algorithm is aware of the length of the upcoming idle periods beforehand, and is consequently capable of choosing the true optimal power mode for each idle period.

Assuming t is the length of the upcoming idle period, the chosen power mode S_i minimizes the following expression:

$$C_i = E \cdot (p_i \cdot T + e_i) + L \cdot t_i \text{, where } i \in [\![0, n]\!].$$

where parameters E and L are the weights of energy and latency respectively, as described in the previous chapters.

This algorithm switches to the optimal state at the start of the idle period and, according to the wake-up method selected, powers-up to the active state at the arrival of the next process request, or switches ahead to be active at the time of the input request.

7.1.2. Lower-Envelope Algorithm (DET)

This deterministic algorithm is presented in detail in [1]. Its strategy uses the approach of the optimal offline algorithm to determine at each time t the optimal state for the elapsed idle period.

Assuming t is the length of the upcoming idle period, the optimal state S_i satisfies the cost expression:

$$LE(t) = \min_i (p_i \cdot t + e_i) \text{, where } i \in [\![0, n]\!].$$

The Lower Envelope Algorithm follows the LE curve, remaining in the same state while in the linear parts of the function and transitioning to the next lower power mode at its discontinuities.

The power mode is updated periodically until the arrival of the next process request, time at which the model powers-up to the active state.

7.1.3. Online Probability Based Algorithm (OPBA)

The OPBA is the main approach investigated in [1]. This method is adapted from the Probability-Based Lower-Envelope Algorithm, which is a strategy based on the Lower Envelope Algorithm explained earlier.

a) Probability-Based Lower-Envelope Algorithm (PLEA)

The PLEA assumes that the idle period length is generated by a fixed distribution $\pi(t)$ known to the algorithm and determines the optimal switching threshold for deciding when to transition between two consecutive states.

In the two-state case, assuming that t is the length upcoming idle period, the optimal threshold τ minimizes the cost expression:

$$\int_0^{\tau} p_0 \cdot t \cdot \pi(t) \cdot dt + \int_{\tau}^{\infty} (p_0 \cdot \tau + e_1) \cdot \pi(t) \cdot dt$$

where p_0 is the power consumption in the active state, and e_1 is the start-up energy of the sleep state.

Similarly, in the n-state case, the PLEA determines the series of optimal thresholds to determine when to transition between two consecutive power modes, such as to minimize the cost:

$$\sum_{i=0}^{n} \int_{\tau_i}^{\tau_{i+1}} \left(p_i \cdot t + e_i \right) \cdot \pi(t) \cdot dt$$

where $\tau_0 = 0$, $\tau_{n+1} = \infty$, and τ_i is the threshold to transition between S_{i-1} and S_i.

To adapt this formula from the two-state case, the energy parameters of the two consecutive states are adjusted such that the power consumption of the lower power mode is assumed null, such as is the start-up energy of the other state. From the above formula, p_i is considered as the power consumption of the higher power mode, and e_i is the start-up energy of the other state.

b) Online Probability-Based Algorithm (OPBA)

Following this approach, the OPBA attempts to learn the input distribution $\pi(t)$ online. According to a preliminarily chosen window size w, the last w idle period lengths are collected by the software in a histogram. The algorithm subsequently analyzes the information recorded to predict the future length of the idle period, and consequently switches to the optimal power mode for the predicted idle period. Particularly, the histogram is sectioned in k partitions, grouping intervals of idle lengths. At each new request arrival, the counter $c_j \quad \forall j \in [\![0, k-1]\!]$ of the histogram bin r_j that enfolds the value of the latest idle-period length is incremented and the counter of the bin corresponding to the oldest idle-time value is decremented. The chosen threshold r_t is determined by:

$$\min_{r_t}\left[\sum_{j=1}^{t-1}\frac{c_j \cdot r_j}{w}\cdot\left(p_i - p_{i-1}\right)+\sum_{j=t}^{k}\frac{c_j}{w}\cdot\left[r_t\cdot\left(p_i - p_{i-1}\right)+\left(e_{i-1}-e_i\right)\right]\right]$$

where $r_0 = 0$, $r_k = \infty$, and τ_i is the threshold to transition between S_{i-1} and S_i.

7.1.4. Last-Period Algorithm (LAST)

The Last-Period algorithm is a single-valued prediction strategy. It predicts the length of the upcoming idle period according to the length of the latest idle-time.

7.1.5. Exponential Decay Algorithm (EXP)

The Exponential Decay algorithm, also a single-valued prediction algorithm, keeps a weighted average of the past idle period lengths.

Assuming at time t, $p(t)$ to be the current prediction and l to be the last idle-time, the prediction value is updated according to the following:

$$p(t+1) = \lambda \cdot p(t) + (1-\lambda)\cdot l$$

where $\lambda \in (0,1)$.

7.1.6. Adaptive Learning Tree Algorithm (TREE)

The Adaptive Learning Tree Algorithm develops an adaptive learning tree from the sequence of recent idle period lengths. It then employs the generated tree to predict the length of the next idle period.

7.2. Experimental Parameters

Experiments for these algorithms were realized similarly to the Real Trace Analyses described in Chapter 6. The software developed in [1] was simulated with the same traces as examined with the SLHA model. Both "Preemptive" wake-up and wake-up "On Demand" methods were simulated in each scheme, and only energy was considered for the reduction of the cost expenditure: $E = 1$, $L = 0$. Particularly, the following parameters of the outlined algorithms were employed.

7.2.1. Online Probability Based Algorithm (OPBA)

The histogram is divided in $w = 50$ bins and the thresholds are updated every ten requests. These parameters were determined as optimal in [1].

7.2.2. Exponential Decay Algorithm (EXP)

The coefficient used for updating the prediction value is $\lambda = 0.5$.

The results of these algorithms are given in Chapter 8, along with a comparison of their performance to the performance of the SLHA model.

Chapter 8

Conclusions and Future Work

8.1. Assessment of the SLHA model

In this final section of the study of Stochastic Learning Hybrid Automata for Dynamic Power Management, the results of the previously described former DPM strategies are compared to the performance of the SLHA model, given real-trace input distributions.

The total cost ratio histograms of the best SLHA models, classified by the two wake-up methods, are illustrated in Appendix G, along with the results of the former algorithms examined. The algorithms were run with parameters $E = 1$ and $L = 0$ to optimize energy for the DPM problem.

When comparing the results of the SLHA to the expenditures of the former DPM strategies, it can be observed that the costs of the SLHA systems are significantly lower in the "Preemptive" wake-up strategy. Indeed, the systems with configuration $\{C_{36-T=1}, C_{33-T=10}, C_{33-T=100}\}$, using the first non-linear reinforcement scheme with high

reward and penalty parameters, yield the lowest results among all the studied former strategies. Moreover, configuration C_{454}, using the second non-linear updating algorithm with a high reward parameter and a high degree of non-linearity, yields the lowest total latency cost while keeping the energy ratio around unity. In addition, these configurations outperform the former algorithms LAST and TREE in total energy expenditure.

Furthermore, the total energy ratios of the selected SLHA systems are all less than or equal to unity, reaching as low as 0.59. This indicates a superior performance in energy efficiency than the offline algorithm. This phenomenon is once more due to the fact that consumption is a weighted sum of energy and latency. A reduced consumption can hence be attained by either a reduction in energy or in latency, potentially retaining the other figure high.

When examining the results of the "On Demand" method, it can be observed that the SLHA systems with configurations $\{C_{36-T=1}, C_{33-T=10}, C_{33-T=100}\}$ also yield lower energy expenditures than all of the former algorithms except EXP, while presenting significantly higher latency costs. The energy ratios of these systems however reach as low as 0.6. Furthermore, the SLHA system with configuration $C_{454-T=1}$ presents the lowest energy cost among all the studied models, except DET. Its energy expenditure is however much higher than any of the other models examined.

Figures 8.1 and 8.3 illustrate the total energy ratios versus total latency ratios for all the studied systems for "Preemptive" wake-up and wake-up "On Demand" respectively. Figure 8.2 is a detailed view of Figure 8.1 around the result of the optimal offline algorithm.

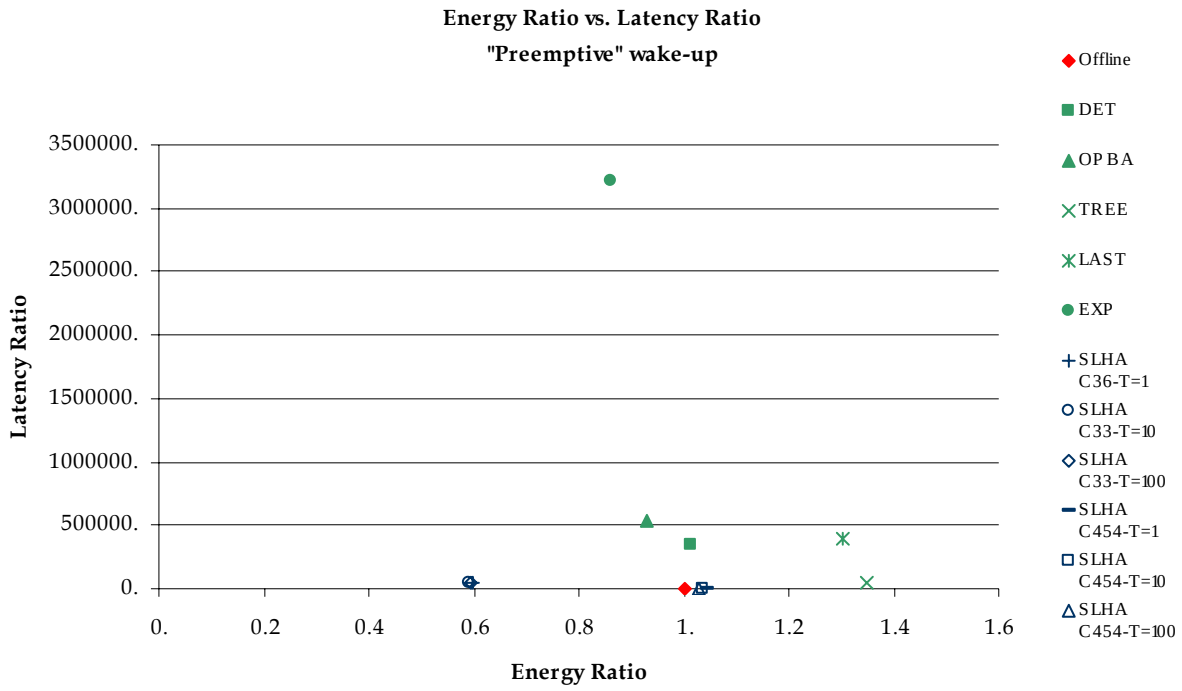


Figure 8.1: Total Energy Ratio vs. Total Latency Ratio for "Preemptive" wake-up

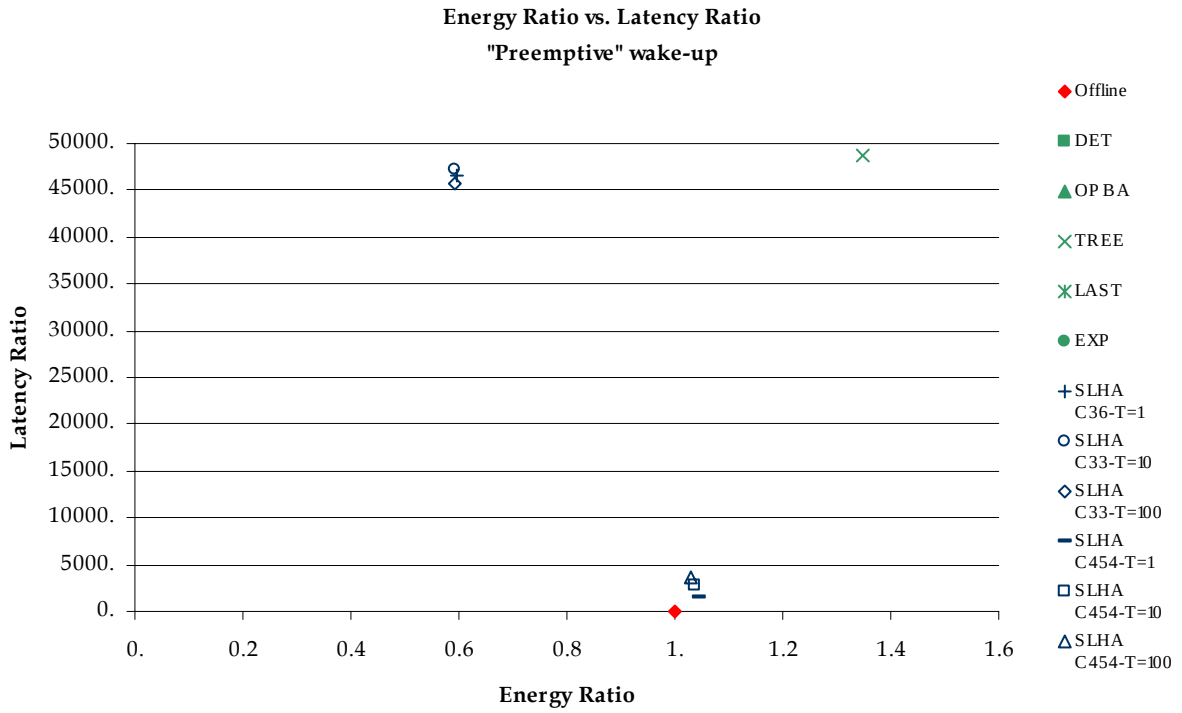


Figure 8.2: Close-up of the Total Energy Ratio vs. Total Latency Ratio for "Preemptive" wake-up

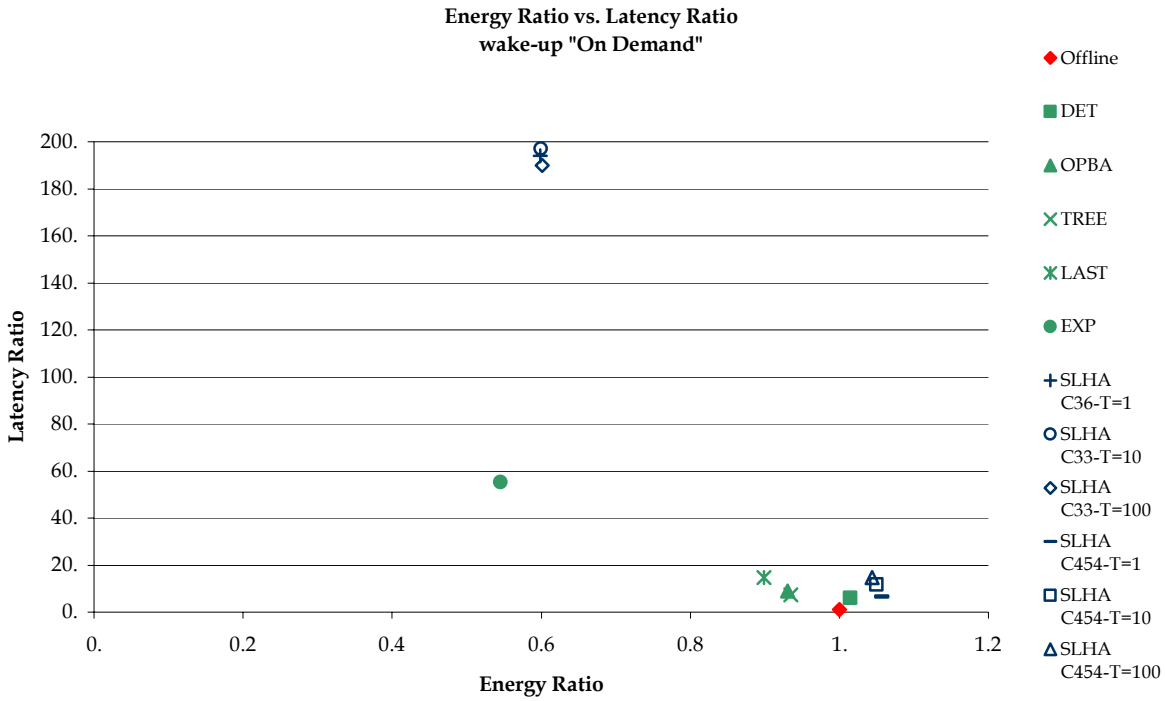


Figure 8.3: Total Energy Ratio vs. Total Latency Ratio for "On Demand" wake-up

8.2. Conclusion

In conclusion, the Stochastic Learning Hybrid Automata mathematical model for Dynamic Power Management of Embedded Systems proved its superiority compared to the former strategies presented in literature with "Preemptive" wake-up, for the examined input patterns. For wake-up "On Demand", results were observed to be enhanced either for the conservation of energy or the prevention of latency, but optimality was not reached for both figures simultaneously.

Given a Dynamic Power Management problem for embedded systems, choice should therefore be made on selecting the presented Stochastic Learning Hybrid Automata

model, configured with either the first non-linear learning algorithm with a low reward parameter, or with the second non-linear learning scheme with a high reward parameter and a high degree of nonlinearity. Selection should also be made on performing with the "Preemptive" wake-up strategy and optimization should only be performed for the minimization of energy expenditure.

8.3. Future Work

Stochastic Learning Automata has demonstrated to be significantly functional for the dynamic management of power in embedded system. Its high versatility presents considerable capabilities to adapt to different structures and problems. Further work may be done in developing the hybrid automaton to manipulate additional parameters that would assist in the learning process. Additionally, work may be extended in the area of the reinforcement schemes, to analyze the finest configuration parameters for certain environments. This would lead to the learning being done more rapidly and more accurately. All these adjustments may lead to a finer learning curve that would mimic more closely the behavior of the optimal offline algorithm.

Bibliography

[1] S. Irani, S. Shukla, and R. Gupta, "Online Strategies for Dynamic Power Management in Systems with Multiple Power Saving States", ACM Transactions on Embedded Computing Systems, Special Issue on Power-Aware Embedded Computing, Aug. 2003.

