Newcastle University

School of Computing

Modelling Energy Efficiency and Performance Trade-offs

Ali Alssaiari

Submitted for the degree of Doctor of Philosophy in the School of Computing, Newcastle University

November 2021

Dedication

I would like to dedicate this thesis to my loving parents and my family (my wife Eman), my son (Alhassan) and my daughters (Enas, Zainab and Kindah).

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification at Newcastle University, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Ali Abdullah Y Alssaiari

November 2021

Acknowledgements

First of all, praises and thanks be to Allah, who has granted me countless blessings, patience and perseverance during this research project, and indeed, throughout all my life. I would also like to express my gratitude and thanks to my supervisor Dr Nigel Thomas, for his invaluable guidance, enormous support and constant encouragement throughout my PhD study. Without your great continuing support and encouragement, this research would never be a reality. I also would like to thank my colleagues at the School of Computing at Newcastle University for their inspiration and support over the years. Special thanks, to Osamah Alrajeh, Said Kamil, Talal Alharbi and Amjad Aldewish for their support.

I wish to also acknowledge the support of my friends Idris Musa, Abdullah Faqihi, Ahmmad Alhatrushi and the various people who has touch my life during my time of study. In addition, I would like to give my special thanks to all the children in Saudi School, they made Saturday a valuable day in my life during my volunteering in the school since 2016.

Finally, and most importantly, I wish to thank my mother (Shethra Alsaiar), my father, my brothers and my sister. I also wish to express my sincere thanks to my wife (Eman), my son (Alhassan) and my daughters (Enas, Zainab and Kindah) for their great patience, sacrifices, encouragement, infinite love and enormous support. They all were the real motivation for facing all the difficulties during my study.

Abstract

Power and energy consumption in data centres is a huge concern for data centre providers. As a result, this work considers the modelling and analysis of policy and scheduling schemes using Markovian processing algebra known as PEPA. The first emphasis was on modelling an energy policy in PEPA that dynamically controls the powering servers ON or OFF. The focus is to identify and reflect the trade-off between saving energy (by powering down servers) and performance cost. While powering down servers saves energy, it could increase the performance cost. The research analyses the effect of the policy on energy consumption and performance cost, with different combinations of dynamic and static servers used in the policy against different scenarios, including changes in job arrival rate, job arrival duration and the time needed by servers to be powered On and start process jobs. The result gave interesting outcomes because every scenario is unique, and therefore, no server combinations were found to give low energy and high performance in all situations.

The second focus was to consider the impact of scheduler's choice on performance and energy under unknown service demands. Three algorithms were looked at: task assignment based on guessing size (TAGS), the shortest queue strategy and random allocation. These policies were modelled using PEPA to derive numerical solutions in a two servers system. The performance was analysed considering throughput, average response time and servers' utilisation. At the same time, the energy consumption was in terms of total energy consumption and energy consumption per job. The intention was to analyse the performance and energy consumption in a homogeneous and heterogeneous environment, and the environment was assumed to be homogeneous in the beginning. However, the service distribution was considered either a negative exponential (hence relatively low variance) or a two-phase hyper-exponential (relatively high variance) in each policy. In all cases, the arrival process has been assumed to be a Poisson stream, and the maximum queue lengths are finite (maximum size is 10 jobs). The performance results showed that TAGS performs worse under exponential distribution and the best under two-phase hyper-exponential. TAGS produce higher throughput and lower job loss when service demand has an H2 distribution. Our results show that servers running under TAGS consume more energy than other policies regarding total energy consumption and energy per job under exponential distribution. In contrast, TAGS consumes less energy per job than the random allocation when the arrival rate is high, and the job size is variable (two-phase hyper-exponential).

In a heterogeneous environment and based on our results on the homogeneous environment, the performance metrics and energy consumption was analysed only under two-phase hyper-exponential. TAGS works well in all server configurations and achieves greater throughput than the shortest queue or weighted random, even when the second server's speed was reduced by 40% of the first server's in TAGS. TAGS outperforms both the shortest queue and weighted random, whether their second server is faster or slower than the TAGS second server. The system's heterogeneity did not significantly improve or decrease TAGS throughput results. Whether the second server is faster or slower, even when the arrival rate is less than 75% of the system capacity, it approximately showed no effect. On the other hand, heterogeneity of the system has a notable effect on the throughput of the shortest queue and weighted random. The decrease or increase in throughput follows the trend of the second server performance capability. In terms of total energy consumption, for all scheduling schemes, when the second server is slower than the first server, the energy consumption is the highest among all scenarios for each arrival rate. TAGS was the worst and consumed higher energy than both the shortest queue strategy and weighted random allocation. However, in terms of energy per job, when servers are identical, or server2 is faster, it was observed that the shortest queue is the optimal strategy as long as the incoming jobs rate does not exceed 70% of the system capacity (arrival rate <15). Furthermore, the TAGS was the best strategy when the incoming task rate exceeds 70% of the system capacity. So, as more jobs are produced, the energy per job decreases eventually. Choosing the energy policy or scheduling algorithm will impact energy consumption and performance either negatively or positively.

Contents

List of Figures

List of Tables

Nomenclature

List of Acronyms

TOC:	Total Ownership Cost
PEPA:	Performance Evaluation Process Algebra
DPM:	Dynamic Power Management
DVFS:	Dynamic Voltage and Frequency Scaling
TOC:	Total Ownership Cost
DPM:	Dynamic Power Management
DFS:	Dynamic Frequency Scaling
CMOS:	Complementary Metal Oxide Semiconductor
CPU:	Central Processing Unit
FCFS:	First-Come-First-Served
CTMC:	Continuous Time Markov Chain
PUE:	Power Usage Effectiveness
DCiE:	Data Centre Infrastructure Effciency
ODEs:	Ordinary Differential Equations
TAGS:	Task Assignment Based on Guessing Size
CDN:	Content Delivery Network
MTBF:	Mean Time Between Failures
MTTR:	Mean Time To Repair
SPN:	Stochastic Petri Net
GSPN:	Generalized Stochastic Petri Net
SLA:	Service Level Agreement
SLO:	Service Level Objective
VMs:	Virtual Machines
AQTMCC:	Applying Queue Theory for Modelling of Cloud Computing
ERP:	Energy-Response Time Product
SPA:	Stochastic Process Algebra
ACPI:	Advanced Configuration and Power Interface

Acronym

1

Introduction

Contents

1.1 Introduction

The rapid growth in new technological breakthroughs and manufacturers' enormous production processes provide cheap and useful user-friendly products that are more accessible to many people, regardless of their economic situation. This growth has led to increased worldwide consumer usage of evolving technologies. These technologies mostly rely on electricity. The global success of technological development and usage has resulted in a dramatic increase in power demands and consumption worldwide. Studies show that global energy demands have increased by 100% between 1990 and 2014. In 1990, the power consumption was 10k TWh, which increased to 20k TWh worldwide in 2014 [13]. The U.S. Energy Information Administration (EIA) estimates that future demand and consumption of energy will increase by 2.2% per year, which would presumably be about 40k TWh in 2040 [9]

The fast-paced growth of global power consumption raises concerns about energy-related carbon dioxide emissions as many countries still rely on fossil fuels (gas, oil, and coal) to produce electricity. In 2010, the carbon dioxide emissions were estimated to be around 31.2 billion metric tons, which is anticipated to increase by 16%, equivalent to 36.4 billion metric tons, in 2020 [9]. Many countries have introduced diverse environmental requirements to reduce CO_2 emissions. For example, the British Government aims to reduce greenhouse gas emissions by at least 80% in 2050 from a 1990 baseline, according to the Climate Change Act 2008 requirements [14].

1.2 Motivation

The cost of energy is one of the many challenges facing large-scale computing. According to [42], data centre owners expect to spend more capital on energy than their IT infrastructure, which currently contributes more to the Total Ownership Cost (TOC). The Environmental Protection Agency (EPA) has issued a report to the U.S. Congress regarding the energy efficiency of servers and data centres. The report highlighted several important points related to the energy consumption of data centres. According to the report, data centres' electricity demands grew 100% between 2000 and 2006. Data centres in the U.S. consumed 61 billion kWh in 2006, representing 1.5% of total electrical consumption in the country [4].

Reducing energy consumption in data centres cannot be achieved without a deep understanding of their needs and energy consumption mechanisms, which necessitates understanding how data centres operate and the amount of energy each component consumes. In data centres, one of energy management's main objectives is reducing energy consumption by servers. Researchers introduce many techniques and policies to achieve this goal. One of such is the dynamic server allocations, where servers are dynamically switched on or off according to a specific policy. Moreover, the Dynamic Voltage and Frequency Scaling (DVFS) is another approach used to control servers energy consumption. The impact on performance is the primary concern associated with these energy reduction techniques. In contrast, the improvement of performance techniques directly affects increased energy consumption. For example, a technique for improving system throughput or average response time under variable service demand might increase energy consumption.

1.3 Problem Statement

Selecting an energy policy or a job scheduler based on energy efficiency and performance is vital to obtaining maximum performance with minimum energy consumption. However, the selection process needs an understanding of each policy and scheduler impact on energy and performance. In this thesis, we initially consider evaluating the high/low heuristic policy introduced by Slegers *et al.* [52]. The policy dynamically switches on or off servers to reduce energy consumption. Moreover, we evaluate and study the Task Assignment Based on Guessing Size (TAGS) algorithm, introduced by Harchol-Balter [23]. TAGS purpose is to improve the performance where the service demand is not known. These systems will be modelled by Performance Evaluation Process Algebra (PEPA) to derive performance metrics and evaluate energy consumption. As a result, this research investigates the impact of dynamically switching off servers to minimise power consumption on performance. Also, this study investigates the high variability in service demand and the benefits of redirecting large jobs to specific servers and the effect these decisions have on energy and performance.

The research questions become:

1. Can dynamically shut down a group of servers in the data centre minimise energy consumption without degrading the overall performance?
2. Can the TAGS mechanism of forwarding jobs to specific servers when job service demand is unknown improves performance without increasing energy consumption?

1.4 Aims and objectives of the research

This research aims to evaluate energy consumption and performance trade-off. To achieve this aim, the following objectives will be undertaken:

1. To identify the current energy-efficient approaches and techniques.
2. To evaluate the energy consumption and performance in the high/low energy policy that has been introduced to reduce energy consumption. **To fulfil the requirement to answer question 1.**
3. To evaluate and compare the energy consumption of some current algorithms that increase performance but do not consider energy consumption. **To fulfil the requirement to answer question 2.**
   (a) To investigate and compare the energy consumption and performance of TAGS, the shortest queue and random allocation algorithms in a homogeneous environment. There are many scheduling algorithms we can use to allocate work. TAGS is the focus of the study as an unusual type of scheduling algorithm which is suitable for variable demand. We study the performance and energy consumption of TAGS to compare that to other well-known algorithms. We chose the shortest queue as provably the best strategy for exponential traffic, while random allocation is chosen because it is the simplest job allocation strategy.
   (b) To construct PEPA Models for TAGS, the shortest queue and random allocation algorithms in a heterogeneous environment.

(c) To investigate and compare the energy consumption and performance of TAGS, the shortest queue and random allocation algorithms in a heterogeneous environment.

4. To evaluate and compare existing approaches by using performance evaluation processing algebra (PEPA).

1.5 Contributions

This section highlights the main contributions achieved throughout the work presented in this thesis. These contributions are detailed as follows:

1. Contributions from Chapter 3 to answer question 1:
    (a) Developed and analysed a PEPA model for the high/low policy. Which dynamically controls switching servers on or off according to two service demand periods. The high period represents the excessive demand in a period, while the low period represents a period where the service demand is reduced.
    (b) Presented an investigation of the energy and performance trade-off of the high/low policy by different server combinations in four different scenarios. These scenarios include changing the arrival rate, changes in period duration, changing the switching time needed by servers to switch from one state to another and changing the cost difference between holding the job in the queue and the cost of saving energy.
2. Contributions from Chapters 4, 5 and 6, respectively, to answer question 2:
    (a) Developed an energy model based on P-state value.
    (b) Presented an energy and performance evaluation of TAGS, the shortest queue and random allocation algorithms in a homogeneous environment. The performance metrics considered are throughput, average response time, job loss and servers utilisation. Simultaneously, the energy consumption evaluation observed total energy consumption by servers and energy per job.
    (c) Developed PEPA models for TAGS, the shortest queue and random allocation algorithms, in a heterogeneous environment.
    (d) Evaluated energy and performance for TAGS, the shortest queue and random algorithms in a heterogeneous environment using the developed PEPA models.

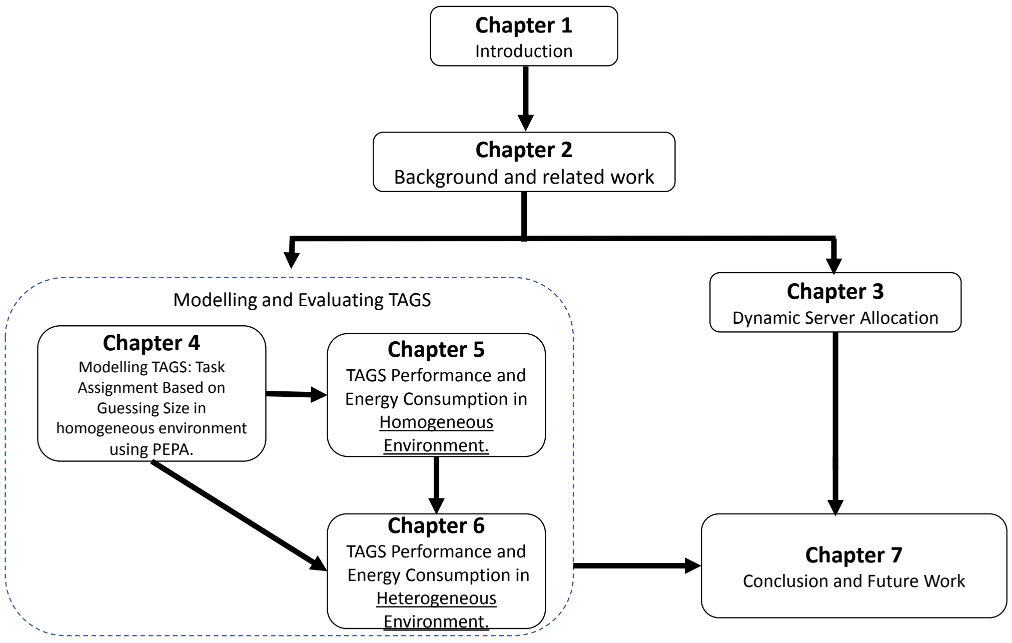

Chapter 1
Introduction
Chapter 2
Background and related work
Modelling and Evaluating TAGS
Chapter 3
Dynamic Server Allocation
Chapter 4
Modelling TAGS: Task Assignment Based on Guessing Size in homogeneous environment using PEPA.
Chapter 5
TAGS Performance and Energy Consumption in Homogeneous Environment.
Chapter 6
TAGS Performance and Energy Consumption in Heterogeneous Environment.
Chapter 7
Conclusion and Future Work

Figure 1.1: Thesis structure

1.6 Thesis structure

This thesis consists of 7 chapters that cover two main concepts (i) dynamic server allocation energy policies in Chapter 3 and (ii) the TAGS scheduling algorithm in Chapters 4, 5 and 6. The aim is to evaluate each concept in terms of performance and energy consumption trade-off. It is worth noting that, as there are two different concepts, the experiment requirements and setups for each concept are different. So, the dynamic server allocation experiment in Chapter 3 is different from the TAGS experiments in Chapters 4, 5 and 6. Figure 1.1 presents an overall vision of the research that shows how the various chapters are connected.

This thesis organised as follows:

Chapter 2 presents an overview of the related background topics and relevant literature to this thesis. In addition, the PEPA modelling language, which we adopted as the modelling method throughout this thesis, is introduced.

Chapter 3 investigates the performance and energy cost of dynamically switching on or off servers. In this chapter, we modelled in PEPA a high/low policy. In addition, we evaluate the energy and performance cost under four scenarios.

Chapter 4 presents TAGS, the shortest queue and random allocation algorithms, and their PEPA models specifications in a homogeneous environment. The system is modelled under two service demand types: (i) exponential service demand and (ii) a two-phase hyper-exponential service demand. Moreover, it describes the timeout optimisation mechanism for TAGS.

Chapter 5 presents performance and energy consumption analysis of TAGS, the shortest queue and random allocation in a homogeneous environment. The results showed TAGS under two-phase hyper-exponential service demand consumes less energy per job than the shortest queue and random allocation.

Chapter 6 presents PEPA models and analysis of performance and energy for TAGS, the shortest queue and random allocation in a heterogeneous environment. In this chapter, we adjust the PEPA models to adopt the heterogeneity of the system.

Chapter 7 presents the overall conclusions of the thesis and propose some areas of further investigation.

1.7 Related Publications

The following papers represent the thesis published contributions:

1. Ali Alssaiari and Nigel Thomas. ”Performance modelling of dynamic server allocation for energy efficiency using PEPA.” 32nd UK Performance Engineering Workshop, University of Bradford. 2016.

   In this paper, we considered modelling the high/low policy in PEPA to control powering on or off servers dynamically. The main focus was identifying and reflecting on the trade-off between saving energy (by powering down servers) and performance cost. While powering down servers save energy, it could increase the performance cost. The experiment analysed the effect of the policy on energy consumption and performance cost. Multiple combinations of dynamic and static servers in different scenarios were used in the experiment. These scenarios include changes in job arrival rate, job arrival duration, the time needed by servers to power on fully and serve jobs. The results give an interesting outcome because every scenario is unique, and therefore, no server combination gives low energy and high performance in all scenarios. This paper forms Chapter 3.

2. Ali Alssaiari Ray Adderley Jm Gining, and Nigel Thomas. ”Modelling energy efficient server management policies in PEPA.” Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion. 2017.

3. Ali Alssaiari and Nigel Thomas. ”Performance and Energy Consumption of TAGS in a Heterogeneous Environment Under Unknown Service Demand.” 35th UK Performance Engineering Workshop 16 December 2019. .

   In this paper, TAGS a job allocation algorithm is modelled in PEPA. The working environment is assumed to be heterogeneous, and the job size distribution is assumed to be H2 hyper exponential. Furthermore, the queues are bonded. We use a two nodes system with exponentially distributed incoming tasks. We analysed the performance metrics and energy consumption under different arrival rates. The results show that TAGS can perform well for a different range of performance metrics. In contrast, TAGS increases total energy consumption. Finally, we calculate the energy per job to analyse the benefit of using TAGS in a heterogeneous environment. This paper and paper 5 forms Chapter 6.

4. Ali Alssaiari and Nigel Thomas. ”Energy Consumption by Servers under Unknown Service Demand.” Electronic Notes in Theoretical Computer Science 353 (2020): 21-38.

   In this paper, we evaluate energy consumption under unknown service demands using three strategies: TAGS, the shortest queue strategy and the random allocation in a homogeneous environment. We modelled these policies using performance evaluation processing algebra (PEPA) to derive numerical solutions. Our results show that servers running under TAGS consumes more energy than other policies in terms of total energy consumption. In contrast, TAGS consumes less energy than the random allocation in terms of energy per job when the arrival rate is high, and the job size is variable. This paper forms the basis of Chapters 4 and 5.

5. Ali Alssaiari and Nigel Thomas. ”Energy Consumption of TAGS in a Heterogeneous Environment under Unknown Service Demand”. Sustainable Computing: Informatics and Systems, Volume 30, 2021.

   In this paper, we modelled the TAGS job allocation algorithm using PEPA. The working environment is assumed to be heterogeneous, and the job size distribution is assumed to be a two-phase hyper-exponential. Furthermore, the queues are bounded. A two nodes system implemented with exponentially distributed incoming tasks. We analysed the performance metrics and energy consumption under different arrival rates. We found that TAGS can perform well and improve performance, although it increases total energy consumption. Finally, we calculated the energy per job to evaluate TAGS in a heterogeneous environment. We demonstrated that TAGS reduces energy consumption per job when the system is under a heavy load. This paper along with Paper 3 forms Chapter 6.

2

Background and related work

Contents

2.1 Introduction

This chapter presents an overview of the related background topics and relevant literature to this thesis. Section 2.2 discuss the power and energy and the difference between them. Section 2.3 contains a brief introduction to energy consumption in data centres, including primary power metrics used to assess the efficient use of energy in data centres. Section 2.4 briefly shows some mechanisms used to reduce energy and power consumption. Related previous studies are presented in Section 2.5.

Finally, Section 2.6 presents the PEPA modelling language, which we adopted as the modelling method throughout this thesis. We present the syntax of PEPA and demonstrate the Continuous Time Markov Chain (CTMC) analysis techniques.

2.2 Power and energy

It is imperative to differentiate between the terms power and energy specifically. Power and energy could be defined in terms of the work being performed by a system. The energy (E) is a measure of how much work was completed over a time duration, while the power (P) defined by [17] as " the time rate of doing work". In computing, this could be defined as the rate of electricity consumption. Power measured in Watts (W) and energy measured in Joules or Watt-hour (Wh). The relation between the power and energy defined as in equations 2.1

$$E = P.T \tag{2.1}$$

Where E is the energy, T refers to the period of time, and P is the power.

The Central Processing Unit (CPU) power consumption consists of dynamic power consumption, short circuit power consumption, and power leakage. The CPU dynamic power is created by the logic gates within the CPU chip. Dynamic power can be calculated using Equation 2.2 [10].

$$P = CV^2F \tag{2.2}$$

where P is the power, C is the capacitance, V is the supply voltage, and F is the CPU clock frequency.

In order to reduce the power consumption, one or more of the parameters must be decreased. The C value is related to the low-level system design. The voltage V and clock frequency F could be reduced by using DVFS, which is well known Dynamic Power Management (DPM) technique. The DVFS technique will be discussed in more detail in Section 2.4.3.

It is clear that the main difference between power and energy is the relation between energy and time. Increasing the time required to process a task will increase the energy. Thus, reducing power consumption by servers does not necessarily reflect a reduction in its energy consumption. For example, consider a job that needs a time duration T to be processed successfully at a power rate P; if the processor performance reduces to save power, the processing time to accomplish the job will increase, resulting in an energy increase or remains the same.

2.3 Energy consumption in data centres

The energy consumed by a data centre can be divided into two main categories: (i) energy consumed by IT equipment (e.g. servers, networks, and storage), and (ii) energy consumed by infrastructure facilities (e.g. cooling system and supporting infrastructure components) [55].

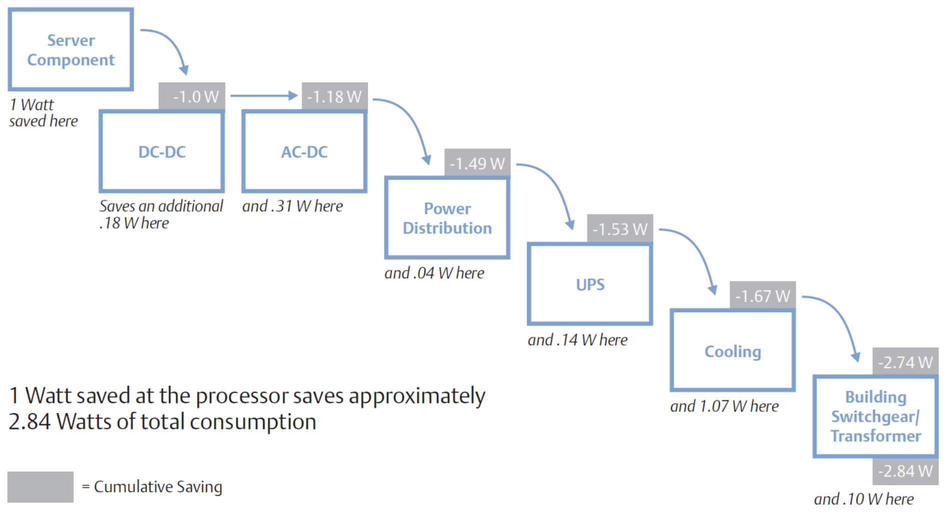


Figure 2.1: With the Cascade Effect, a 1-Watt savings at the server-component level creates a reduction in facility energy consumption of approximately 2.84 Watts [46].

2.3.1 Server energy consumption

In the context of a data centre, ’server ’ refers to computing and storage servers. This does not include communication equipment, which falls under the network domain category [37]. Servers in data centres are considered to be the main consumer of energy. Studies have shown that servers have become the dominant IT source of power consumption in data centres. According to [61], servers are responsible for nearly 56% of power consumption in data centres. Thus, it can reasonably conclude that running servers in idle mode or at low levels of utilisation represent a considerable waste of power [3]. A study by Emerson Network Power [46] found that a 1-Watt reduction in power at the server level reduces total facility energy consumption by approximately 2.84 Watts, as illustrated in Figure 2.1. In 2010, the estimated total number of installed servers in the United States alone was 15.8 million [5].

Many studies have focused on low-level power models that consider reducing energy consumption in each component in the server, e.g. CPU, Memory and storage disk or considered network components such as switches and routers. In contrast, there is another approach which, considers a more abstract modelling level. The abstract modelling level considers the job distributed between servers and shut down idle servers to save energy. The second approach is more generic and not focused on the application, operating system or hardware component.

2.3.2 Network energy consumption

The term “network” comprises all the communication equipment in a data centre used by servers to send or receive data, such as routers. The energy consumed by a network in a data centre is about 4% of the total IT equipment consumption of power, as stated in [46]. However, the network energy consumption accounts for around 10% of total IT equipment energy use [5].

2.3.3 Cooling system energy consumption

Cooling systems in data centres are responsible for 38% of total facility energy consumption [46]. However, this figure is said to be higher, maybe as much as higher 50% [48]. There is a direct relationship between the energy consumption of servers and cooling system energy consumption. When the server energy consumption increases, this leads to a rise in temperature in the room housing the servers. Consequently, to reduce the server room temperature, the cooling system works harder and thus consumes more energy.

2.3.4 Power metrics in data centre

Data centre owners and operators need a clear and effective mechanism to measure and calculate actual IT energy consumption and energy efficiency in their data centres. Accordingly, to calculate the efficient use of energy in data centres, it is essential to establish agreed standardized measurement metrics. Several energy metrics have been introduced and applied for different scales and components in data centres [60]. Prominent among these metrics is Power Usage Effectiveness (PUE) and Data Centre Infrastructure Efficiency (DCiE). Both metrics are considered and applied as primary metrics and are used to measure two parameters: total energy input to the data centre and IT equipment power. Malone *et al.* [36], introduced PUE in 2006, and defined it as ”the ratio of the total facility power to the IT equipment power. The Green Grid adopted PUE in 2007 [22]. PUE is illustrated in Equation 2.3, while DCiE is defined as the inverse of PUE, as seen in Equation 2.4

$$PUE = \frac{The\ total\ facility\ energy}{The\ IT\ equipment\ energy} \tag{2.3}$$

$$DCiE = \frac{The\ IT\ equipment\ energy}{The\ total\ facility\ energy} \tag{2.4}$$

From the previous equations, IT equipment energy is the energy consumed by IT equipment to operate required tasks (e.g. processing and storing data). Whereas the Total Facility Energy is the energy that is specifically dedicated for the data centre facility (i.e. the measured energy at the utility meter) [2]. The total facility energy includes the IT equipment energy and any systems that used energy to support the IT equipment. Such as Power delivery components (e.g. UPS systems, switchgear, generators, power distribution units (PDUs) and batteries), Cooling systems and other miscellaneous component loads [2].

Avelar *et al* [2], conducted a comprehensive examination of PUE. The optimal data centre efficiency is achieved when the PUE value is equal to 1. That means all power that goes into the facility is consumed or used by the IT equipment for its operation. However, any PUE value of more than 1 means more power in the system can support the IT load, which implies the existence of data centre overhead. The overhead occurs because not all power that enters the data centre is used by the IT equipment (e.g. servers, network and storage). Realistically, some of the total power used by the data centres are used by supporting systems such as cooling, lighting and other infrastructure systems. Moreover, in reality, there is a consequent loss of power that occurs in the power system [8].

It is essential to look at the PUE value considering the total energy consumed in the data centre. Some data centre managers sometimes misunderstand the PUE value because the PUE value only shows the efficient use of power, not the total decrease or increase of power usage. For instance, if the total input power to the facility was 300 kWh and the IT equipment consumed 150 kWh, meaning the PUE value is:

$$PUE = \frac{300}{150} = 2$$

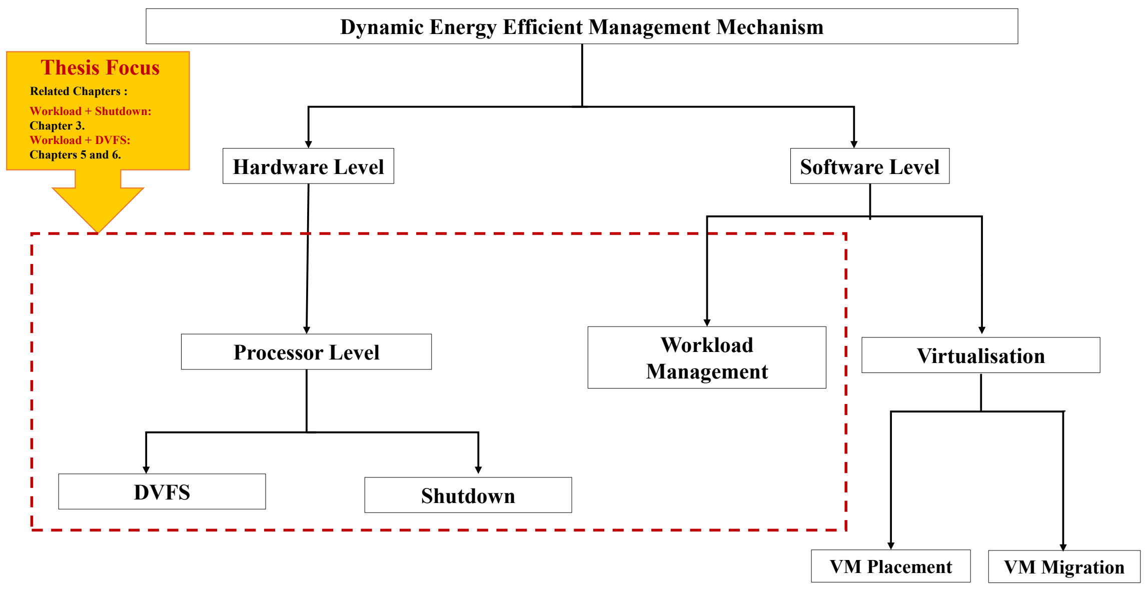


Figure 2.2: Taxonomy of dynamic energy-efficient management mechanism

The data centre manager may have intended to use virtualisation effectively to reduce the IT power consumption by 50%, so he had the IT usage of power reduced by 75kwh. However, because there are overhead management or power losses in other parts of the data centre, the PUE value might not reflect this decrease in power consumption resulting in the calculation below:

$$PUE = \frac{255}{75} = 3$$

So, given the anomaly that arises from such a situation, the data centre managers must investigate further or understand the cause of the rise or fall in the expected PUE value, such as UPS losses.

2.4 Servers energy and power saving mechanism

Reducing energy consumption in data centres involves a range of approaches. It must consider the multiple elements of the data centre (e.g. the IT equipment and support systems such as cooling systems). However, in this thesis, we will only discuss some of the power and energy-saving methods used on servers, as our models and analysis in Chapters 3, 4, 5 and 6 considers servers only. There are different mechanisms used to reduce servers' energy consumption, such as virtualisation, dynamic server allocation, and voltage and frequency scaling approach.

2.4.1 Server virtualisation

In cloud computing, virtualisation is an approach whereby multiple independent virtual operating systems run on one computer. Virtualisation is achieved by increasing server utilisation so that the same amount of processing in multiple servers is achieved using fewer servers. As a result, the number of running servers in the data centre can be considerably reduced, leading to a decrease in power and energy consumption by servers and cooling systems [59]. Multiple studies analyse the energy-saving potential of virtualisation in data centres, such as [6, 32, 61]. A comparison study by Buchanan *et al.* [6] compared PC energy consumption and that of Virtual Machines (VMs) in data centres equipped with 500 PCs, showed that there is a saving in power consumption when using virtualisation. A Mapper framework introduced in [61] used to explore the dynamic allocation of resources in virtualised machines to optimise energy consumption and performance. Also, Sharma and Sharma [49] introduced a new VMs load balancing algorithm applied to reduce costs and response time. The drawback of virtualisation is the energy overhead in virtualised servers. A virtualised server consume more energy than a physical server due to an increase in utilisation. This situation can be attributed to the energy overhead, which highly depends on the hypervisor used. Therefore, it becomes essential to have a trade-off between shutting down idle servers and energy overhead due to virtualisation [31].

2.4.2 Workload variation optimisation

The workload in computing refers to application utilisation, for example, works imposed on a computer, the number of task requests arrived in the system and the load due to task processing. Fehling *et al.* [16] categorised the application workload pattern in cloud data centres into five categories. First, *Static Workload*, where the utilisation over time has slightly fluctuated within specific boundaries or nearly constant. Second, *Periodic Workload*, when system utilisation increases to a peak level at reoccurring time intervals. Third, *Once-in-a-lifetime Workload* represents a particular case of periodic workload where the consistent utilisation of the system over time is impacted by an outburst of workload that occurs just once. Forth, *Unpredictable Workload*, where resources experience random workload over time, leading to volatile utilisation. Fifth, *Continuously Changing Workload* where system utilisation facing continuous growth or decline over time due to continuous change in the workload arrives at the system.

Workloads in the cloud frequently change over time, as do the resource needs, arrival rates, and operating duration. Therefore, It is essential to take into account the effect of workload variability on energy usage. As a result, comprehending the distribution of different server workloads and their corresponding hardware needs is important. Several workload optimisation techniques might be employed to mitigate the impact on energy usage. Additionally, identifying the workload differences across different servers would aid in improving server utilisation and lowering energy usage. Workload prediction is one technique that can be used to adjust the server resources to save energy. In this approach, resource consumption is recorded based on workload so that the resources required in the future can be predicted[28].

2.4.3 Voltage and frequency scaling mechanism

Processors nowadays feature energy-saving techniques that preserve a low power consumption by decreasing the operating frequencies of the processors. The used mechanism of voltage and frequency scale determines the relationship between power and frequency. Gandhi *et al.* [19] categorised the scaling mechanism under three broad categories. **(i) Dynamic Frequency Scaling (DFS) a.k.a. T-states** is a power-saving strategy in which the CPU is operated at a lower-than-maximum clock frequency. **(ii) Dynamic Voltage and Frequency Scaling (DVFS) a.k.a. P-states** power-saver technique that lowers power consumption by decreasing the processor's voltage and frequency at various levels. Finally **(iii) DVFS + DFS** ,this technique investigates the advantages of using both techniques by implementing the DFS at the lowest performance state in DVFS.

2.4.3.1 Dynamic Voltage and Frequency Scaling (DVFS)

The DVFS is the processor's capability to change its voltage and frequency in run time. A processor with a DVFS functionality can adjust its frequency up or down dynamically based on the number of jobs currently operating and the energy efficiency policy [7]. Therefore, DVFS allows the processor to reduce its power consumption by reacting to the change in workload demand in real-time by reducing its speed (i.e. scale down the frequency) when the workload is low.

The processors with DVFS-feature are classified into two types: ideal and non-ideal. The ideal processor can scale up or down the voltage and frequency continuously, which means the frequency and voltage can be any value between the processor's minimum and maximum supported voltage and frequency. On the contrary, a non-ideal processor has only supported some predefined frequency levels, and the processor is constrained to switching between these certain levels [7, 12].

Section 5.3 presents our energy model, which depends on the assumption that the CPU is equipped with the DVFS capability, and the CPU moves from one performance state (P-state values) to another based on the system utilisation. Therefore, the CPU requires the DVFS to adjust its voltage and frequency accordingly.

2.5 Related work

Slegers *et al.* [52] introduced a model to examine the cost of holding the job in the queue and the energy consumption cost by evaluating different heuristics of powering servers on or off. Six heuristics were introduced, including idle, static, threshold, semi-static, high/low arrival period and average flow heuristic. Heuristics control powering on or off servers according to job demand with different criteria. However, the model in [52] does not consider the server setup time (i.e. the time needed by a server to be fully powered on or down). Moreover, the benefit of powering down servers considered only the direct impact on the power consumption by servers and ignored the cascade effect (i.e. indirect energy saving in other IT components) [46]. Furthermore, it does not consider different locations of servers and assumes all servers are in one data centre location. Likewise, it assumes that all servers are homogeneous, meaning they are identical in their components and energy consumption, which is not always accurate in practice.

Mitrani [40] proposed a policy to reduce power consumption in a data centre by powering down a block of servers when the service can meet the job demand without that block of servers. The model assumed the data centre consists of N servers where n is permanent and always on and ready to serve the job while N - n is reserved servers that can dynamically be powered on or off according to the demand. Two thresholds control the availability of the reserved servers, 'U' and 'D', where U refers to Up and D refers to Down. Reserved servers powered on as a block if the job demand increased from U to $U + 1$ and powered off in the same fashion if job demand dropped from $D + 1$ to D. Reserved servers consume energy while powering on or off but cannot serve the job until fully powered on. The author assumes that a job cannot be lost when powered off reserved servers as the job will be transferred to another server.