[2] van der Schaft, and H. Schumacher, "An Introduction to Hybrid Dynamical Systems", 2000.

[3] K. S. Narendra, and M. A. L. Thathachar, "Learning Automata: An Introduction", 1989.

[4] Technical specifications of hard drive IBM Travelstar VP 2.5inch, available at, http://www.hgst.com/hdd/support/table.htm#Travelstar, 1996. – 2004.

[5] Auspex File Traces from the NOW project, available at, http://now.cs.berkeley.edu/Xfs/AuspexTraces/auspex.html, 1993.

[6] T. Erbes, "SLHA and Offline simulation software for DPM". http://fermat.ece.vt.edu/teodora, 2003.

[7] Microsoft, "OnNow Power Management Architecture for Applications". http://www.microsoft.com/whdc/hwdev/tech/onnow/OnNowApp.mspx, Aug. 1997.

[8] Intel, Microsoft, and Toshiba, "Advanced Configuration and Power Interface Specification". http://www.acpi.info, Dec. 1996.

[9] C. Pereira, R. Gupta, P. Spanos, and M. Srivastava, "A Power Aware API". Power Aware Computing, 2002.

[10] D. Ramanathan, "High-Level Timing and Power Analysis for Embedded Systems", PhD thesis, University of California at Irvine, Sept. 2000.

[11] Chi-Hong Hwang, C. Allen, and H. Wu, "A Predictive System Shutdown Method For Energy Saving of Event-Driven Computation", in Proceedings of the IEEE/ACM International Conference on Computer Aided Design, pp. 28-32, Nov. 1996.

[12] M. B. Srivastava, A. P. Chandrakasan, and R. W. Broderson, "Predictive Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation", IEEE Transactions on VLSI Systems, 4(1):42--54, Mar. 1996.

[13] L. Benini, and G. De Micheli, "Dynamic Power Management: Design Techniques and CAD Tools", Kluwer Academic Publishers, 1998.

[14] Q. Qiu, and M. Pedram, "Dynamic Power Management Based on Continuous Time Markov Decision Processes", In Proceedings of the Design Automation Conference (DAC), 1999.

[15] Q. Qiu, Q. Wu, and M. Pedram, "Stochastic Modeling of Power Managed Systems: Construction and Optimization", in Proceedings of the International Symposium of Low Power Electronics and Design, pp. 194-199, Aug. 1999.

[16] R. Karlin, M. Manasse, L. McGeoch, and S. Owicki, "Randomized competitive algorithms for non-uniform problems", in 1st Annual ACMSIAM Symposium on Discrete Algorithms, pp. 301-309, Jan. 1990.

[17] D. Ramanathan, S. Irani, and R. Gupta, "An Analysis of System Level Power Management Algorithms and their effects on Latency", Accepted to be published at the IEEE Transactions on Computer Aided Design.

[18] L. Benini, A. Bogliolo, G. Paleologo, and G. D. Micheli. "Policy Optimization for Dynamic Power Management", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 18(6): 813-833, 1999.

[19] E. Y. Chung, L. Benini, and G. D. Micheli, “Dynamic Power Management using Adaptive Learning Tree”, In proceedings of ICCAD, 1999.

[20] Y. Lu, E. Chung, T. Simunic, L.Benini, and G. De Micheli, “Quantitative Comparison of Power Management Algorithms”, Proceedings of Design Automation and Test, Europe, 2000.

[21] T. Simunic, "Energy Efficient System Design and Utilization", Ph.D Thesis, Stanford University, 2001.

Appendix A

Preliminary Input Histograms

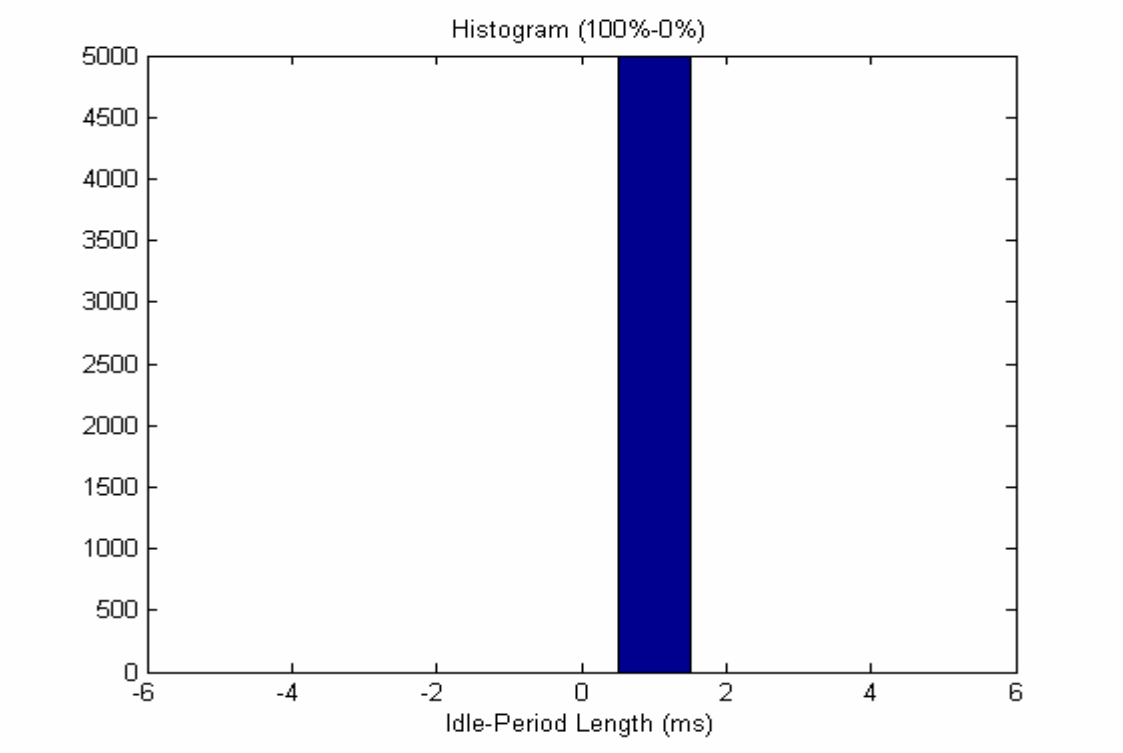


Figure A.1: Histogram 100%-0%

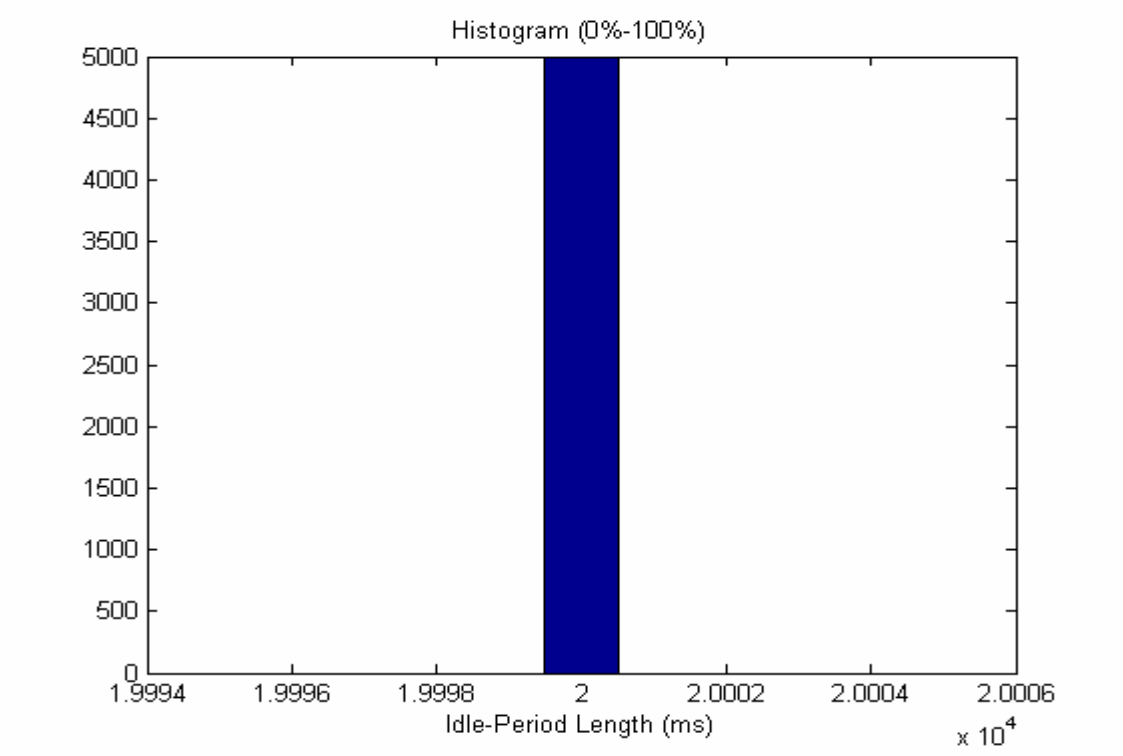


Figure A.2: Histogram 0%-100%

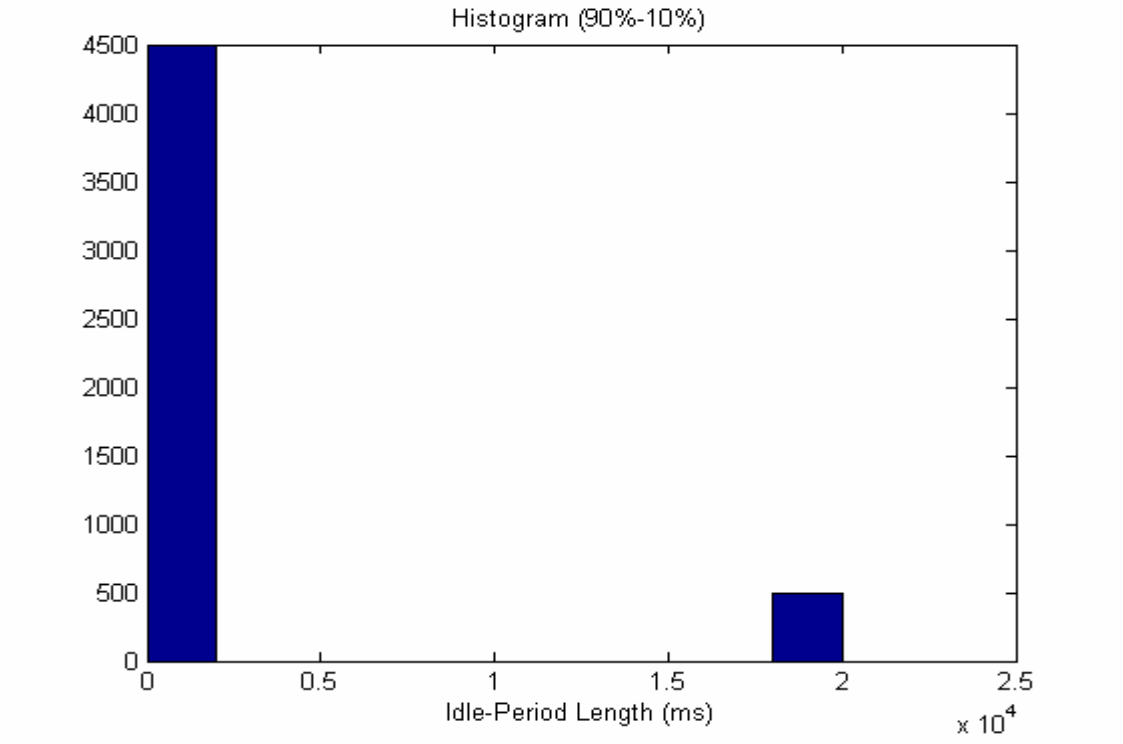


Figure A.3: Histogram 90%-10%

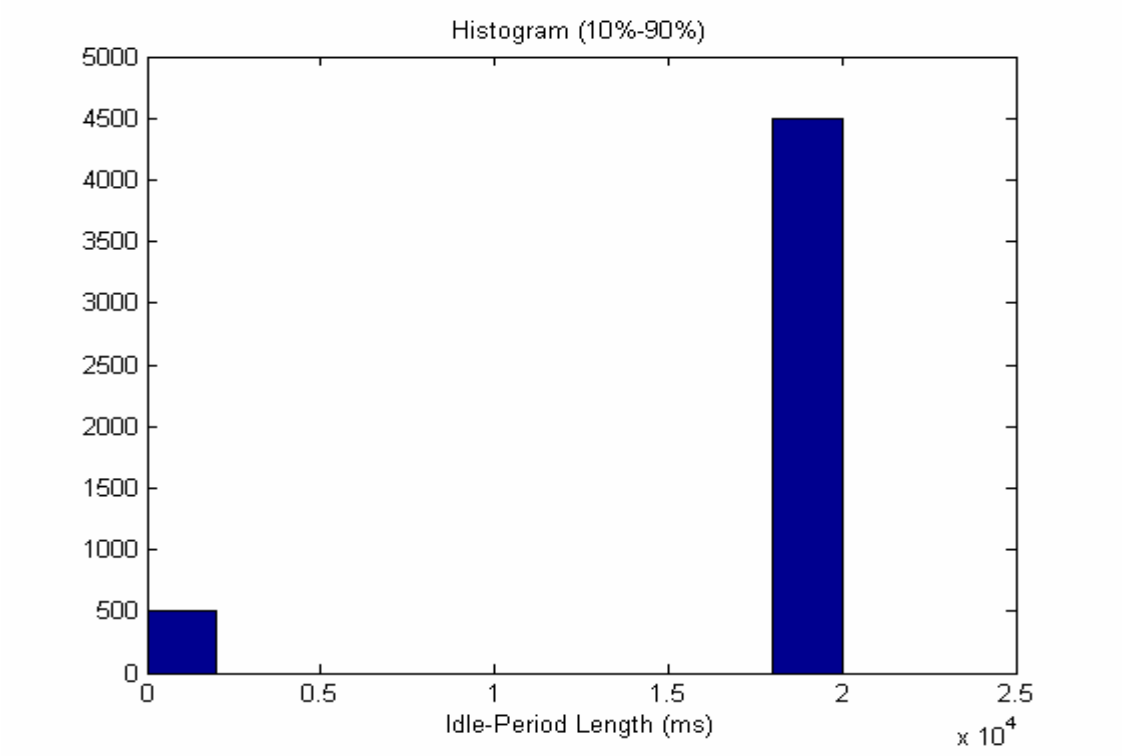


Figure A.4: Histogram 10%-90%

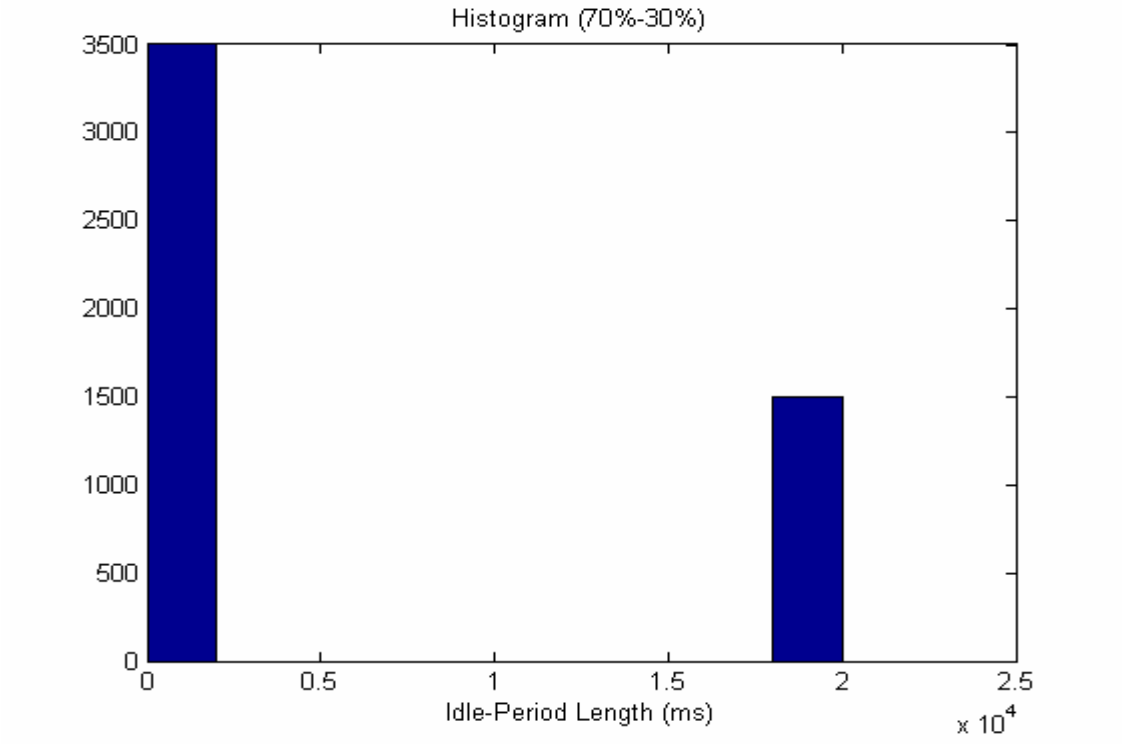


Figure A.5: Histogram 70%-30%

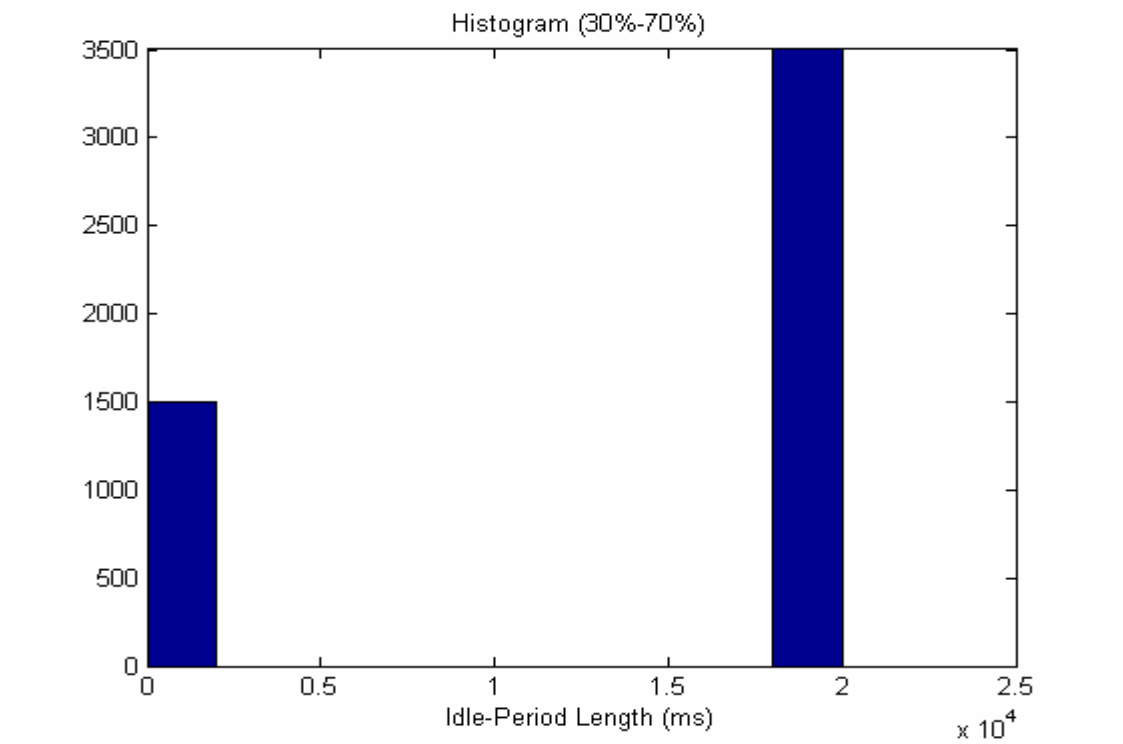


Figure A.6: Histogram 30%-70%

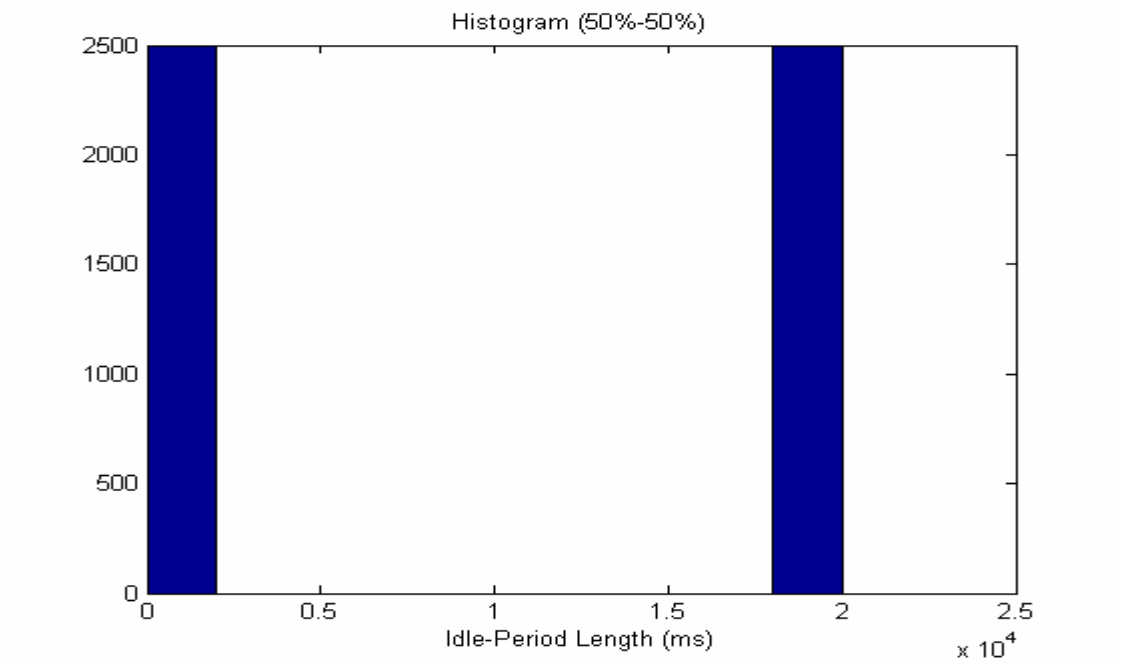


Figure A.7: Histogram 50%-50%

Appendix B

Configuration Files

```
//Config c11  A=B
A                0.1
B                0.1
D                  1
O                  1
Q
```

```
//Config c12  A=B
A                0.5
B                0.5
D                  1
O                  1
Q
```

```
//Config c13  A=B
A                0.9
B                0.9
D                  1
O                  1
Q
```

```
//Config c14  A<B
A                0.1
B                0.9
D                  1
O                  1
Q
```

```
//Config c15  A<B
A                0.3
B                0.7
D                  1
O                  1
Q
```

```
//Config c16  A>B
A                0.9
B                0.1
D                  1
O                  1
Q
```

```
//Config c17  A>B
A                0.7
B                0.3
D                  1
O                  1
Q
```

```
//Config c21  B=0
A                0.1
B                  0
D                  1
O                  2
Q
```

```
//Config c22  B=0
A                0.5
B                  0
D                  1
O                  2
Q
```

```
//Config c23  B=0
A                0.9
B                  0
D                  1
O                  2
Q
```

```
//Config c31  A=B
A                0.1
B                0.1
D                  1
O                  3
Q
```

```
//Config c32  A=B
A                0.5
B                0.5
D                  1
O                  3
Q
```

```
//Config c33  A=B
A                0.9
B                0.9
D                  1
O                  3
Q
```

```
//Config c34  A<B
A                0.1
B                0.9
D                  1
O                  3
Q
```

```
//Config c35  A<B
A                0.3
B                0.7
D                  1
O                  3
Q
```

```
//Config c36  A>B
A                0.9
B                0.1
D                  1
O                  3
Q
```

```
//Config c37  A>B
A                0.7
B                0.3
D                  1
O                  3
Q
```

```
//Config c421  B=0
A                0.1
B                  0
D                  2
O                  4
Q
```

```
//Config c422  B=0
A                0.5
B                  0
D                  2
O                  4
Q
```

```
//Config c423  B=0
A                0.9
B                  0
D                  2
O                  4
Q
```

```
//Config c424  B=0
A                  1
B                  0
D                  2
O                  4
Q
```

```
//Config c431  B=0
A                0.1
B                  0
D                  3
O                  4
Q
```

```
//Config c432  B=0
A                0.5
B                  0
D                  3
O                  4
Q
```

```
//Config c433  B=0
A                0.9
B                  0
D                  3
O                  4
Q
```

```
//Config c434  B=0
A                  1
B                  0
D                  3
O                  4
Q
```

```
//Config c451  B=0
A                0.1
B                  0
D                  5
O                  4
Q
```

```
//Config c452  B=0
A                0.5
B                  0
D                  5
O                  4
Q
```

```
//Config c453  B=0
A                0.9
B                  0
D                  5
O                  4
Q
```

```
//Config c454  B=0
A                  1
B                  0
D                  5
O                  4
Q
```

```
//Config c51   B=0
A                0.1
B                  0
D                  1
O                  5
Q
```

```
//Config c52  B=0
A                0.5
B                  0
D                  1
O                  5
Q
```

```
//Config c53  B=0
A                0.9
B                  0
D                  1
O                  5
Q
```