Mitrani [41] extended the previous model in [40] by introducing multiple reserve blocks that can be turned on and off dynamically in response to different loading conditions. The aim was to investigate whether this approach reduces energy costs more than the single reserve-block approach. The results showed minimal advantages of using multiple reserve blocks instead of a single reserve block. Moreover, although the small amount of saving in large-scale systems can be valuable, a single reserve-block policy is sufficient in contrast to a complicated process of finding the optimal energy saving policy.

In another study, Van Do [58] proposed a simple energy-aware policy that controls the energy consumption of physical servers and moves to a low-power consumption level (e.g. sleep state) when no virtual machines are allocated to the physical server. In addition, when virtual servers are assigned to a physical server, they start operating at a high-power consumption level. The model consists of three different dynamic mechanisms to control the allocation request of virtual servers. The first mechanism allocates the request to the physical machine with the most significant number of virtual machines, but it is not entirely loaded. In contrast, the second mechanism maps the virtual machine request to the least loaded physical server. The last scheme prioritises physical servers and numbers them from lowest to highest priority. Then, when the job request arrives, it automatically chooses the fully loaded physical server that meets the prioritising scheme to place the request and activate the virtual machine.

Lent [35] studied the optimal energy consumption in an energy proportional data centre, where servers became available on demand. The main objective was to analyse the optimal energy requirements by servers to avoid violating the performance service level objective. Three different scenarios were considered, including servers running at or below the maximum utilisation, controlling the average response time up to a specific limit and reducing the probability of job response time exceeding a time limit deadline.

On the other hand, Ricciardi *et al.* [47] investigated a case considering the turning off of a subset of servers in the data centre to save power. The decision of switching on or off servers depend on fluctuation in service demand and the available service. Consequently, when the difference between the demand and the available service increased, more servers turned on to meet the increase in service demand. On the contrary, more servers turned off when the difference between service demand and available service decreased.

In terms of evaluating the trade-off between reducing energy consumption in data centres and job performance to obtain the maximum revenue Ghamkhari *et al.* [20] proposed a systematic approach to maximise the profit in data centres. A practical service level agreement (SLA) is considered in the model, and other factors include the availability of local renewable power at the data centre and the workload.

Phung-Duc [44] considered a multi-server queuing model with setup time and impatient customers to analyse the power saving and the performance trade-off in data centres. In his model, the server shut down immediately if it has no job to do. In contrast, off servers are switched on when a new job arrives in the system.

Vimal *et al.* [38] presented an algorithm for cluster shutdown that turns off servers in an entire cluster of a Content Delivery Network (CDN) in a data centre. A real-world trace from a group of commercial CDN was used to evaluate the cluster shut down technique, and results show that a reduction by 67% can be achievable.

Gandhi *et al.*[18] studied the issue of the topic energy-performance trade-off in server farms by utilising the Energy-Response Time Product (ERP) to find the optimal server farm management policy. They looked at three policies. The first policy is NEVEROFF, where any server that goes idle remains in the idle state until a new job arrives. The second policy is INSTANTOFF, which in contrast to the previous policy, any server that goes to state idle turns off as long as there are no jobs in the queue. However, the server turns on as soon as the job arrives. The final policy is SLEEP, where any server in an idle state moves to a sleep state instead of being turned off. However, every job arrival to the system turns on a sleeping server; when available, working servers can not serve the number of jobs in the queue. They prove that for a single server under Poisson arrivals, the optimal policy with respect to ERP is either to always keep the server on or idle (NEVEROFF policy) or always to turn the idle server off and to turn it back on when the job arrives (INSTANTOFF policy), or to put the server in one of the sleep states when idle (SLEEP policy). For a multi-server system, based on their finding, they suggest the previous policies generalisation sufficient to find a near-optimal policy.

Phung-Duc [45] proposed a mathematical model in order to reduce the waiting time needed by a server to switch on. The number of jobs in the system, states of servers, job distribution, and the generating functions are all utilised to derive the mean queue length and the mean power consumption.

Table 2.1: Comparison of related work in terms of techniques, limitations and methods.

Reference	Used Technique	Limitation	Method
Slegers *et al.* [52]	Heuristics control powering on or off homogeneous servers dynamically according to job demand with different criteria.	Servers are homogeneous..	Simulation of a pre-existing mathematical model by same authors
Mitrani [40]	Powering down a block of servers when the service can meet the job demand without that block of servers. Servers are turned on and off, not individually, but in a block. The job can not be lost when powered down the reserved servers, as the job will be transferred to other servers. Jobs are impatient and liable to defect.	Single block only. Homogeneous only.	Mathematical model and numerical experiments.
Mitrani [41]	Multiple reserve blocks can be turned on and off dynamically in response to different loading conditions. Customers can be patient or defect.	Minimal advantages over single block method. Complicated process to find the optimal saving policy. Homogeneous only.	Mathematical model and numerical experiments.
Van Do [58]	An energy-aware policy that controls the energy consumption of physical servers and moves to a low-power consumption level(e.g. sleep state) when no virtual machines are allocated to the physical server.	Did not look at the impact of servers environment (Homogeneous or Heterogeneous).	545454
Lent [35]	Analysed the optimal energy requirements by servers to not violate the performance Service Level Objective (SLO). Servers became available on-demand. SLO-1 limiting the utilisation level of servers. SLO-1constraints the average job response time.	Homogeneous servers only. Servers can hibernate but not sleep or be switched off. The characteristics of the workload such as average time to complete or waiting time average are known.	Mathematical model and real experiment on real hardware
Ricciardi *et al.* [47]	A data centre energy manager, which turning off a subset of servers to save power. Switching on or off servers depends on fluctuating service demand and the available service resources.	Did not look at the impact of servers environment (Homogeneous or Heterogeneous).	Simulation of the model against available real data traces from Google and Naples LHC Tier2Grid site
Ghamkhari *et al.* [20]	An analytical model to calculate profit in large data centres. Evaluating the trade-offs between reducing energy consumption in data centres and job performance to obtain the maximum revenue. Considered in the model practical Service Level Agreement (SLA), availability of renewable power and stochastic nature of data centres workload.	Did not look at the impact of servers environment (Homogeneous or Heterogeneous).	Simulation
Phung-Duc [44]	A multi-server queuing model with setup time and impatient customers to analyse the power saving and the performance trade-off in data centres. The server is shut down immediately if it has no job to do. Jobs leave without receiving the service if the waiting time is larger than the timer.	No consideration on different servers specification impact on energy and performance.	Mathematical model and numerical experiments.
Vimal *et al.* [38]	An algorithm for cluster shutdown that turns off servers in an entire cluster of a Content Delivery Network (CDN).	Turn off entire clusters or leave them entirely. Cannot turn off servers individually. Servers are Homogeneous with identical capacities.	Linear energy model. The algorithm Simulation result was validated against real load traces from a large set of Akami clusters (data centres) in the USA.
Gandhi *et al.*[18]	Energy-performance trade-off in server farms by utilising the Energy-Response Time Product (ERP) to find the optimal server farm management policy. Three Policies govern the switch between server states: NEVEROFF: Servers goes Idle when there is no job in the queue. INSTANTOFF: turn off any server when there is no job in the queue. SLEEP: moves idle server to sleep state.	Servers are homogeneous	A discrete event simulator is written in the C++ language to verify the theoretical results
Phung-Duc [45]	A mathematical model to reduce server setup time and power consumption. Servers states and job distribution were utilised to derive the mean queue length and mean power consumption.	Servers are homogeneous	Mathematical model and numerical experiments.
Hotta *et al.* [28]	Profile-based power-performance optimisation by using DVS. Workload prediction. Adjust the server resources to save energy by using the DVS technique.	Servers are homogeneous	Real Experiment
Buchanan and Yampolsky [6]	Compared energy consumption in data centre equipped with 500 PCs. The result proved virtualisation reduce energy consumption.	Unused servers move to idle states, not shutdown. No analysis of heterogeneity impact on energy consumption.	Real Experiment
Warkozek*et al.* [61]	Introduced Mapper framework dynamically allocate resources in virtualised machines to optimise energy consumption and performance.	Servers are homogeneous	Real Experiment
Sharma [49]	Proposed a VM load balancing algorithm to reduce costs and response time.	Virtual machines are identical.	Simulation-based on CloudSim 11 and Cloudsim based tools 12

As seen in Table 2.1, the literature employs a variety of approaches and methodologies. Most prior research has concentrated on conducting experiments and drawing conclusions in a homogeneous environment. Moreover, some conducted the experiments in a heterogeneous environment but without examining the effect of heterogeneity on energy use.

The purpose of this study is to conduct a new investigation into performance and energy consumption trade-offs. This study examines the influence on energy consumption of (1) single queue multi-servers with dynamically powered on or off servers in a homogeneous environment, (2) individual queue server pairs, and how work is distributed amongst queues in a homogeneous and heterogeneous environment. Unlike previous research, we considered the impact of heterogeneity on energy and performance.

It is well established that various research methodologies are required to create novel insights or avoid skewed outcomes [29, 39]. As a result, we employed a modelling language distinct from those used in previous investigations. PEPA [24] was used to model and analyse the systems under consideration. Further details of PEPA are presented in Section 2.6.

In summary, in this research, we (1) used PEPA to build models in Chapters 3, 4, 6 (2) investigated the techniques of powering on or off servers dynamically and their impact on performance and energy trade-off in a homogeneous environment in Chapter 3, (3) investigated the impact of scheduling algorithms on performance and energy trade-offs in a homogeneous environment in Chapter 5, and in a heterogeneous environment in Chapter 6.

2.6 Performance Evaluation Process Algebra (PEPA)

Performance modelling is the modelling of the dynamics of systems. In performance modelling, we are interested in timing and probability because we want to ensure the resources in the system are used reasonably and efficiently. Given the difficulty of constructing Markov processes for big systems despite their broad application, an intermediate system description language is frequently employed. High-level modelling formalisms such as queuing networks, Stochastic Petri Nets (SPNs) and Stochastic Process Algebra (SPA) are used to build models more easily. Queuing networks are restricted in expressiveness and lack formal interpretation. In contrast, SPNs models have formal interpretation but lack explicit structure, making the model building much easier but the analysis can be more difficult. However, In contrast to SPA SPNs are not compositional [26]. Different research papers address SPA and SPNs. Donatelli *et al* [11] presents a comparison between an SPA, PEPA, and Generalized Stochastic Petri Net (GSPN), Gilmore *et al* [21] introduced PEPA nets formalism which uses PEPA as the inscription language for labelled stochastic Petri.

As mentioned previously in Chapter 1, we will use PEPA to carry out the modelling and the analysis throughout this thesis. PEPA is an extension to classical process algebras, which was introduced by Jane Hillston [24]. PEPA was developed to be a high-level description language for Markov processes. Therefore, The model specifications written in PEPA are Markovian and can be mapped to a Continuous Time Markov Chain (CTMC). PEPA is used to effectively define, construct and analyse models in various systems for the performance evaluation. These models included but not limited to Healthcare systems [63] , resource allocation [53], network protocols [43], and security protocols [64] In PEPA, systems are represented as a set of *components* that can be individually or multiply engaged in *activities*. Each *component* represents an element of the system or behaviour. The occurrence of each *activity* in the system is determined by the associated rate for each action in that activity. An activity *arrival* represented as a pair $(arrival, \lambda)$ consist of the action type *arrival* associated with the activity rate λ, where $arrival \in \mathcal{A}$ and $\mathcal{A}$ is a set of activities.The activity rate is a random parameter representing the activity duration. The supported rate of any action by PEPA follows the negative exponential distribution. Thus, the rate λ can be any positive real number $\mathbb{R}_{\geq 0} = \{\lambda \in \mathbb{R} \mid \lambda \geq 0\}$ or unspecified (represented as $\top$). Unspecified, or passive, rate means the component does not have control over the rate of this action. Therefore, while the cooperation of the components might be necessary to carry out such an activity, the component is not engaged in the work. So, the rate of the activity is determined by another component of the system. Various features of performance modelling, including compositionality, formality and abstraction, are supported in the PEPA. Following is a brief description of the most important supported features in PEPA.

- **Parsimony:** PEPA is a simplified language with a few basic elements, which are components and activities. This parsimonious allows the modeller to understand the language easily and have great flexibility when building the model.
- **Formal Definition:** The simple structured operational semantics of the language presents a formal description of all expressions. Equivalence notions were developed based on these semantic rules to provide a formal basis for models and components comparison and manipulation.
- **Compositionality:** In PEPA, forming the interaction between the model's components can be done by the *cooperation combinator*. This cooperation combinator gives the modeller flexibility to simplify any part of the model in isolation of other components interaction in the model. The model simplification and aggregation techniques can be developed, which are complementary to this combinator.

2.6.1 The Syntax of PEPA.

A comprehensive formal overview of PEPA syntax is given in [24], but for brevity, the following informal overview will suffice in this thesis. As mentioned earlier, components and activities are the primary building blocks of the language PEPA, so models are structured based on them. Furthermore, PEPA has a small collection of combinators. The combinators of the language can be used to construct expressions that describe and define the behaviour of components through their activities and interactions. The syntax for terms and describing components interactions in PEPA can be defined as:

$$P ::= (\alpha, f).P \mid P \underset{L}{\bowtie} Q \mid P + Q \mid P/L \mid A$$

Prefix: $(\alpha, f).P$

The prefix is the mechanism used to represent and construct the behaviour of the system components. The component $(\alpha, f).P$ performs the activity (α, f) of action type α and an exponentially distributed duration with rate f. After triggering the activity (α, f), the components behave as (P). The rate f could be just a constant or a functional system's current state to allow a certain amount of that activity. Moreover, the passive rate $\top$ can be used instead of f in shared activities.

Choice: $P + Q$ The notion of choice means the system may behave as component P or component Q. $P + Q$ enables all current activities in P and Q. A race condition governs which component will be triggered. The probabilistic choice depends on the speed of the first action in both P and Q.

Cooperation: $P \underset{L}{\bowtie} Q$

The components can interact with each other over shared activity. The cooperation combinator represents a set of shared action types L, where $L \subseteq \mathcal{A}$. The set of shared action called *cooperation set* defines the interaction between both components P and Q. So, if components P and Q interact over a different set $P \underset{K}{\bowtie} Q$, they will have different behaviour from $P \underset{L}{\bowtie} Q$, if $L \neq K$.

If there are other activities for P or Q not in the *cooperation set*, they will not be affected and processed independently.

A component P cooperates over shred activity will not be able to process the shared action without the involvement of the other component Q. Therefore, the component P might be blocked waiting for component Q to engage. It is worth mentioning that the cooperation between the components will happen over shared activities with the same action type, but the rate will be for the slowest activity.

In PEPA, if components do not have shared activity $P \underset{K}{\bowtie} Q$ where $K = \varnothing$, they will proceed independently. As there is no shared activity between P and Q the set K

could be replaced by $\varnothing$, which refers to the empty set, $P \underset{\varnothing}{\bowtie} Q$. Also, the shorthand $\|$ parallel combinator represents this case as $P \parallel Q$. Moreover, if there are many identical components, synchronised over an empty set $\underset{\varnothing}{\bowtie}$, i.e. N copies of component P instead of represent them as $P_i \parallel ... \parallel P_N$, we could represent them with a shorthand notation $P[N]$, which means there are N instance of component P.

A component in PEPA can cooperate with one or more components; this is called multi-way cooperation. Therefore, all engaged components need to be synchronised in order to fulfil the activity. The cooperation set will have an impact on the behaviour of the model. For Example, by considering the following system which consists of three components P, Q and S and some α type activities.

$$\Big((\alpha, r).P \underset{L}{\bowtie} (\alpha, m).Q\Big) \underset{L}{\bowtie} (\alpha, n).S$$

In this system, P, Q and S would provide three-way synchronisation between them on the activity of type α.

If the cooperation set changed we will have different possibilities of the model behaviour, considering the following changes:

$$((\alpha, r).P \parallel (\alpha, m).Q) \underset{L}{\bowtie} (\alpha, r).S$$

In this case, components P and Q will run in parallel as they do not have shared activity, but both of them can cooperate with component S over set $\underset{L}{\bowtie}$. However, because both components are inside brackets, the parallelism here means both components will compete to cooperate with component S. As a result, two activities of type α will be possible, but only one of them can be processed. However, if the cooperation set changed to be as follow:

$$((\alpha, r).P \underset{L}{\bowtie} (\alpha, m).Q) \parallel (\alpha, r).S$$

The behaviour will be different. The new cooperation set will mean there are two α type activities one synchronise P and Q and one in S; in contrast to the previous cooperation set, both activities can proceed.

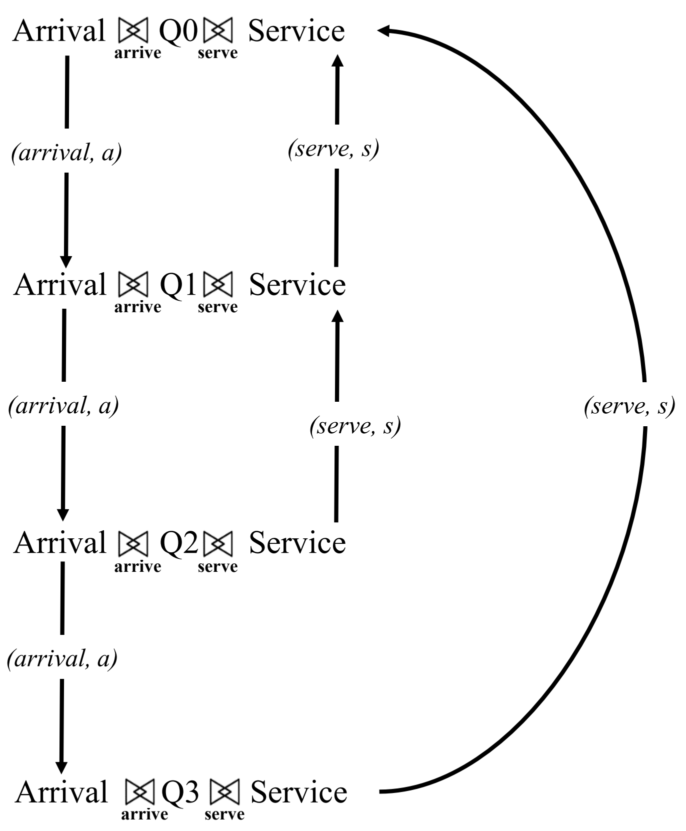
Hiding: P/L

The hiding means that the component P/L will behave as P with all activities except hidden activities in set L. Therefore, the hidden activity is not observed externally. Moreover, such an activity cannot be involved in any cooperation with other components.

Constant: $\boldsymbol{A} \overset{\text{def}}{\equiv} P$

The behaviour of component P can be given to a constant $\mathcal{A}$. So, whenever $\mathcal{A}$ occurs, the P component will replace it.

2.6.2 Continuous Time Markov Chain CTMC



Arrival ⋈arrive Q0 ⋈serve Service
(arrival, a)
(serve, s)
Arrival ⋈arrive Q1 ⋈serve Service
(arrival, a)
(serve, s)
(serve, s)
Arrival ⋈arrive Q2 ⋈serve Service
(arrival, a)
Arrival ⋈arrive Q3 ⋈serve Service

Figure 2.3: Underlying CTMC of the simple PEPA model of M/M/1/3

PEPA support two analysis techniques CTMC and Ordinary Differential Equations (ODEs). Hillston [24] stated a system could be represented as a stochastic process employing a derivation graph. The derivative graph represents the state transition diagram of a continuous-time Markov chain. By translating a PEPA model to a

Markovian process, a negative exponential distribution is assumed to rule the duration of all activities. The CTMC can be used to analyse and evaluate small systems due to the state space expansion problem. Therefore, Hillston in [25] introduces the ODEs to allow PEPA to approximate a massive discrete-state system as a continuous state system.

In this thesis, we are using the CTMC to analyse and evaluate our models.

A simple Example

Figure 2.3 illustrates the derivation of CTMC by presenting a simple queue system in PEPA. In this example, we considered the $M/M/1/N$ Queue, where the arrival process is a Poisson process, and the service time is exponentially distributed. The number of servers in the system is one server. The buffer capacity is finite $N = 3$. In this example, jobs arrive at a single queue and are then served by the server on an FCFS basis. All places in the queue are initially empty Q_0. When an *arrival* action happens with rate a, then one place in the queue will become full. Then, the system can serve this job by the action *serve* at rate s or continuous receiving jobs. If at least one place in the queue is occupied, the server can do the *service* action with rate s. When the server serves the job, one place in the queue will become empty. The job arrival to the system will continue until all places in the queue become full, then the job arrival will be blocked by the system. This is done by removing the *arrival* action from the last position in the queue Q_3 and only keeps the serve action.

The model in PEPA is shown in Figure 2.4 and alternative PEPA specification is shown in Figure 2.5 . The model generates four states of the underlying process. Let assume these states can be labelled as x_0,..., X_4, which is identified as follows:

$$\begin{aligned}
X_0 &\longleftrightarrow Arrival \underset{arrive}{\bowtie} Q0 \underset{serve}{\bowtie} Service\\
X_1 &\longleftrightarrow Arrival \underset{arrive}{\bowtie} Q1 \underset{serve}{\bowtie} Service\\
X_2 &\longleftrightarrow Arrival \underset{arrive}{\bowtie} Q2 \underset{serve}{\bowtie} Service\\
X_3 &\longleftrightarrow Arrival \underset{arrive}{\bowtie} Q3 \underset{serve}{\bowtie} Service
\end{aligned}$$

by using the global balance equation, $\Pi \mathrm{Q} = 0$, the steady-state distribution of the system can be obtained. When applying the global balance equation on the states of

$$\begin{aligned}
\mathit{Q0} &\overset{\text{def}}{=} (arrival, \top).\mathit{Q1},\\
\mathit{Q1} &\overset{\text{def}}{=} (arrival, \top).\mathit{Q2} + (serve, \top).\mathit{Q0},\\
\mathit{Q2} &\overset{\text{def}}{=} (arrival, \top).\mathit{Q3} + (serve, \top).\mathit{Q1},\\
\mathit{Q3} &\overset{\text{def}}{=} (serve, \top).\mathit{Q0},\\
\\
Arrival &\overset{\text{def}}{=} (arrive, a).Arrival,\\
Service &\overset{\text{def}}{=} (serve, s).Service,
\end{aligned}$$

$$Arrival \underset{arrive}{\bowtie} \text{Q0} \underset{serve}{\bowtie} \text{Service}$$

Figure 2.4: The PEPA model for $M/M/1/3$

the underlying process, we will get:

$$\begin{gathered}
\text{a}\Pi(\text{X}_0) = s\Pi(\text{X}_3) + s\Pi(\text{X}_1)\\
(\text{a} + \text{s})\Pi(\text{X}_1) = a\Pi(\text{X}_0) + s\Pi(\text{X}_2)\\
(\text{a} + \text{s})\Pi(\text{X}_2) = a\Pi(\text{X}_1)\\
(\text{s})\Pi(\text{X}_3) = a\Pi(\text{X}_2)
\end{gathered}$$

The normalisation equation will be as :

$$\sum_{i=0}^{3} \Pi(X_i) = 1$$

Metrics such as throughput and average response time and utilisation can be calculated from these steady states probabilities. More details can be found in [24].

Alternative PEPA model

It is worth mentioning, the PEPA model presented in Figure 2.4 can be rewritten as in Figure 2.5. The two models are isomorphic - meaning that they have a 1:1 equivalence in the underlying CTMC. As such, their steady-state solutions are identical by definition. The point of this form of specification is that it is more concise to specify, especially when the maximum queue size is large. Adding other places is easy by just changing the system equation. For example, if N= 100, that can be specified by changing the system equation to be:

$Arrival \underset{arrive}{\bowtie} \text{QEmpty[100]} \underset{serve}{\bowtie} \text{Service}$

$$\begin{aligned}
QEmpty &\overset{\text{def}}{=} (arrival, \top).QFull, \\
QFull &\overset{\text{def}}{=} (serve, \top).QEmpty \\
Arrival &\overset{\text{def}}{=} (arrive, a).Arrival, \\
Service &\overset{\text{def}}{=} (serve, s).Service,
\end{aligned}$$

$$Arrival \underset{arrive}{\bowtie} \text{QEmpty[3]} \underset{serve}{\bowtie} \text{Service}$$

Figure 2.5: Alternative PEPA model for $M/M/1/3$

However in the original model Figure 2.4 each place has to be specified in the Queue component, i.e. Q1, Q2, ..., Q_N. However, sometimes we need to use this model if there are triggers which occur at specific queue sizes (e.g. threshold levels for change of behaviour).

2.7 Research methodology

To fulfil the research aims and objectives and provide answers to the research questions, we studied two main concepts: (i) job allocation in a single queue multi servers system and (ii) job allocation in a single queue and single service system.

First, for job allocation in a single queue multi servers system, and based on the work we have previously discussed in Section 2.5, we select the dynamic server allocation scheme [52] to be modelled in PEPA. In Chapter 3 we explained the system in-depth and modelled one of its policies in PEPA to demonstrate how it works. The next step was solving the PEPA model, which yielded the steady-state solution. The experiment's results were utilised to make a trade-off between performance and energy consumption. Chapter 3 experiment and results represent the attempt to answer research question1.

Second, for the job allocation in a single queue and single service system. Whereas queues are separate, and each queue is connected to a single server. TAGS scheme a job allocation scheme [23] was chosen to be modelled in PEPA along with shortest queue strategy and random allocation. Chapter 4 discussed the TAGS scheme in detail and presented how the TAGS, shortest queue and random allocation PEPA models work. Chapter 5 studied performance metrics such as throughput, average response time and servers utilisation. Additionally, the energy model that was used to determine energy usage in terms of overall energy consumption and energy consumption per task is presented in section 5.3. Furthermore, the experiment results were utilised to trade-off performance and energy usage in a homogeneous setting. On the other hand, we studied TAGS in a heterogeneous environment; Chapter 6 describes the modification to the PEPA models to implement system heterogeneity. Furthermore, it shows the experiment's performance and energy consumption results. It is worth mentioning that the same energy model is applied to homogeneous and heterogeneous environments. Chapters 4, 5 and 6 represent an effort to address research question 2.

2.8 Experiments

There are three experiments conducted to answer the research questions: (i) dynamic server allocation in chapter 3, (ii) TAGS in homogeneous environments in chapter 5 and (iii) TAGS in a heterogeneous environment in chapter 6.

2.8.1 Experiments objectives

In this thesis, we conducted three main experiments to fulfil the requirements to answer research questions in Section 1.3 and meet the research objectives in Section 1.4. The experiments and their related chapters are as follow:

Dynamic server experiment in Chapter 3: In a single queue multi-server system, this experiment evaluated the trade-off between performance and energy usage. The primary objective of this experiment is to determine the energy and performance consequences of a management policy that dynamically powers on or off servers.

TAGS experiment in a homogeneous environment in Chapter 5: In this experiment, job allocation in a single queue and single service system is the main focus. Queues are separate, and each queue is connected to a single server. TAGS scheme a job allocation scheme [23] was chosen to be modelled in PEPA. The main aim of this experiment is to investigate the scheduling scheme impact on performance and energy in a homogeneous environment. Particularly the TAGS as it is the focus of the study as a particularly different type of scheduling algorithm that is suitable for variable demand.

TAGS experiment in a heterogeneous environment in Chapter 6 The same technique is used in this experiment as in the preceding experiment in Chapter 5. However, the focus is on the heterogeneous environment to analyse the effect of changing the environment on performance and energy trade-off.

2.8.2 Experiments environments

The following are the experimental conditions in terms of hardware and software environments in which these experiments were completed:

Hardware environments: All experiments in this thesis were performed in a PC with the following specifications: Intel Core i7 (3770) 3.4GHz Quad-Core Processor and Installed memory (RAM) 8.0 GB.

Software environments: The operating system is Windows 8.1 Enterprise. The Software to analyse and simulate PEPA models is Eclipse IDE for Java Developers, versions Neon and Oxygen.3a equipped with version v25 of the PEPA Eclipse Plugin [27, 57].

Queuing Concept The idea of "queueing" in the context of this thesis is based on the queuing theory. Many research publications make wide use of the queuing system in their cloud computing modelling. Ghomi *et al.* [30], conducts a systematic literature review, which categorises cloud computing modelling techniques based on the queuing theory into seven categories based on their focus area: (1) performance, (2) quality of service, (3) workflow scheduling, (4) energy savings, (5) resource management, (6) priority-based servicing, and (7) reliability. The author focus in the literature is on queuing theory for cloud environment modelling. So queuing models were classified into two categories: (1) single-queue models, such as M/M/1, G/M/1, and M/M/k, and (2) queuing network models, such as a Jackson network and an open/closed network.

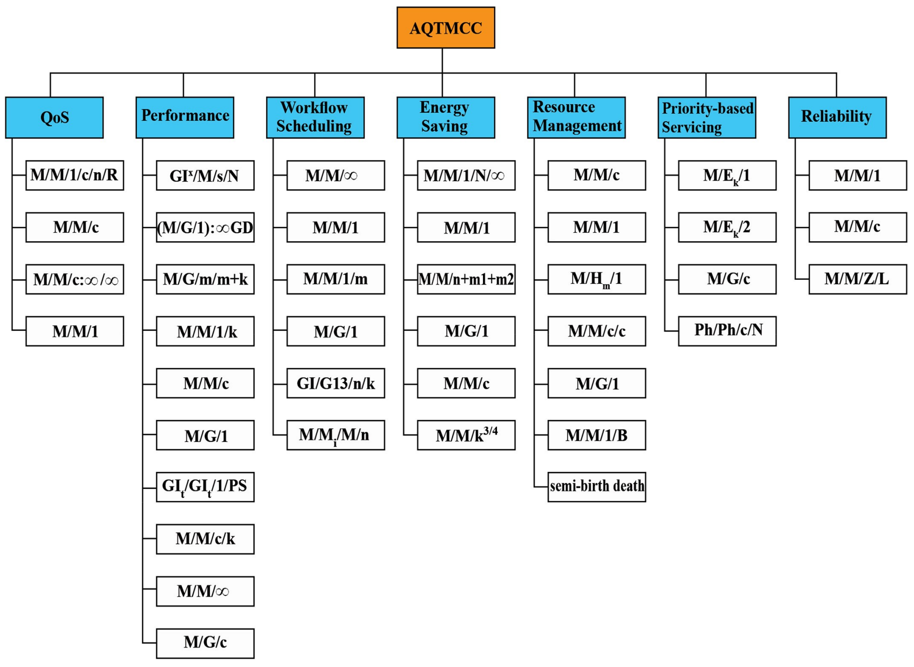


Figure 2.6: A taxonomy of queue models for AQTMCC by [30].

In Figure 2.6, Ghomi *et al* [30] summarised the queuing models used in modelling cloud computing in the Applying Queue Theory for Modelling of Cloud Computing (AQTMCC) survey by categorising them into seven groups.

The M/M/1 is being utilised to model performance and energy consumption in a cloud environment. The research in this thesis focuses on task allocation in two scenarios: (1) a single queue multi-server system with dynamically powered-on or powered-off servers; and (2) a single queue and single service system with distinct queues and each queue connected to a single server.

Validation: There was not a specific validation of the PEPA models was undertaken within this thesis. Nevertheless, there is a relation of the results to pre-existing results within the field.

Reliability: Regarding the reliability of results, the Eclipse Plugin is a tool used to construct and solve PEPA models for many years and was used in many research papers. So, it is proven that its performance results are reliable. Performance metrics, including (throughput, utilisation, and population) are calculated automatically by the

PEPA Eclipse tool. However, as with any numerical calculation, there will be some rounding errors and formation errors due to the software's finite representation of real numbers. In CTMC solver, there are just floating-point errors in the calculation, but they are consistent; they are the same every time we solve the model. Nevertheless, there is not an explicit error tolerance controller.

Reproducibility: PEPA models and experiment findings may be replicated in various environments without affecting the accurate results, as long as PEPA eclipse v25 is used with the same model parameters.

General Limitation: The major limitation in all experiments in this thesis is related to PEPA and available computing resources. The state-space expansion in PEPA leads to an unsolvable model under the available computing resources. So, to overcome this issue, the queue size, the number of the queues and servers in each PEPA model were limited. However, the servers and queues numbers and the queue size could be increased easily without significant changes to the model under higher computing resources

2.8.3 Notations and Terminologies

It is necessary to specify notations and terminology before utilising them in the thesis to minimise ambiguity and inaccuracies. As a result, Table 2.2 contains the notation and terminology for each chapter. The fact that some notations are used across all chapters while others are only used in specific chapters should be noted.

Table 2.2: Notations and Terminologies

Notation used in all chapters	
Notation	**Meaning**
λ	The job arrival rate.
μ	The average service rate.
$Q1$	The first queue.
$Q2$	The second queue.
Q_0	Component represent the initial state of the queue.
Chapter 3 Notations	
Notation	**Meaning**
$Arrival_{high}$	Component refer to the high arrival period.
$arrivalH$	The high arrival action.
$Arrival_{low}$	Component refer to the low arrival period.
$arrivalL$	The low arrival action.
ε	The rate of low arrival action.
$highPeriodEnd$	Action refer to the end of high arrival period.
β	The rate of $highPeriodEnd$ action.
$lowPeriodEnd$	Action refer to the end of low arrival period.
γ	The rate of $lowPeriodEnd$ action
$\overline{\lambda}$	The average arrival .
$Server_{static}$	Component refers to a server that is always available and cannot change state.
$Server_{On}$	Component refers to a server when its state is on.
$Server_{Off}$	Component refers to a server when its state is Off.
$ServerPowering_{On}$	Component refers to a server when its state is powering on.
$powerup$	Action of powering up the server.
$ServerPowering_{Off}$	Component refers to a server when its state is powering off.
$poweroff$	Action of powering off the server.
$Server_{failOn}$	Component refers to a server when its state is failing during powering on.
$Server_{failOff}$	Component refers to a server when its state is failing during powering off.
$C1$	The weight assigned to energy.
$C2$	The weight assigned to the performance.
Chapters 4,5,6 Notations	
Notation	**Meaning**
$Timer1$	The first timer which governs the decision to terminate the job at the first server after a specific processing time.
$Timer2$	The timer in the second server used to model the repeated service.
$K1$	The maximum length of the first queue.
$K2$	The maximum length of the second queue.
$arrival$	The arrival process.
$service1$	The service process at first server.
$service2$	The service process at second server.
$timeout$	The timeout action which kills the job at $server1$ when triggered.
$tick1$	The tick action of the timeout clock.
$repeateservice$	the repeat service action, that repeats the amount of service that timed out previously in $server1$.
$\mu1$	The service rate at $Server1$
$\mu2$	The service rate at $Server2$
t	The timeout rate
α	The proportion of short job.
α'	The proportion of long job.
σ'	The difference in performance between $Server1$ and $Server2$.

3

Dynamic Server Allocation in a homogeneous system

Contents

3.1 Introduction

In this chapter, we present a PEPA model of a policy for managing the turning on and off of servers, previously studied by Slegers *et al.* [50, 51, 52] and Nguyen*et al.* [42]. Various assumptions and modifications to the original model have been made to help model the policy in PEPA. The main focus was identifying and reflecting on the trade-off between saving energy (by powering down servers) and performance cost. Powering down servers saves energy, but it could increase the performance cost. The experiment in this chapter analysed the effect of the policy on energy consumption and performance cost. Different combinations of dynamic and static servers are used in this policy against different scenarios, including changes in job arrival rate, job arrival duration, and the time needed by servers to fully power on and serve the jobs.

This chapter consists of 4 main sections. Section 3.2 presents related work of dynamic server allocation that forms the basis of our PEPA model in this chapter. Section 3.3 presents the PEPA model of high/low policy. Section 3.4 presents the formulas of energy and performance costs and shows the evaluation results of the model in four scenarios. Section 3.5 presents the chapter conclusion and future work.

3.2 Dynamic server allocation model

The problem of unnecessary power consumption in the data centre by servers has been the subject of increasing focus; however, the amount of energy that can be saved by switching servers on or off according to demand has not been given more consideration. Slegers *et al.* [52] proposed several heuristics regarding organise server usage according to demand. This approach is further examined in [42] by using JAVA to implement the heuristics policies and simulate dynamic servers allocation for power efficiency management according to demand. The heuristics referred to in [42] were simulated experimentally. The heuristics were individually tested by running through the circumstances established in the design of the experiments. A variety of experimental scenarios were established so that the heuristics could be adequately appraised; this was of particular importance as the conclusion presented in [2] determined that no heuristic is universally applicable but that each heuristic characteristic that is appropriate in only certain circumstances.