Appendix C

Real-Trace Input Histograms

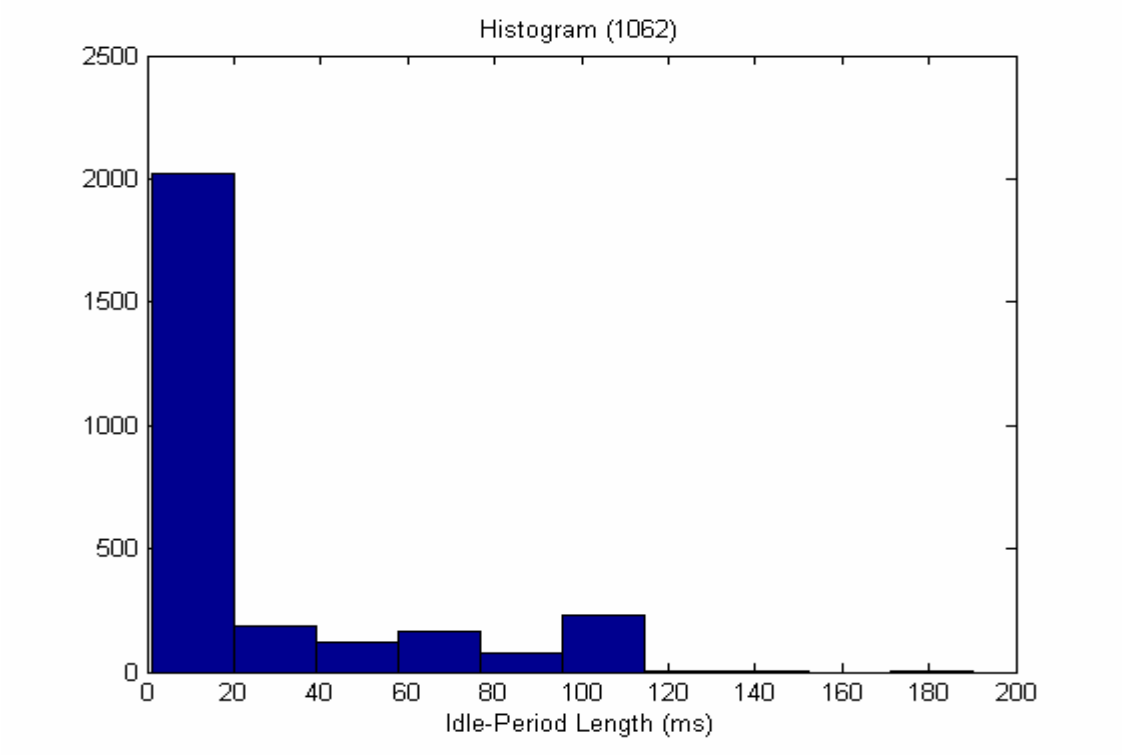


Figure C.1: Histogram H1062

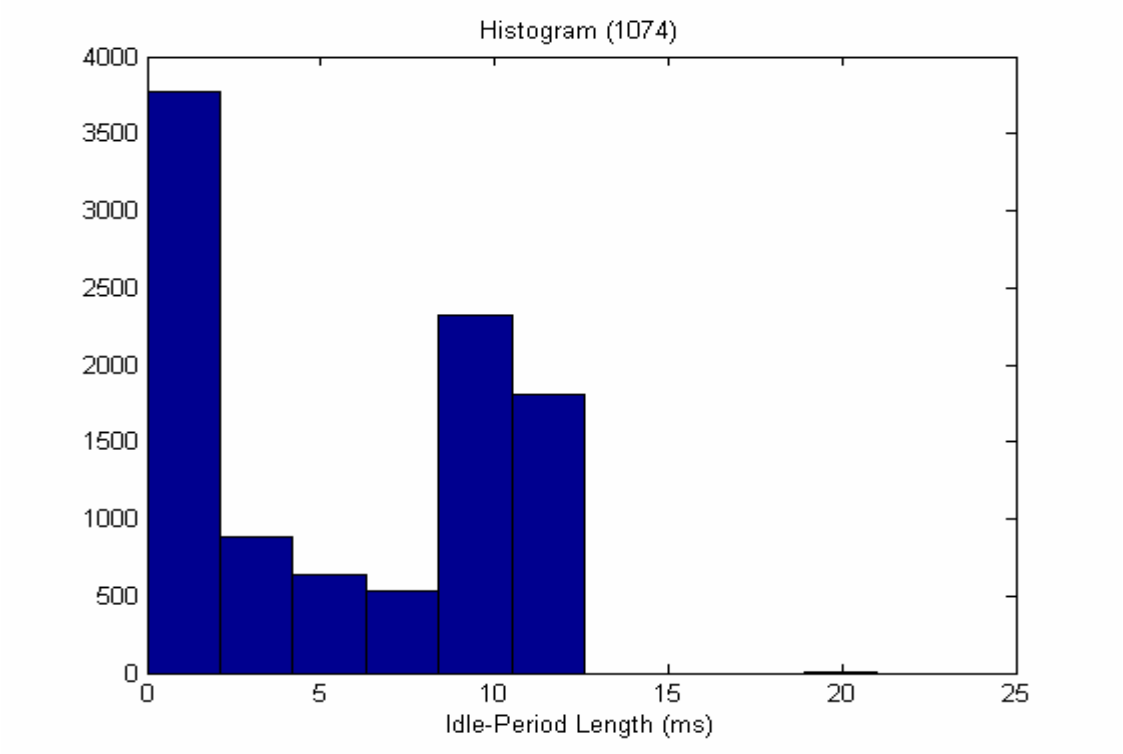


Figure C.2: Histogram H1074

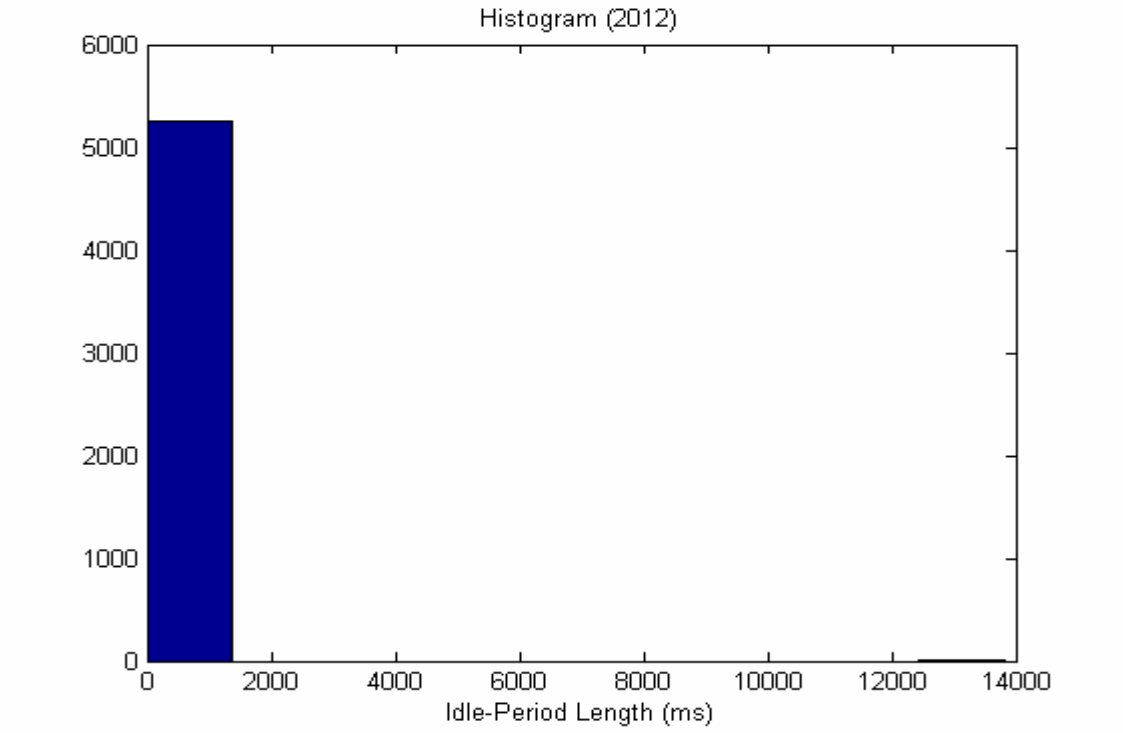


Figure C.3: Histogram H2012

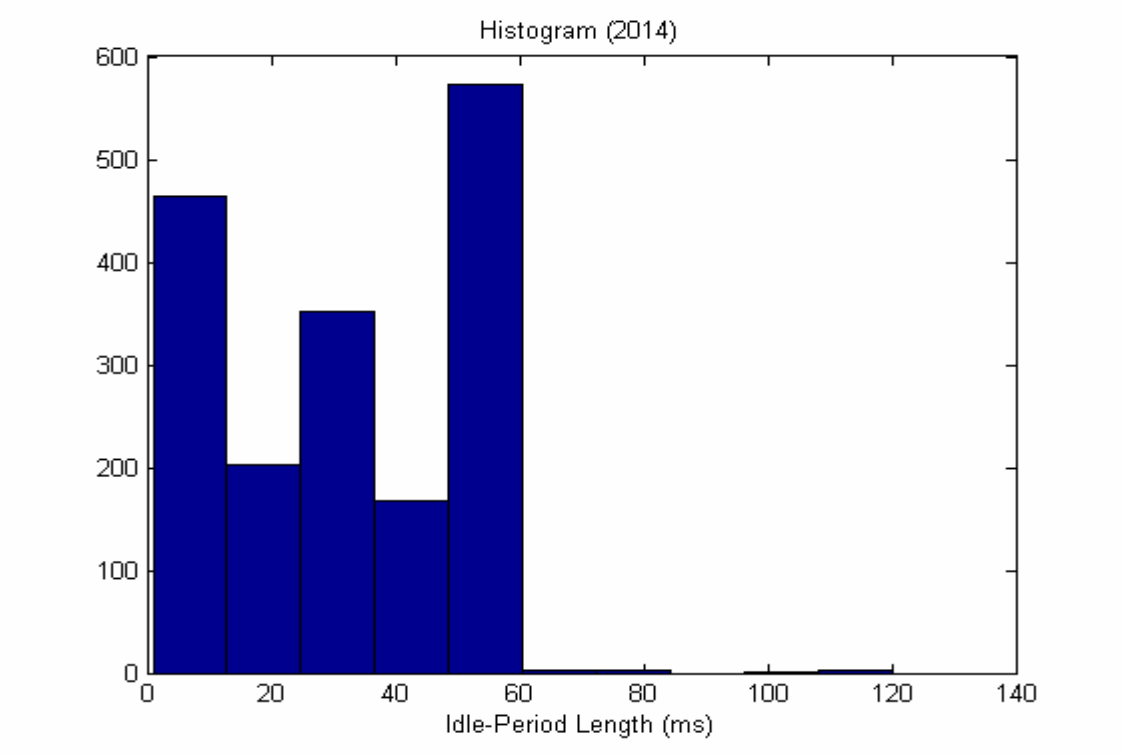


Figure C.4: Histogram H2014

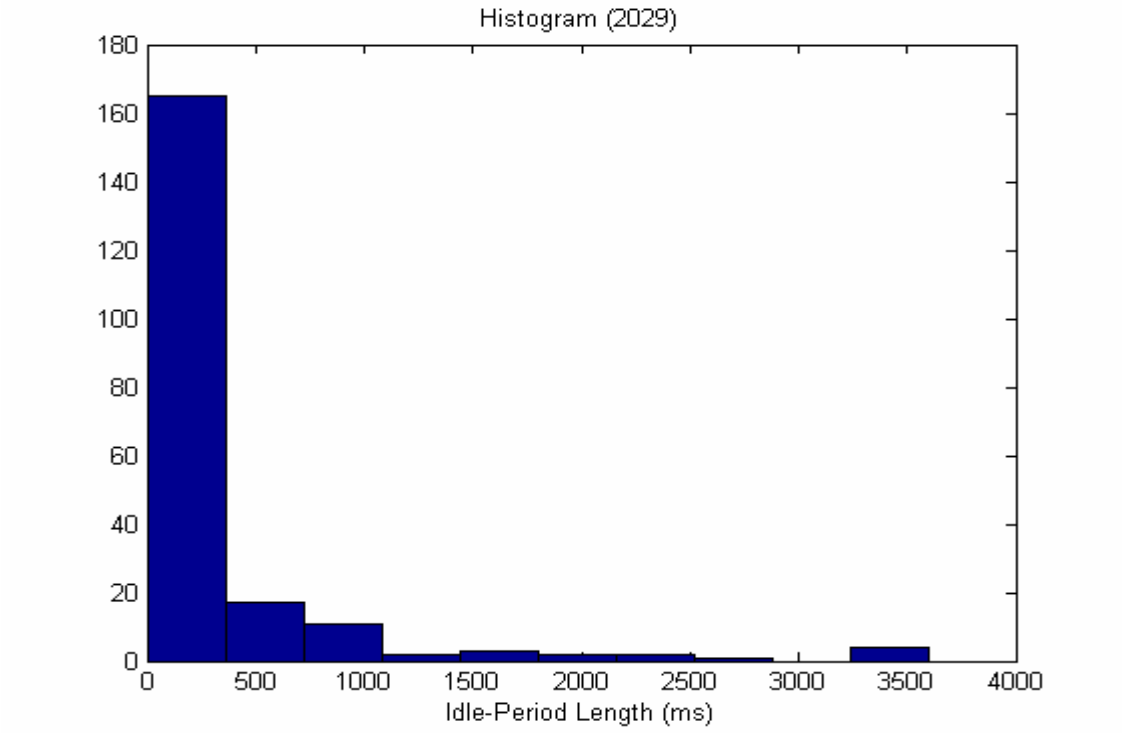


Figure C.5: Histogram H2029

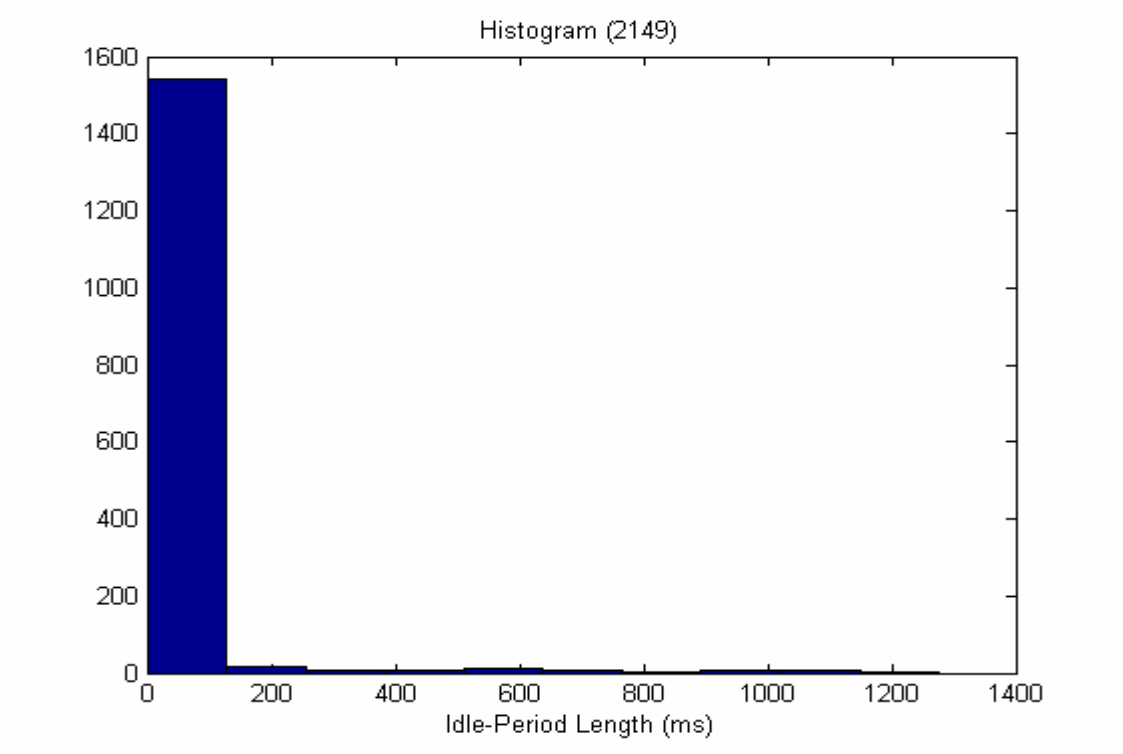


Figure C.6: Histogram H2149

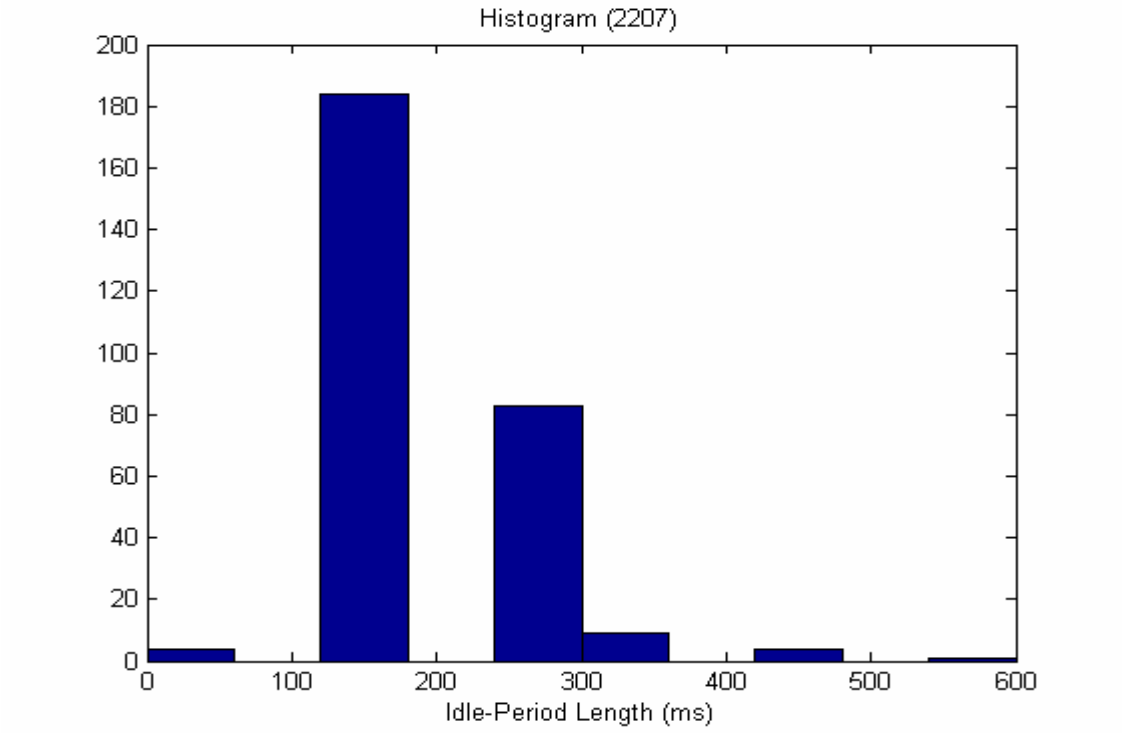


Figure C.7: Histogram H2207

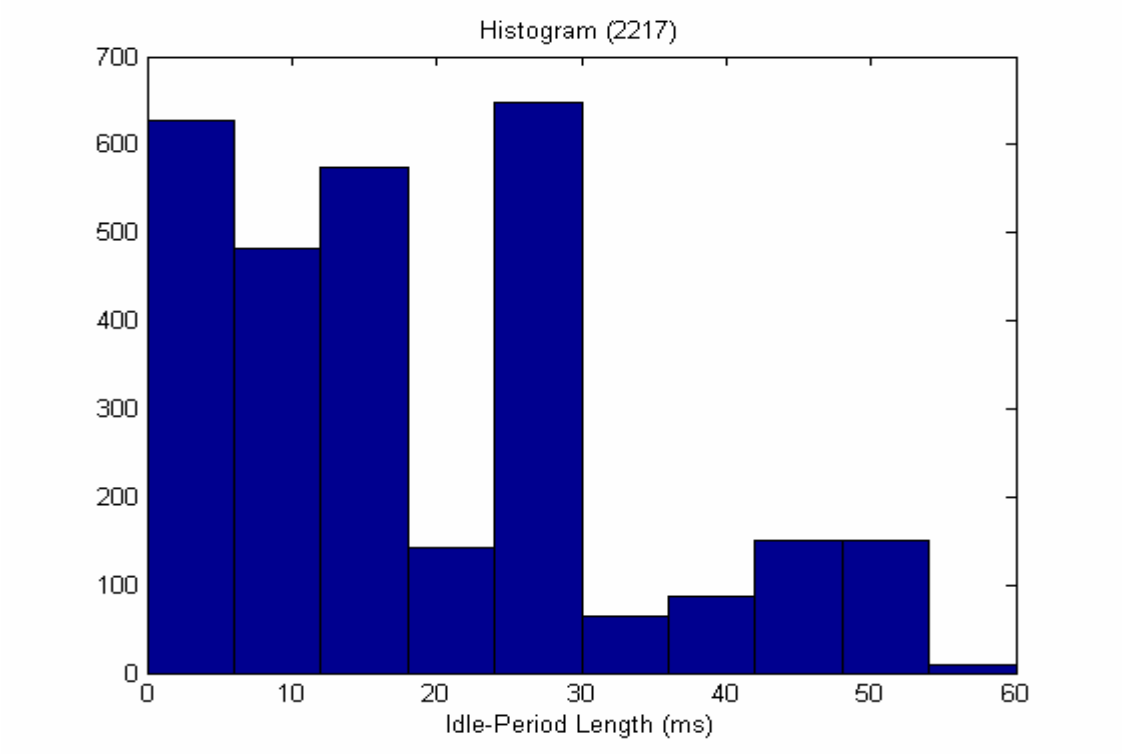


Figure C.8: Histogram H2217