The system in [42] consists of a given number (N) of homogeneous servers. These servers may be in any of five conditions; namely, “powered up”, wherein the server may be either active or idle, “powering up”, “powered down”, “powering down”, and “fault”. While in the “powered down” condition, the server may be in a quiescent condition or completely turned off, although, whichever happens, to be the case, its power consumption would be little or nothing. In the transitional conditions, i.e. “powering down” or “powering up”, and the “fault” condition, servers are unable to respond to service demands but would nevertheless still be consuming power. Although the “fault” condition could coincide with any of the other four conditions, the author of [42] only considered the fault in the transitional states (Powering up and Powering down) due to the increased likelihood of faults becoming apparent then. This, it is surmised, should be sufficient to indicate how consistently the system is performing. In order to analyse the relative costs inherent in system operation, costs were assigned to each system condition. These were as follows: “Powered up”: $\mathrm{C_{up}}$, “Powering up”: $\mathrm{C_{pow}Up}$,“Powered down”: $\mathrm{C_{down}}$,“Powering down”: $\mathrm{C_{pow}Down}$ and “Fault”: $\mathrm{C_{fault}}$. The first four of these represent relative power costs per time unit. In contrast, $\mathrm{C_{fault}}$ represents the proportionate detriment suffered by the system as a result of a server being faulty and, consequently, inoperative.

An additional cost was identified in respect of system operation, this being $\mathrm{C_{job}}$ the cost of holding jobs in the queue for over one time unit. This represented a requirement of increasing the rate at which jobs were handled to meet increased service demand. The final additional cost identified was $\mathrm{C_{pow}}$ the beneficial consequence of holding a server in the “powered down” condition per unit time. These costs are expressed comparatively, so if $\mathrm{C_{pow}} = 1$ and $\mathrm{C_{job}} = 2$, it may be surmised that the cost associated with holding a job in the queue is twice that of the benefit accrued as a result of holding a server in the “powered down” condition.

There are several heuristics noted in [42], and [52], each of which has its advantages and disadvantages with regard to power conservation, performance and consistent performance. In total, there are six policies introduced in [52] and simulated in JAVA in [42].

The first heuristic is the Static Allocation Heuristic, where servers never turn off or on; instead, a fixed number of servers are permanently available. The second heuristic is the Semi-static Allocation Heuristic, which is a modification to the Static Heuristic by adding some servers that are always available, and some that power off when the arrivals turn off or power on when more jobs arrive. The third heuristic is the Idle Heuristic, a straightforward method that depends on the number of waiting jobs in the queue. This heuristic turns off any idle server and turns on more servers if jobs are waiting in the queue. The fourth heuristic is the Threshold Heuristic, in contrast to the Semi-static Allocation Heuristic, which responded to changes in the behaviour of arrivals; this heuristic used a more straightforward mechanism to employ a threshold on the number of jobs in the queue to determine when to turn servers on or off. The fifth heuristic is the High/Low Heuristic, which depends on the current arrival period of the system and takes many factors into its decision of switching servers on or off. The Sixth heuristic, similar to the High/Low Heuristic but instead of using two types of arrival periods, this policy averages out the high and low arrival rates into one arrival rate.

In this chapter, only one heuristic, the High/Low Heuristic, was considered out of the six heuristics to be discussed in Section 3.2.1 and modelled using PEPA in Section 3.3.

3.2.1 The High/Low heuristic

The high/low heuristic bears comparison with the semi-static allocation heuristic, but in this instance, transition periods are included in the heuristic's determinations. The high/low heuristic also builds arrival periods, job accomplishment times and job queue length into its decision-making processes, and in consequence, is the most complex. Despite its inherent complexity, the high/low heuristic maintains its stability as a result of job arrival rates still being categorised in high λ_{high} or low λ_{low} binary form and the job time μ being maintained as a constant. The high/low heuristic is a complex application that incorporates system performance and stability into its determinations.

3.3 The High/Low policy model in PEPA

To meet the implementation environment requirements in PEPA, several modifications have been made to the original policy formulated in [42]. This modification was necessary as implementing the policy as described by [42] in PEPA necessitated computing resources that are not available to this project. Running a model in PEPA requires a machine with relatively large memory. This experiment has been completed on a personal computer with limited memory space. Thus, as a graceful compromise, a small-scale model was undertaken, sufficient to meet the experiment's requirements. The policy outlined in [42] examines numerous factors in the system, including the switching time, processing time, queue size and the arrival period. This policy depends on the arrival period and considers the switching time to determine how many servers should be switched on or off. Unlike [42] that used JAVA to simulate and model this policy, the model in this chapter uses PEPA. As a result, policy behaviour has been modified. Because some JAVA features helped implement this policy in [42], which are not available in PEPA. Some noteworthy features affected include flow control statements that help repeat the processes and conditional statements that facilitate flexible decision making. For this project's purpose, the experiment repeated different values for the arrival periods, the switching time between servers and the number of dynamic servers. Also, the job duration had a fixed rate of μ in each period. These policy modifications were conducted using the arrival period as an indicator to switch the servers on or off. So, there are two arrival periods with specific arrival rates assigned to each period that indicate the arrival job. Moreover, to control switching between periods, periods end rates indicate each period's duration. Figure 3.1 illustrates PEPA model for this policy.

The number of jobs in the system at the current time, i, is specified by the current state of Q_i. The maximum number of jobs is bounded at N. Arrivals into the system occur at either a high rate λ, by action *arrivalH,* or at a low rate ε (typically zero) by action *arrivalL*. The arrival stream switches between the high and low rate through the actions *highPeriodEnd* and *lowPeriodEnd* at rates β and γ respectively. Hence, a high period has a duration $1/\beta$ and the low period has a duration $1/\gamma$. Thus the

$$\begin{aligned}
Q_0 &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q1 + (arrivalL, \varepsilon).Q1, \\
Q_i &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q_{i+1} + (arrivalL, \varepsilon).Q_{i+1} + (service, \mu).Q_{i-1}, 1 \leq i < N, \\
Q_n &\stackrel{\text{def}}{=} (service, \mu).Q_{n-1}, \\
\\
Arrival_{high} &\stackrel{\text{def}}{=} (arrivalH, \lambda).Arrival_{high} + (highperiodEnd, \beta).Arrival_{low}, \\
Arrival_{low} &\stackrel{\text{def}}{=} (arrivalL, \varepsilon).Arrival_{Low} + (lowperiodEnd, \gamma).Arrival_{high}, \\
\\
ServerPowering_{On} &\stackrel{\text{def}}{=} (powerup, \eta * (1-\rho).Server_{On} + (powerup, \eta * \rho).Server_{failOn}, \\
Server_{On} &\stackrel{\text{def}}{=} (service, \mu).Server_{On} + (hperiodEnd, \beta).ServerPowering_{Off}, \\
ServerPowering_{Off} &\stackrel{\text{def}}{=} (poweroff, \xi * (1-\rho).Server_{Off} + (poweroff, \xi * \rho).Server_{failOff}, \\
Server_{Off} &\stackrel{\text{def}}{=} (lperiodEnd, \gamma).ServerPowering_{On}, \\
\\
Server_{failOn} &\stackrel{\text{def}}{=} (repair, \omega).Server_{On}, \\
Server_{failOff} &\stackrel{\text{def}}{=} (repair, \omega).Server_{Off}, \\
Server_{static} &\stackrel{\text{def}}{=} (service, \mu).Server_{static},
\end{aligned}$$

$$(Arrival_{high} \underset{\mathcal{K}}{\bowtie} Server_{On} \underset{\mathcal{K}}{\bowtie} ... \underset{\mathcal{K}}{\bowtie} Server_{On}) \underset{\phi}{\bowtie} Server_{static}[m] \underset{\mathcal{L}}{\bowtie} Q_0$$

$$\begin{aligned}
Where \;\; \mathcal{K} &= \{lowperiodEnd, highperiodEnd\} \\
and \;\; \mathcal{L} &= \{arrivalH, arrivalL, service\}
\end{aligned}$$

Figure 3.1: High/Low policy PEPA Model

average arrival rate is given as:

$$\overline{\lambda} = \frac{\lambda\gamma + \varepsilon\beta}{\gamma + \beta}$$

Jobs leave the system according to the service process, which is determined by the number of active servers. M servers are static and remain permanently available to serve the jobs. The remaining servers turn on and off in response to the high and low periods of arrivals. Thus, when a high period ends, these dynamic servers will become unavailable for service, but they will turn back on when a low period ends. Static servers and dynamic servers are distinguished in the PEPA model with two components, $Server_{static}$ and $Server_{On}$. A dynamic server's status is indicated by the component $Server_{On}$ or $Server_{Off}$, which is part of the model. A dynamic server may change state from $Server_{On}$ to $Server_{Off}$ and back again. The $Server_{static}$ component, on the other hand, refers to a server that is always available and cannot change state.

It is assumed that there is a delay in turning servers on and off (rates η and ξ respectively). Therefore when a high period begins, there will be a delay until the dynamic servers become available to serve the jobs. Suppose this delay is considerable, and the high arrival rate dramatically exceeds the service capacity of the static servers. In that case, there may be a significant increase in the number of waiting jobs in the system during this time. During the turning on and turning off periods, servers will consume power but not provide a service. Hence, if these periods are long, they would represent a potentially significant inefficiency in the system.

It is further assumed that servers may fail when switching on and off with probability p. Following failures, servers undergo repair at a rate ω. Typically, failures would be expressed by considering the metrics Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR) in continuous operation. However, in our model, we have only a probability that a server can fail. Moreover, in the PEPA model, as mentioned earlier, the failure only occurs during powering up or down the server. Thus, the MTBF would depend on the probability p and the frequency that a server switches on and off. Following failures, servers undergo a repair at a given rate. In this case, the repair rate is equivalent to 1/MTTR. Krasich [33] discussed, with examples, the uses of terms MTBF and MTTR in a variety of contexts.

We distinguish whether servers fail in the turning on or turning off state so that following repair, the server will return to the same state as other dynamic servers. It is assumed that all dynamic servers must begin to turn on or off simultaneously. Therefore a *highPeriodEnd* action may only occur if all the dynamic servers are on. Likewise, a *lowPeriodEnd* action may only occur when all the dynamic servers are off. Clearly, such synchronisation is not ideal. However, typically the switching rates of the arrival on and off periods are much longer than the switching periods needed to power servers on and off; if they were not, powering off would not be a sensible option. Hence, this synchronisation between the server state and the arrival state would not affect the average arrival rate unless the failure rate is high and the average repair time is excessive.

The problem associated with this model is to find the best number of static and dynamic servers needed to minimise the energy usage for a given set of parameters (arrival rates, service rate, switching rates, failure probability and repair rate). We will explore this problem in the next section.

3.4 Evaluation

Experiments are conducted based on several scenarios. These scenarios represent various data centre environments and preferences. The first scenario compares different combinations of dynamic and static servers against different high arrival rates. The second scenario compares the same combinations of servers applied in the first scenario against different durations of high arrival periods, where the arrival rate is high, but the durations are varied. The third scenario varies the switching time for powering the servers on or off. The fourth scenario considered variations in the costs of holding jobs in the queue and the benefits of saving energy by powering down servers while varying different ratios. Finally, the total fault in all server combinations was examined and presented to describe which combination could be the best in terms of stability.

Two costs are considered in this chapter, energy cost and performance cost. The probability that the queue is not empty represents the performance cost. The energy cost is the cost of energy consumed by all servers, except when they are powered down or in a repair state. Building the model in PEPA spared the use of a lot of the calculation involved in obtaining the queue size. The queue size in this chapter is limited to 16 jobs. The queue size is bounded because PEPA does not support unbounded queues. Moreover, the available computing resources can not handle more than 16 jobs in the queue before the model suffers from the state-space expansion, as mentioned previously in Section 2.8. The energy and performance cost formulas are presented below:

- Energy Cost:

$$C1 * (Server_{on} + ServerPowering_{on} + ServerPowering_{off} + Server_{static}$$

- Performance Cost:

$$C2 * (1 - Prob(Q_0))$$

$C1$ represents the weight assigned to energy, while $C2$ is the weight assigned to the performance. These two weights are used to indicate the relative costs that are used for trading off the merits of performance and energy usage. As such, where a data centre values performance over energy-saving, $C2$ will be assigned a higher value than $C1$. Rates $C1$ and $C2$ are arbitrary weights; individually, they are not necessarily proportional to anything, but they combine to give a relative performance-energy metric. For example, in a system where we are only interested in performance, $C2$ might equal one, and $C1$ might be zero. But in another situation, $C1$ and $C2$ can take any positive value that depends on how you look at the system.

The total of both costs (energy and performance) are used to represent the overall performance cost. The aim is to have an indicator for comparison between different combinations of servers. Where total costs are low in any server combination, it is said to have the lowest overall performance cost, making it a better preference for data centres.

The experiment was repeated for each combination of servers against each specified value of arrival rates, length of period and switching time. The findings from each combination of servers in each scenario were then compared to analyse their performance and energy saving.

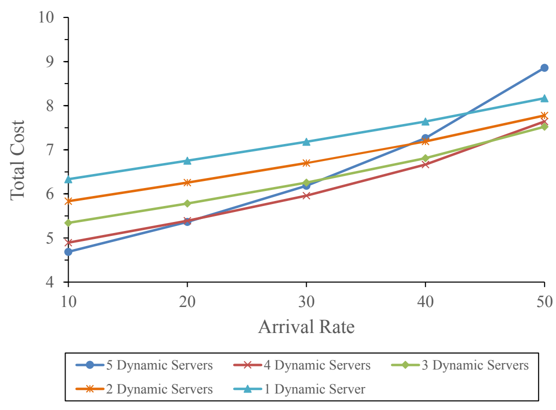


Figure 3.2: Total cost based on number of dynamic servers at different arrival rates

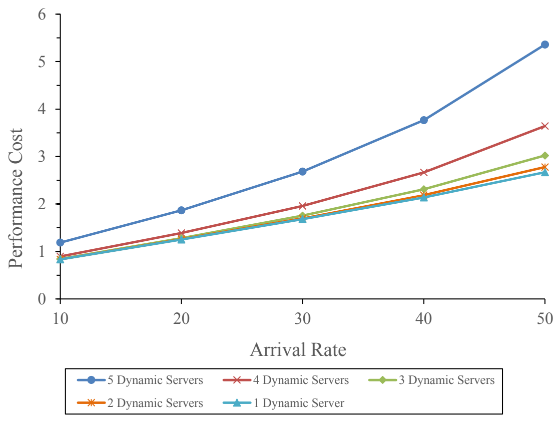


Figure 3.3: Performance cost based on number of dynamic Servers at different arrival rates

It's important to note that the figures in this chapter do not have precise units of measurement. They were scaled by the use of weights.

3.4.1 Increasing arrival rate

In this scenario, the experiment configuration and settings for the average request processing time is set to 12. The high arrival rate was increased from 10, representing the low arrival rate, to 50, which uses high server capacity (resources) at the highest arrival rate. The low arrival rate was set to 10. In addition, the high arrival time and low arrival time were set to 10, meaning both periods are neither too short nor too long. The holding job cost $C2$, and the benefit of servers being down $C1$ is set to 1. In this case, the data centre does not prioritise energy over performance and vice versa. The probability of failure during powering up or down servers was set up at 10%, which is sufficiently high for this experiment environment and system model to monitor the effect of servers' failure on the system. Finally, the rates of powering up and down servers were set to 100 for both states, which means switching servers on or turning them off will take a short period of time.

The experiment has been carried out by increasing the high arrival rate in a repetitive fashion on each combination of dynamic and static servers. Figure 3.2 shows the lowest total cost at the highest arrival rate of 50 is given 3 dynamic servers and 3 static servers. The lowest total cost at the arrival rate of 10 occurs where there are

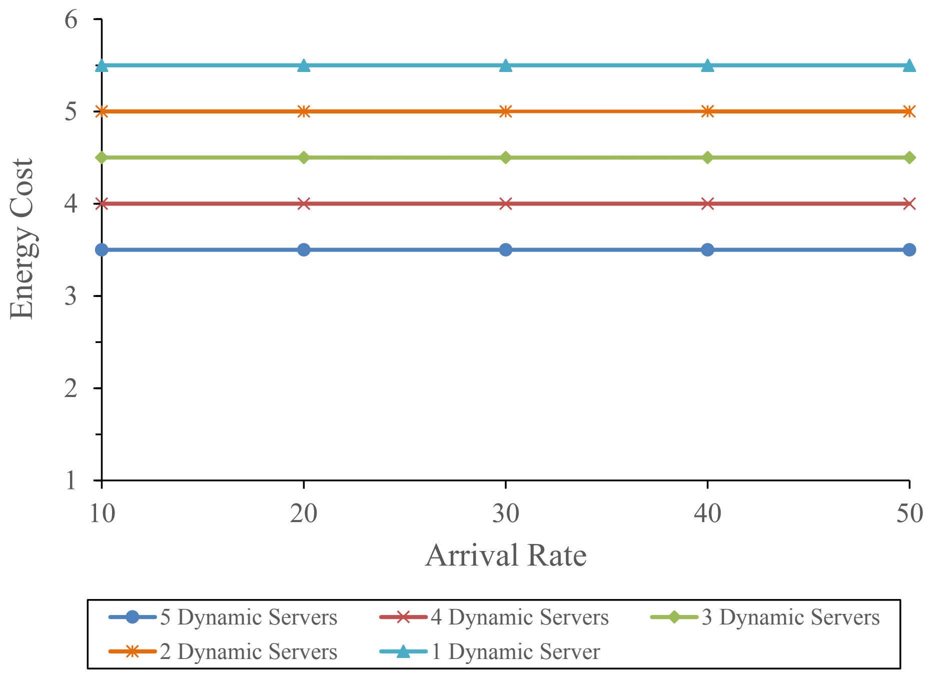


Figure 3.4: Energy cost based on number of dynamic servers at different arrival rates

5 dynamic servers and 1 static server. The reason for the 3 dynamic servers' and 3 static servers' better performance than the other server combinations in total cost can be attributed to the decline in the performance cost at a high arrival rate with this combination. From Figure 3.3 it can be observed that performance cost increases when there are more dynamic servers, say five, at a higher arrival rate of 50. The reason for that is the increase in waiting jobs in the queue. This is evident because, in this model, the decision to switch on additional servers to complete the job depends on the duration of the period. In fact, an investigation shows the 5 dynamic servers perform better in terms of energy saving in this experiment compared to other combinations of dynamic and static servers Figure 3.4. It is worth mentioning, in this scenario, the energy cost at all arrival rates for each server combination is the same because, as we mentioned earlier, the decision of switching on or off servers depends on the rate of period duration, not on the arrival rate. Section 3.4.2 takes an in-depth look at the impact of different duration of high arrival periods on the system.

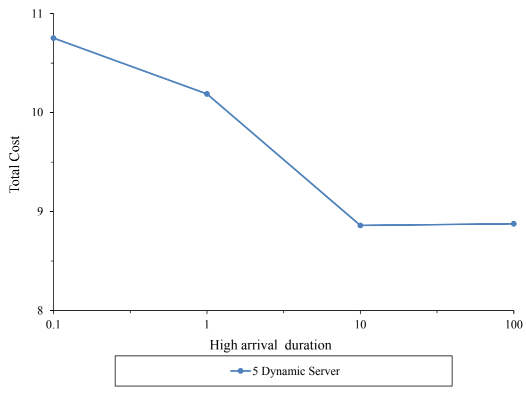


Figure 3.5: Total cost at different period length

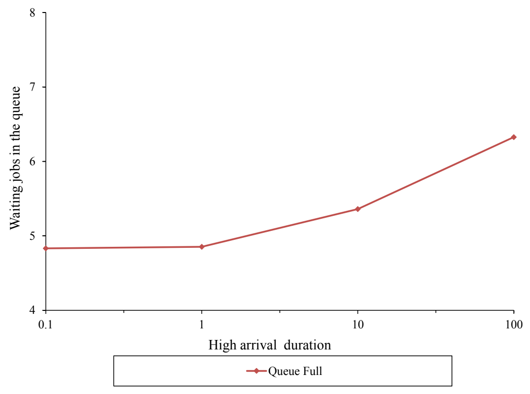


Figure 3.6: Queue length at different period length rates

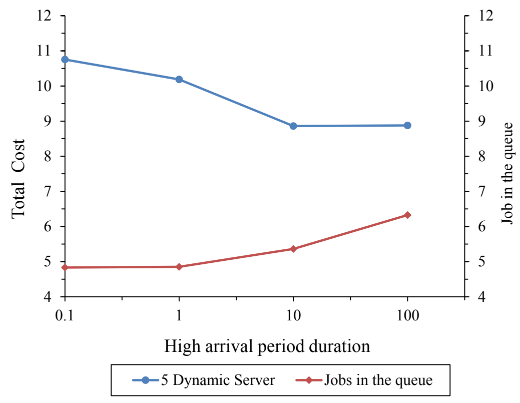


Figure 3.7: Total cost and queue length for different period rates

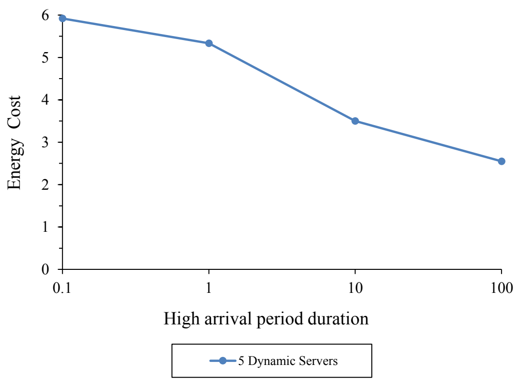


Figure 3.8: Energy cost for different high arrival period duration rates

3.4.2 Changes in period duration

In this experiment, the high arrival and low arrival rates were set at fixed values of 50 and 10, respectively. Six (N=6) servers were used in this model, with five turned on or off depending on the duration of the high arrival period, which the last is a static server that is always on to serve jobs in high and low arrival periods. With an average processing time of job requests of 12, the rates for powering up and down servers were set to 100 for both states. The probability of system failure during transition states were 10%, which can be considered acceptable for this small model.

In this scenario, the high arrival period duration rate is varied at 0.1, 1, 10, and 100. These rates represent a longer period at a low rate and a short and faster period at a bigger large rate. The objective of altering the duration is to show the impact of unstable durations of high arrivals. The idea is to analyse the impact of a short high

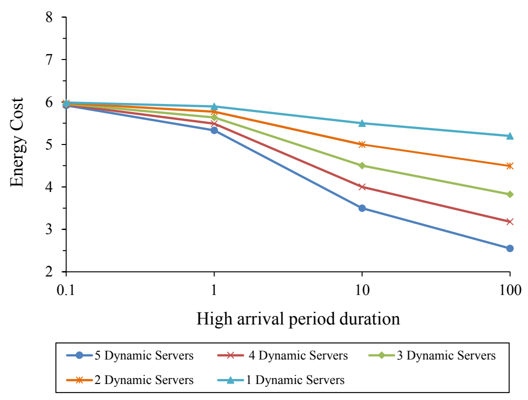


Figure 3.9: :Energy cost based on number of dynamic servers at different period length rates

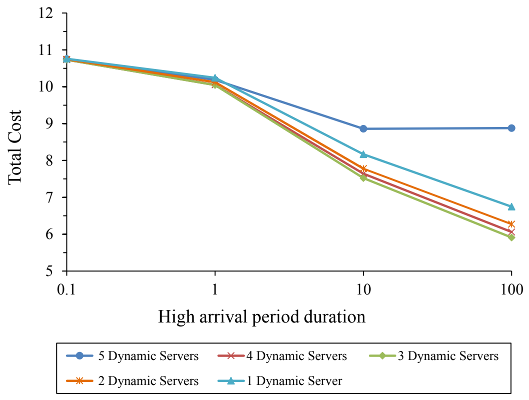


Figure 3.10: Total cost based on number of dynamic servers at different period rates

arrival period on system performance. For instance, a high arrival period could be reached before deciding whether to power on additional servers to serve the job. As a result, the waiting jobs in the queue will grow, leading to increased performance costs. The job holding cost $C2$ was considered in this scenario to be 1, and the energy benefits of powering down servers $C1$ as 1. This setup means the data centre does not prioritise energy saving over performance and vice versa.

Figure 3.5 shows that total cost decreases when the duration is shorter, e.g. 100, while Figure 3.6 shows that an increase in the job queue leads to a corresponding increase in performance cost. This increase in performance cost has not been reflected in the total cost due to the higher decrease in energy cost. Figure 3.7 combines the two preceding graphs to show the correlation between the factors. Figure 3.8 shows that when the arrival period is short, the system does not have sufficient time to switch on additional servers compared with a long arrival duration. This leads to a decrease in energy cost.

A further analysis was conducted by varying the combination of servers that are dynamic and static for each period length rate. The objective is to map out the impact of different durations with different server combinations to highlight the best energy and total cost savings. Figure 3.9 below indicates that at the longest high arrival period (e.g. 0.1), the energy cost for all server combinations is similar. However, at shorter periods, the energy cost becomes lower when there are more dynamic servers involved. Figure 3.10 below shows the total cost for the same settings, confirming that at the shortest duration, a combination of 1 static and 5 dynamic servers leads to higher total costs. However, this does not mean that having the least number of dynamic servers will reduce total cost. For example, 5 static servers and 1 dynamic have higher costs than 3 static and 3 dynamic servers. This is because the short period ceases before sufficient servers can be switched on to serve the jobs. So, the energy cost for 3 dynamic servers is much less than 5 static servers. Hence, the total cost with 5 static servers is more than the total cost with 3 dynamic servers and 3 static.

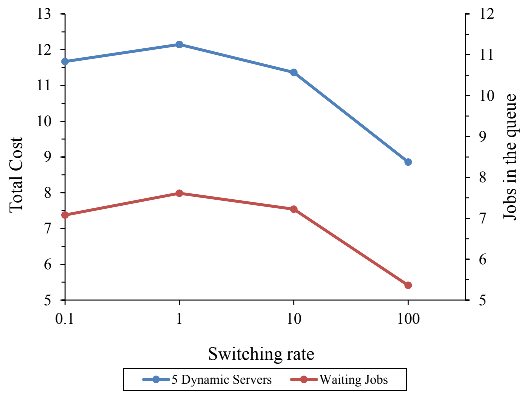


Figure 3.11: Total cost and waiting jobs for 5 dynamic servers at different switching time Rate

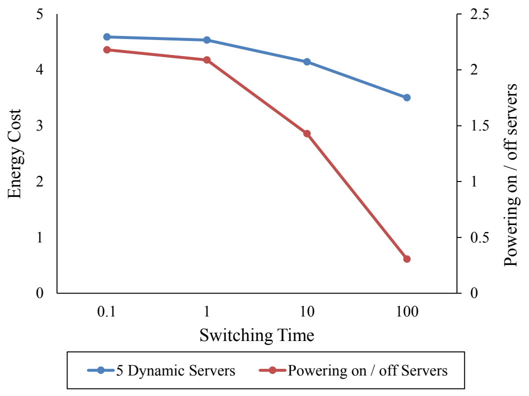


Figure 3.12: Energy cost and number of servers in powering on / off states at different switching time rates.

3.4.3 Changing the switching time

In this case, the experimentation was configured in the same way as in Section 3.4.1. However, instead of varying the high arrival periods, the switching time between having the servers on and off vary at 0.1, 1, 10, and 100. Here, the numbers are ordered from a slower to a faster switching time. The high arrival period duration was set to 10, which is neither too short nor too fast. For this configuration, we considered five dynamic and one static server because it is helpful to observe servers switching states (on or off) in relation to switching time. Hence, no other combinations of servers are used. At faster switching times, the overall performance cost is lower. Interestingly, Figure 3.11 shows that a long switching time leads to an increase in performance cost. This is because servers with slower switching times are unable to respond to more job requests, which causes a backlog of jobs waiting in the queue. Consequently, having more jobs in the queue increases the performance cost, which increases the total cost as a result. On the other hand, the performance cost is lowered with faster switching times as the server can be switched on and respond to job requests quickly, which avoids any delay that causes a queue backlog. The energy cost declines with a fast switching time, as shown in 3.12 below. Further investigation reveals that a fast switching time reduces the number of servers that are in powering on or off states, helping to lower energy costs.

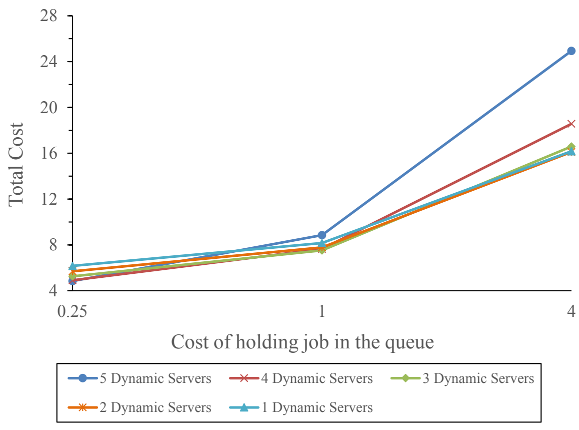


Figure 3.13: Changing cost difference

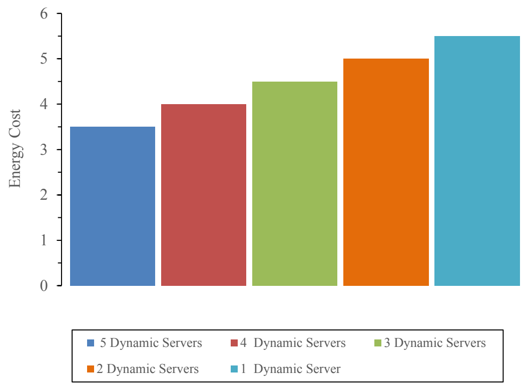


Figure 3.14: Energy cost for different server combinations

3.4.4 Changing the cost deference

In all previous scenarios in this chapter, the setup does not prioritise energy over processing the jobs in the queue. However, here we look at various data centre options for deciding between energy and performance. The same setup used in Section 3.4.1 is applied in this scenario for the arrival rate and powering on or off state rates. However, the period length was set to 10. In addition, the benefits of saving energy $C1$ is constant at 1 while the cost of holding a job in the queue or performance cost increased from 0.25, 1 and 4.

When the cost of holding a job is higher than the cost of saving energy, the total cost is higher with a larger number of dynamic servers. Figure 3.13 shows the combination of 5 dynamic servers and 1 static server performs worse than other combinations and even more poorly than a combination of 5 static servers and 1 dynamic. This is because having more dynamic servers during a period duration that is not too long leads to increased performance cost by having more waiting jobs in the queue, which increases the total cost. Even though there are energy savings when there are more dynamic servers used, Figure 3.14 shows the energy saved is not as big as the relative increase in performance cost; thus, energy cost becomes less profound in the total cost.

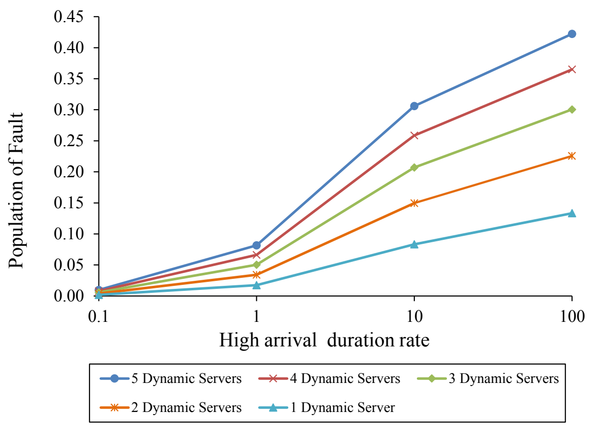


Figure 3.15: Fault rate at different duration rate

3.4.5 The fault rate

Fault rates are considered in order to determine data centre performance accurately due to the impact of switching state failures (on or off). In this scenario, the same setup as Section 3.4.2 was used, with a fault rate of 10%.

The fault population increases dramatically when the duration of the high arrival period is short. This can be understood because when the period length is short, more switching on requests is initiated in a short time. However, before servers are switched on, the period ends and sends a request to switch off servers. Hence, there is an increased risk of a server failure at this moment. This can be clearly seen when there are more servers set up to be dynamic servers.

3.4.6 Evaluation summary

The experiment results from each scenario demonstrate that no one combination performs well in all different scenarios. However, each has its own strengths and weaknesses. The combination of 5 static servers and one dynamic is the most stable combination because it does not involve a high level of switching. The combination of 5 dynamic servers and 1 static performs well in decreasing energy costs when the period is short, but very poorly in terms of performance as it increases the total cost by increasing the waiting job in the queue. However, this combination performs well when the switching time between servers is too fast. The combination of 3 dynamic servers and 3 static servers provides better overall performance than the other combinations with an increased arrival rate, giving us a mid-position in terms of energy saving.

There is clearly no one combination of servers that suits every condition, as each is adaptable in one scenario but not necessarily good in another. The performance of each combination is affected by many factors. These factors include the difference between arrival rates, length of period, probability of faults, and switching time. So, the choice of which combination to use for data centres depends on their preference for performance or energy saving. So, the operator of the data centre can configure the combination of servers according to the situation to have a balance between performance and energy consumption.

3.5 Conclusions and future work

In this chapter, a policy to limit power consumption by servers in data centres was discussed and modelled in PEPA. Numerical experiments were carried out under different scenarios. In each scenario, there was more than one combination of servers in order to find out which combination is better under each operating condition. The result of the conducted experiments shows that there is no one combination of servers under this policy that can perform well in all situations.

Our model in PEPA has some limitations. Firstly, the model depends only on the length of the arrival period to take the decision of switching on or off more servers. Secondly, we assumed all servers in any state consume the same amount of energy.

However, it is not true in practice as servers processing jobs consume more energy than servers in a powering-on or powering-off state. Moreover, not all servers in data centres are identical in hardware which means each server consume a different amount of energy. Finally, the experiment setup in some scenarios changed only one factor and fixed the others, which in practice may not be the situation. One particular scenario worth considering is where all servers are overloaded during short, intense high arrival periods, and then there are no arrivals for a very long time. In this situation, we would want to save power during the long low periods, but we would not want to switch off all the dynamic servers immediately as there would be a lengthy backlog of jobs to process.

Future work in this direction could look at an extended model to consider more servers with extended queue size. In addition, each server state could be assigned different energy costs to represent as close as possible the actual cost of energy consumption. Finally, it would be useful to change more than one factor at a time instead of changing only one factor with other factors fixed, as in some scenarios. Clearly, there are many other options for managing servers that remain to be explored using the approach presented in this chapter.

4

Modelling TAGS: Task Assignment Based on Guessing Size in homogeneous environment using PEPA

Contents

4.1 Introduction

In this chapter, the TAGS, the random allocation policy and the shortest queue strategy has been modelled in PEPA. An introduction to the PEPA language was given in Section 2.6, and a formal presentation of which can be found in [24]. In all cases, it can be assumed that jobs arrive into the system in a Poisson stream and receive a single service before leaving the system. Jobs are assumed to be independent and identically distributed. The system has been modelled under two service demand types: (i) exponential service demand and (ii) a two-phase hyper-exponential service demand. There are two queues, one per server. The queues are bounded with a maximum capacity of 10 jobs each, as PEPA does not support infinite queues [57]. As a result, queues can be full, leading to rejecting new arrival jobs from joining the queue. We assume that all queueing is First-Come-First-Served (FCFS).

There is no direct relationship between this chapter and Chapter 3. Having a single queue and multiple servers is the topic of discussion in Chapter 3, which is an attempt to answer the first research question (Question 1). In contrast, in Chapter 4 and the subsequent Chapters 5 and 6, we looked at individual queue server pairs and how we distribute work between queues in an attempt to address the second research question (Question 2). Therefore, experiments settings and parameters such as the number of servers and number and size of queues are distinct from those in Chapter 3.

The specification of PEPA models and parameters for each algorithm have been presented in more detail in the following sections.

Section 4.2 presents the shortest queue models under exponential and hyper exponential service demand. Section 4.3 shows the PEPA models for random allocation. Section 4.4 presents TAGS and its models in PEPA. Finally, Section 4.5 illustrates the timeout optimisation mechanism for TAGS.

4.2 Shortest queue model in PEPA

The shortest queue strategy overcomes the problem of load balancing between the queues. The shortest queue strategy is known to be optimal when the arrival and service distributions are negative exponential. However, this does not hold for hyper-exponential service, as we will see. When jobs arrive, the policy forwards them to the queue with the least waiting jobs, thus leading to no queue becoming full while other queues still have available capacity. So, the probability of losing jobs is less significant as long as the arrival rate does not exceed the system capacity. However, a short job might be delayed for a long time when getting stuck behind a long job, as this strategy only counts the waiting jobs and not their service demands. The shortest queue involves a management overhead as the policy has to have knowledge of the states of the queues. If the latency in polling queues is long, then this overhead might be significant, leading to poorer performance in practice. However, In the thesis, the aspect of management overhead has not been included.

4.2.1 The model under exponential service

The first section of the model represents the first queue $Q1$. Component $\mathit{Q1_0}$ symbolise the initial state of the queue. In that state, the queue is empty and waiting for a job to arrive. The *activity* $(\mathit{arrival1}, \top).\mathit{Q1_i}$ consist of action type $arrival1$ and rate T, the action will be carried out, and then the processed component will behave as the component $\mathit{Q1_i}$ which represents the following state of the queue. Component $\mathit{Q1_i}$ will have two different activities to choose of them, the first *activity* $(\mathit{arrival1}, \top).\mathit{Q1_i} + \mathit{1}$ which means the queue can do the action $arrrival1$ and then behave as $\mathit{Q1_i} + \mathit{1}$ which represents the following position in the queue. The second *activity* is $(\mathit{service1}, \top).\mathit{Q1_{i-1}}$ which means the queue can serve the job and return back to previous state. The component $\mathit{Q1_N}$ represents the state where the queue is full, and the only available action is to serve the job, and there is no possibility to receive a new job. When performing the activity $(\mathit{service1}, \top)$ then the queue will return to previous state $\mathit{Q1_{N-1}}$.

$$
\begin{aligned}
&\mathrm{Q1}_0 \overset{\mathrm{def}}{=} (\mathit{arrival1}, \top).\mathit{Q1_i},\\
&\mathit{Q1_i} \overset{\mathrm{def}}{=} (\mathit{arrival1}, \top).\mathit{Q1_{i+1}} + (\mathit{service1}, \top).\mathit{Q1_{i-1}}, \qquad 1 \le i \le N,\\
&\mathit{Q1_N} \overset{\mathrm{def}}{=} (\mathit{service1}, \top).\mathit{Q1_{N-1}},
\end{aligned}
$$

The following section represents the second queue $Q2$ in the model. It works in the same fashion as $Q1$.

$$\mathrm{Q2}_0 \overset{\text{def}}{=} (\mathit{arrival2}, \top).\mathit{Q2}_i,$$
$$\mathit{Q2}_i \overset{\text{def}}{=} (\mathit{arrival2}, \top).\mathit{Q2}_{i+1} + (\mathit{service2}, \top).\mathit{Q2}_{i-1}, \qquad 1 \leq i < N,$$
$$\mathit{Q2}_n \overset{\text{def}}{=} (\mathit{service2}, \top).\mathit{Q2}_{n-1},$$

The following specification represents the servers. The component $Server1$, has only one activity to perform action $service1$ at rate μ_1 then acts as $Server1$. $Server2$ components act in the same fashion, but with action $service2$ and rate μ_2.

$$\text{Server1} \overset{\text{def}}{=} (\mathit{service1}, \mu_1).\mathit{Server1}$$
$$\mathit{Server2} \overset{\text{def}}{=} (\mathit{service2}, \mu_2).\mathit{Server2}$$

The following section of the model represents the dispatcher.

$$S_0 \overset{\text{def}}{=} (\mathit{arrival}1,\lambda_1).\text{S1}+(\text{arrival2},\lambda_2).\mathit{S2}_1 + (\mathit{service1}, \mathit{u1}).\mathit{S2}_1 + (\mathit{service2}, \mathit{u1}).\mathit{S1}$$

$$\text{S1}_j \overset{\text{def}}{=} (\mathit{arrival2}, \lambda_1{+}\lambda_2).\mathit{S1}_{j-1}{+}(\mathit{service1}, \mu_1).\mathit{S1}_{j-1}{+}(\mathit{service2}, \mu_2).S_{j+1}, \; 1 \leq j < N$$
$$\mathit{S1}_N \overset{\text{def}}{=} (\mathit{arrival2}, \lambda_1 + \lambda_2).\mathit{S1}_{N-1} + (\mathit{service1}, \mu_1).\mathit{S1}_{N-1} + (\mathit{service2}, \mu_1).\mathit{S1}_N, \; N = 10$$

$$\text{S2}_j \overset{\text{def}}{=} (\mathit{arrival1}, \lambda_1{+}\lambda_2).\mathit{S2}_{j-1}{+}(\mathit{service1}, \mu_1).\mathit{S2}_{j+1}{+}(\mathit{service2}, \mathit{u1}).\mathit{S2}_{j-1}, 1 \leq j < N$$
$$\mathit{S2}_N \overset{\text{def}}{=} (\mathit{arrival1}, \lambda_1 + \lambda_2).\mathit{S2}_{N-1} + (\mathit{service1}, \mu_1)\text{S2}_N{+}(\text{service2},\mu_1).\mathit{S2}_{N-1} \quad N = 10$$

Finally, the system equation that shows the interaction between the model components.

$$\text{Q}_{10} \parallel \mathit{Q}_{2\,0} \underset{K}{\bowtie} \mathit{S0} \underset{L}{\bowtie} \mathit{Server1} \parallel \mathit{Server2}$$

4.2.2 The model under hyper exponential service

The previous model has been modified to support the hyper-exponential service distribution. At each service action, there is a probabilistic branch of the next service action to be performed as either $Server1$ or $Server1'$. $Server1$ represents a short service duration, and $Server1'$ represents a long service duration. $\alpha' = (1 - \alpha)$. The modified components have been highlighted in the model as shown in Figure 4.1

$$
\begin{aligned}
Q1_0 &\stackrel{\text{def}}{=} (arrival1, \top).Q1_i, \\
Q1_i &\stackrel{\text{def}}{=} (arrival1, \top).Q1_{i+1} + (service1, \top).Q1_{i-1}, \qquad 1 \le i \le N, \\
Q1_N &\stackrel{\text{def}}{=} (service1, \top).Q1_{N-1}, \\
\\
Q2_0 &\stackrel{\text{def}}{=} (arrival2, \top).Q2_i, \\
Q2_i &\stackrel{\text{def}}{=} (arrival2, \top).Q2_{i+1} + (service2, \top).Q2_{i-1}, \qquad 1 \le i < N, \\
Q2_n &\stackrel{\text{def}}{=} (service2, \top).Q2_{n-1}, \\
\\
Server1 &\stackrel{\text{def}}{=} \boxed{(\text{service1}, \alpha * \mu_1).Server1 + (service1, \alpha' * \mu_1).Server1'} \\
Server1' &\stackrel{\text{def}}{=} \boxed{(\text{service1}, \alpha * \mu_2).Server1 + (service1, \alpha' * \mu_2).Server1'} \\
Server2 &\stackrel{\text{def}}{=} \boxed{(\text{service2}, \alpha * \mu_1).Server2 + (service2, \alpha' * \mu_1).Server2'} \\
Server2' &\stackrel{\text{def}}{=} \boxed{(\text{service2}, \alpha * \mu_2).Server2 + (service2, \alpha' * \mu_2).Server2'} \\
\\
S0 &\stackrel{\text{def}}{=} (arrival1, \lambda_1).S1 + (arrival2, \lambda_2).S2_1 + (service1, u1).S2_1 + (service2, u1).S1 \\
\\
S1_j &\stackrel{\text{def}}{=} (arrival2, \lambda_1 + \lambda_2).S1_{j-1} + (service1, \mu_1).S1_{j-1} + (service2, \mu_2).S_{j+1} \quad 1 \le j < N \\
S1_N &\stackrel{\text{def}}{=} (arrival2, \lambda_1 + \lambda_2).S1_{N-1} + (service1, \mu_1).S1_{N-1} + (service2, \mu_1).S1_N \quad N = 10 \\
\\
S2_j &\stackrel{\text{def}}{=} (arrival1, \lambda_1 + \lambda_2).S2_{j-1} + (service1, \mu_1).S2_{j+1} + (service2, u1).S2_{j-1} \quad 1 \le j < N \\
S2_N &\stackrel{\text{def}}{=} (arrival1, \lambda_1 + \lambda_2).S2_{N-1} + (service1, \mu_1).S2_N + (service2, \mu_1).S2_{N-1} \quad N = 10
\end{aligned}
$$

$$
\mathrm{Q}_{10} \parallel Q_{2\,0} \underset{K}{\bowtie} S0 \underset{L}{\bowtie} Server1 \parallel Server2
$$

Where $K = (arrival1, arrival2, service1, service2)$ and $l = (service1, service2)$

Figure 4.1: Shortest Queue PEPA hyper exponential model in Homogeneous Environment

$$\begin{aligned}
Q1_0 &\stackrel{\text{def}}{=} (arrival1, \lambda_1).Q1_1 \\
Q1_j &\stackrel{\text{def}}{=} (arrival1, \lambda_1).Q1_{j+1} + (service1, \mu_1).Q1_{j-1}, \quad 1 \le j \le N \\
Q1_N &\stackrel{\text{def}}{=} (service1, \mu_1).Q1_{N-1}
\end{aligned}$$

$$\begin{aligned}
Q2_0 &\stackrel{\text{def}}{=} (arrival2, \lambda_2).Q2_1 \\
Q2_j &\stackrel{\text{def}}{=} (arrival2, \lambda_2).Q2_{j+1} + (service2, \mu_2).Q2_{j-1} \\
Q2_N &\stackrel{\text{def}}{=} (service2, \mu_2).Q2_{N-1}
\end{aligned}$$

$$\begin{aligned}
Server1 &\stackrel{\text{def}}{=} (service1, \mu_1).Server1 \\
Server2 &\stackrel{\text{def}}{=} (service2, \mu_2).Server2
\end{aligned}$$

$$Q_{1\,0} \parallel Q_{2\,0} \underset{K}{\bowtie} Server1 \parallel Server2$$

Where $K = (service1, service2)$

Figure 4.2: Random allocation PEPA model under exponential demand

4.3 Random allocation model in PEPA

The random allocation policy assigns arrival jobs to a queue randomly. So, it does not take into consideration how many jobs are already waiting in the queue. As a result, one queue might be overflowing with jobs while the other queue is empty or half full. Moreover, the probability of losing jobs is high, as sending a job to a full queue results in dropping that job permanently. Furthermore, short jobs might be delayed for a long time when getting stuck behind a long job, which is not detected by the random scheduler. However, random allocation can be an attractive option as no knowledge about the system is needed. Hence it is relatively trivial to implement.

4.3.1 The model under exponential demand

Queues are modelled in a similar way to the shortest queue. The difference between the random allocation and the shortest queue models is queues in the random allocation model are parallel. There is no dispatcher forwarding the job to the shortest of them. The full model is illustrated in Figure 4.2

4.3.2 The model under hyper exponential service

The components $Server1$ and $Server2$ modified to support hyper exponential demand service in the same way as in Section 4.2.2. Other component in the model stay unchanged the modification shown below:

$$\begin{aligned}
\text{Server1} &\stackrel{\text{def}}{=} (\mathit{service1}, \alpha * \mu_1).\mathit{Server1} + (\mathit{service1}, \alpha' * \mu_1).\mathit{Server1'} \\
\mathit{Server1'} &\stackrel{\text{def}}{=} (\mathit{service1}, \alpha * \mu_2).\mathit{Server1} + (\mathit{service1}, \alpha' * \mu_2).\mathit{Server1'}
\end{aligned}$$

$$\begin{aligned}
\text{Server2} &\stackrel{\text{def}}{=} (\mathit{service2}, \alpha * \mu_1).\mathit{Server2} + (\mathit{service2}, \alpha' * \mu_1).\mathit{Server2'} \\
\mathit{Server2'} &\stackrel{\text{def}}{=} (\mathit{service2}, \alpha * \mu_2).\mathit{Server2} + (\mathit{service2}, \alpha' * \mu_2).\mathit{Server2'}
\end{aligned}$$

4.4 Task Assignment Based on Guessing Size model

The TAGS scheme was initially introduced by Harchol-Balter [23] in order to address the problem of jobs with long service demands unduly delaying jobs with a short service. The main justification of this algorithm is to allocate jobs where the service demand is unknown before execution. In this approach, a job is sent to a single server queue. The server starts processing the first job in the queue until the job is completed and departed successfully or until a fixed time out is reached. If the timeout is reached before the job is completed, then the job is transferred to the next server. When the job arrives at the next server, the same steps are repeated, but the timeout for this level increases. The process is repeated with a longer timeout each time until the last server is approached. The job in this final stage receives an uninterrupted service until completion. It is assumed that there is no checkpointing, and so any service accrued at the previous stage must be repeated.

The main difference between TAGS and multi-level feedback queuing is that in the TAGS approach, the job is killed if it reaches the end of the time out period on a server. Afterwards, it is transferred to the next server and starts from the beginning. In contrast, in multi-level feedback queuing, the service resumes on the next server from where it stopped on the previous server. Thus, the effort is not lost, but the resource is consumed in recording the execution state, which can be significant. Figure 4.3 illustrates the concept of TAGS.

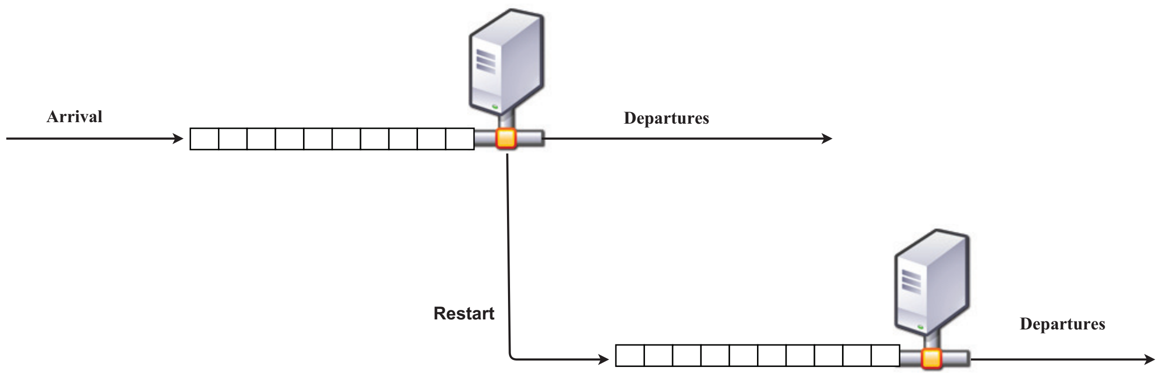


Figure 4.3: Jobs allocation flow in TAGS

When compared to the random allocation and the shortest queue, TAGS can overcome the problem of short jobs getting stuck behind a long job. However, there is an overhead in the repeated service, which can affect performance. For example, consider a system consisting of 2 servers with their bounded queues. If the timeout is too short at the first server, then too many jobs will be killed and transferred to repeat the service from scratch in the subsequent server. As a result, the jobs will be delayed leading to degrading the performance (increasing the response time).

Furthermore, the next queue might become full, leading to an increase in job losses. On the contrary, if the time out is too long, most jobs will be served at the first server, and fewer jobs will be transferred to the second server. However, the possibility of losing jobs in the first queue will increase as many short jobs will be stuck behind longer jobs, and the second server will be underutilised. As a consequence, the throughput of the system will be decreased. So, optimising the time out value is crucial to have the most advantages of TAGS.

Thomas [56] studied and modelled TAGS in PEPA in a homogeneous environment and showed that TAGS could perform well for various performance metrics compared to random allocation and the shortest queue strategy when the job size variability is high; however, the timeout values need to be optimised.

TAGS mechanism indicates that there is an energy overhead as repeating the same job from scratch means the processing time will become longer. As we discussed in Section 2.2 the energy has linear relationship with the processing time. So, the energy consumption for the same job in TAGS compared to the shortest queue random allocation will be more. We investigate the performance and energy in Chapter 5 in a homogeneous environment and in Chapter 6 in a heterogeneous environment.

Thomas's study [56] has been extended to consider the energy consumption level and performance using the TAGS scheme. The energy consumed were compared with this consumed by the random allocation and the shortest queues strategies.

In this thesis, the models and analysis have been extended to consider a heterogeneous environment. Chapter 5 shows the results in a homogeneous environment while Chapter 6 describes the changes in the PEPA models to add the system heterogeneity. In addition, it shows the results of performance and energy in a heterogeneous environment.

4.4.1 TAGS model in PEPA under exponential service demand

As PEPA is a Markovian Process Algebra, the deterministic timeout used in TAGS is modelled by an Erlang distribution. The service distribution is considered as either a negative exponential (hence relatively low variance) or a two-phase hyper-exponential (relatively high variance). In all cases, the inter-arrival periods are negative exponentially distributed, and the maximum queue lengths are finite. PEPA does not support unbounded queues. Therefore, we have to use a finite queue. The model specification can be generalised or extended by adding more components. Hence, the queue capacity can be extended to any number of jobs. However, the constraint in this model evaluation is related to the available computing resource to generate and solve the PEPA model's steady-state, as stated previously in Section 2.8.

The notations of the PEPA model have been summarised in Table 4.1. Figure 4.4 illustrates the TAGS model in PEPA under exponential demand while Figure 4.5 illustrate the model under hyper-exponential demand.

For numerical tractability and ease of understanding, the number of nodes is restricted to two, as this is sufficient to investigate the consequences of using the TAGS scheme on energy consumption. The queue size is bounded, and hence a job can be lost at arrival by being dropped from the first node or at the subsequent node after completing

Notation	Meaning
$Q1$	the first queue.
$Server1$	the first server.
$Timer1$	the first timer which governs the decision to terminate the job at the first server after a specific processing time.
$Timer2$	the timer in the second server used to model the repeated service. the remaining part of the job receives service from action $service2$ until completion.
$Q2$	the second queue.
$K1$	the maximum length of the first queue.
$K2$	the maximum length of the second queue.
$arrival$	the arrival process.
$service1$	the service process at first server.
$service2$	the service process at second server.
$timeout$	the timeout action which kills the job at $server1$ when triggered.
$tick1$	the tick action of the timeout clock.
$repeateservice$	the repeat service action, that repeats the amount of service that timed out previously in $server1$.
λ	the job arrival rate.
μ	the average service rate.
$\mu1$	The service rate at $Server1$
$\mu2$	The service rate at $Server2$
t	The timeout rate
α	The proportion of short job.
α'	The proportion of long job.
σ'	The difference in performance between $Server1$ and $Server2$.

Table 4.1: PEPA Model Notation

a timeout service. Thus, a proportion of jobs might be lost from the second node if the load is high and the timeout at the first node is too short. In contrast, a long timeout at the first node will increase the probability of losing a job at the first node as the queue becomes full, rejecting new arrivals from joining the queue.

The queues are modelled in such a way that each job is represented as a separately named derivative of the queue. The timeout at the first node is modelled using an Erlang distribution, and the number of ticks is fixed. While the queue is not empty, the timeout clock starts at the beginning of each derivative of the queue. This is done by introducing the *tick* action at each derivative. A race exists between the timeout action and the service process $service1$. If the timeout action wins, the job is killed and transferred to the second node to restart the service from the beginning. Otherwise, the task departs the system since it is finished before the timeout action is triggered.

In both situations, the timeout clock is reset. If a job is waiting in the queue, the race starts again; otherwise, if the queue is empty, the server enters an idle state until a new job arrives.

After timing out, the job restarts at the second node and receives a repeated process of the amount of service (the same number of ticks) that timed out in the first node. To overcome the resampling problem, this is represented by introducing the *repeatservice* action in *timer*2, while the remaining part of the job receives service from action *service*2.

$$\begin{aligned}
Q1_0 &\overset{\text{def}}{=} (arrival, \lambda).Q1_i;\\
Q1_i &\overset{\text{def}}{=} (arrival, \lambda).Q1_{i+1} + (service1, \top).Q1_{i-1}\\
&\quad +(tick1, \top).Q1_i + (timeout, \top).Q1_{i-1}; \quad 1 \le i < K1\\
Q1_n &\overset{\text{def}}{=} (service1, \top).Q1_{n-1} + (tick1, \top).Q1_n + (timeout, \top).Q1_{n-1}\\
\\
Server1 &\overset{\text{def}}{=} (service1, \mu).Server1;\\
\\
Timer1_0 &\overset{\text{def}}{=} (timeout, t).Timer1_n + (service1, \top).Timer1_n\\
Timer1_i &\overset{\text{def}}{=} (tick1, t).Timer1_{i-1} + (service1, \top).Timer1_n \quad 1 \le i \le n\\
\\
Q2_0 &\overset{\text{def}}{=} (timeout, \top).Q2_i\\
Q2_i &\overset{\text{def}}{=} (timeout, \top).Q2_{i+1} + (tick2, \top).Q2_i\\
&\quad +(repeatservice, \top).Q2'_i, \qquad\qquad 1 \le i < K2\\
Q2_{K2} &\overset{\text{def}}{=} (timeout, \top).Q2_{K2} + (tick2, \top).Q2_{K2}\\
&\quad +(repeatservice, \top).Q2'_{K2},\\
Q2'i &\overset{\text{def}}{=} (timeout, \top).Q2'_{i+1} + (service2, \top).Q2_{i-1}, \quad 1 \le i < K2\\
Q2'_{K2} &\overset{\text{def}}{=} (timeout, \top).Q2'_{K2} + (service2, \top).Q2'_{K2-1},\\
\\
Timer2_0 &\overset{\text{def}}{=} (repeatservice, t).(service2, \mu).Timer2_n\\
Timer2_i &\overset{\text{def}}{=} (tick2, t).Timer2_{i-1}, \quad 1 \le i \le n
\end{aligned}$$

$$((\mathrm{Q1}_0 \underset{service1}{\bowtie} Server1) \underset{K}{\bowtie} Timer1_n) \underset{timeout}{\bowtie} (Queue2_0 \underset{L}{\bowtie} Timer2_n)$$

where $K = (service1, timeout, tick1)$ and $L = (repeatservice, service2, tick2)$

Figure 4.4: A PEPA TAGS exponential model

4.4.2 TAGS model in PEPA under hyper-exponential service demand

When considering TAGS, the exponential distribution is not the most interesting to use, as the main motivation for TAGS is to enable the throughput of short jobs in the presence of long-running jobs. Hence TAGS will perform best with a mixed workload where there are lots of short jobs and a few very long-running jobs. Modelling the hyper-exponential distribution in PEPA involves the implementation of some extra factors to produce the required probabilistic branching. Each *timeout* and *service*1 action must, therefore, take place twice with rates multiplied by α and (1- α) to determine whether the next job will be served at the appropriate rate, $\mu1$ or $\mu2$ (in $Server1$ and $Server1'$ respectively). In the second node, the branching process is less complex, with the branching taking place at the *repeatservice* action. Clearly, the probability that a short job time out will be less than the probability that a long job times out, which necessitates computing the resultant probability α'. Figure 4.5 shows the TAGS model after the H2 distribution support change has been implemented.

$$
\begin{aligned}
Q1_0 &\stackrel{\text{def}}{=} (arrival, \lambda).Q1_i;\\
Q1_i &\stackrel{\text{def}}{=} (arrival, \lambda).Q1_{i+1} + (service1, \top).Q1_{i-1}\\
&\quad +(tick1, \top).Q1_i + (timeout, \top).Q1_{i-1}; \qquad 1 \le i < K1\\
Q1_n &\stackrel{\text{def}}{=} (service1, \top).Q1_{n-1} + (tick1, \top).Q1_n + (timeout, \top).Q1_{n-1}\\
\\
Server1 &\stackrel{\text{def}}{=} (service1, \alpha * \mu_1).Server1 + (service1, (1-\alpha) * \mu_1).Server1'\\
&\quad +(timeout, \alpha * t).Server1 + (timeout, (1-\alpha) * t).Server1'\\
&\quad +(tick1, t).Server1\\
\\
Server1' &\stackrel{\text{def}}{=} (service1, \alpha * \mu_2).Server1 + (service1, (1-\alpha) * \mu_2).Server1'\\
&\quad +(timeout, \alpha * t).Server1 + (timeout, (1-\alpha) * t).Server1'\\
&\quad +(tick1, t).Server1'\\
Timer1_0 &\stackrel{\text{def}}{=} (timeout, t).Timer1_n + (service1, \top).Timer1_n\\
Timer1_i &\stackrel{\text{def}}{=} (tick1, t).Timer1_{i-1} + (service1, \top).Timer1_n \quad 1 \le i \le n\\
\\
Q2_0 &\stackrel{\text{def}}{=} (timeout, \top).Q2_i\\
Q2_i &\stackrel{\text{def}}{=} (timeout, \top).Q2_{i+1} + (tick2, \top).Q2_i\\
&\quad +(repeatservice, \top).Q2'_i, \qquad\qquad 1 \le i < K2\\
Q2_{K2} &\stackrel{\text{def}}{=} (timeout, \top).Q2_{K2} + (tick2, \top).Q2_{K2} + (repeatservice, \top).Q2'_{K2},\\
\\
Q2'i &\stackrel{\text{def}}{=} (timeout, \top).Q2'_{i+1} + (service2, \top).Q2_{i-1}, \quad 1 \le i < K2\\
Q2'_{K2} &\stackrel{\text{def}}{=} (timeout, \top).Q2'_{K2} + (service2, \top).Q2'_{K2-1}\\
\\
Timer2_0 &\stackrel{\text{def}}{=} (repeatservice, \alpha' * t).(service2, \mu_1).Timer2_5\\
&\quad +(repeatservice, (1-\alpha') * t).(service2, \mu_2).Timer2_5\\
Timer2_5 &\stackrel{\text{def}}{=} (tick2, t).Timer2_4\\
Timer2_4 &\stackrel{\text{def}}{=} (tick2, t).Timer2_3\\
Timer2_3 &\stackrel{\text{def}}{=} (tick2, t).Timer2_2\\
Timer2_2 &\stackrel{\text{def}}{=} (tick2, t).Timer2_1\\
Timer2_1 &\stackrel{\text{def}}{=} (tick2, t).Timer2_0
\end{aligned}
$$

$$((\mathrm{Q1}_0 \underset{K}{\bowtie} Server1) \underset{K}{\bowtie} Timer1_5) \underset{timeout}{\bowtie} (Queue2_0 \underset{L}{\bowtie} Timer2_5)$$

where $\boxed{K = (service1, timeout, tick1) \text{ and } L = (repeatservice, service2, tick2)}$

Figure 4.5: A PEPA TAGS hyper exponential model

4.5 TAGS timeout optimisation

In contrast to random allocation and shortest queue, TAGS involved time out factor. So to have a standardised comparison between the three algorithms, the timeout rate values for the TAGS algorithm have to be optimised to obtain the maximum throughput for each arrival rate. It is worth mentioning the optimisation in this context refers

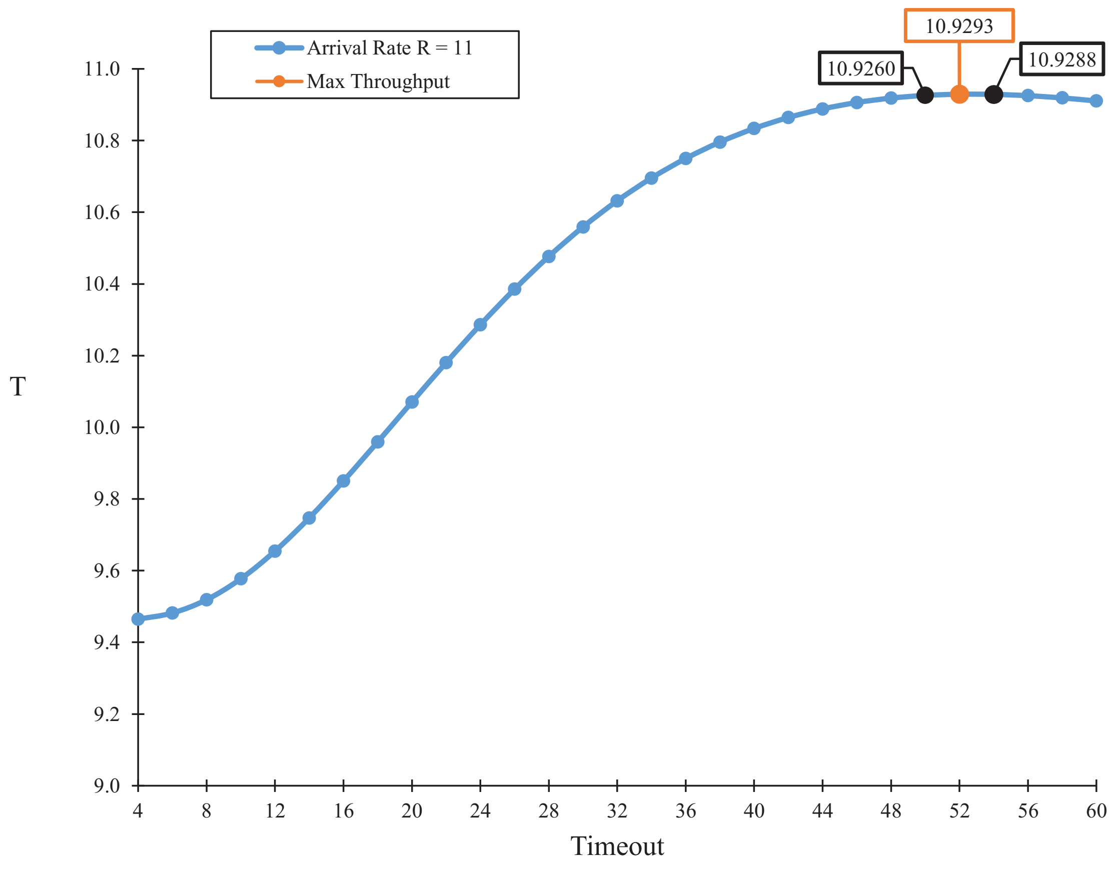


Figure 4.6: Throughput varied against timeout rate t when the arrival rate λ =11, and the average service μ= 10 for exponential service demand in homogeneous environment

to the best timeout rate achieved, producing the maximum throughput under the experiment setup. Consequently, this cannot be generalised for different experiment setups or environments.

Subsequently, the maximum throughput is used as a comparison factor in all following chapters. For example, if the timeout value t that gives the maximum throughput is t =8 for the arrival rate λ =12. Consequently, when we compare the three algorithms for performance or energy for the arrival rate 12, we considered the TAGS throughput when the timeout rate is 8 for the arrival rate λ =12.

In Figure 4.6 The timeout optimisation process for TAGS to obtain the maximum throughput has been presented. It can be seen timeout rates varied from 4 to 60 for the arrival rate $\lambda = 11$. The timeout rate that gives the highest throughput is $t = 52$. So, for any comparison with other algorithms at arrival rate $\lambda = 11$ we considered the throughput when the timeout rate $t = 52$.

4.6 Conclusion

This chapter illustrated the PEPA models of TAGS and the shortest queue and random allocation Schemes. The three algorithms in models in PEPA are presented under two service demand distributions. These service demands are: (i) the exponential service demand, and (ii) the two-phase hyper exponential demand.

TAGS mechanism relies on the timeout factor in contrast to the other two policies. Therefore, a timeout rate optimisation mechanism for TAGS is expected before analysing or comparing the three schemes. We showed how to optimise time out values for TAGS that produce the highest throughput.

In the next chapter, Chapter 5 will present the results of the performance and energy consumption of this chapter PEPA model in a homogeneous environment.

5

TAGS Performance and Energy Consumption in Homogeneous environment

Contents

5.1 Introduction

In Chapter 3, we studied one energy policy and modelled it in PEPA. Subsequently, we analysed it in different scenarios regarding its energy consumption and performance. In contrast, in Chapter 4, we discussed and presented some scheduling algorithms introduced to improve the system's performance. These algorithms have been modelled in PEPA under two service demand types: (i) exponential service demand and (ii) two-phase hyper-exponential service demand.

The primary goal of this chapter is to investigate each of the algorithms described in Chapter 4 in terms of performance and energy consumption and to produce comparable results. Both chapters, Chapter 3 and this chapter evaluated a homogeneous environment. The main difference between this chapter and Chapter 3 could be summarised as the chapter 3 focused on energy policies and their impact on performance. In contrast, this chapter focused on some performance scheduling algorithms and compared them in terms of their performance's potential benefits and their energy consumption. It is essential to emphasise that this chapter's experiment setting and technique are distinct from those in Chapter 3. As a result, the queue sizes, number of servers and arrival and service rates are not comparable to the experiment described in Chapter 3. In summary, the experiments described in this chapter and Chapter 6 are separate from the experiment described in Chapter 3, as they are trying to answer the research question 2.

This chapter, along with Chapters 3 and 6, make use of performance metrics such as throughput, utilisation, and average response time. However, there are three primary metrics in terms of energy consumption: cost (discussed in Chapter 3), total energy consumption and energy per job (discussed in this chapter and Chapter 6). Chapter 3's energy cost is a relative measure for energy consumption. By contrast, we introduce the energy consumption metric associated with the P-state value in Chapters 5 and 6. Thus, the energy measurements are expressed in terms of Watt per time unit. Additionally, energy per job was included to link energy usage with throughput and serve as a benchmark for trading offs performance and energy consumption and comparing algorithms. Section 5.2, presents the performance analysis. Section 5.3 illustrates the energy model and analysis.

5.2 Performance analysis

This section represents the performance evaluation of TAGS, shortest queue and random allocation under two service demand types: (i) exponential service demand in Section 5.2.1 and (ii) two-phase hyper-exponential service demand in Section 5.2.2. The performance metrics we have been studied are the throughput of the system and job loss. Furthermore, we looked at the utilisation of servers as well as the average response time.

5.2.1 Performance analysis under exponential service demand

The TAGS model specified in Figure 4.4 represents the case of exponential demand analysed to obtain the throughput and the average response time.

Figure 5.1 shows the throughput varied against arrival rate λ. The TAGS algorithm is optimised for the maximum throughput, the best values of t being 72, 62, 56, 54,52, 54, 54 and 54 (for λ = 19, 17, 15, 13, 11, 9, 7 and 5, respectively). The random allocation and the shortest queue strategies results are included for contrast as well. Figure 5.4 shows the utilisation of both servers.

Figure 5.3 shows the job loss at the shortest queue strategy is nearly insignificant at all arrival rates (0.5 at the highest arrival rate and 10^{-12} at the lowest arrival). At the same time, the random assignment is slightly higher at high arrival rates, and the TAGS is the worst when the arrival rate $\lambda > 11$. In addition, Figure 5.3 shows that the average response time by TAGS is the worst in all arrival rates. These findings indicate that TAGS is not very useful compared to the random and shortest queue strategies under exponential service demand, notably as the service demand increases resulting in an increase in the incomplete jobs rate in TAGS. This is expected as it is well known that the optimal strategy for exponential arrivals and service demands is the shortest queue. Winston [62] has shown join the shortest queue is optimal when the queue size

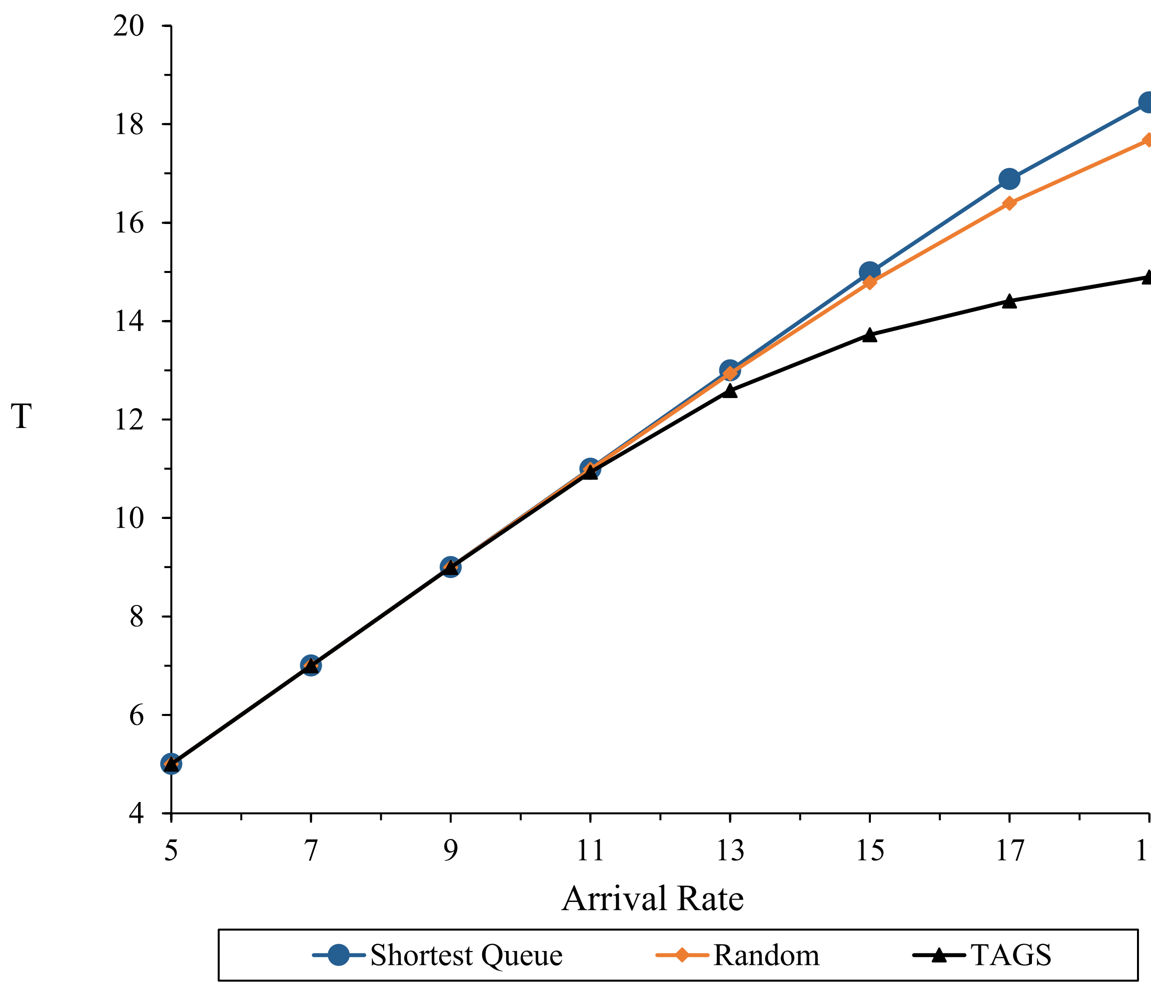


Figure 5.1: Throughput varied against arrival rate λ, when the average service μ= 10 for exponential service demand in homogeneous environment.

is finite, and jobs are exponentially distributed in a homogeneous environment and servers serving jobs on FCFS basis.