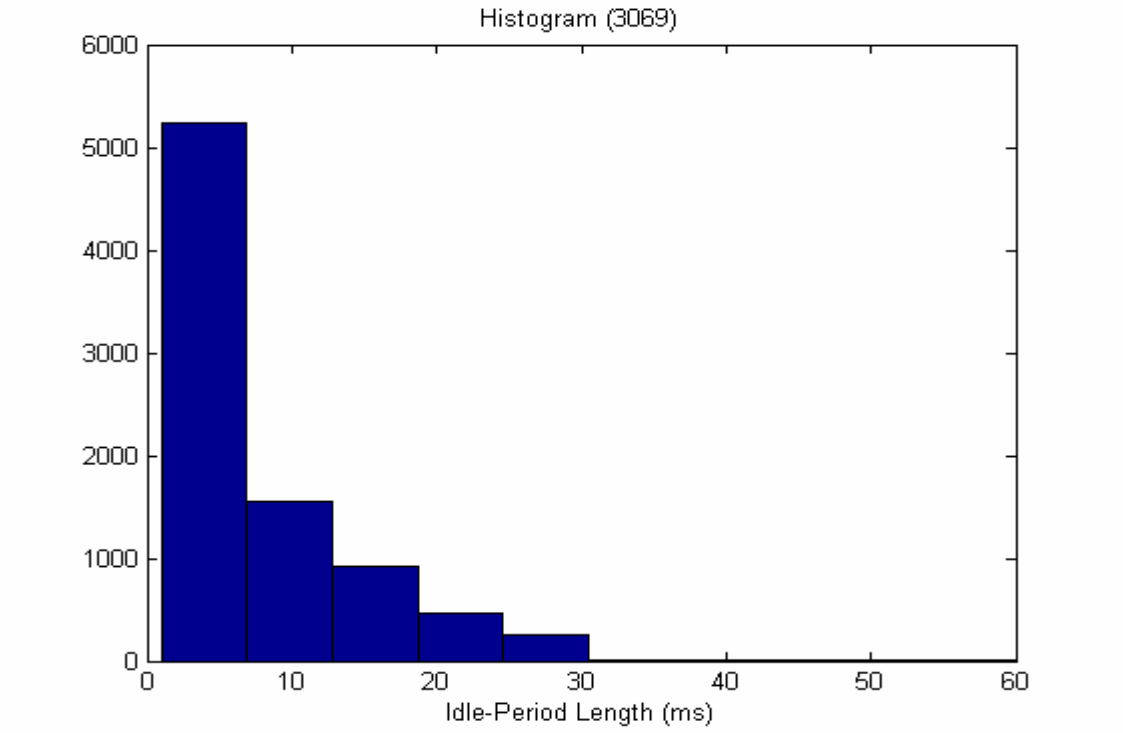


Figure C.9: Histogram H3069

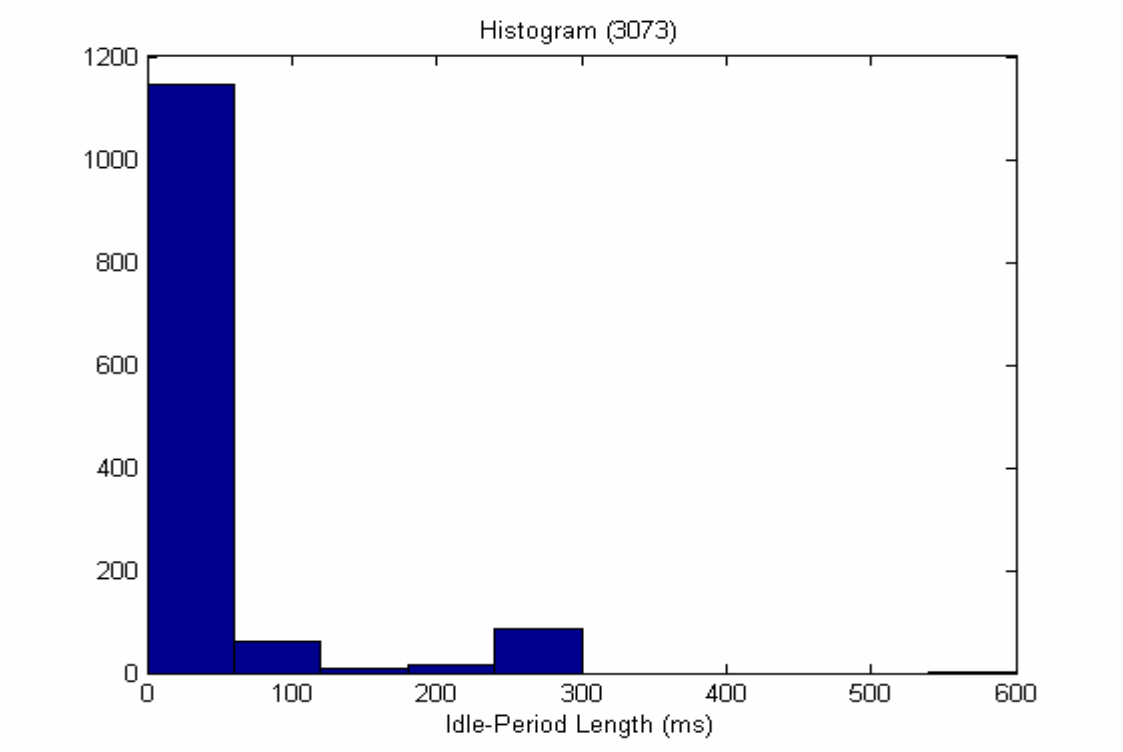


Figure C.10: Histogram H3073

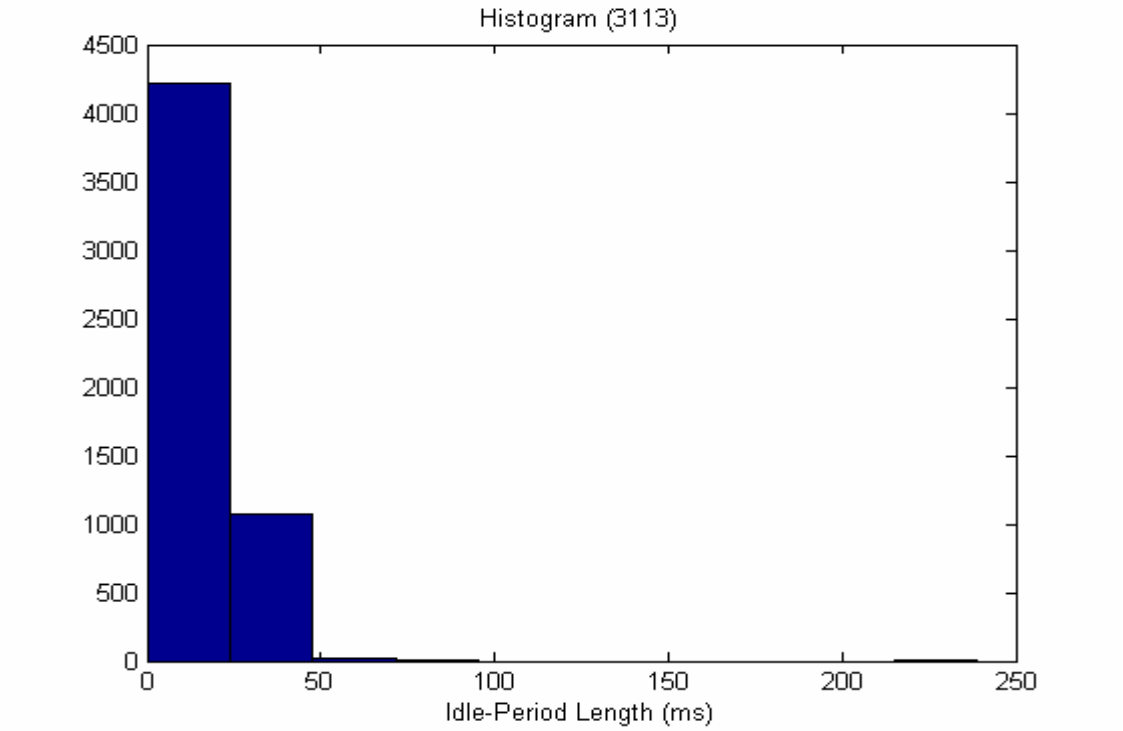


Figure C.11: Histogram H3113

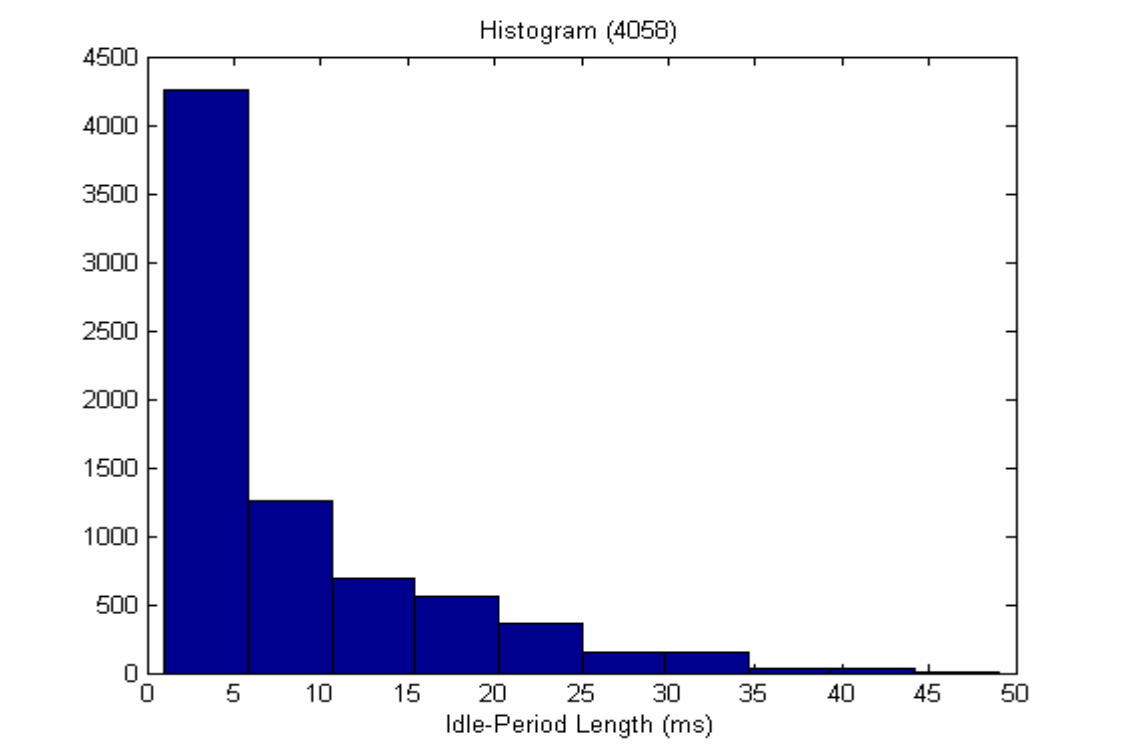


Figure C.12: Histogram H4058

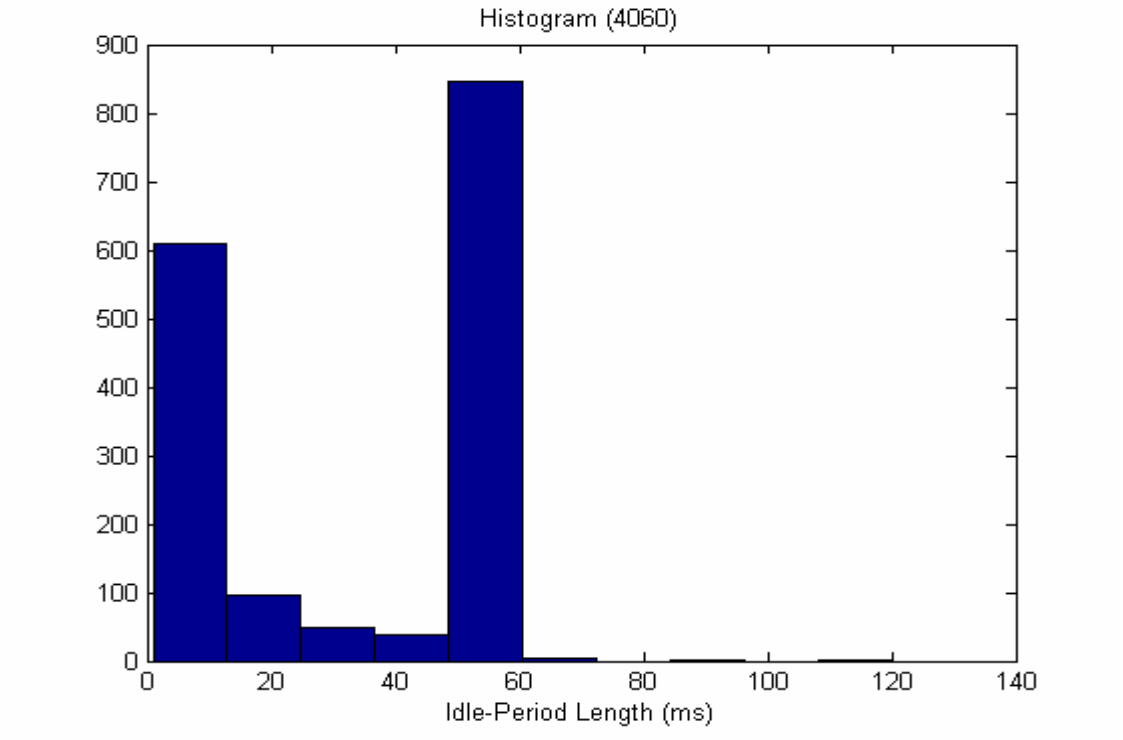


Figure C.13: Histogram H4060

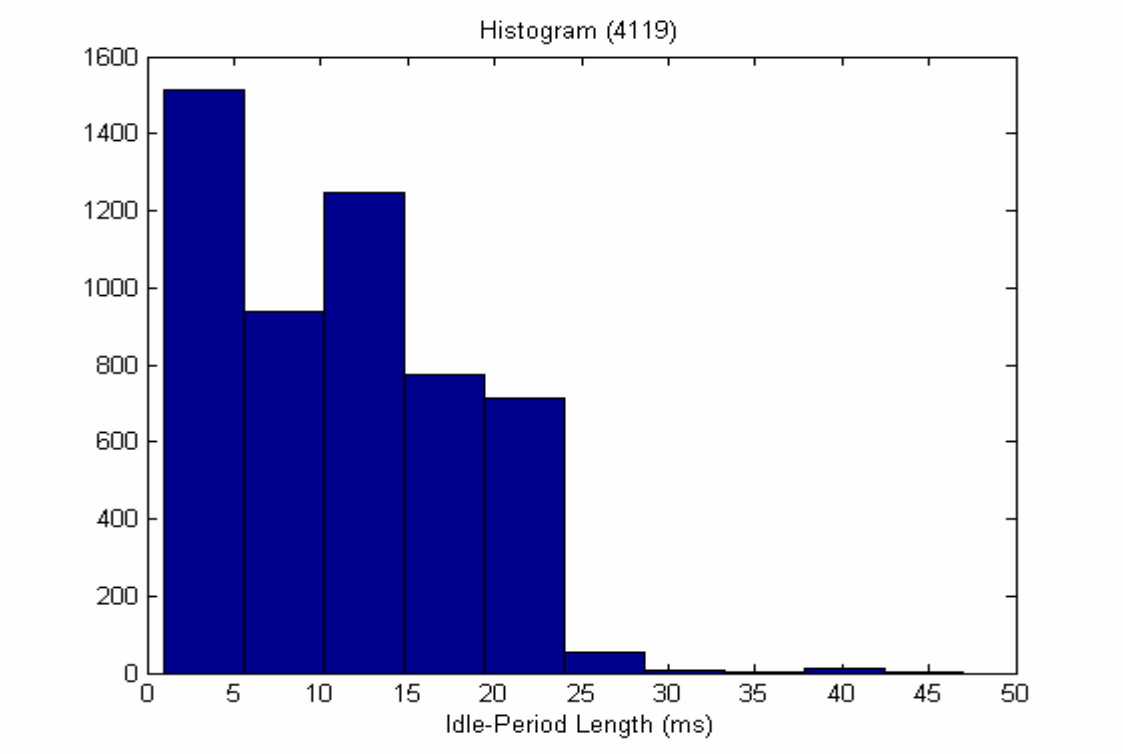


Figure C.14: Histogram H4119

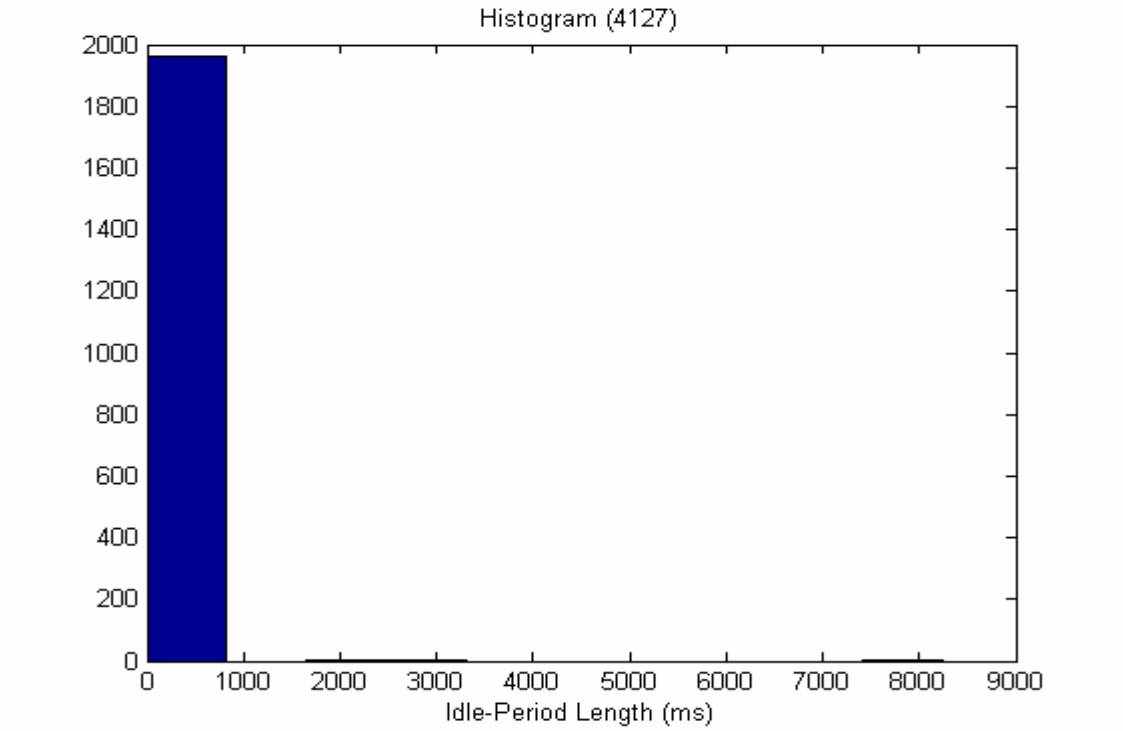


Figure C.15: Histogram H4127

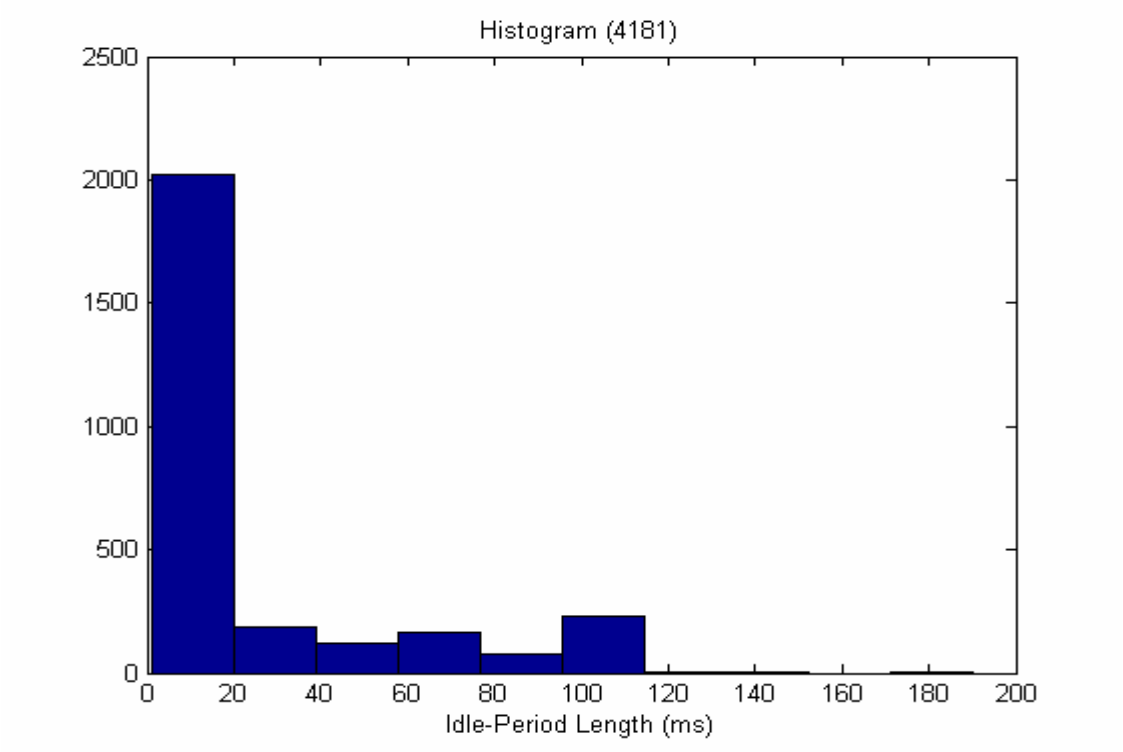
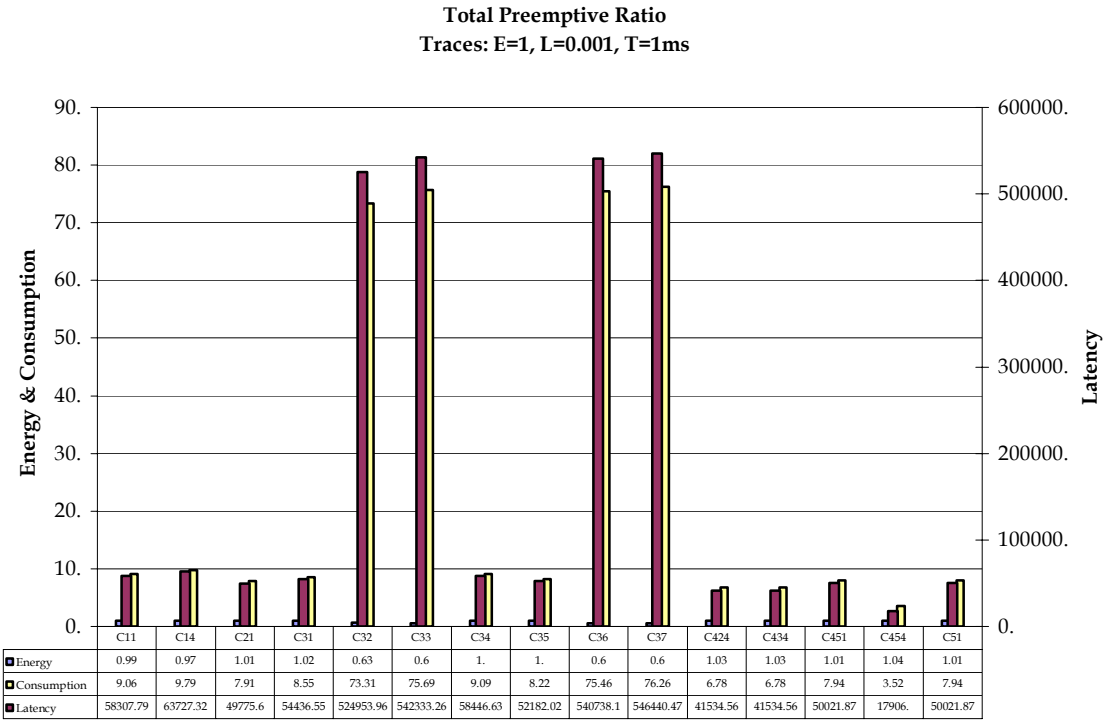