Therefore, analysing these algorithms under a service demand with higher variance would give better results for TAGS. Section 5.2.2 will investigate the system under a two-phase hyper exponential service demand.

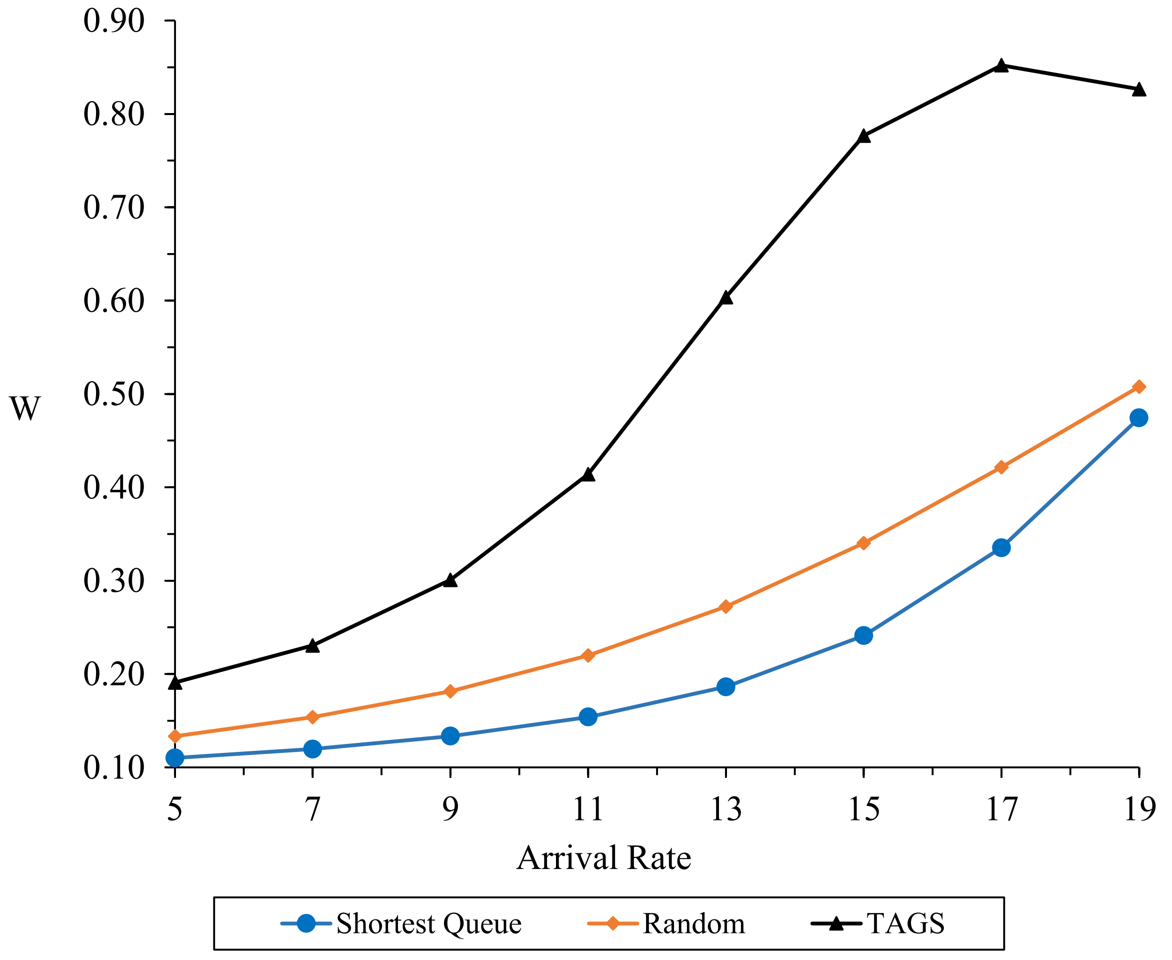


Figure 5.2: Average response time, varied against arrival rate λ, when the average service μ= 10 for exponential service demand in homogeneous environment.

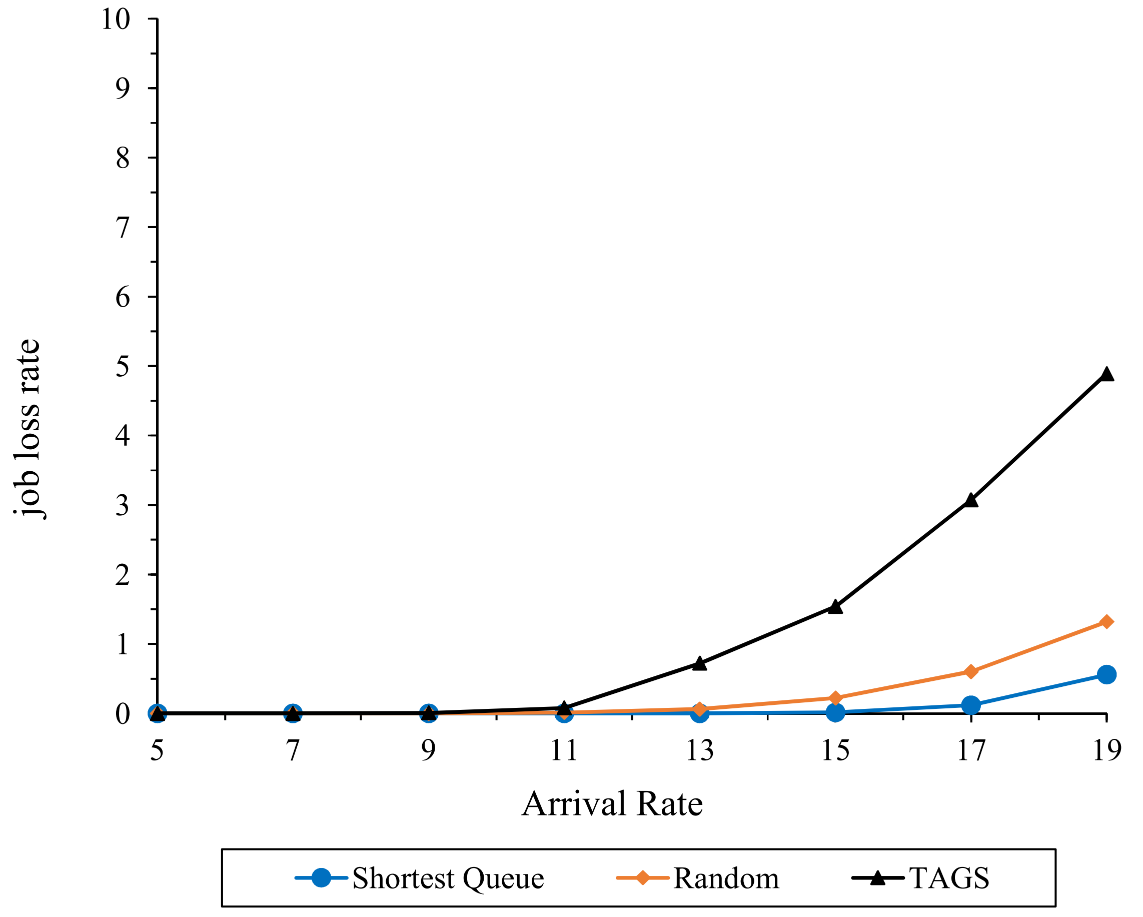
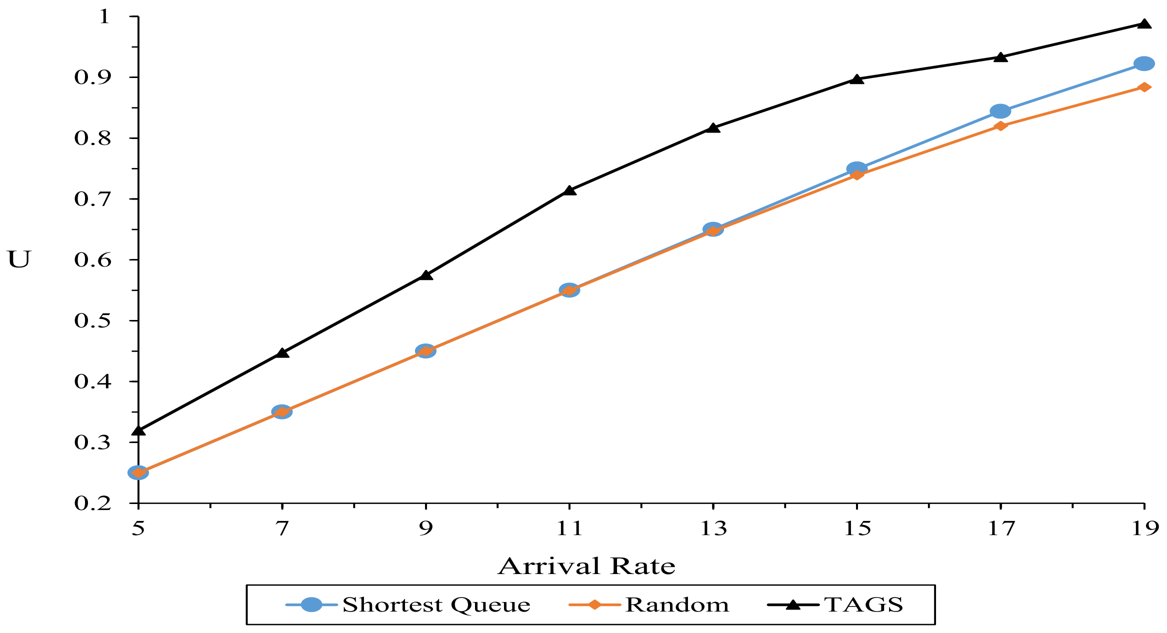
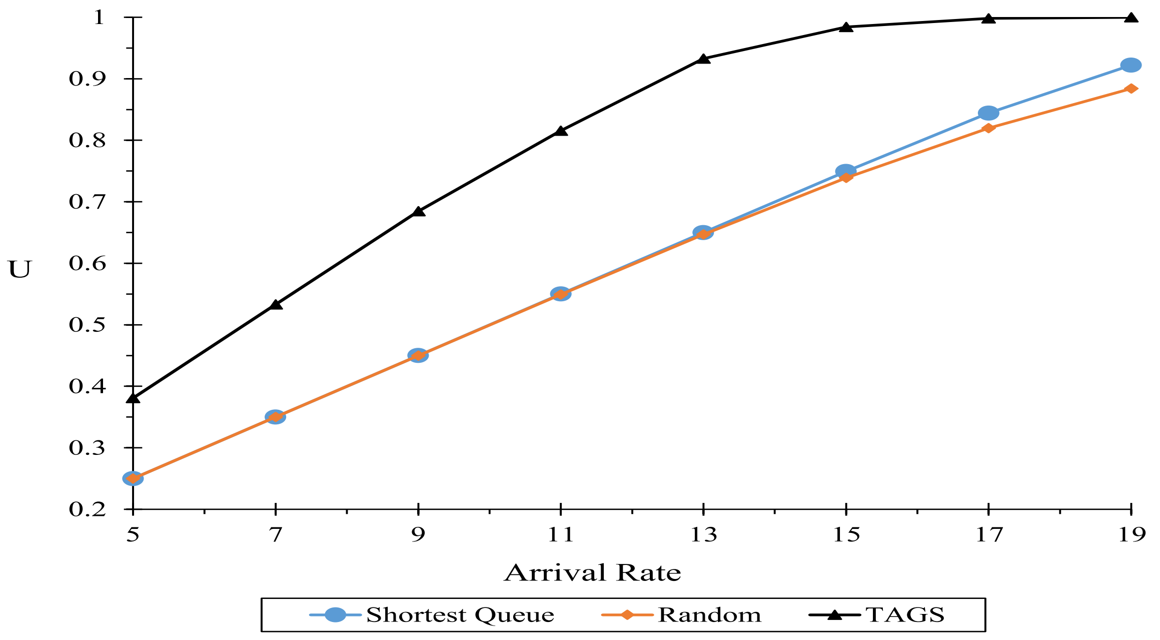


Figure 5.3: Job loss, varied against arrival rate λ, when the average service μ= 10 for exponential service demand in homogeneous environment.



(a) Server1



(b) Server2

Figure 5.4: (a) $server_1$ utilisation and (b) $server_2$ utilisation, varied against arrival rate λ, when the average service μ= 10 for exponential service demand in homogeneous environment.

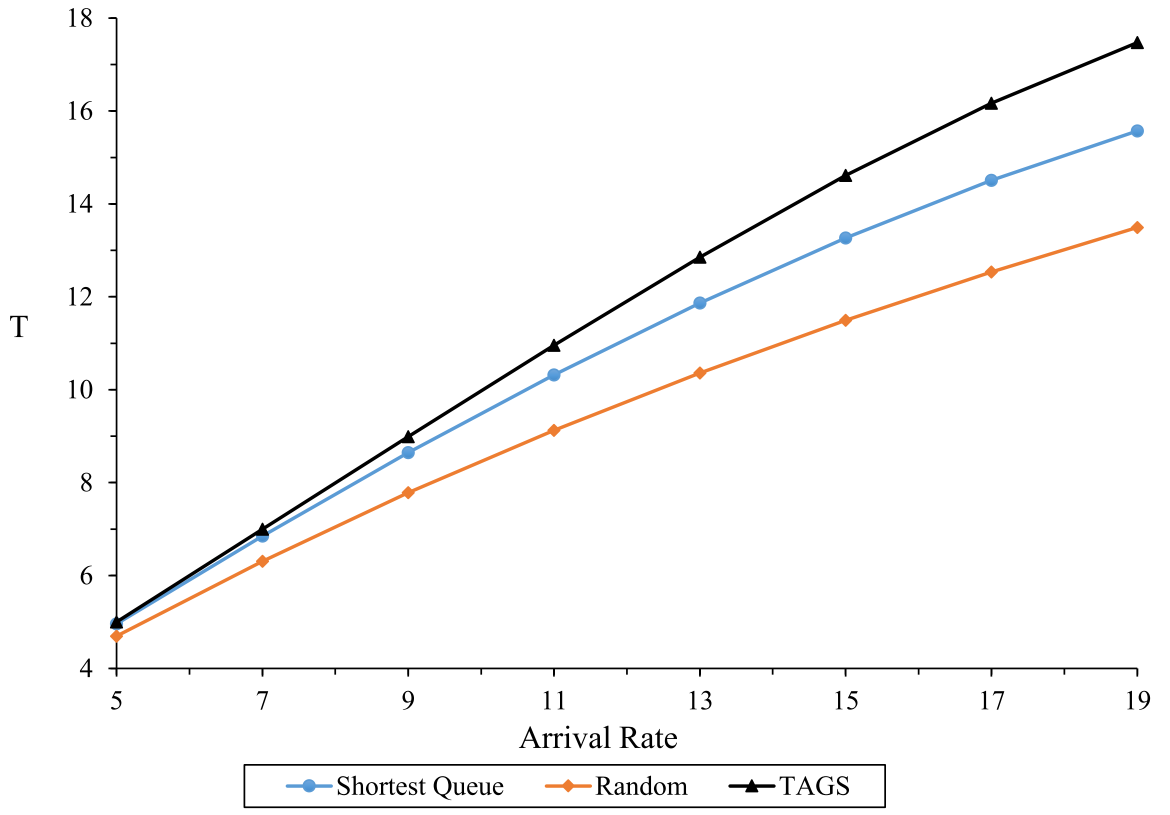


Figure 5.5: Throughput varied against arrival rate λ, when the average service μ= 10, and proportion of short job α = 0.99, and the service rate μ_1=100μ_2.

5.2.2 *Performance analysis under hyper-Exponential service demand*

Hyper-exponential distribution has a greater variance than exponential demand, which makes it an appropriate distribution to investigate the performance metrics for TAGS. Figure 5.5 shows the throughput varied against arrival rate when service demand has an H_2 distribution. Results are presented for TAGS, shortest queue and random allocation. The proportion of short jobs was set to α= 0.99 and the average service to $\mu = 10$. The long job size was set to be 100 times longer than the short job by specifying the service rate μ_2 at the second node to be 100 times less than the service rate at the first node: $\mu_1 = 100\mu_2$.

TAGS outperforms the shortest queue and random allocation when service demand increases and load variability is high. The explanation of why TAGS is better than the shortest queue is easy to clarify. The shortest queue strategy will lose jobs when

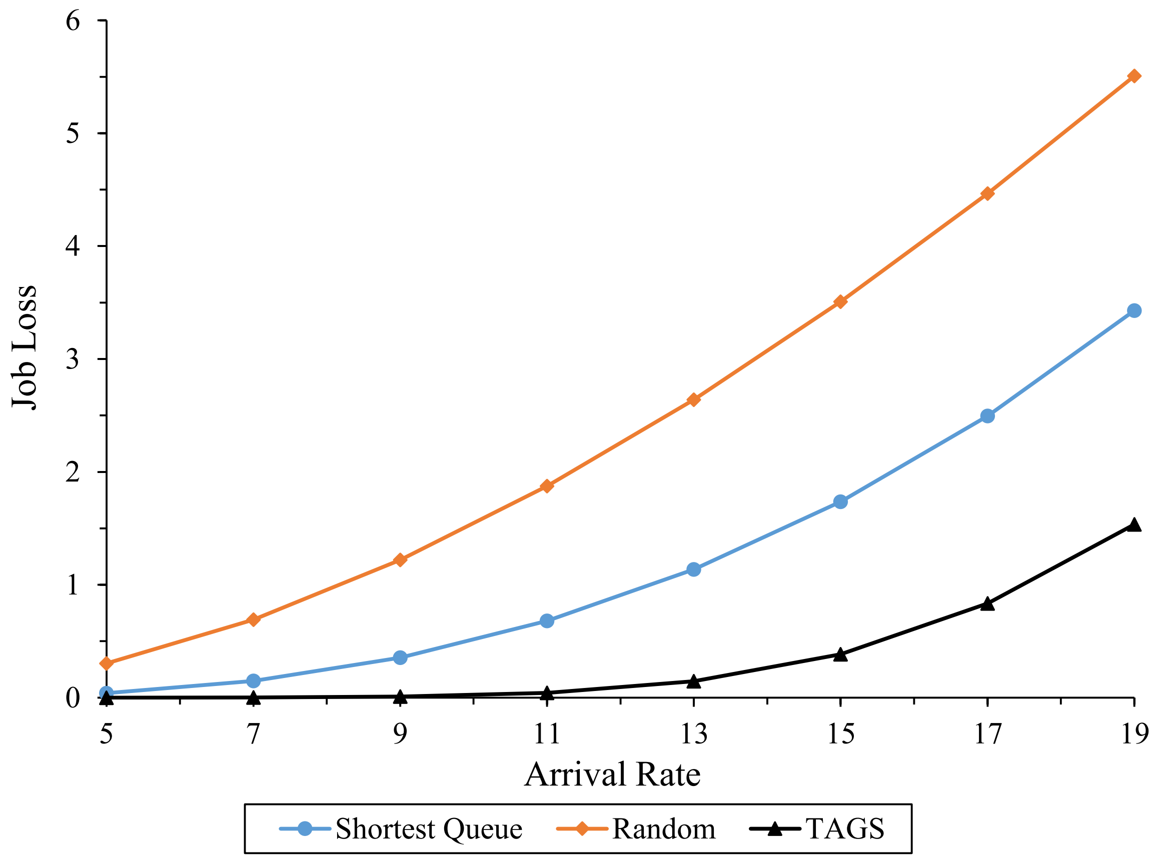


Figure 5.6: Job loss varied against arrival rate λ, when the average service μ= 10, and proportion of short job α = 0.99, and the service rate μ_1=100μ_2

a long job occupies both queues. This can happen when one long job arrives in the system and is forwarded to the first server. Subsequently, If another long job arrives, it will be forwarded to the second server as the first one is already occupied. As a result, both queues will become full, and if any new job arrives in the system will be dropped from the queue leading to an increase in the job loss rate. In contrast, TAGS reduces the chance that both queues become full if the timeout is well-tuned. The first queue is unlikely to become full as the timeout mechanism will kill long jobs and transfer them to the second server. Even though the processing time for each long job is 100 times longer than any short job, the probability of the second queue becoming full is relatively small as there are too few long jobs. Figure 5.6 shows the job loss at each arrival rate for TAGS, the shortest queue and the random allocation.

The results become interesting when we look at the average response time. Figure 5.7 shows the average response time varied against arrival rate. It can be seen that at low arrival rates ($\lambda < \mu = 10$), TAGS outperforms the shortest queue and random

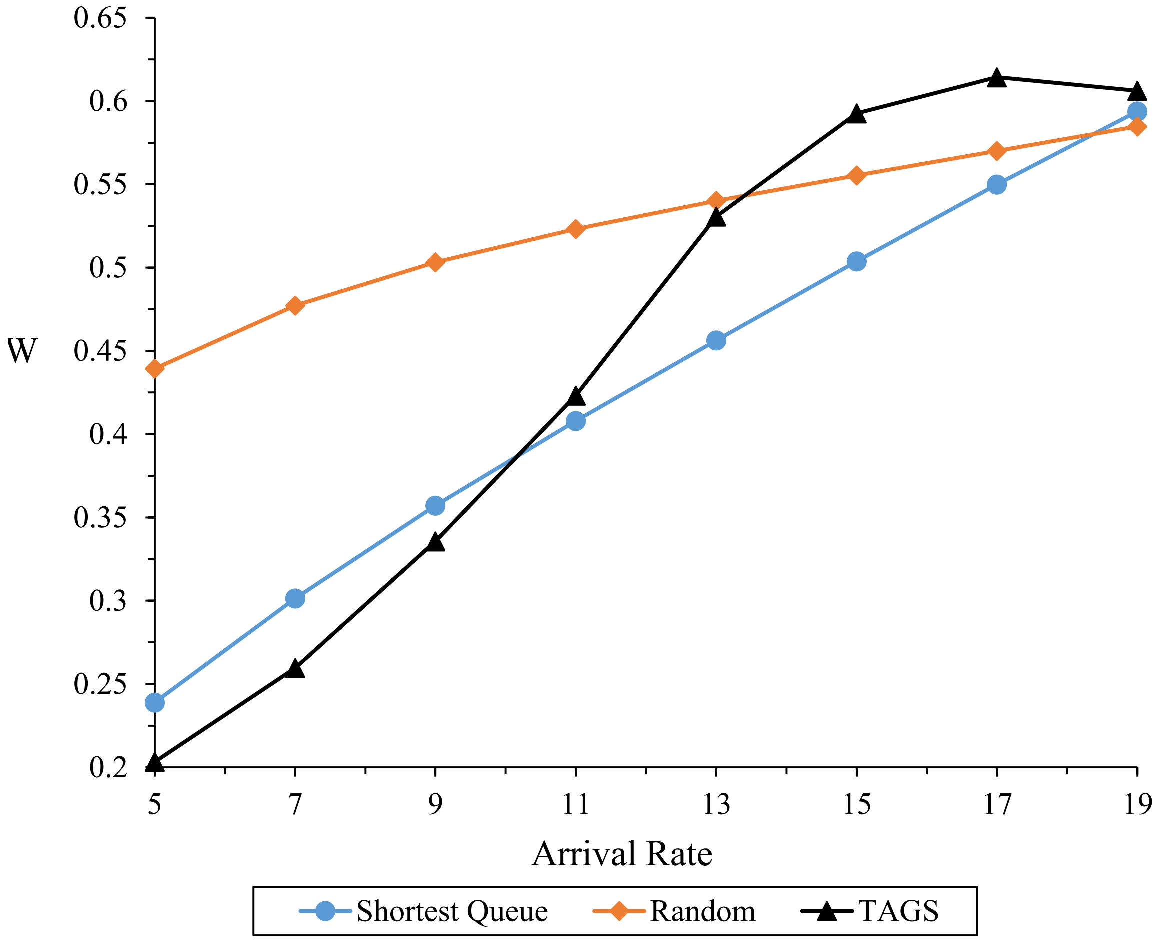


Figure 5.7: Average response time varied against arrival rate λ, when the average service μ= 10, and proportion of short job α = 0.99, and the service rate μ_1=100μ_2

allocation. In contrast, when arrival rates increased to be more than the average service ($\lambda > \mu = 10$), TAGS performs worse than the shortest queue. This is simply because TAGS processes more jobs and, in particular, more long jobs, which by their nature have a longer response time. It is worth noting, the timeout values t for TAGS are the best values for the highest throughput as mentioned previously in Section 5.2.1. So, to reduce the average response time for TAGS, the most important is to consider the t optimisation, in that different values for t from the current values might be obtained.

5.3 Energy model and analysis

The choice of the energy model is an integral part of the study that demonstrates how performance values convert into energy. We focus on the case where most of the server power consumption is due to the CPU, so other components such as memory and hard disk power consumption have been ignored. In this study, it has been assumed that the processor is equipped with DVFS capability. Therefore, the process we propose to estimate the energy consumed Ec by a server has an underlying assumption that this energy is essentially the processor performance states (P-states) value multiplied by utilisation. The P-states are defined by the Advanced Configuration and Power Interface (ACPI) specification as the capability of a processor to switch between different supported operating frequencies. The P_0 state represents the highest performance state which achieves maximum performance and consumes maximum power. States from P_1 to P_n are lower performance states, where n is the maximum P-state implemented by the processor, not to exceed 16. The higher P-state number refers to lower utilisation of the CPU and lower energy consumption. The number of the P-state is processor-specific. For example, the AMD Opteron CPU has six performance levels with frequencies ranging from 1000 to 2600 MHz [54].

In our energy model and based on previous studies [15, 34], the server's power consumption is assumed to follow a linear relationship with power consumption at any given time and the CPU utilisation at the same time. To estimate the amount of energy per job, the system's throughput is used, from which an estimation of the cost of each job in Watt and the utilisation of P-state value can be estimated.

In this chapter, the throughput, utilisation and P-state were considered in calculating the total and the average energy consumption per job. The system was assumed to be homogeneous, equipped with AMD Opteron CPUs. The value for each P-state in AMD Opteron CPUs was obtained from [1]. To specify the P-state, the utilisation of the system U at each arrival rate is calculated by Equation 5.1.

$$U = 1 - Prob(Q_{Empty}) \tag{5.1}$$

where $Prob(Q_{Empty})$ is the probability of the event that the queue being empty.

Table 5.1: AMD opteron CPU specifications

P-state	Power(W)	Clock (GHz)	Voltage (V))
P0	95	2600 MHz	140
P1	90	2400 MHz	135
P2	76	2200 MHz	130
P3	65	2000 MHz	125
P4	55	1800 MHz	120
P5	32	1000 MHz	110
Idle	15	-	-

Table 5.1 shows all P-state values. When obtaining the system utilisation and its corresponding P-state value, the CPU total energy consumption can be calculated by Equation 5.2.

$$Ec = \sum_{i=1}^{n}(SU_i \times SP_{i,active}) + ((1 - SU_i) \times SP_{i,idle}) \tag{5.2}$$

Wher SU_i is the $Server_i$ utilisation , $SP_{i,active}$ is the $Server_i$ P-state value and $SP_{i,idle}$ is the $Server_i$ P-state idle value.

The average energy consumption per job is calculated by equation 5.3, where Ec is the CPU total energy consumption and T is the system throughput.

$$AverageEnergyPerJob = Ec/T \tag{5.3}$$

The document for the processor does not specify at which utilisation level each P-state should be triggered [1]. However, it was mentioned that these P-states could be pro-

Table 5.2: The equivalent utilisation level for each P-state

P-state	Utilization	P-state value in Watts
P0	100	95
P1	80	90
P2	60	76
P3	40	65
P4	20	55
P5	10	32
Idle	0	15

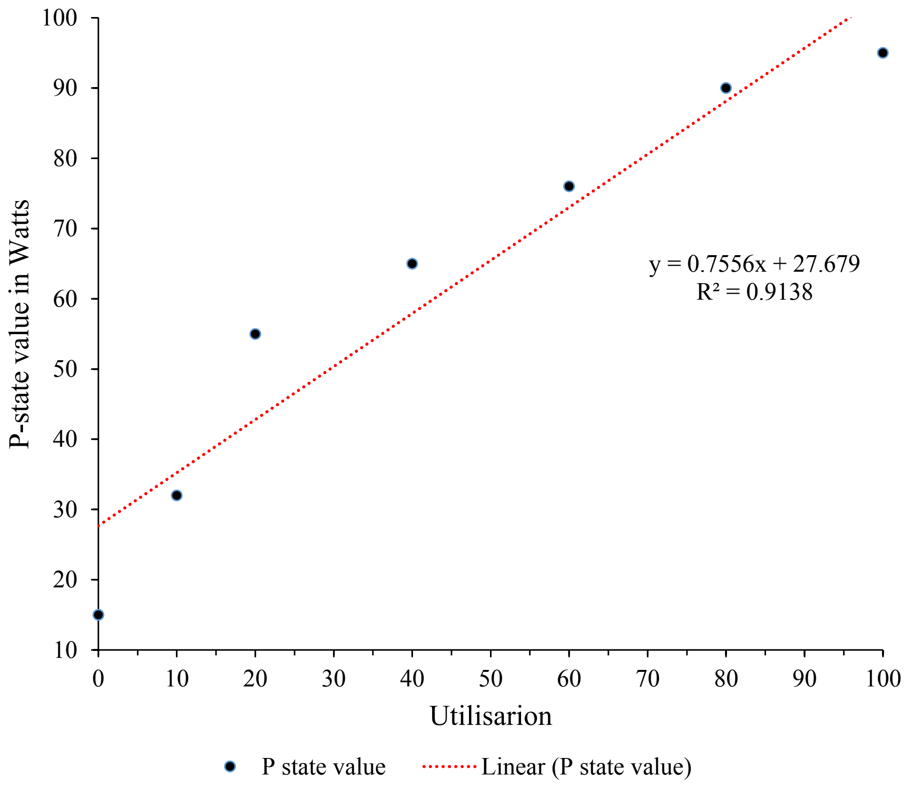


Figure 5.8: Linear regression

grammed to trigger at a specific utilisation level. Therefore, the equivalent utilisation level for each P-state was obtained from [54] and depicted in Table 5.2. However, by looking at these values, it will be noticed that some utilisation levels fall between other levels not connected to any P-state value. For example, P4 triggered when utilisation reached 20% and P3 triggered when utilisation reached 40%. However, if the utilisation is 35%, there is no specific P-state value connected to it, and its equivalent P-state value will be considered P4. This will have an impact on energy consumption calculation according to the method used. Therefore, there are two methods to use P-state value. The first is assuming each P-state covers a range of utilisation. For example, P4 is triggered when the utilisation is 20%, and the system keeps in this P-state until the utilisation increases to 40%; then, the system moves to P3. Notwithstanding, this approach does not accurately depict the rise or reduction in energy consumption that occurs when varying utilisation levels. The second is using the linear regression to obtain the equivalent P-state value in Watts for each utilisation. We used the linear regression to find a formula of the energy consumption for any utilisation value not in Table 5.2. Figure 5.8 shows the linear regression values and formula

5.3.1 *Energy consumption under exponential service demands*

Results presented in this section are discussed in terms of total energy consumption and energy per job under exponential demand. It is worth observing that the difference in total energy consumption between the shortest queue and random allocation is insignificant under the exponential demand. This is especially the case when the arrival rate $\lambda \leq 11$ which is very close to the average service μ, as shown in Table **??**

Table 5.3: Total energy consumption results and percentage difference.

Arrival Rate	**19**	**17**	**15**	**13**	**11**	**9**	**7**	**5**
Total Energy _Random	192.433	184.659	174.863	163.712	151.898	139.852	127.749	115.638
Total Energy _Shortest	197.051	187.595	176.108	164.079	151.974	139.862	127.750	115.638
Total Energy _TAGS	205.775	202.326	199.299	191.352	178.001	161.647	144.743	127.777
% difference between Random and TAGS	6.7%	9.1%	13.1%	15.6%	15.8%	14.5%	12.5%	10.0%
% difference between Random and Shortest	2.4%	1.6%	0.7%	0.2%	0.0%	0.0%	0.0%	0.0%

Figure 5.9 shows the effect of varying the arrival rate on energy consumption by the TAGS algorithm, the shortest queue strategy and random allocation. It can be noticed that the energy consumption of TAGS is higher than random allocation and the shortest queue in terms of total energy consumption under high and low arrival rates. The reason is related to the fact that the TAGS timeout mechanism assigns long jobs to the second node. The second server utilisation increases as it repeats the part of the service that has been processed in the first node and then processes the subsequent part of the service. In total, the same job receives processing time equal to timeout at the first node plus the processing time at the second node. This behaviour excuses $server_1$ from long jobs, allowing more short jobs to be served, which increases the utilisation of $server_1$ as well. See Figure 5.4 for more clarification. based on that, the repetition of the process for the timed-out jobs along with the extra jobs processed in the first server leads to an increase in the energy consumption by TAGS. On the other hand, the shortest queue strategy assigns the job to the server with the shortest queue, and the job receives service until completion without interruption. Hence, the system utilisation at both nodes is balanced; there is no repeat as the job is only processed at one node.

Interestingly, while the difference in total energy consumption between random allocation and the shortest queue strategy is relatively small at each arrival rate, the

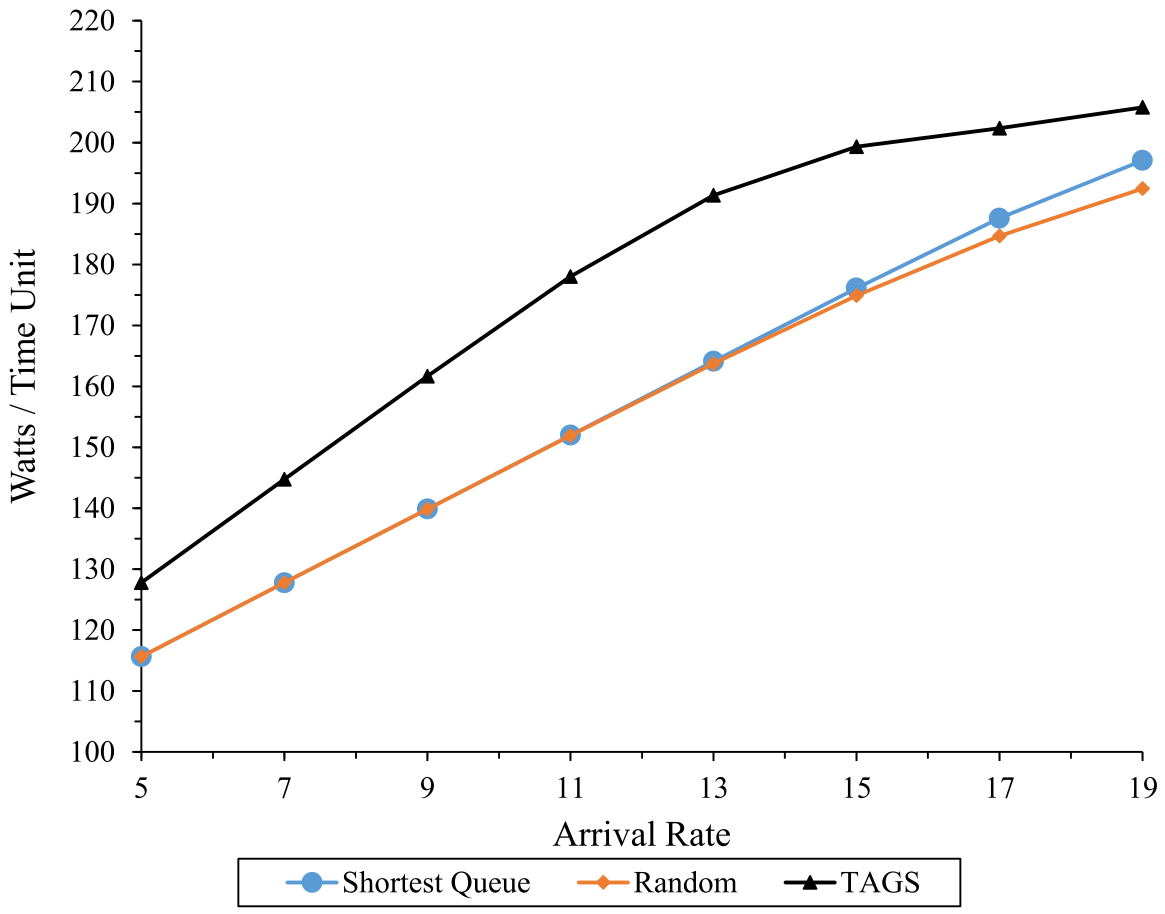


Figure 5.9: Total energy consumption varied against arrival rate λ, μ= 10.

difference between random allocation (the least energy consumption) and TAGS (the highest) follows a different trend. At the lowest arrival rate ($\lambda = 5$), the difference is approximately 10%, increasing to approximately 15.8% when the load increased to 50% of the system capacity. Afterwards, the difference decreases to 6.7% at the highest arrival rate ($\lambda = 19$). This behaviour can be explained like that TAGS mechanism from the beginning consuming more energy because the longer job receives processing in two servers with repeating of processing a part of the same job in both servers. In contrast, in the random allocation, the job receives the processing only once at one server, So the energy at a low arrival rate by random is less than TAGS. However, when the load increases, the random allocation starts consuming more energy, leading to decreased energy consumption difference between TAGS and random allocation. All three policies continuously increase energy consumption when the arrival rate increases, but the increase ratio by each one is different.

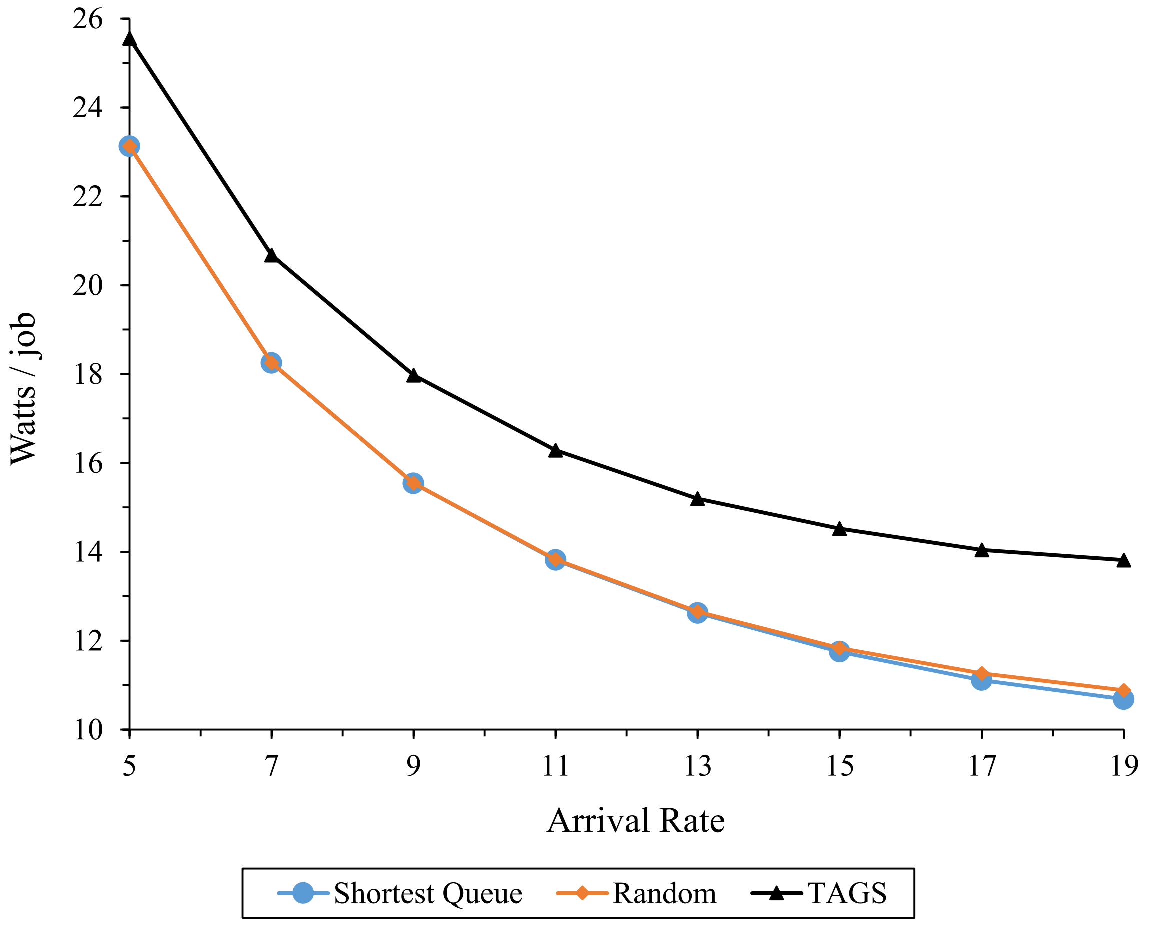


Figure 5.10: Average energy consumption per job varied against arrival rate λ, μ= 10.

In terms of average energy consumption per job, the TAGS algorithm with the best timeout values at each arrival rate also consumes higher energy than the shortest strategy and random allocation. See Figure 5.10. It is also worth pointing out that the TAGS algorithm can cost more energy per job when the arrival rate is relatively low. This can happen because the utilisation in both nodes at low arrival rates is less than the utilisation at higher arrival rates. Thus, the throughput is relatively small, but the energy reduction percentage is not as much as the reduction in the throughput percentage. Thus, it leads to an increase in the average energy per job at a low arrival rate. For example, when the arrival rate $\lambda = 5$, the average energy consumption per job is more than when the arrival rate $\lambda = 7$. Table 5.4 illustrates the total energy consumption at rates $\lambda = 5$ and 7, which is 127.78 and 144.47 Watts per time unit, respectively, and the throughput is 5 and 7 per time unit. Moreover, the energy per job is 25.56 and 20.68 Watts for the same arrival rates values. Thus, while the increase

Table 5.4: P-state value at different arrival rates

	Utilisation		**Throughput**		**Energy Consumption**	
Rates	**5**	**7**	**5**	**7**	**5**	**7**
Server 1	31.96	44.74	3.29	4.47	62.03	69.77
Server 2	38.09	53.32	1.80	2.53	65.74	74.97
Throughput	-	-	5	7	-	-
TAGS Total	-	-	-	-	**127.78**	**144.47**
TAGS Energy per Job	-	-	-	-	**25.56**	**20.68**

in total energy consumption at $\lambda = 7$ is approximately 13%, the throughput increased by 40%, decreasing the average energy consumption per job by $\approx$ 19% compared to energy per job at $\lambda = 5$.

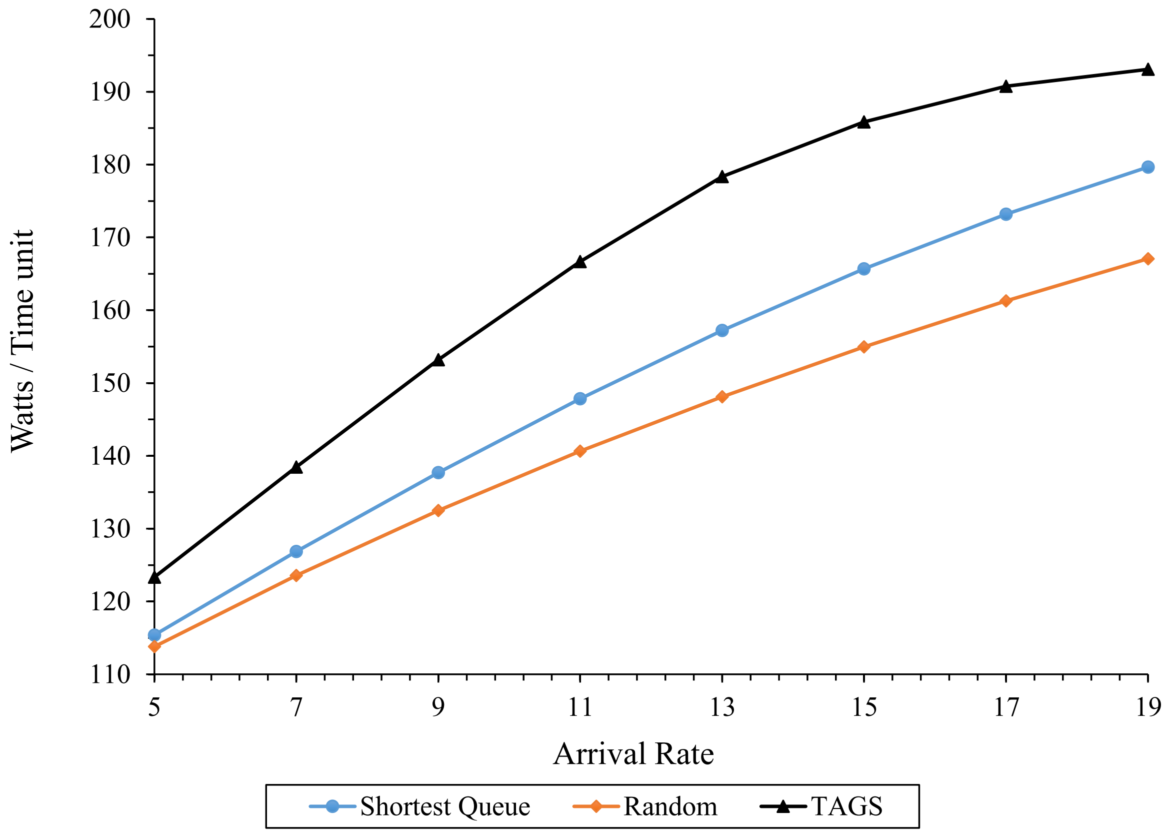


Figure 5.11: Total energy consumption varied against arrival rate λ, when the average service μ= 10, and proportion of short job $\alpha = 0.99$, and the service rate μ_1=100μ_2

5.3.2 *Energy Consumption Under Hyper-Exponential Service Demands*

The same methodology to calculate the energy consumption under exponential demand has been implemented in this section to calculate energy under hyper exponential demand. In order to evaluate the TAGS energy consumption under the hyper-exponential distribution, we considered the scenario when the long job is 100 times longer than the short job $\mu_1 = 100\mu_2$, and the proportion of the short job is $\alpha = 0.99$. Figure 5.11 shows total energy consumption varied against arrival rate λ when service demand has the H_2 distribution. Results for energy consumption are shown for TAGS, shortest queue and random allocation strategies.

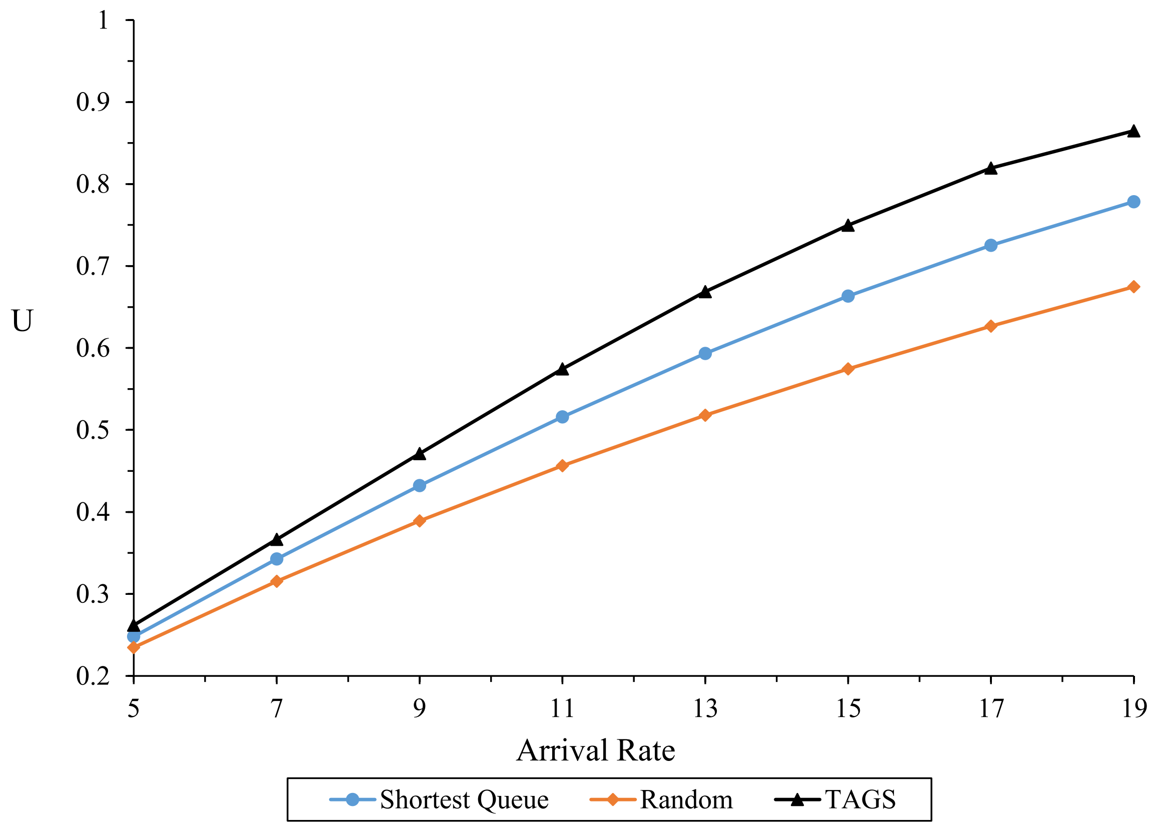
Figure 5.11, shows that TAGS consumed more energy than the shortest queue and random allocation at all arrival rates in terms of total energy consumption. It is also interesting to note that, while the difference in total energy consumption between

TAGS and random allocation at a higher arrival rate is as high as ≈ 14.44%, it is reduced to ≈ 8.4% at a low arrival rate λ=5. A possible explanation for this might be that at high arrival rate, the utilisations of $server_1$ and $server_2$ under TAGS are higher than the utilisation under the random allocation by 25% for $server_1$ 30% for $server_2$, As shown in Figure 5.12. Both servers under the TAGS mechanism consume more energy than any server running under the random allocation policy. While under the low arrival rate, the difference in the utilisation of $server_1$ between TAGS and the random allocation was reduced to ≈ 11% for $server_1$ but for $server_2$ the difference increased to 43%. It should also be noted that to calculate energy consumption, we use the performance state (P-state value), which increases gradually when utilisation is increased. Hence, at a high utilisation level, more energy is consumed.

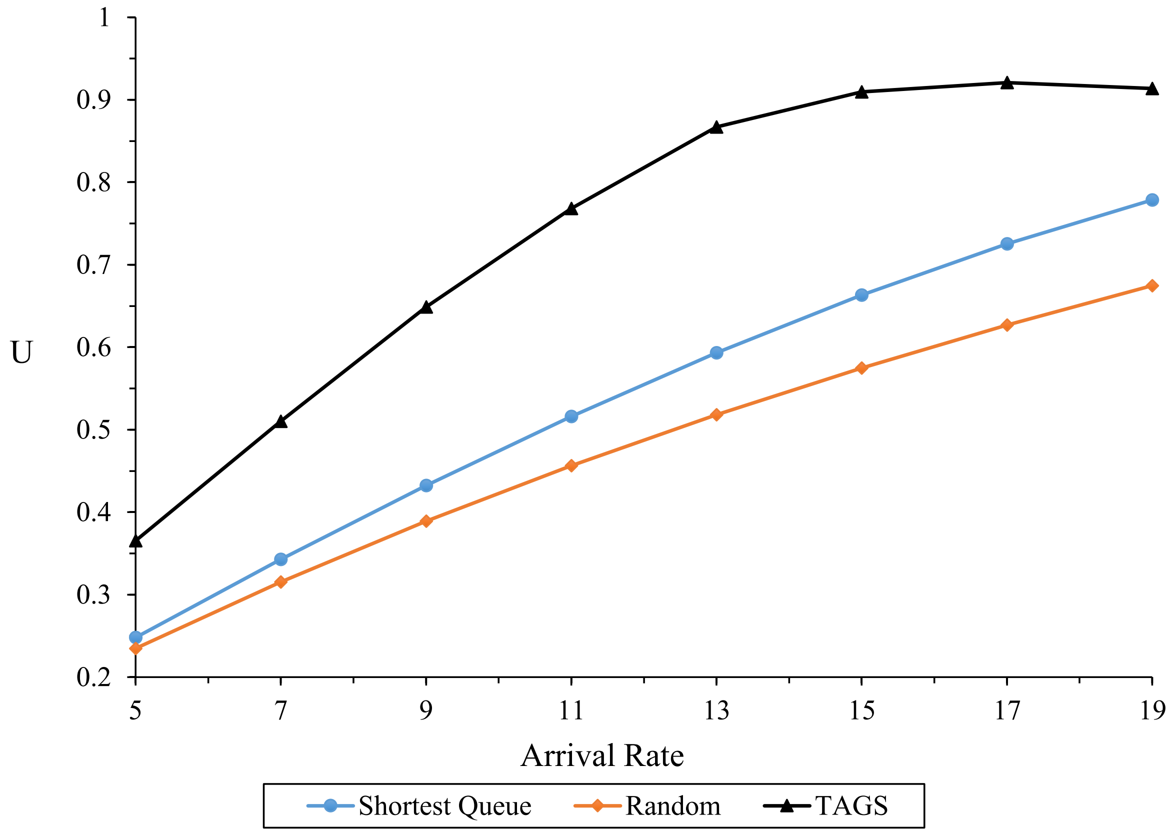
In terms of energy consumption per job, TAGS consumes less energy than random allocation and the shortest queue when service demand exceeds 75% of the system capacity (arrival rate λ >15) as shown in Figure 5.13. We think the reason is related to the TAGS mechanism, which sends long jobs from the first server to the second server, clearing the way to serving more short jobs at the first server. That approach produces a higher throughput compared to random allocation strategies, as shown in Figure 5.5. In contrast, TAGS consumes more energy per job at a low arrival rate than the shortest queue and random allocation. In fact, TAGS performs poorly in energy consumption when the service demand is low as the TAGS mechanism involves repeating the amount of service from the timeout period at the second server.

The shortest queue is the best strategy in terms of energy per job under hyper exponential service demand as long as the service demand does not exceed 75% of the system capacity.

For all three algorithms, energy per job consumption shows an opposite trend to total energy consumption. While total energy consumption decreases at a low arrival rate, the energy per job increases. This behaviour is related to the fact that the utilisation increases at a higher arrival rate, leading to more energy consumption. At the same time, the throughput of the system increases, as shown in Figure 5.5. Consequently, the energy per job decreases at high arrival rates and increases at low arrival rates.



(a) Server1



(b) Server2

Figure 5.12: (a) $server_1$ utilisation and (b) $server_2$ utilisation, varied against arrival rate λ, $\mu = 10$, $\alpha = 0.99$, μ_1=100μ_2

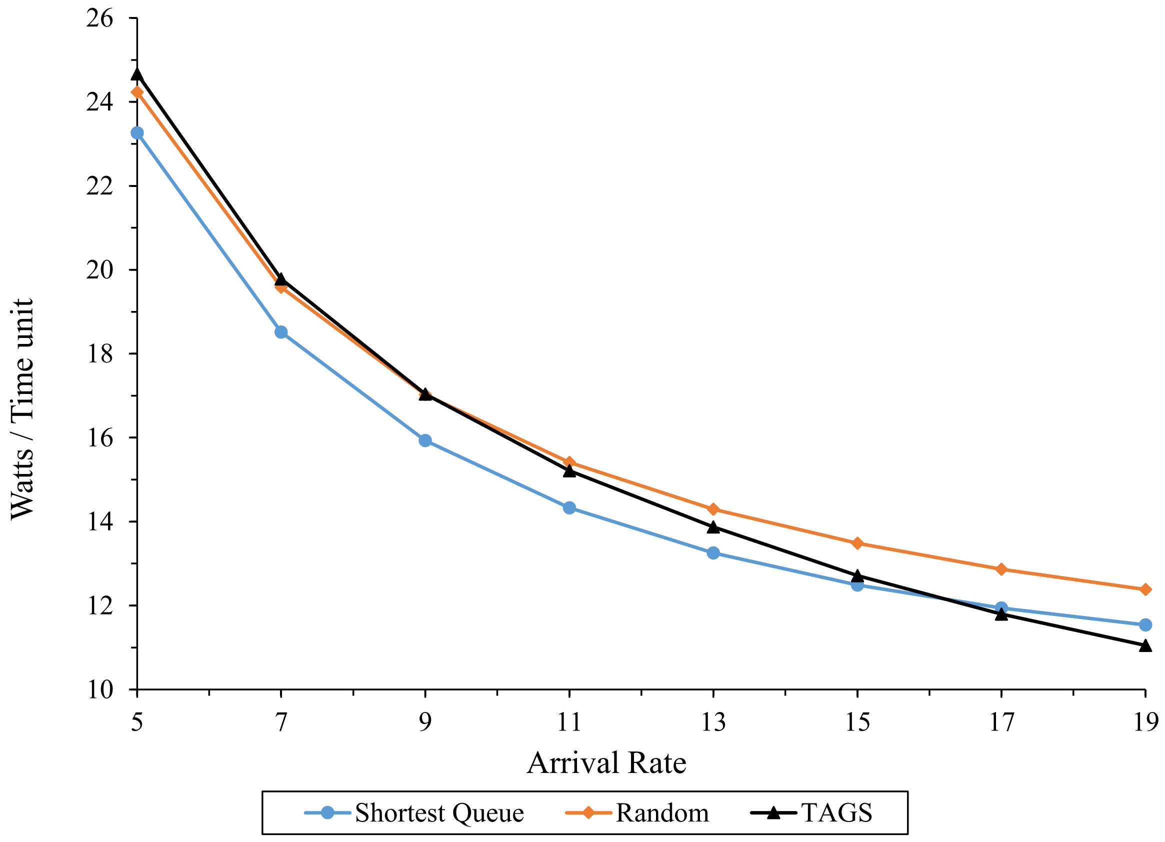


Figure 5.13: Energy per job varied against arrival rate λ, $\mu = 10$, $\alpha = 0.99$, μ_1=100μ_2

There is a correlation between energy consumption, throughput and task size. The processing of more jobs leads to a higher throughput. However, producing higher throughput will increase the total energy consumption. In contrast, it will reduce the energy per job. Although increasing throughput increases total energy consumption, the energy per job will depend on the amount of longer jobs that arrive in the system. If the system receives numerous long jobs, the throughput will be decreased as the time to process these jobs will be longer. As a result, the energy consumption per job will increase. On the other hand, if more short jobs arrive into the system are successfully completed, the throughput increases, resulting in a drop in energy consumption per job.

5.4 Conclusion

We have studied energy consumption by the TAGS policy and compared it with that of the shortest queue strategy and random allocation. We have focused on these policies in the case of the high variability in workload. Our model assumed we have a two-node system where servers are identical regarding energy consumption and performance. In practice, the downside is that data centres are heterogeneous environments in which energy consumption may differ from server to server. We rely on the processor performance states (P-states) value to calculate energy consumption. The main downside of this is that we neglect energy consumption by other server components, such as hard disk and memory. Our analysis of energy consumption under the exponential distribution and hyper-exponential distribution concluded that the TAGS mechanism consumes more energy than the other two policies regarding total energy consumption. The energy consumed per job followed the same trend under the exponential distribution. In contrast, when the arrival rate was high, TAGS consumed less energy per job under the hyper-exponential distribution than random allocation. The shortest queue was the best policy as long as the arrival rate was less than 75% of the system capacity. This chapter's primary focus was to evaluate and compare energy consumption by TAGS, shortest queue strategy and random allocation.

Variations among server specifications and processing capabilities are other factors that should be taken into consideration. We assumed servers are homogeneous regarding performance and energy, while in reality, the data centre has a heterogeneous environment. It is worthwhile to study energy consumption when servers are not identical. Suppose we have two servers with different performance capabilities. In that case, it will be valuable to investigate this combination and its impact on the performance and energy of TAGS.

The three strategies performance and energy consumption will be investigated in a heterogeneous environment in the next chapter.

6

TAGS Performance and Energy Consumption in Heterogeneous environment

Contents

6.1 Introduction

Chapters 4 and 5 considered modelling and evaluating the performance and energy consumption of TAGS in a system with groups of physical hosts having a homogeneous environment. Therefore, the PEPA model was designed to represent the interactions in such an environment. The system environment's heterogeneity is an essential factor to be considered in modelling servers performance and energy consumption in data centres. In this case, the PEPA model needs to be redesigned to adapt to the system environment changes.

In this chapter, we adjust the PEPA models to adopt the heterogeneity of the system. Section 6.2 introduces the methodology to add heterogeneity to the system. Moreover, it shows the updated PEPA models. Section 6.3 presents the experiment setup.

In Section 6.4, we investigated the TAGS performance and energy under different timeout values to show the effect of timeout on TAGS performance and energy. Section 6.5 compared TAGS, the shortest queue and weighted random in terms of performance metrics (i.e. throughput, job loss, average response time and servers utilisation). Finally, Section 6.6 compared the three mechanisms in terms of energy consumption, focusing on total energy consumption and energy per job.

6.2 Model in PEPA

Chapter 5, modelled and evaluated TAGS in a homogeneous environment under two service demand types: (i) exponential service demand in Section 5.2.1 and (ii) two-phase hyper-exponential service demand in Section 5.2.2. Results showed that TAGS performs well under hyper exponential service demand. Therefore, this chapter modelled and analysed TAGS, shortest queue and weighted random under hyper exponential demand in a heterogeneous environment.

The heterogeneity context in this chapter refers to the difference in performance between the first and second nodes. So, new parameters were introduced to represent the difference in performance between the two nodes in the model. The parameter σ is used to represent the difference in performance between Server2 and Server1. The value of σ equals 1 in case both servers are identical in performance. In contrast, if the second server performance is less than the first server, the σ value is less than one or more than one if the second server performance is better than the first server performance. The exact value of σ depends on the difference percentage between the two servers, e.g. if the second server is faster than the first server by 10% then σ = 1+(1*10%) = 1.1. In the same fashion, if server2 is slower than server1 by 10% then σ = 1-(1*10%) =0.9. So, the σ value in this model is used to control the second server performance.

It is worth mentioning that we assume servers are identical in terms of their energy consumption regardless of the performance difference. An argument might be made that it is unrealistic to assume that both servers consume the same amount of energy regardless of their performance differences. Nevertheless, this assumption forms because newer servers often tend to be faster and more energy-efficient.

6.2.1 TAGS PEPA model in heterogeneous environment

This chapter reused the same models introduced in Chapter 4. However, the models were updated to reflect the change in the working environment. As mentioned in the previous Section 6.2, a new parameter σ introduced to represent the difference in performance between $Server1$ and $Server2$. The σ parameter used in $Timer2$ component in PEPA model as this component incorporate $Server2$ representation, refer back to Section 4.4.1. The other components in the model remain without changes. Figure 6.1, shows the updated model.

$$
\begin{aligned}
Q1_0 &\overset{\text{def}}{=} (arrival, \lambda).Q1_i;\\
Q1_i &\overset{\text{def}}{=} (arrival, \lambda).Q1_{i+1} + (service1, \top).Q1_{i-1}\\
&\quad +(tick1, \top).Q1_i + (timeout, \top).Q1_{i-1}; \qquad 1 \le i < K1\\
Q1_n &\overset{\text{def}}{=} (service1, \top).Q1_{n-1} + (tick1, \top).Q1_n + (timeout, \top).Q1_{n-1}\\
Server1 &\overset{\text{def}}{=} (service1, \alpha * \mu_1).Server1 + (service1, (1-\alpha) * \mu_1).Server1'\\
&\quad +(timeout, \alpha * t).Server1 + (timeout, (1-\alpha) * t).Server1'\\
&\quad +(tick1, t).Server1\\
Server1' &\overset{\text{def}}{=} (service1, \alpha * \mu_2).Server1 + (service1, (1-\alpha) * \mu_2).Server1'\\
&\quad +(timeout, \alpha * t).Server1 + (timeout, (1-\alpha) * t).Server1'\\
&\quad +(tick1, t).Server1'\\
Timer1_0 &\overset{\text{def}}{=} (timeout, t).Timer1_n + (service1, \top).Timer1_n\\
Timer1_i &\overset{\text{def}}{=} (tick1, t).Timer1_{i-1} + (service1, \top).Timer1_n \quad 1 \le i \le n\\
Q2_0 &\overset{\text{def}}{=} (timeout, \top).Q2_i\\
Q2_i &\overset{\text{def}}{=} (timeout, \top).Q2_{i+1} + (tick2, \top).Q2_i\\
&\quad +(repeatservice, \top).Q2'_i, \qquad 1 \le i < K2\\
Q2_{K2} &\overset{\text{def}}{=} (timeout, \top).Q2_{K2} + (tick2, \top).Q2_{K2} + (repeatservice, \top).Q2'_{K2},\\
Q2'i &\overset{\text{def}}{=} (timeout, \top).Q2'_{i+1} + (service2, \top).Q2_{i-1}, \quad 1 \le i < K2\\
Q2'_{K2} &\overset{\text{def}}{=} (timeout, \top).Q2'_{K2} + (service2, \top).Q2'_{K2-1}\\
Timer2_0 &\overset{\text{def}}{=} \boxed{(\text{repeatservice},\alpha' * t * \sigma).(service2, \mu_1 * \sigma).Timer2_5}\\
&\quad \boxed{+(\text{repeatservice},(1\text{-}\alpha') * t * \sigma).(service2, \mu_2 * \sigma).Timer2_5}\\
Timer2_5 &\overset{\text{def}}{=} (tick2, t).Timer2_4\\
Timer2_4 &\overset{\text{def}}{=} (tick2, t).Timer2_3\\
Timer2_3 &\overset{\text{def}}{=} (tick2, t).Timer2_2\\
Timer2_2 &\overset{\text{def}}{=} (tick2, t).Timer2_1\\
Timer2_1 &\overset{\text{def}}{=} (tick2, t).Timer2_0
\end{aligned}
$$

$$((Q1_0 \underset{K}{\bowtie} Server1) \underset{K}{\bowtie} Timer1_5) \underset{timeout}{\bowtie} (Queue2_0 \underset{L}{\bowtie} Timer2_5)$$

where $K = (service1, timeout, tick1)$ and $L = (repeatservice, service2, tick2)$

Figure 6.1: A PEPA TAGS hyper exponential model in Heterogeneous environment

6.2.2 Shortest queue PEPA model in heterogeneous environment

The only modification to the previous shortest queue hyper exponential model in chapter 4 is adding the σ parameter to the components $Server2$ and $server2'$. The full model in a heterogeneous environment is shown in Figure 6.2.

$$
\begin{aligned}
Q1_0 &\overset{\text{def}}{=} (arrival1, \top).Q1_i, \\
Q1_i &\overset{\text{def}}{=} (arrival1, \top).Q1_{i+1} + (service1, \top).Q1_{i-1}, \qquad 1 \le i \le N, \\
Q1_N &\overset{\text{def}}{=} (service1, \top).Q1_{N-1}, \\[1ex]
Q2_0 &\overset{\text{def}}{=} (arrival2, \top).Q2_i, \\
Q2_i &\overset{\text{def}}{=} (arrival2, \top).Q2_{i+1} + (service2, \top).Q2_{i-1}, \qquad 1 \le i < N, \\
Q2_n &\overset{\text{def}}{=} (service2, \top).Q2_{n-1}, \\[1ex]
Server1 &\overset{\text{def}}{=} (service1, \alpha * \mu_1).Server1 + (service1, \alpha' * \mu_1).Server1' \\
Server1' &\overset{\text{def}}{=} (service1, \alpha * \mu_2).Server1 + (service1, \alpha' * \mu_2).Server1' \\
Server2 &\overset{\text{def}}{=} \boxed{(\text{service2}, \alpha * \mu_1 * \sigma).Server2 + (service2, \alpha' * \mu_1 * \sigma).Server2'} \\
Server2' &\overset{\text{def}}{=} \boxed{(\text{service2}, \alpha * \mu_2 * \sigma).Server2 + (service2, \alpha' * \mu_2 * \sigma).Server2'} \\[1ex]
S0 &\overset{\text{def}}{=} (arrival1, \lambda_1).S1 + (arrival2, \lambda_2).S2_1 + (service1, u1).S2_1 + (service2, u1).S1 \\[1ex]
S1_j &\overset{\text{def}}{=} (arrival2, \lambda_1 + \lambda_2).S1_{j-1} + (service1, \mu_1).S1_{j-1} + (service2, \mu_2).S_{j+1} \quad 1 \le j < N \\
S1_N &\overset{\text{def}}{=} (arrival2, \lambda_1 + \lambda_2).S1_{N-1} + (service1, \mu_1).S1_{N-1} + (service2, \mu_1).S1_N \quad N = 10 \\[1ex]
S2_j &\overset{\text{def}}{=} (arrival1, \lambda_1 + \lambda_2).S2_{j-1} + (service1, \mu_1).S2_{j+1} + (service2, u1).S2_{j-1} \quad 1 \le j < N \\
S2_N &\overset{\text{def}}{=} (arrival1, \lambda_1 + \lambda_2).S2_{N-1} + (service1, \mu_1).S2_N + (service2, \mu_1).S2_{N-1} \quad N = 10
\end{aligned}
$$

$$Q1_0 \,\|\text{Q}2_0 \underset{K}{\bowtie} S0 \underset{L}{\bowtie} Server1 \,\|\text{Server2}$$

Where $K = (arrival1, arrival2, service1, service2)$ and $l = (service1, service2)$

Figure 6.2: Shortest Queue PEPA hyper exponential model in Heterogeneous Environment

6.2.3 Weighted random PEPA model in heterogeneous environment

In this chapter, the analysis was performed in a heterogeneous environment. Therefore, as both nodes have different performance capabilities, we should balance the load. So for that, we used the weighted random strategy, not the random strategy. Similar to the shortest queue modification in the previous section. The only modification to the previous random allocation hyper exponential model in chapter 4 components $Server2$ and $server2'$ was updated by adding σ parameter to weight the performance difference.

The updated components are shown below:

$$\text{Server2} \stackrel{\text{def}}{=} (\mathit{service2}, \alpha * \mu_1 * \sigma).\mathit{Server2} + (\mathit{service2}, \alpha' * \mu_1 * \sigma).\mathit{Server2'}$$
$$\mathit{Server2'} \stackrel{\text{def}}{=} (\mathit{service2}, \alpha * \mu_1 * \sigma).\mathit{Server2} + (\mathit{service2}, \alpha' * \mu_2 * \sigma).\mathit{Server2'}$$

while the rest of the model stays the same as in Figure 4.2.

6.3 Experiment design

The analysis is performed over a simplified case with two nodes with exponentially distributed incoming tasks with an arrival rate of λ, with an interval of 5 to 19 tasks per time unit.

In this part of the experiment, we have the following parameters:

- σ: represents the difference in performance between $Server1$ and $Server2$. This parameter is used to control the performance of $Server2$ as we mentioned previously in section 6.2.
- λ: represents the job arrival rate. The interval varied from 5 to 19 tasks per time unit with a step of 2.
- $\mu 1$: represents the average service rate at Server1.
- $\mu 2$: represents the average service rate at Server2.
- α: represents the probability of the short job.

- α': represents the probability of a long job.
- t: represents the time out rate (used with TAGS only). The interval varied from 4 to 60 with step of 2.
- $\mu 1 = 100 \mu 2$: for all scenarios, the long job is 100 times longer than the short one.

We considered different scenarios to study the heterogeneity of the system. The first scenario is when the second server is faster and the speed difference varied by 10%, 20%, 30% and 40%. The second scenario is when the second server is slower by 10%, 20%, 30% and 40%. Finally, we compare these two scenarios with the previous scenario in Chapter 5 where servers were identical.

In Section 6.4, we studied TAGS performance and energy consumption under different timeout rates t to show the effect of timeout value on each metric.

Section 6.5, presents the performance comparison between TAGS, the shortest queue and weighted random schemes, while section 6.6 presents the energy comparison between them.