Figure C.16: Histogram H4181

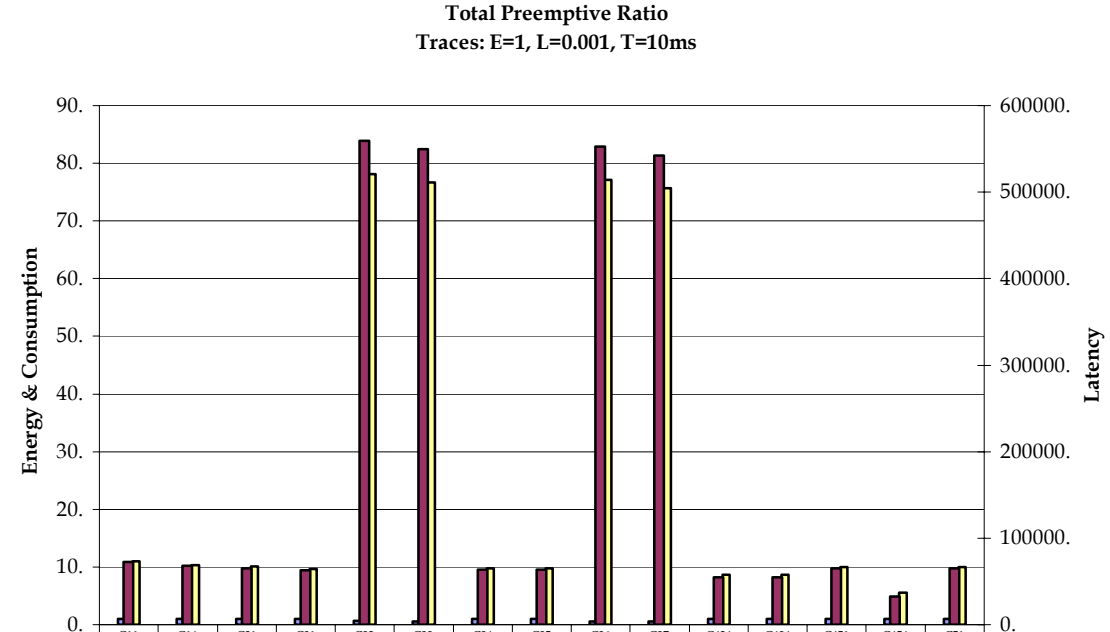
Appendix D

SLHA Results: Optimization of Energy and Latency



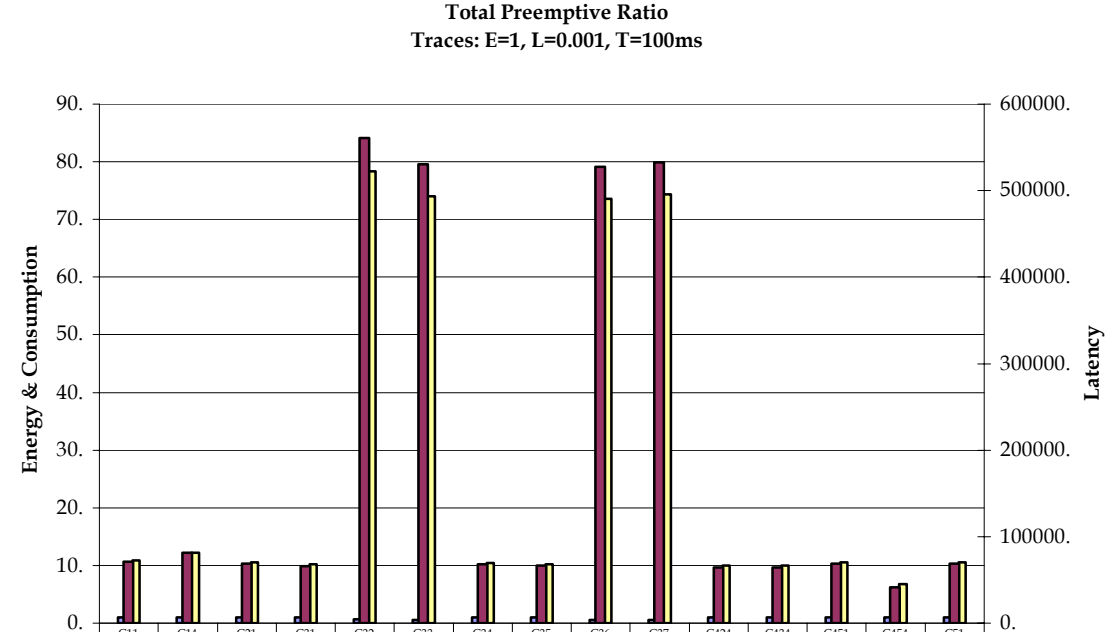
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.99	0.97	1.01	1.02	0.63	0.6	1.	1.	0.6	0.6	1.03	1.03	1.01	1.04	1.01
Consumption	9.06	9.79	7.91	8.55	73.31	75.69	9.09	8.22	75.46	76.26	6.78	6.78	7.94	3.52	7.94
Latency	58307.79	63727.32	49775.6	54436.55	524953.96	542333.26	58446.63	52182.02	540738.1	546440.47	41534.56	41534.56	50021.87	17906.	50021.87

Figure D.1: Total Output Ratio Histogram for T=1ms for "Preemptive" Method



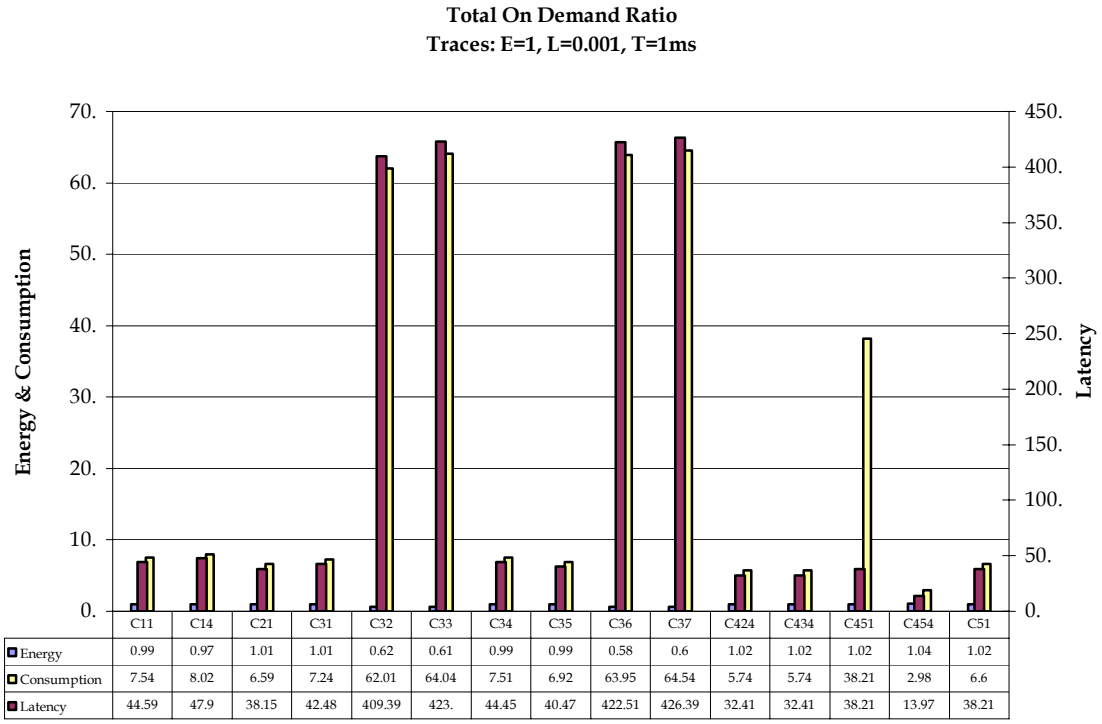
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.97	0.96	1.	1.01	0.64	0.59	1.	0.98	0.6	0.61	1.02	1.02	1.	1.04	1.
Consumption	11.	10.37	10.05	9.69	78.08	76.73	9.81	9.8	77.13	75.7	8.65	8.65	10.03	5.58	10.03
Latency	72427.27	67919.09	65332.85	62696.54	559288.57	549896.05	63608.57	63750.96	552786.34	542387.09	55111.06	55111.06	65234.73	32794.27	65234.73

Figure D.2: Total Output Ratio Histogram for T=10ms for "Preemptive" Method



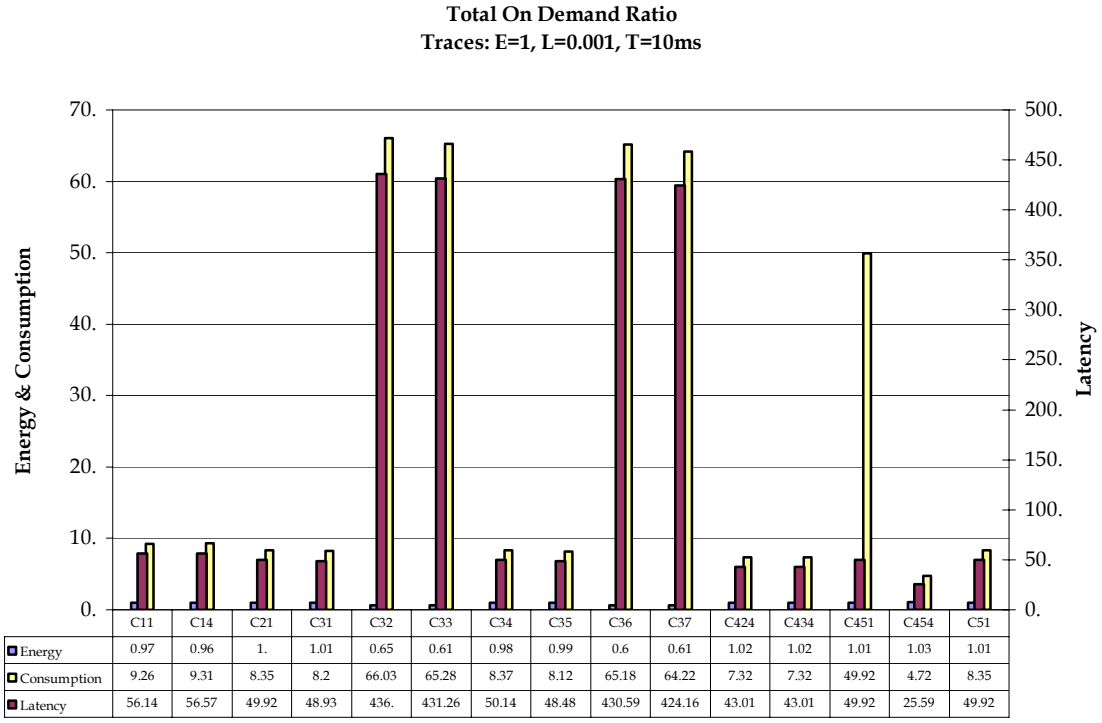
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.97	0.96	1.	1.01	0.64	0.59	0.99	0.99	0.6	0.61	1.01	1.01	1.	1.03	1.
Consumption	10.84	12.25	10.49	10.16	78.31	74.	10.38	10.19	73.59	74.36	9.95	9.95	10.5	6.74	10.5
Latency	71235.02	81567.97	68543.91	66086.	561010.03	530214.1	67862.53	66459.48	527161.06	532710.36	64556.81	64556.81	68652.34	41209.24	68652.34

Figure D.3: Total Output Ratio Histogram for T=100ms for "Preemptive" Method



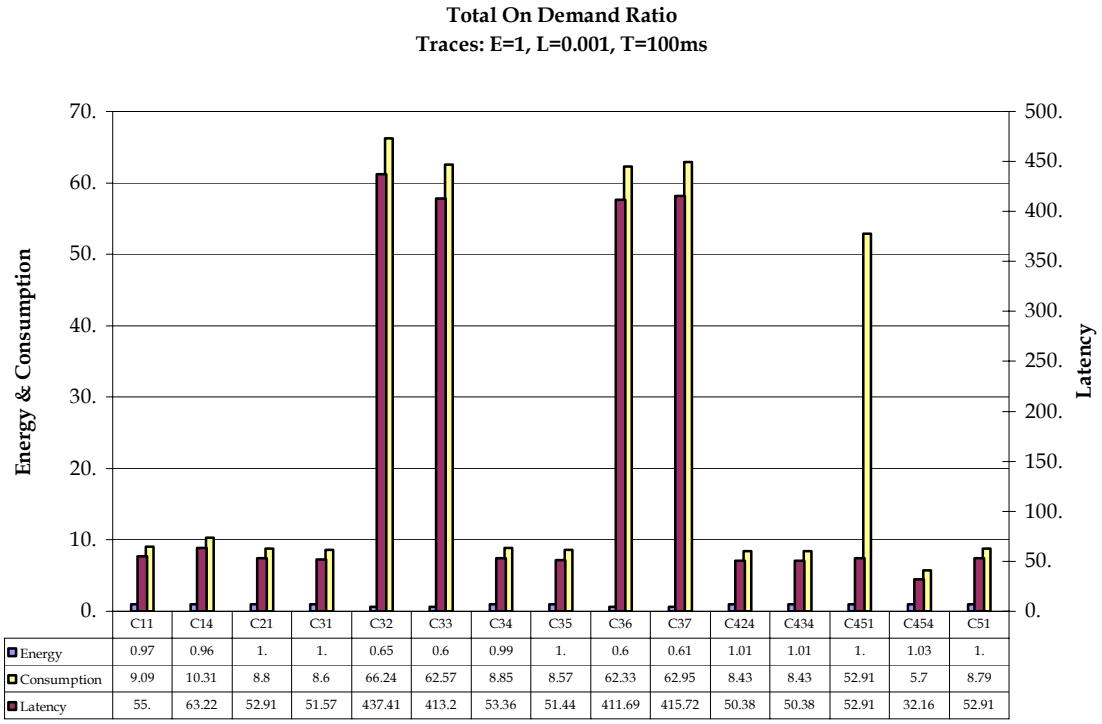
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.99	0.97	1.01	1.01	0.62	0.61	0.99	0.99	0.58	0.6	1.02	1.02	1.02	1.04	1.02
Consumption	7.54	8.02	6.59	7.24	62.01	64.04	7.51	6.92	63.95	64.54	5.74	5.74	38.21	2.98	6.6
Latency	44.59	47.9	38.15	42.48	409.39	423.	44.45	40.47	422.51	426.39	32.41	32.41	38.21	13.97	38.21

Figure D.4: Total Output Ratio Histogram for T=1ms for "On Demand" Method



	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.97	0.96	1.	1.01	0.65	0.61	0.98	0.99	0.6	0.61	1.02	1.02	1.01	1.03	1.01
Consumption	9.26	9.31	8.35	8.2	66.03	65.28	8.37	8.12	65.18	64.22	7.32	7.32	49.92	4.72	8.35
Latency	56.14	56.57	49.92	48.93	436.	431.26	50.14	48.48	430.59	424.16	43.01	43.01	49.92	25.59	49.92

Figure D.5: Total Output Ratio Histogram for T=10ms for "On Demand" Method

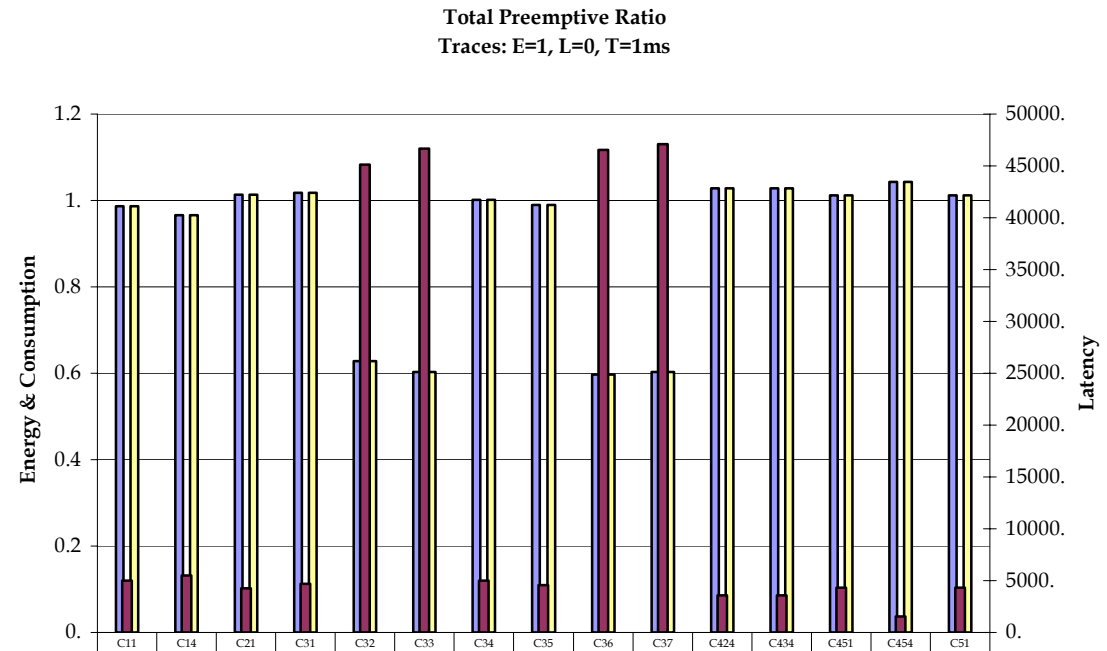


	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.97	0.96	1.	1.	0.65	0.6	0.99	1.	0.6	0.61	1.01	1.01	1.	1.03	1.
Consumption	9.09	10.31	8.8	8.6	66.24	62.57	8.85	8.57	62.33	62.95	8.43	8.43	52.91	5.7	8.79
Latency	55.	63.22	52.91	51.57	437.41	413.2	53.36	51.44	411.69	415.72	50.38	50.38	52.91	32.16	52.91