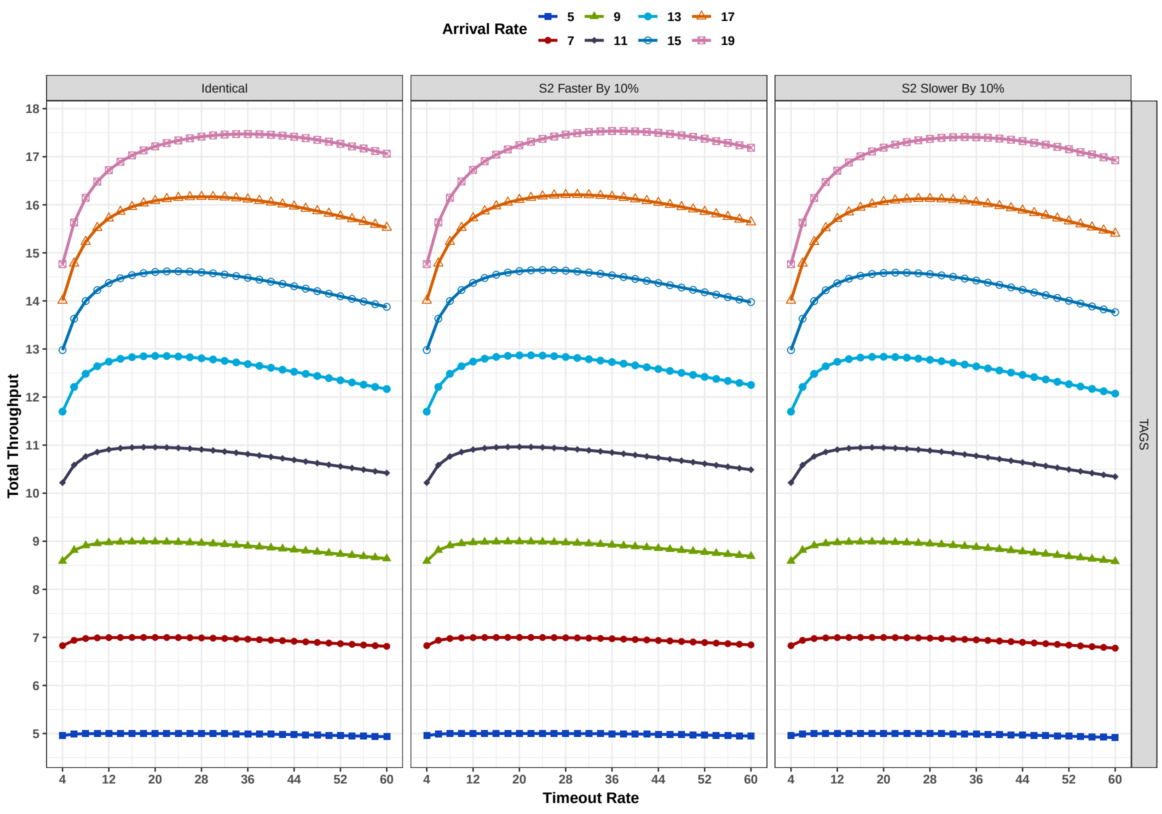


Figure 6.3: TAGS throughput varing against timeout rate

6.4 TAGS analysis in heterogeneous environment

As mentioned previously in Chapter 4 and Chapter 5, in contrast to the shortest queue and random allocation schemes, TAGS algorithm behaviour and working mechanism incorporate the time out factor. Therefore, TAGS results need to be optimised first in terms of throughput, utilisation and energy consumption before comparing with the shortest queue and the weighted random. Optimising timeout values is essential as the best time out value t differs from one performance or energy metric to another.

6.4.1 TAGS Throughput

We first studied the performance metrics starting with the system throughput. Figure 6.3 shows a comparison of throughput among different arrival rates in three different servers combination.

In all server combinations, TAGS throughput increases by increasing the timeout rate. As mentioned before, in PEPA, a race condition controls the behaviour of the model when there is more than one activity. So, by increasing the timeout rate, the probability of triggering timeout action increases. Hence, each time's processing duration in the first server became shorter, resulting in more short jobs processed at the first server and transferring more jobs to the second server as longer jobs. It is worth noting, the percentage increase in throughput is higher with a higher arrival rate. For example, when servers are identical, and the arrival rate is high λ=19, and the timeout rate increases from 4 to 36; consequently, the throughput increases by 18%. In contrast, when the arrival rate is low λ=5, the throughput increased only by less than 1% (0.71) for the same timeout values.

TAGS is sensitive to the timeout rate, so when the timeout rate is low, e.g. t=4, the throughput is the lowest among all timeout values for each arrival rate. The throughput increased each time the timeout rate increased up to a level when increasing the timeout, resulting in a decrease in the throughput. The reason for this behaviour can be connected to the increase in job loss at $Server2$. The timeout rate value that yields peak throughput differs for each arrival rate. The timeout rate value that the throughput starts decreasing at is different for each arrival rate. For example, when $Server2$ is faster by 10% and the arrival rate λ=19, the throughput starts decreasing when the timeout rate $t > 38$. While for arrival rate λ=11 the throughput decreases when the timeout rate $t > 20$. Figure 6.4 shows the total job loss at the first and the second server. A higher timeout rate kills and transfers more short jobs to the second server. While this mechanism reduces the job loss at the first server, it leads to overflow at the second queue resulting in an increase in the job loss rate at the second server, as shown in Figure 6.6. Consequently, the cumulative job loss starts increasing, leading to a decrease in the total throughput.

Table 6.1 shows TAGS optimised time out values for maximum throughput at each arrival rate for all server combinations. We considered the TAGS best timeout value for throughput presented in this section when we compared TAGS, the shortest queue and the random allocation in Section 6.5 and Section 6.6.

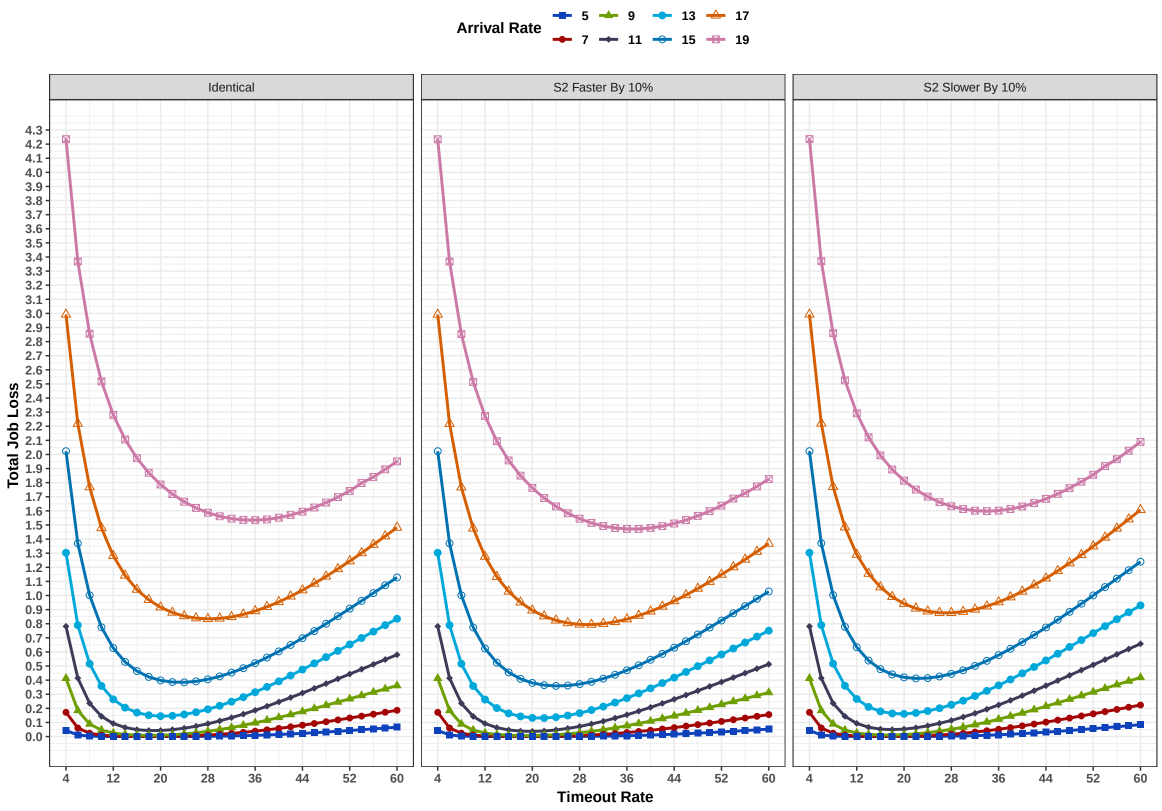


Figure 6.4: Total job loss

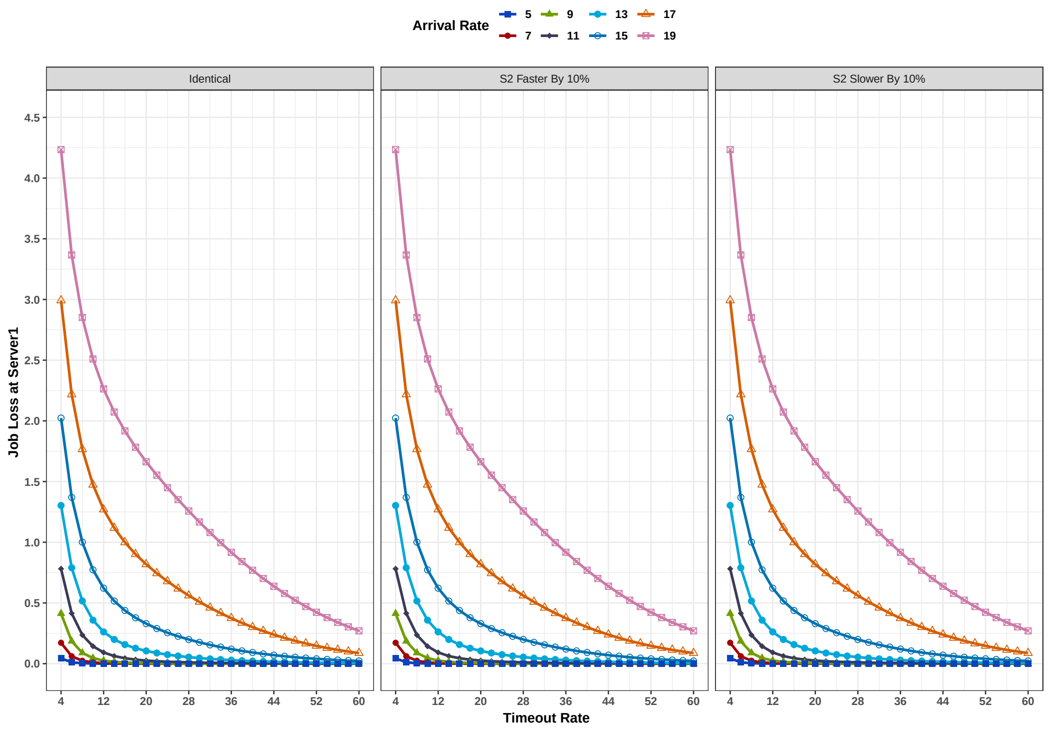


Figure 6.5: Job loss at server1

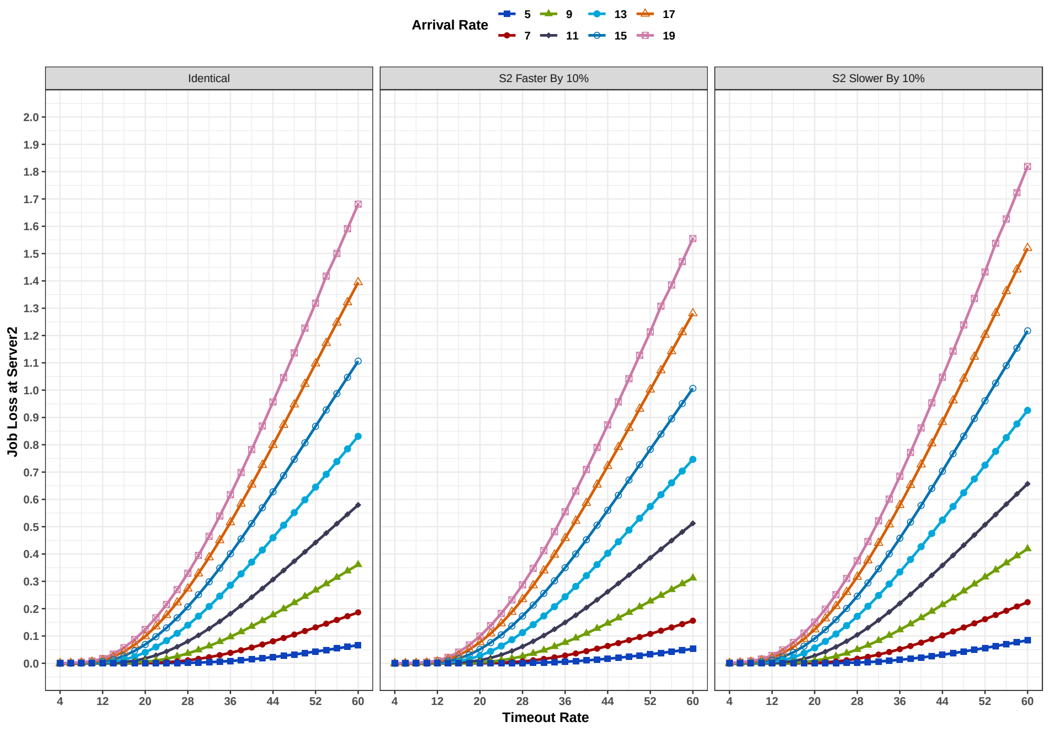


Figure 6.6: Job loss at server2

Table 6.1: The TAGS timeout rate values that maximise throughput at each arrival rate.

Arrival Rate	Identical		Server2 Faster by 10%		Server2 Faster by 20%		Server2 Faster by 30%		Server2 Faster by 40%	
	Timeout rate	Throughput	Timeout rate	Throughput	Timeout rate	Throughput	Timeout rate	Throughput	Timeout rate	Throughput
5	18	5.000	18	5.000	20	5.000	20	5.000	20	5.000
7	18	6.999	18	6.999	20	6.999	20	6.999	22	6.999
9	18	8.991	20	8.993	20	8.994	20	8.995	22	8.995
11	18	10.957	20	10.964	20	10.968	22	10.971	22	10.974
13	20	12.856	22	12.869	22	12.881	24	12.890	24	12.899
15	24	14.616	24	14.642	26	14.665	26	14.686	28	14.705
17	28	16.169	30	16.209	30	16.248	32	16.285	32	16.319
19	36	17.473	38	17.535	38	17.597	40	17.657	42	17.714

Arrival Rate	Server2 Slower by 10%		Server2 Slower by 20%		Server2 Slower by 30%		Server2 Slower by 40%	
	Timeout rate	Throughput	Timeout rate	Throughput	Timeout rate	Throughput	Timeout rate	Throughput
5	18	5.000	16	5.000	16	5.000	14	5.000
7	18	6.999	16	6.998	16	6.997	14	6.996
9	18	8.989	16	8.986	16	8.982	14	8.975
11	18	10.951	18	10.940	16	10.929	16	10.912
13	20	12.839	18	12.820	18	12.799	18	12.774
15	22	14.589	22	14.560	20	14.528	20	14.497
17	28	16.126	26	16.083	26	16.039	24	15.998
19	34	17.409	32	17.344	30	17.280	30	17.220

6.4.2 TAGS utilisation

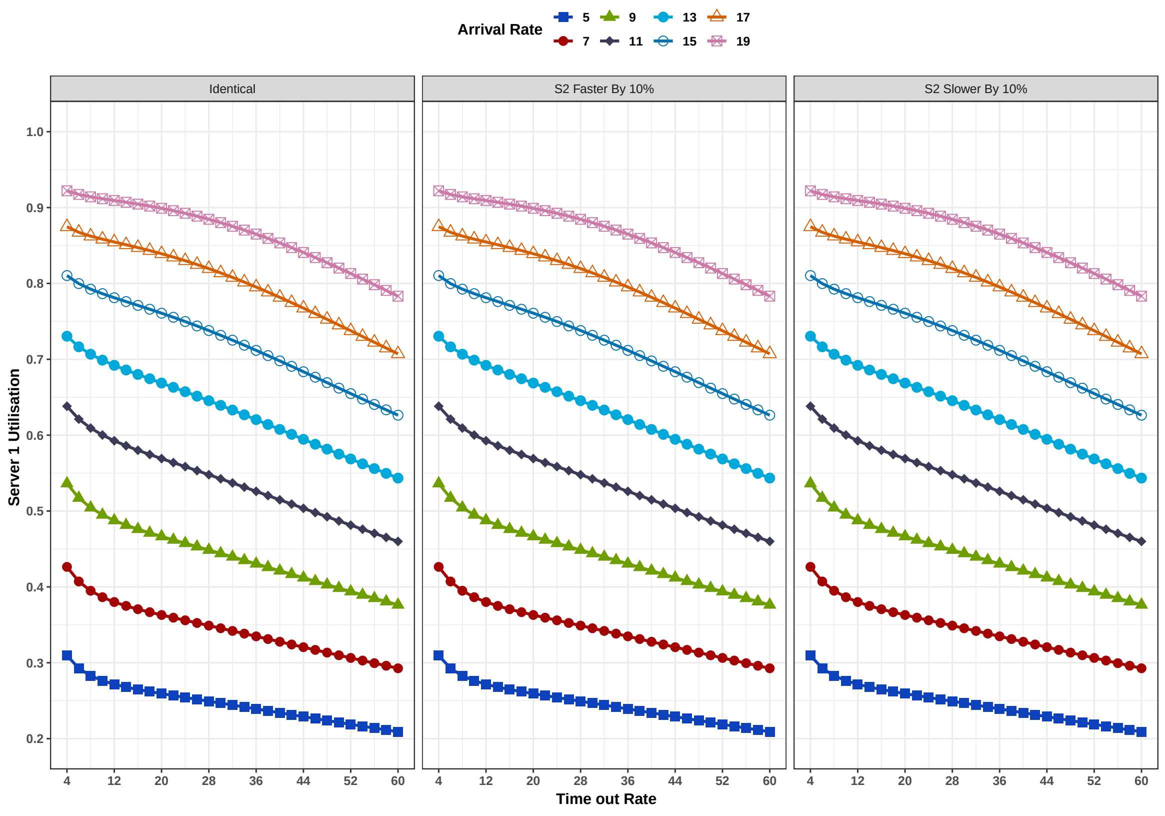


Figure 6.7: TAGS Sever1 utilisation varying against arrival rate

The utilisation of *Server*1 decreases by increasing the timeout rate (faster action). Because more jobs transferred to the second server, leaving only shorter tasks to receive processing at the first server Figure 6.7. If the timeout is very short, then *Server*1 will only transfer jobs to *Server*2 and will not process effectively any jobs. In contrast, the utilisation in the second server increases by increasing the time out rate. The server started serving more jobs, which relatively do not have a long duration. Thus, most of the time, the server is active and not idle, as shown in Figure 6.8. When the time out rate reaches a level leading to overflow, the second server's utilisation starts decreasing as job loss increases.

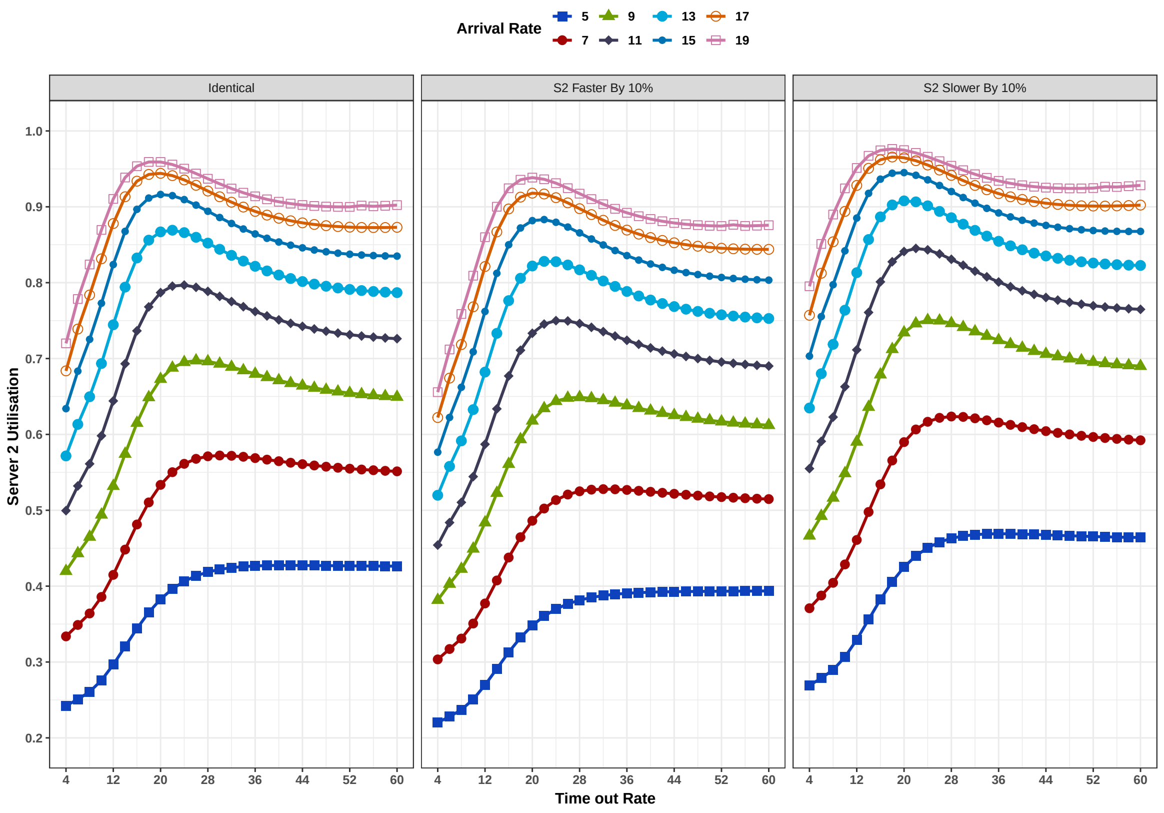


Figure 6.8: TAGS sever2 utilisation varying against arrival rate

6.4.3 TAGS total energy consumption

Less total energy consumption is achieved when the timeout rate is low. However, we illustrated in Section 6.4.1 that this is not ideal for the throughput.

The time out rate increase results in more jobs being transferred to the second server. Accordingly, the second server utilisation increases, leading to an increase in total energy consumption as shown in Figure 6.9. Even though there is a reduction in the utilisation of the first server compared with Figure 6.7, which reduces its active energy consumption in Figure 6.10, this is not enough to reduce the cumulative energy consumption by both servers. However, it is worth noting that the total energy increases, reaching a peak level when further increases in timeout rate, leads to a decrease in energy consumption. The reason for that is related to the first server, as it began processing fewer jobs and moved to an idle state quickly.

Calculating energy consumption in the energy model depends on the P-state value, which depends on server utilisation. So, when the utilisation is high, the P-value is high, leading to increased energy consumption.

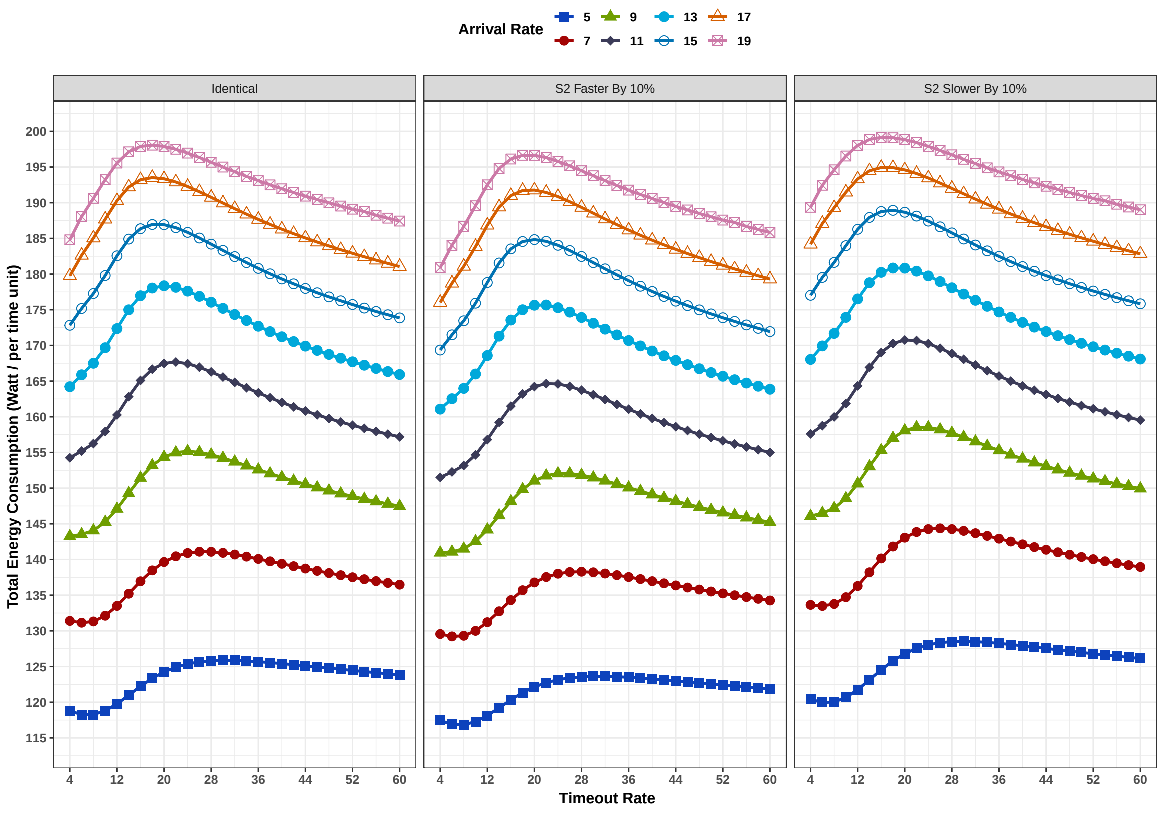


Figure 6.9: TAGS total energy consumption varying against arrival rate

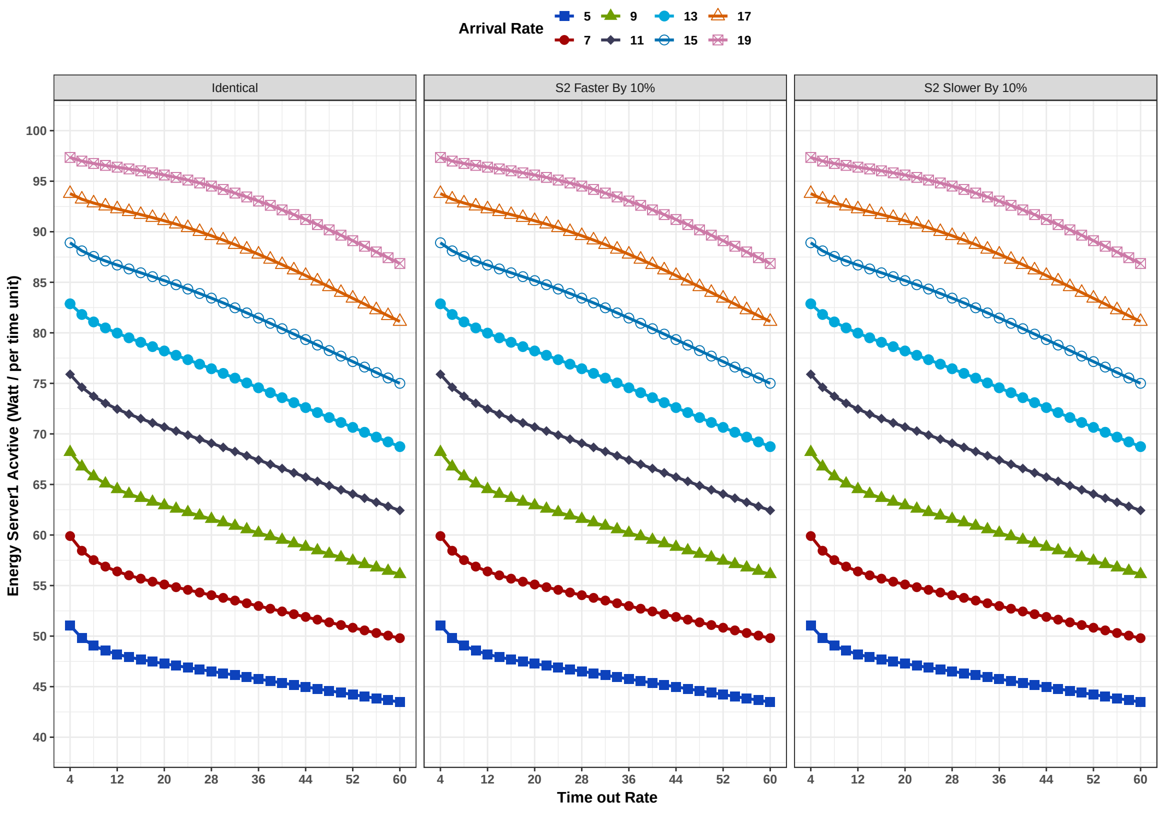


Figure 6.10: TAGS server1 active energy consumption varying against arrival rate

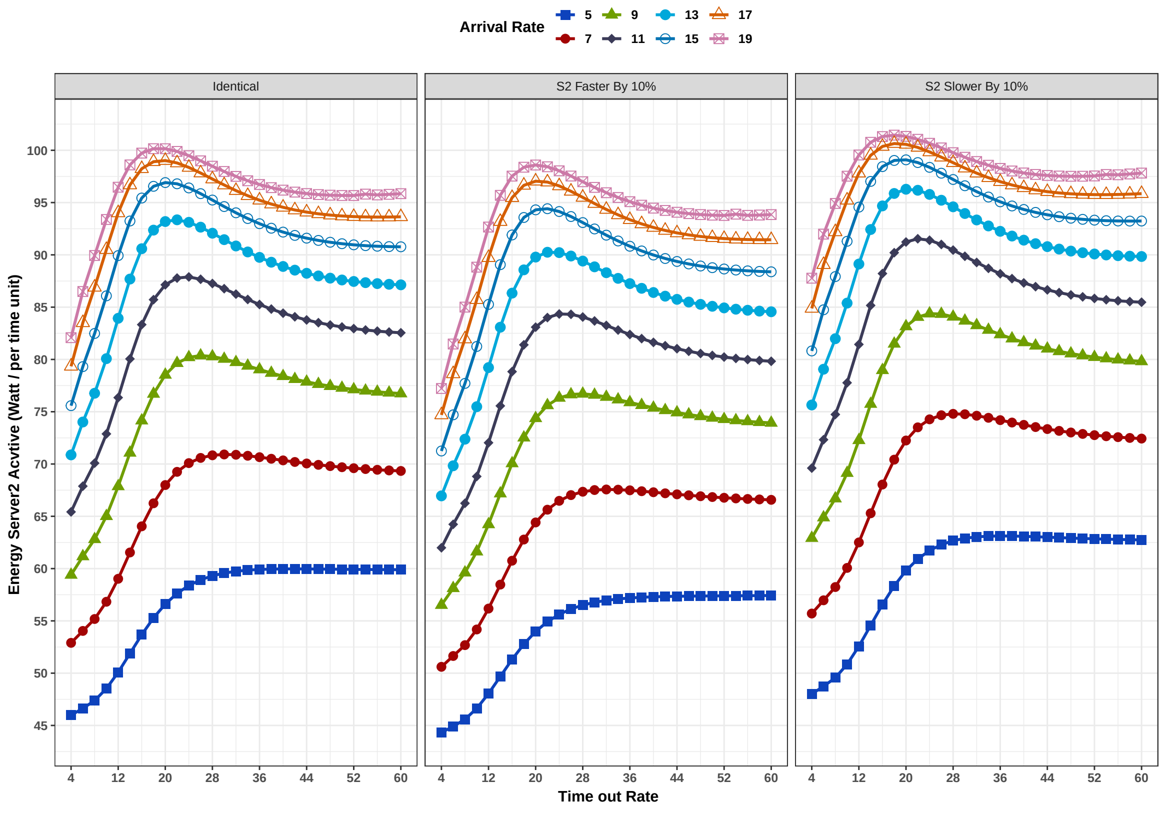


Figure 6.11: TAGS server2 active energy consumption varying against arrival rate

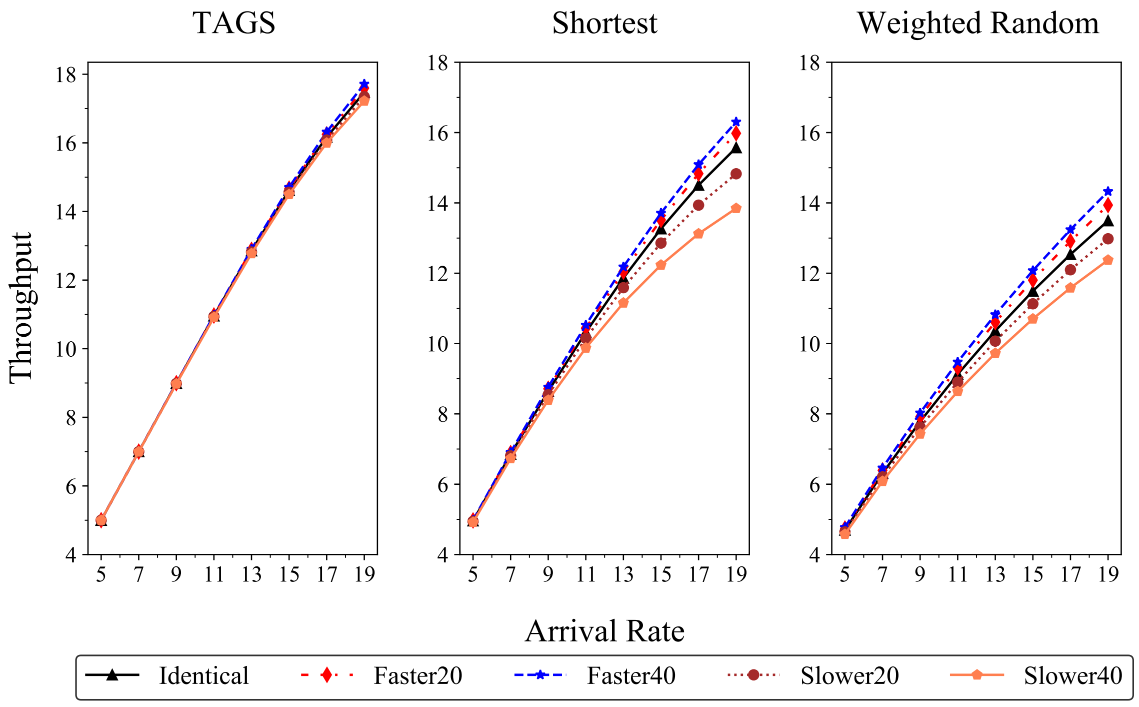


Figure 6.12: Throughput varied against arrival rate.

6.5 Performance analysis of TAGS vs shortest queue and weighted random in heterogeneous environment

This section demonstrates and compares the performance of TAGS, the shortest queue and the weighted random on three performance metrics: (i) Throughput, (ii) Utilisation and (iii) Average response time. TAGS algorithm results obtained for the best timeout values give each server's combinations the maximum throughput at each arrival rate. The best timeout values and their corresponded throughput was given Table 6.1.

For readability and clarity of comparison on graphs, we omitted some servers combinations. The results are shown in graphs for the five servers combination. However, the full results for all servers combinations are given in appendices. Appendix A shows the energy data and Appendix B presents performance data.

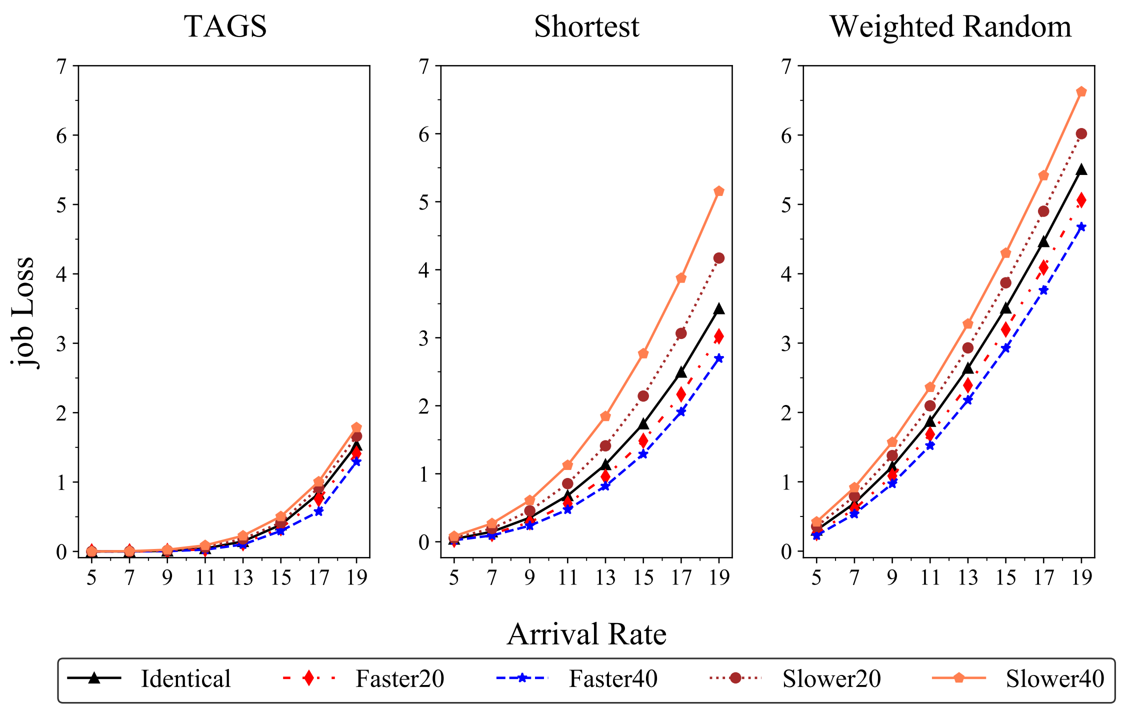


Figure 6.13: Job loss varied against arrival rate.