Figure D.6: Total Output Ratio Histogram for T=100ms for "On Demand" Method

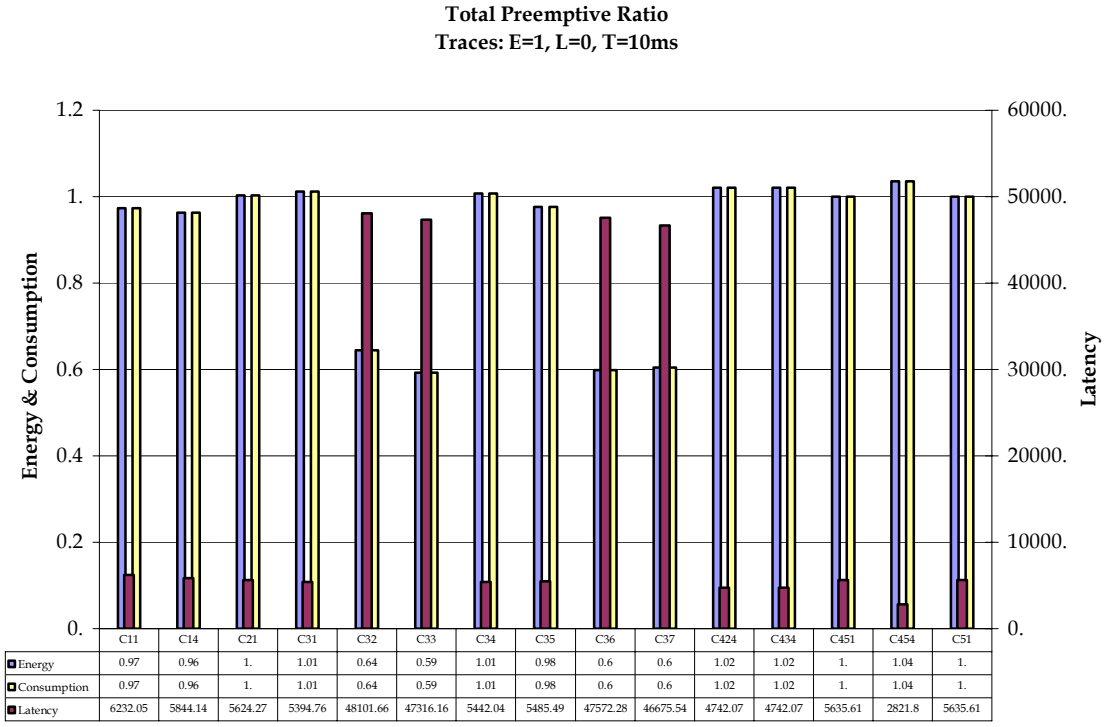
Appendix E

SLHA Output: Optimization of Energy



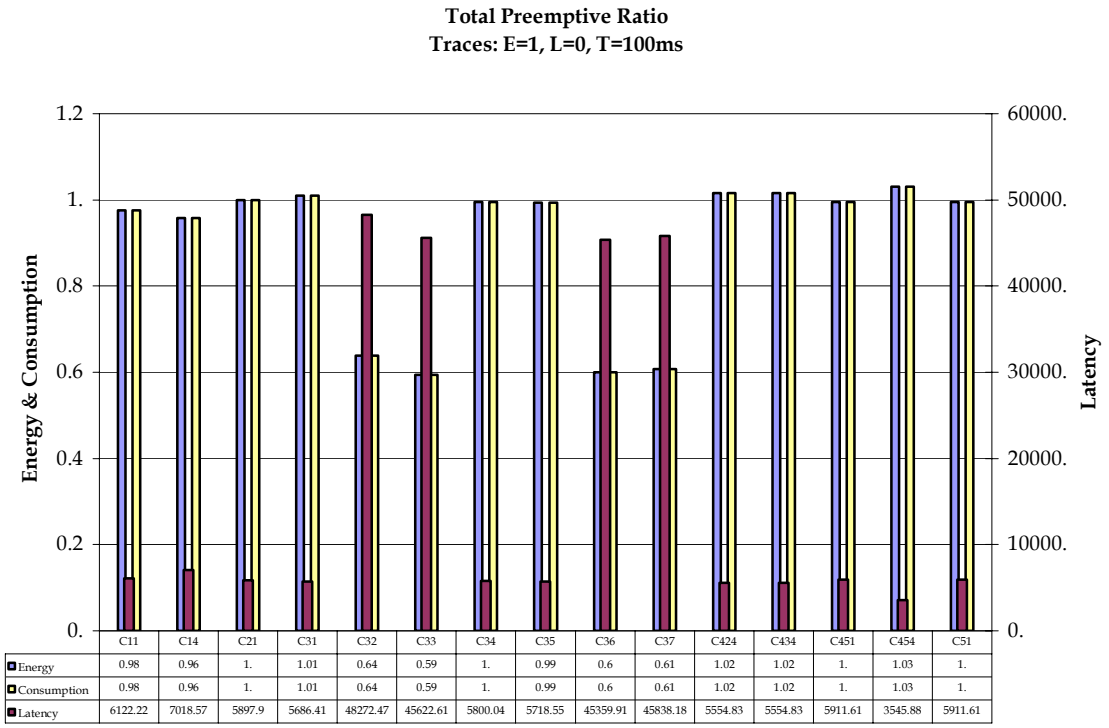
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.99	0.97	1.01	1.02	0.63	0.6	1.	0.99	0.6	0.6	1.03	1.03	1.01	1.04	1.01
Consumption	0.99	0.97	1.01	1.02	0.63	0.6	1.	0.99	0.6	0.6	1.03	1.03	1.01	1.04	1.01
Latency	5017.13	5483.46	4282.85	4684.03	45105.42	46665.41	4999.7	4562.07	46528.16	47068.99	3573.87	3573.87	4301.26	1540.73	4301.26

Figure E.1: Total Output Ratio Histogram for T=1ms for "Preemptive" Method



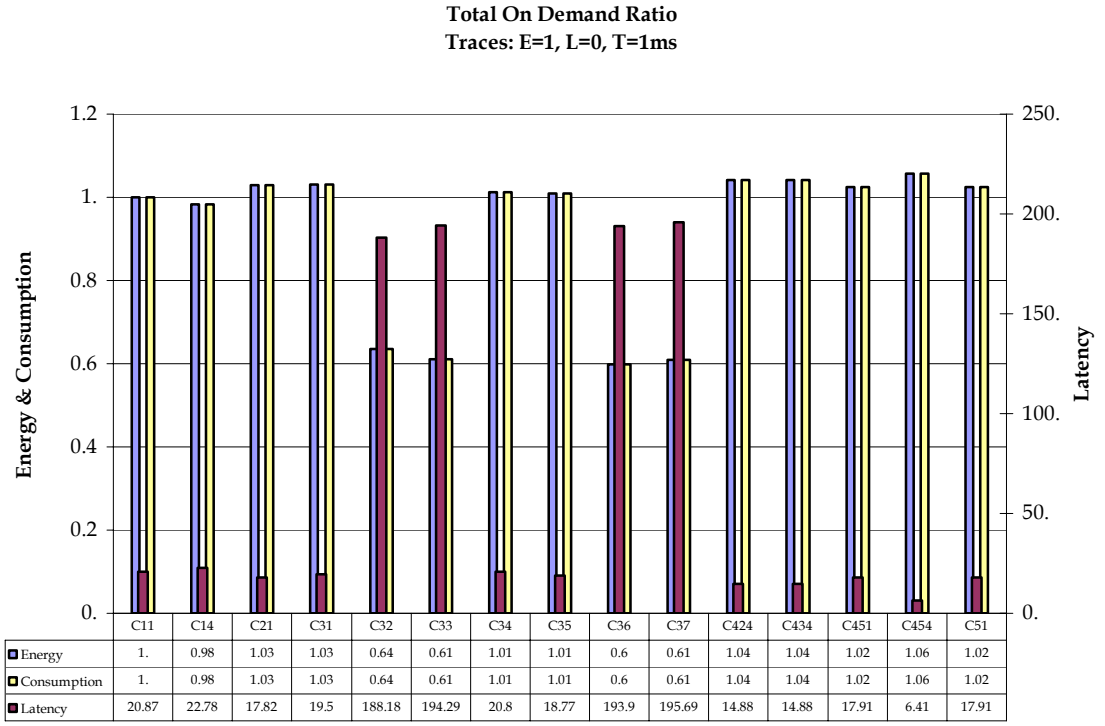
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.97	0.96	1.	1.01	0.64	0.59	1.01	0.98	0.6	0.6	1.02	1.02	1.	1.04	1.
Consumption	0.97	0.96	1.	1.01	0.64	0.59	1.01	0.98	0.6	0.6	1.02	1.02	1.	1.04	1.
Latency	6232.05	5844.14	5624.27	5394.76	48101.66	47316.16	5442.04	5485.49	47572.28	46675.54	4742.07	4742.07	5635.61	2821.8	5635.61

Figure E.2: Total Output Ratio Histogram for T=10ms for "Preemptive" Method



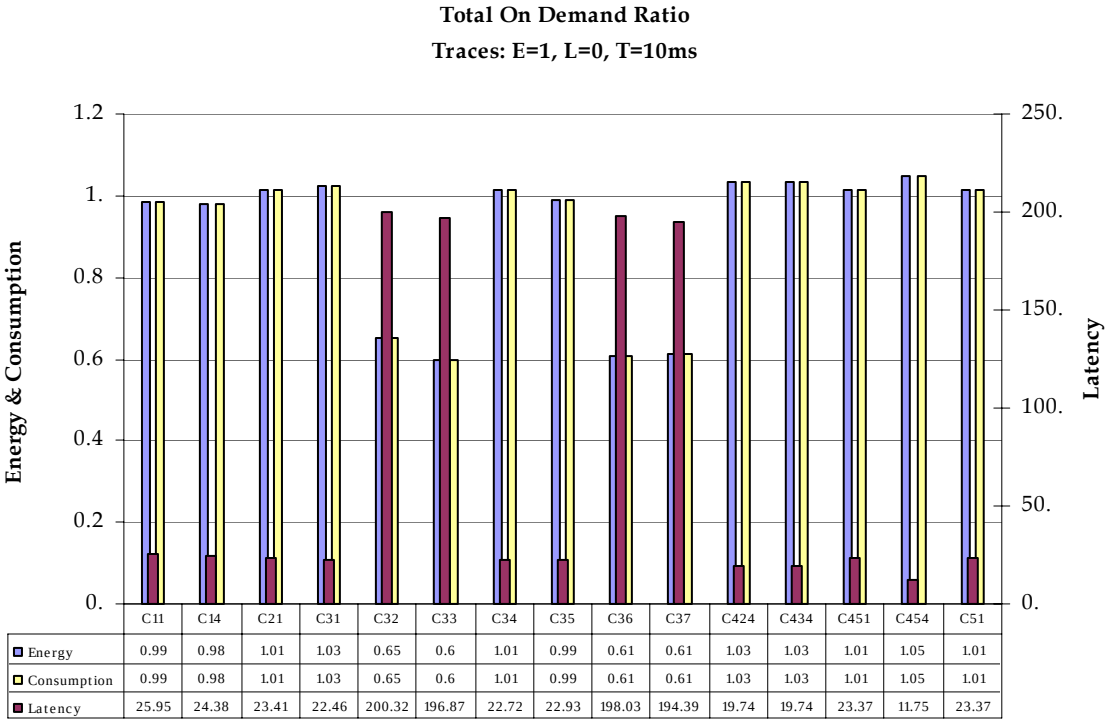
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.98	0.96	1.	1.01	0.64	0.59	1.	0.99	0.6	0.61	1.02	1.02	1.	1.03	1.
Consumption	0.98	0.96	1.	1.01	0.64	0.59	1.	0.99	0.6	0.61	1.02	1.02	1.	1.03	1.
Latency	6122.22	7018.57	5897.9	5686.41	48272.47	45622.61	5800.04	5718.55	45359.91	45838.18	5554.83	5554.83	5911.61	3545.88	5911.61

Figure E.3: Total Output Ratio Histogram for T=100ms for "Preemptive" Method



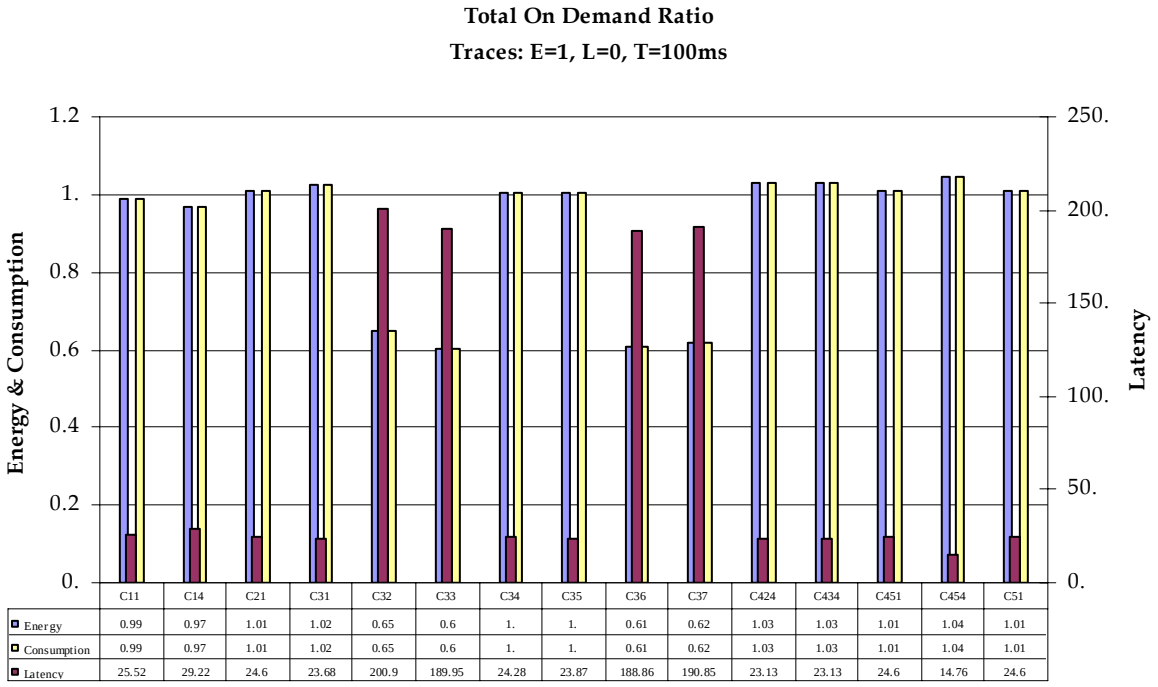
	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	1.	0.98	1.03	1.03	0.64	0.61	1.01	1.01	0.6	0.61	1.04	1.04	1.02	1.06	1.02
Consumption	1.	0.98	1.03	1.03	0.64	0.61	1.01	1.01	0.6	0.61	1.04	1.04	1.02	1.06	1.02
Latency	20.87	22.78	17.82	19.5	188.18	194.29	20.8	18.77	193.9	195.69	14.88	14.88	17.91	6.41	17.91

Figure E.4: Total Output Ratio Histogram for T=1ms for "On Demand" Method



	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.99	0.98	1.01	1.03	0.65	0.6	1.01	0.99	0.61	0.61	1.03	1.03	1.01	1.05	1.01
Consumption	0.99	0.98	1.01	1.03	0.65	0.6	1.01	0.99	0.61	0.61	1.03	1.03	1.01	1.05	1.01
Latency	25.95	24.38	23.41	22.46	200.32	196.87	22.72	22.93	198.03	194.39	19.74	19.74	23.37	11.75	23.37

Figure E.5: Total Output Ratio Histogram for T=10ms for "On Demand" Method

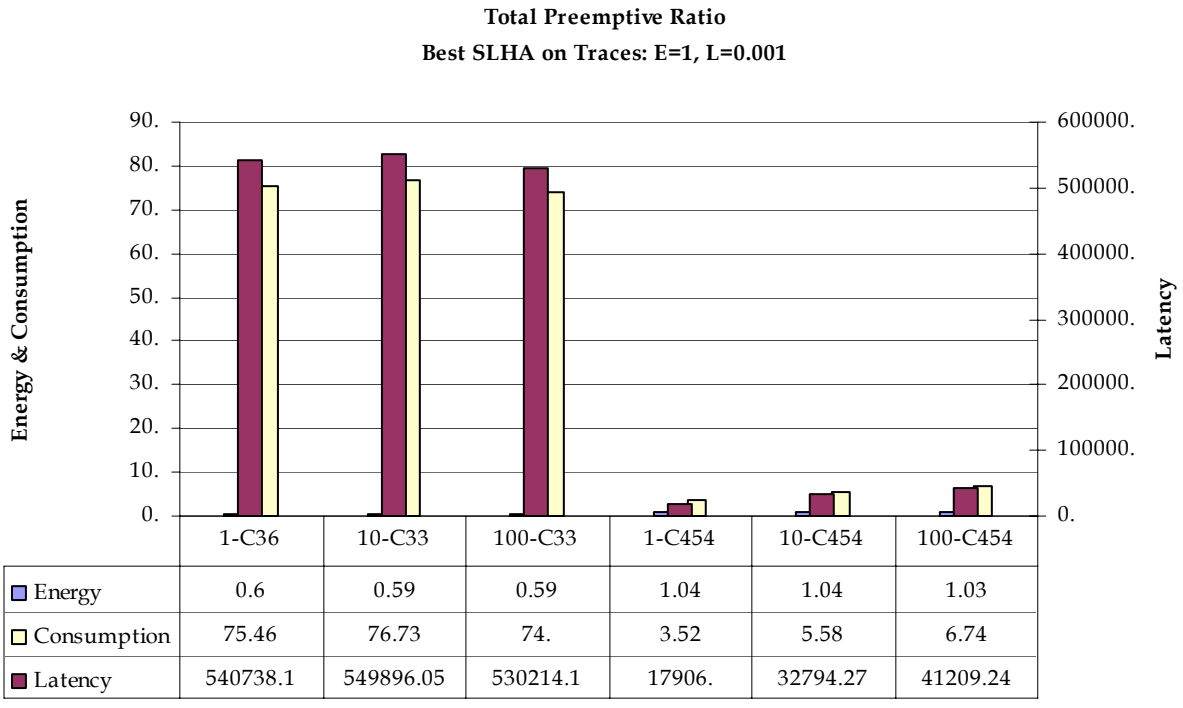


	C11	C14	C21	C31	C32	C33	C34	C35	C36	C37	C424	C434	C451	C454	C51
Energy	0.99	0.97	1.01	1.02	0.65	0.6	1.	1.	0.61	0.62	1.03	1.03	1.01	1.04	1.01
Consumption	0.99	0.97	1.01	1.02	0.65	0.6	1.	1.	0.61	0.62	1.03	1.03	1.01	1.04	1.01
Latency	25.52	29.22	24.6	23.68	200.9	189.95	24.28	23.87	188.86	190.85	23.13	23.13	24.6	14.76	24.6

Figure E.6: Total Output Ratio Histogram for T=100ms for "On Demand" Method

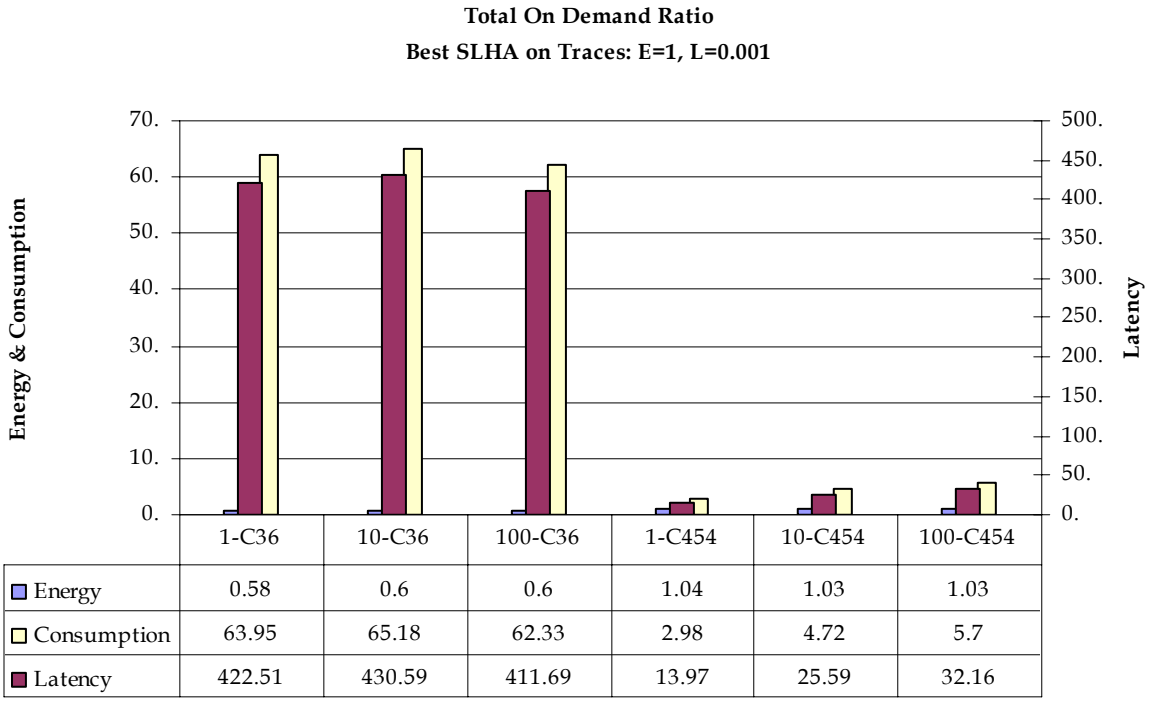
Appendix F

Best SLHA Results



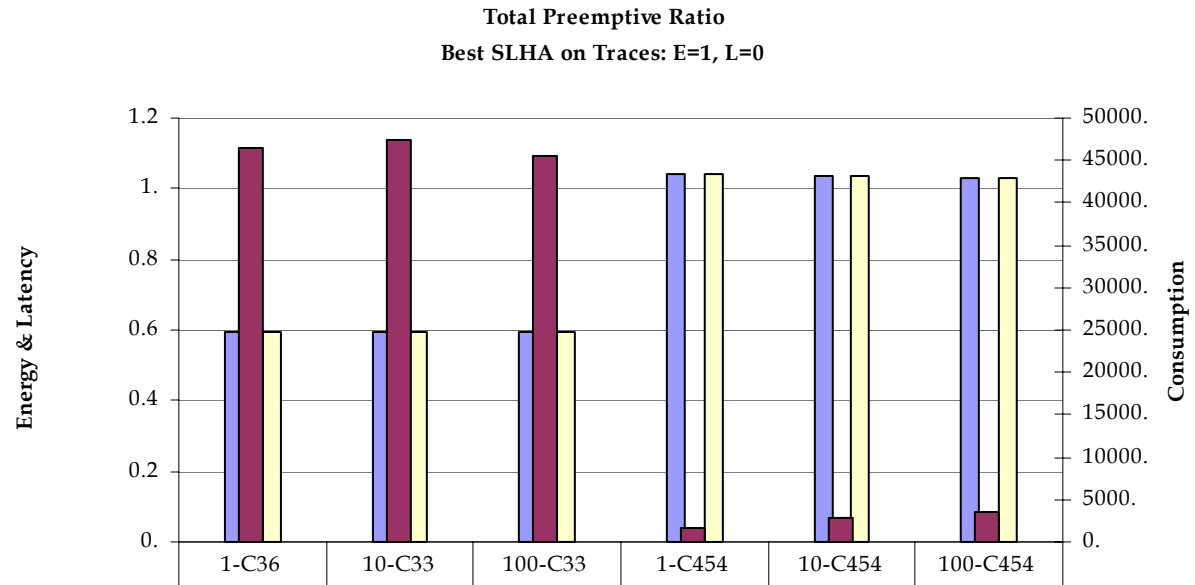
	1-C36	10-C33	100-C33	1-C454	10-C454	100-C454
Energy	0.6	0.59	0.59	1.04	1.04	1.03
Consumption	75.46	76.73	74.	3.52	5.58	6.74
Latency	540738.1	549896.05	530214.1	17906.	32794.27	41209.24

Figure F.1: Total Cost Ratio Histogram for "Preemptive" method with E=1 L=0.001



	1-C36	10-C36	100-C36	1-C454	10-C454	100-C454
Energy	0.58	0.6	0.6	1.04	1.03	1.03
Consumption	63.95	65.18	62.33	2.98	4.72	5.7
Latency	422.51	430.59	411.69	13.97	25.59	32.16

Figure F.2: Total Cost Ratio Histogram for "On Demand" method with E=1 L=0.001



	1-C36	10-C33	100-C33	1-C454	10-C454	100-C454
Energy	0.6	0.59	0.59	1.04	1.04	1.03
Consumption	0.6	0.59	0.59	1.04	1.04	1.03
Latency	46528.16	47316.16	45622.61	1540.73	2821.8	3545.88

Figure F.3: Total Cost Ratio Histogram for "Preemptive" method with E=1 L=0

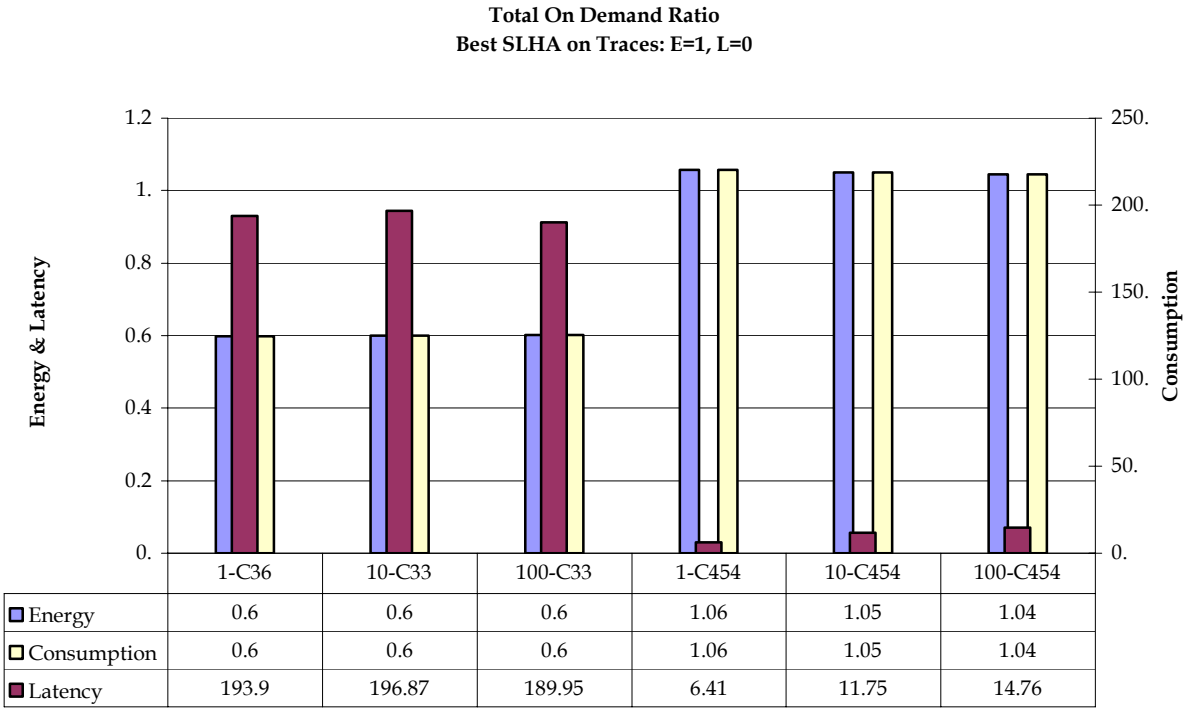
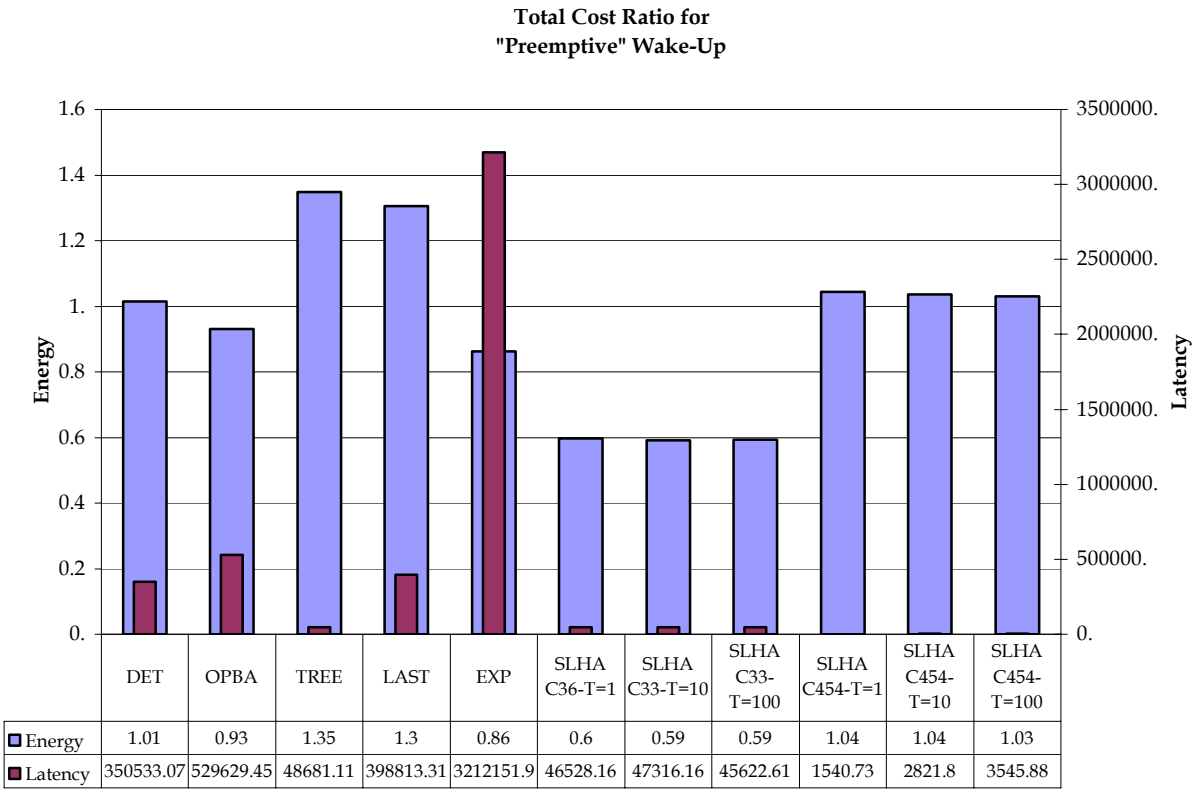


Figure F.4: Total Cost Ratio Histogram for "On Demand" method with E=1 L=0

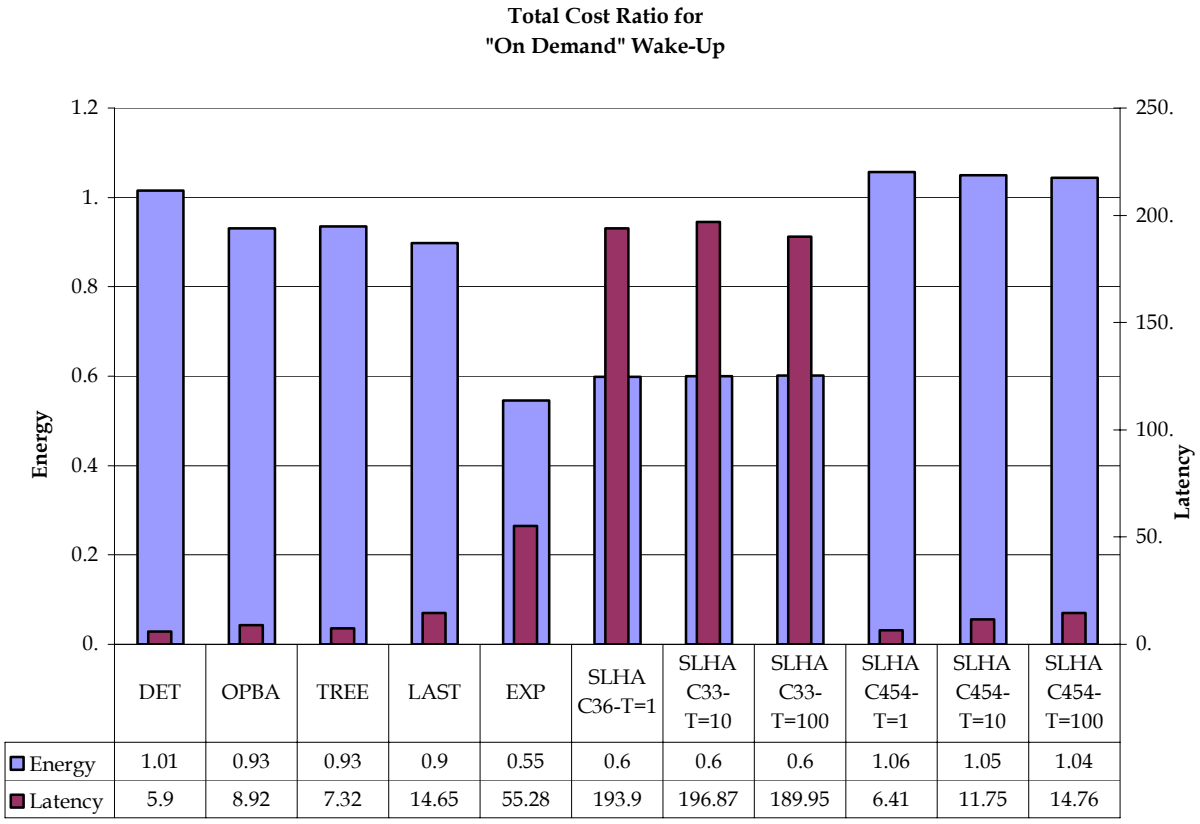
Appendix G

DPM Results



	DET	OPBA	TREE	LAST	EXP	SLHA C36-T=1	SLHA C33-T=10	SLHA C33-T=100	SLHA C454-T=1	SLHA C454-T=10	SLHA C454-T=100
Energy	1.01	0.93	1.35	1.3	0.86	0.6	0.59	0.59	1.04	1.04	1.03
Latency	350533.07	529629.45	48681.11	398813.31	3212151.9	46528.16	47316.16	45622.61	1540.73	2821.8	3545.88

Figure G.1: Total Cost Ratio Histogram for "Preemptive" method



	DET	OPBA	TREE	LAST	EXP	SLHA C36-T=1	SLHA C33-T=10	SLHA C33-T=100	SLHA C454-T=1	SLHA C454-T=10	SLHA C454-T=100
Energy	1.01	0.93	0.93	0.9	0.55	0.6	0.6	0.6	1.06	1.05	1.04
Latency	5.9	8.92	7.32	14.65	55.28	193.9	196.87	189.95	6.41	11.75	14.76

Figure G.2: Total Cost Ratio Histogram for "On Demand" method

Vita

Teodora Erbes was born in Belgrade, Yugoslavia, on the cold night of January 9th, 1981. She spent her early childhood between Belgrade and Njivice, before moving to the Côte d'Azur. After completing an international education at the International Center of Valbonne Sophia-Antipolis in 1998, she sought an American curriculum at the Euro-American Institute of Technology. She finally transferred to Virginia Polytechnic Institute and State University, where she obtained her Bachelor's Degree in Computer Engineering in 2002. Teodora intends to pursue her research interests in dynamic systems and artificial intelligence.