6.5.1 Throughput

Figure 6.12 shows the system throughput for the three algorithms in 5 server combinations. The system's heterogeneity did not significantly improve or decrease TAGS throughput results, whether the second server is faster or slower. When the arrival rate is less than 75% of the system capacity, the effect is approximately nothing. However, these results can not be generalised for TAGS, as we optimised the timeout values for the maximum throughput at each arrival rate for each combination. Hence, we would get different throughput results if timeout values were adjusted or optimised for a particular metric, such as reducing average response time or energy consumption.

On the other hand, the system's heterogeneity has a notable effect on the throughput of the shortest queue and weighted random. The throughput decrease or increase follows the trend of the second server performance capability. TAGS works well in all server configurations and achieves greater throughput than the shortest queue or weighted Random. Even when the second server's performance in TAGS is less by 40% of the first server, TAGS always outperforms both the shortest queue and the weighted random, whether their second server is faster or slower than TAGS second server. Figure 6.13 shows that job loss is the least for TAGS than the shortest queue and random allocation. Moreover, it should also be noted that for any server configurations, the difference in job loss between TAGS and the shortest queue at the lowest arrival rate ($\lambda = 5$ and 7) is imperceptible.

6.5.2 Utilisation

Figure 6.14 shows the average response time. TAGS performs well and outperforms the shortest queue and the random as long as the system load is low ($\lambda \leq 11$), especially when $Server2$ is fast.

The average response time increases for all three algorithms when the arrival rate increases. Notwithstanding, TAGS has a slightly different behaviour when the demand exceeds 75% of the system capacity. The average response time starts decreasing at this level. This behaviour could be explained as TAGS deals with job variability by unbalancing the load to increase the proportion of completed short jobs successfully. So, both servers in TAGS will not become busy processing two long jobs, while other shorter jobs wait for a long time. On the other hand, the shortest queue and weighted random do not have the mechanism that prevents two long jobs of occupied both servers for a long time. So, the probability of both servers being blocked by a long job increases. Consequently, the average response time increases.

Figures 6.15 and 6.16 shows the utilisation of $server1$ and $server2$. While the difference in utilisation between servers combinations for weighted random is noticeable, the system's heterogeneity does not indicate a substantial difference in $server1$ utilisation for TAGS. However, the disparity in utilisation between TAGS' servers combinations is apparent in $Server2$ results. For example, when $Server2$ is faster by 40%, the utilisation is lower than when it is slower by 40%. A slow server serving long jobs will be in high utilisation, while the faster server can serve the same number of long jobs without increasing the utilisation.

TAGS works to maximise the utilisation of both servers by unbalancing the load. So, servers are configured in a way that best suits the time required for job fulfilment. As

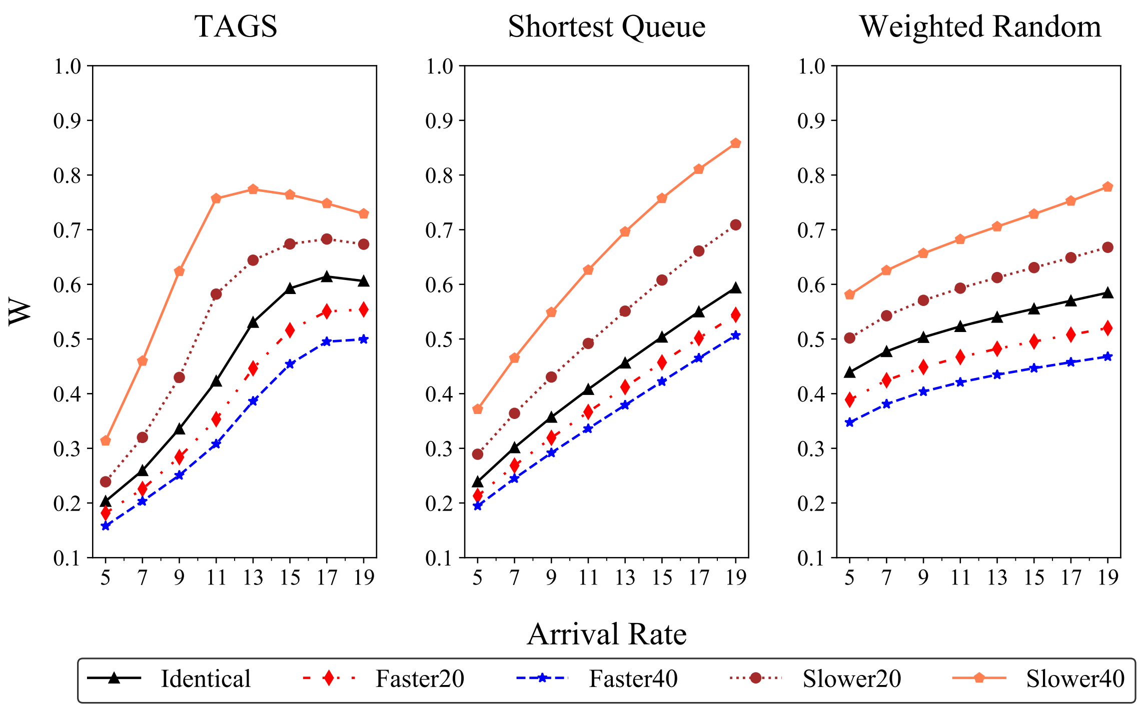


Figure 6.14: Average response time varied against arrival rate.

a result, the first server serves more short jobs, and the second server is devoted to long jobs.

It is worth noting that, in the random allocation scheme, the utilisation of $server1$ or $server2$ is equivalent since the load between the two servers is adjusted by sending additional tasks to the faster server. In contrast, although the shortest queue scheme has a balance mechanism by sending the job to the shortest queue, the utilisation of $server2$ is higher, especially when $server2$ processing speed is slow compared to $server1$.

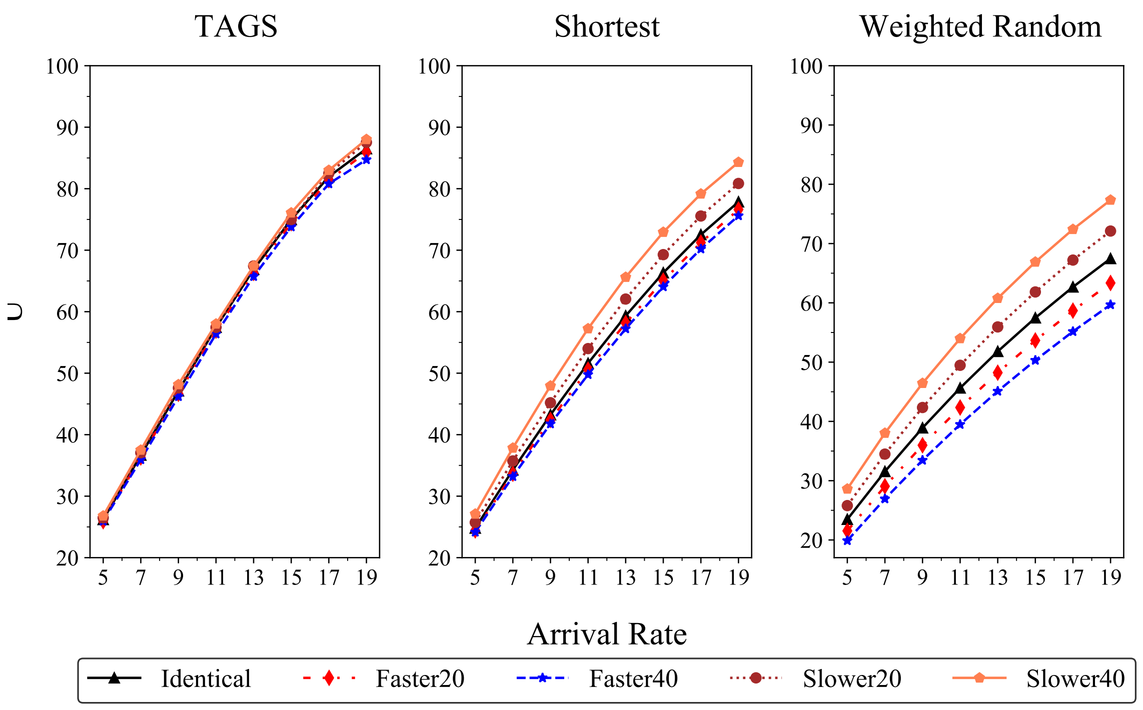


Figure 6.15: S1 Utilisation varied against arrival rate.

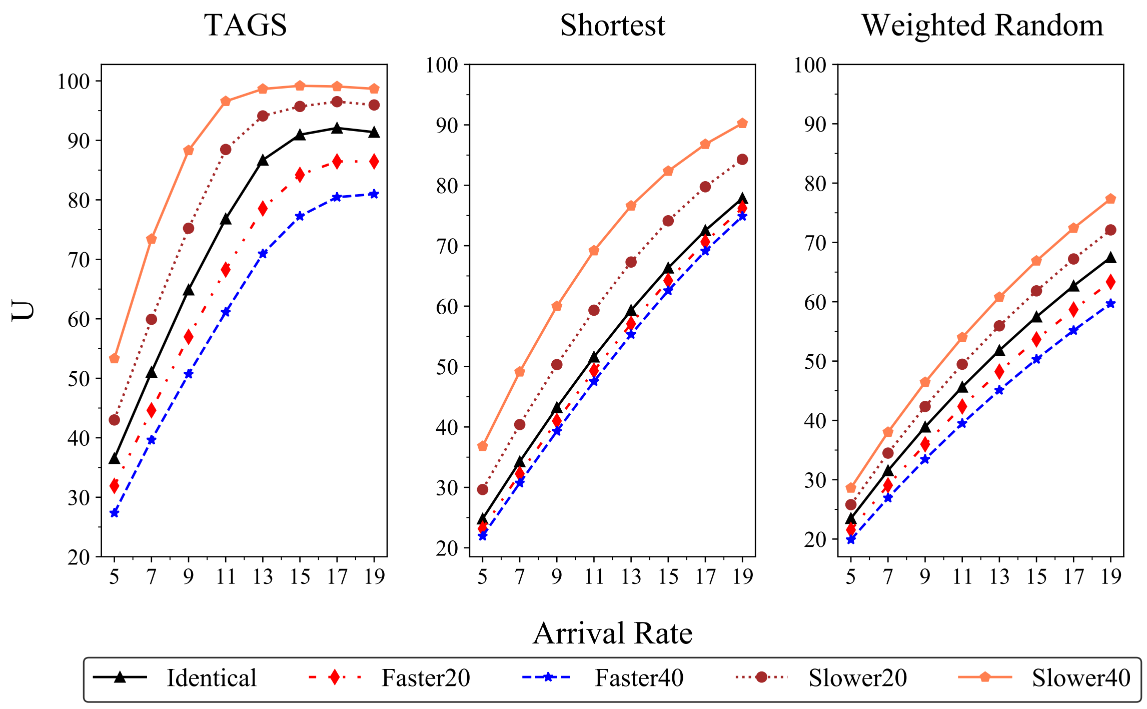


Figure 6.16: S2 utilisation varied against arrival rate.

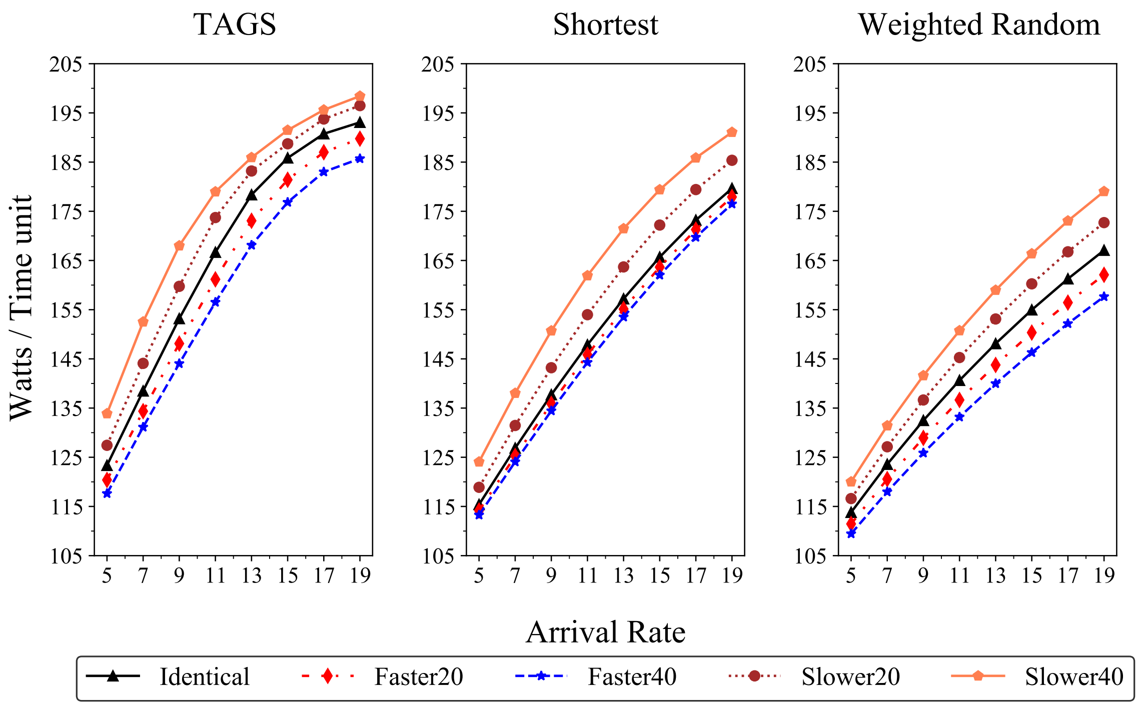


Figure 6.17: Total Energy Consumption varying against arrival rate.

6.6 Energy consumption analysis

This section illustrates the energy consumption by TAGS, the shortest queue and weighted random. We studied the energy consumption of the three algorithms under hyper-exponential service demand in a heterogeneous environment.

6.6.1 Total energy consumption

Calculating energy consumption in the energy model depends on the P-state value, which depends on server utilisation Equation 5.2. So, when the utilisation is increased, the P-state moves the processor to a higher frequency to meet the demand, which leads to increased energy consumption.

From Figure 6.17 It can be noticed that when the second server is slower than the first server, the energy consumption is the highest among all scenarios for each arrival rate. This behaviour is due to a slower server processing the job for a longer time and increasing the utilisation as the job needs more processing resources.

TAGS consume energy more than the shortest queue and weighted random allocation;

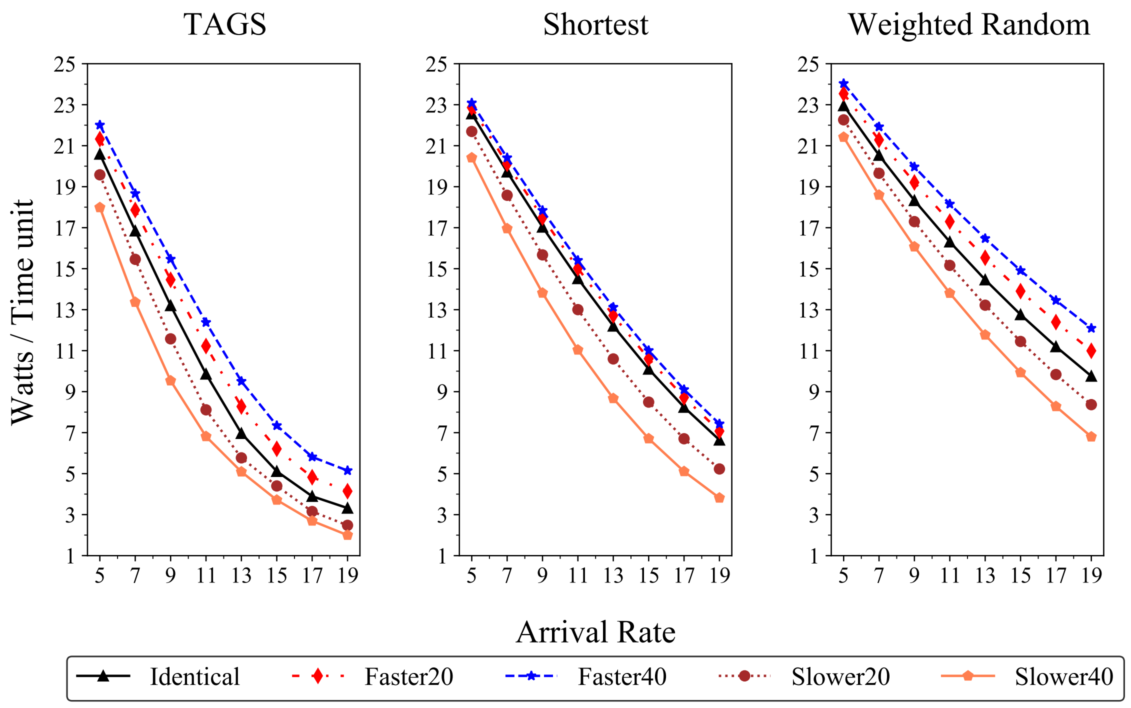


Figure 6.18: Total Idle Consumption varying against arrival rate.

this is expected as TAGS relies on the timeout mechanism to improve the system's throughput. However, this mechanism causes long jobs to be killed and transferred from the first server to the subsequent server after a preset timeout. Furthermore, the job was reinitiated from scratch on the second server. The same job receives service twice, initially at the first server with a processing time equivalent to t, then at the second server from scratch until completion or exhausted. So, the total processing time for long jobs is equal to t along with the processing time in the second server. Thus, serving more short jobs at the first server increased its active energy consumption. So, in total, TAGS consumes more energy than both shortest and weighted random schemes.

Figure 6.18 shows TAGS improves energy usage efficiency by reducing the idle energy consumption compared to both the shortest queue and weighted random. However, the reduced amount of idle consumption does not reduce the total energy being used under TAGS.

An in-depth analysis of each server's energy consumption when it is active or idle will provide a simple indication of each scheme's effect on each server's energy consumption.

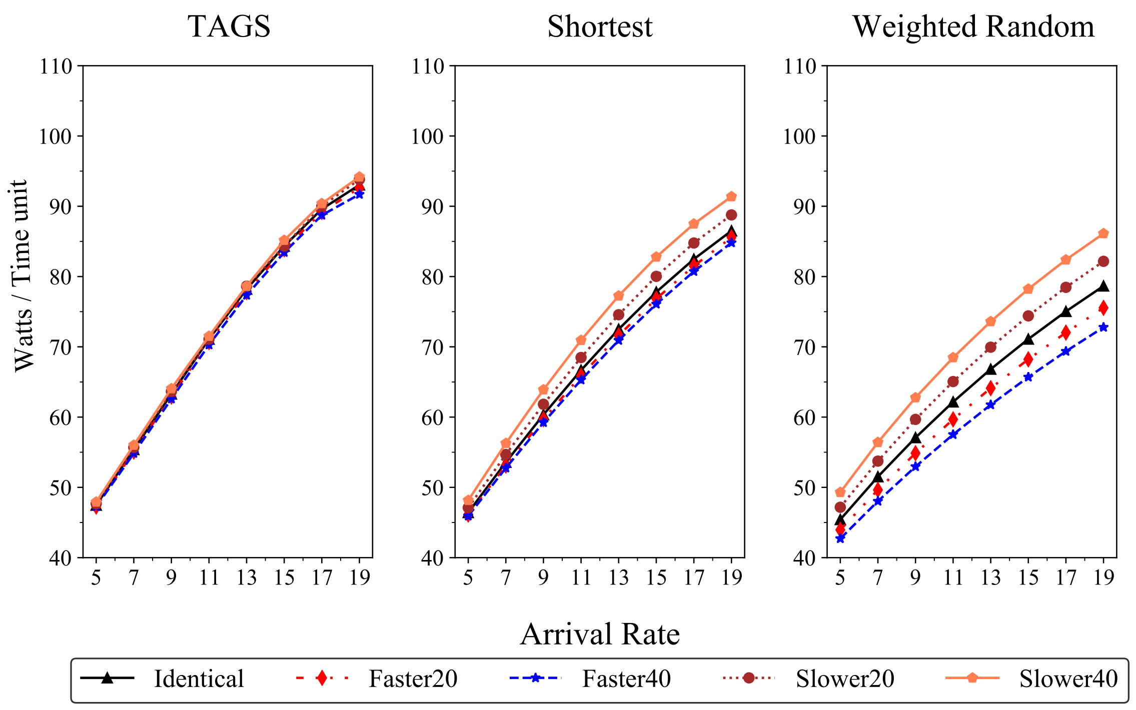


Figure 6.19: Server1 Active Energy Consumption varying against arrival rate

Figures 6.19 and 6.20 show that energy consumption follows the utilisation trend when servers are active, for utilisation refer to Figures 6.15 and 6.16. This behaviour is expected as the utilisation level determines the P-value state, which we used to calculate the energy consumption. The variation of server combinations tested does not significantly impact energy consumption by $server1$ for TAGS. However, the effect is noticeable for $server2$. Increasing the speed of $Server2$ reduced the energy consumed.

Both servers have a noticeable effect on energy consumption for the shortest queue when server2 is slower than server1. Server2's slower processing capability increases the processing time required to complete the job, leading to increased energy consumption. For server1, the increase in energy is related to extra job processed by $Server1$, because when $Server2$ is occupied by a job and takes an uninterrupted duration to process it, the queue becomes longer. Consequently, more jobs dispatch to $Server1$ because its queue became shorter as it processed the job at a higher speed.

When considering the idle energy consumption by servers, the TAGS mechanism of processing more short jobs at the first server and longer jobs at $Server2$ reduces idle consumption when the rate of incoming jobs increases. However, there is no impact of

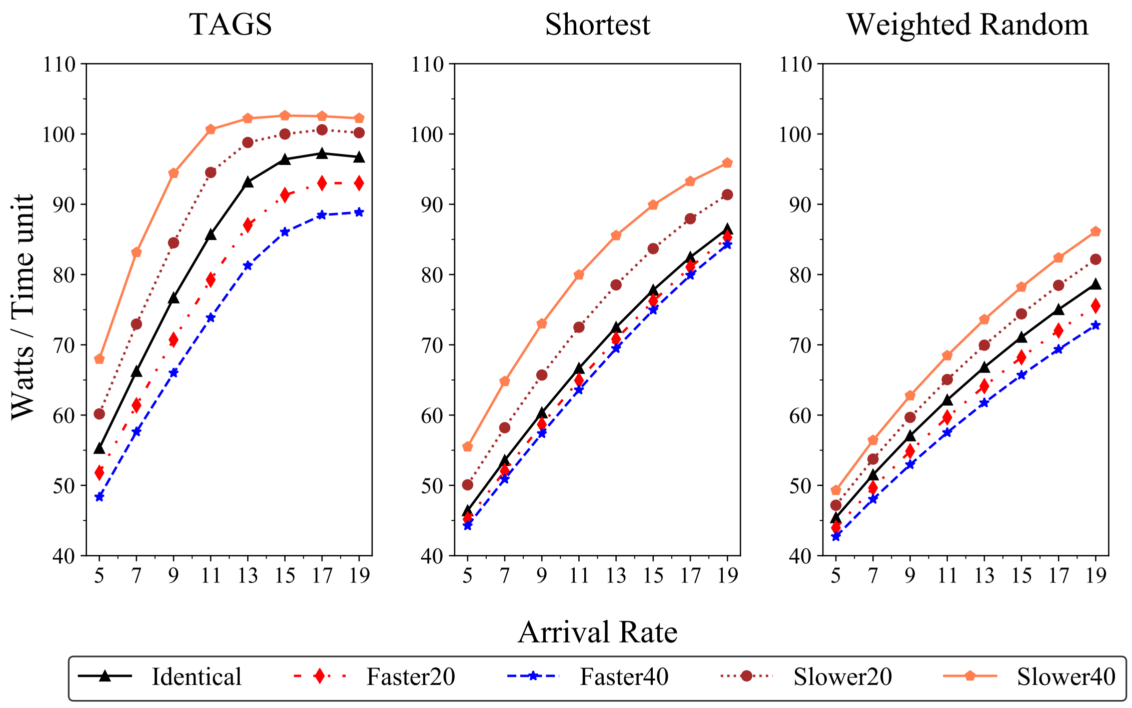


Figure 6.20: Server2 Active Energy Consumption varying against arrival rate

servers combinations on $Server1$ idle consumption, and the difference between combinations is insubstantial. The reason is that the TAGS mechanism keeps $Server1$ busy regardless of $Server2$ processing speed. On the contrary, $Server2$ idle consumptions show a noticeable difference between servers combinations. So, when $Server2$ is faster, the idle consumption is more than when it is slower because a faster server serves the job in a shorter time and moves to an idle state. Figures 6.21 and 6.22 shows the energy consumed by servers in the idle state.

Regardless of the servers' combination, both servers in the weighted random scheme consume the same energy because the load is balanced. When the incoming task rate increases, weighted random is more energy-efficient than shortest queue or TAGS in all server combinations. However, it is the worst in terms of performance.

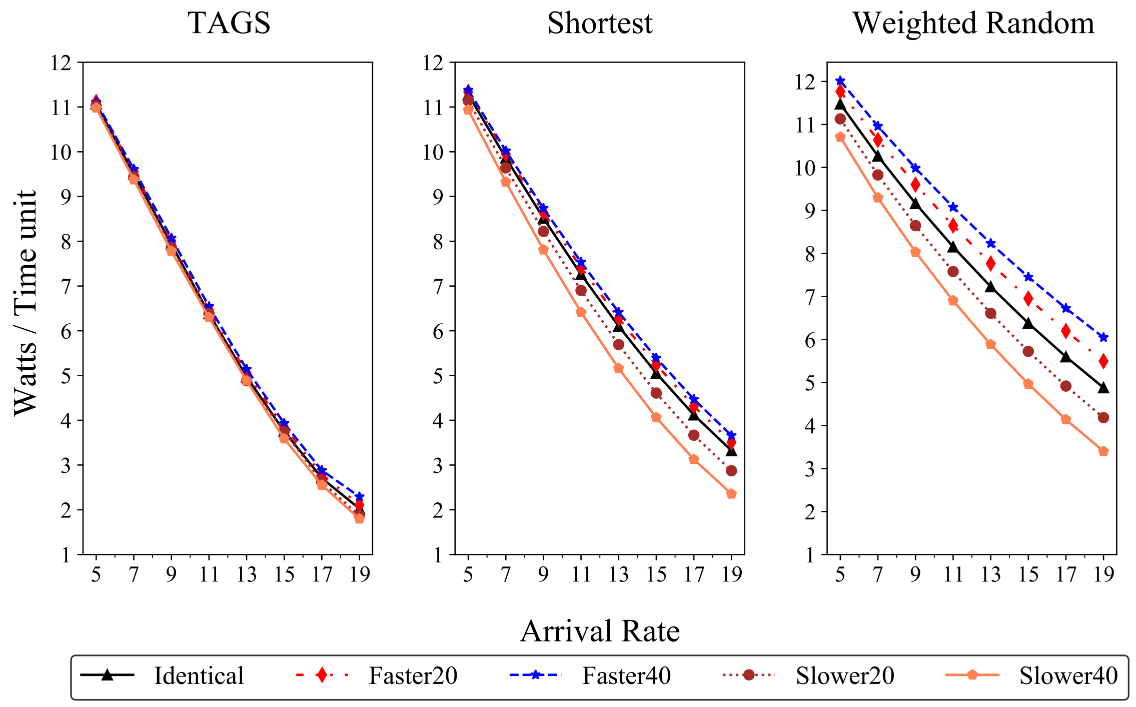


Figure 6.21: Server1 Idle Energy Consumption varying against arrival rate

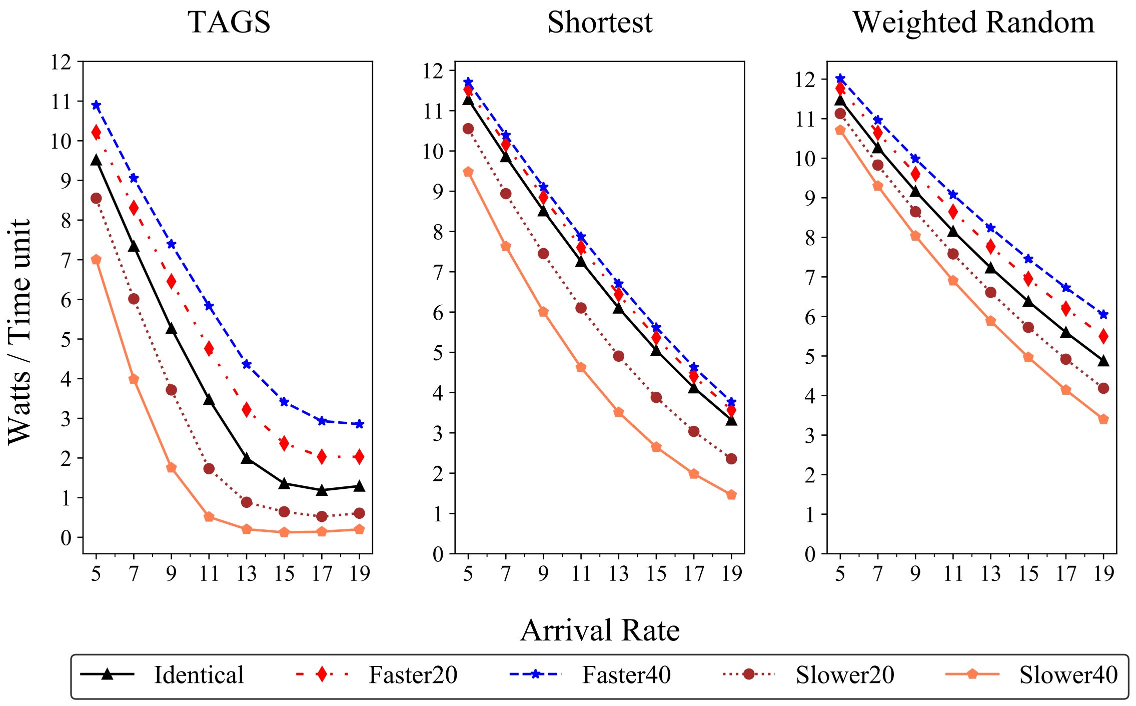


Figure 6.22: Server2 Idle Energy Consumption varying against arrival rate

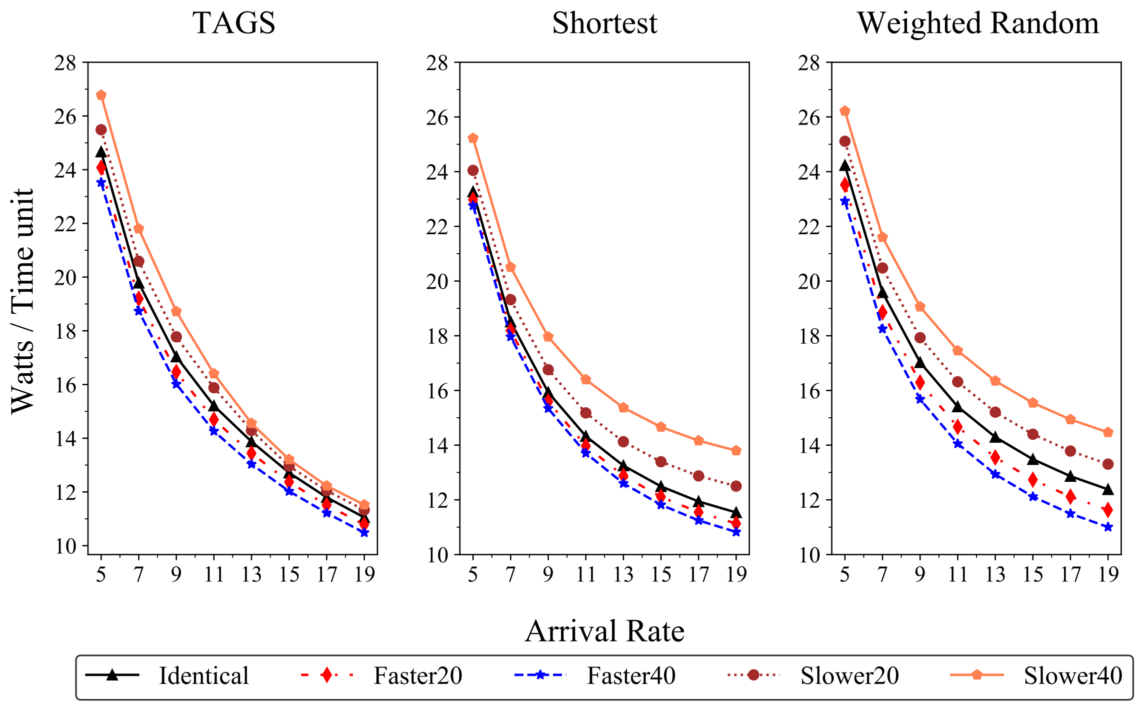


Figure 6.23: Energy Consumption per job varying against arrival rate

6.6.2 Energy per job

While low arrival rates give us the least total energy consumption, it is the worst scenario in terms of throughput. So, we need to use different metrics in order to trade off performance and energy. Hence, we introduce the energy per job as defined in Equation 5.3. Figure 6.23 shows the energy consumption per job in all scenarios. The energy per job is low when we have a high arrival rate. In contrast, energy consumption per job increases when the arrival rate decreases.

In terms of energy per job, when servers are identical, or $server2$ faster, the shortest queue is the best strategy as long as the incoming jobs rate does not exceed 70% of the system capacity (arrival rate $\lambda < 15$). Furthermore, when the incoming task rate is more than 70 % of the system capacity, TAGS will be the best strategy. This is because the timeout mechanism in TAGS reduces the job losses at the first queue. As a result, TAGS produces more jobs at a high arrival rate than the shortest queue and weighted random, so, as more jobs are produced, the energy per job decreases, as per Equation 5.3.

It is worth noting that TAGS performed better than the shortest queue and weighted random when the second server was slower. Figure 6.23 shows that TAGS consumes less energy at arrival rates λ = (15, 17, 19) when $Server2$ speed is less by 30%, but when speed difference increases to 40% TAGS starts to be the best from lower rate (starting from λ =13. In fact, TAGS at a slower speed and higher arrival rates show a higher percentage of energy reduction per job than the shortest queue. For example, for arrival rates λ =(13, 15, 17, 19) the difference between TAGS and shortest queue consumption is approximately (1.66%, 6.12%, 10.44%, 13.49%) respectively when $Server2$ speed less by 30%, but when the $Server2$ speed reduced by extra 10% the difference increased to ≈ (5.44% , 10.44%, 14.66%, 17.989%) for the same arrival rates. This behaviour is related to the TAGS mechanism when $Server2$ is slower that does not prevent TAGS from producing more jobs as the first server will not be blocked by any job that needs processing time longer than the timeout value. So, more jobs will be served by $Server1$. On the contrary, the shortest queue and weighted random do not have a mechanism to mitigate the possibility of having two long jobs occupying both servers, which means the system's throughput reduces.

6.7 Conclusion

In this chapter, we studied the performance and energy consumption of the TAGS algorithm in a heterogeneous environment and compared the performance and energy results with the shortest queue and weighted random results. Our approach for calculating the energy consumption focused on the CPU energy consumption and neglected other server components. We used the processor performance states (P-state) values to find the energy consumption by the server. We considered nine combinations of servers. The first combination is that all servers are identical. In contrast, in the second combination, the second server is slower than the second server by 10%, 20%, 30% and 40%. In the same fashion, the second server is faster than the first server in the last scenario.

The analysis of the result shows that the TAGS algorithm is sensitive to the time out value. TAGS can perform well and increase the system throughput in all scenarios when the time out well-tuned is not very long nor very short. The very long time out causes a low throughput. In the same fashion, if the timeout is very short, the throughput is decreased.

The TAGS algorithm consumes more energy than the shortest queue and the weighted random in all servers combinations concerning the total energy consumption.

The energy per job can be used to identify the best time out value for TAGS that produces the highest possible throughput with minimal impact on energy consumption.

There is a correlation between energy consumption and throughput, as well as the server speed. The same discussion presented previously in Section 5.3.2 regarding the correlation between the throughput, total energy consumption and energy consumption per job in a homogeneous environment is applied to the results in the heterogeneous environment in this chapter. In addition, this chapter shows a correlation between considered metrics (throughput, total energy consumption and energy per job) and the performance difference between servers. The throughput of the system is increased when the second server is faster. On the other hand, under the same setup, the total energy decreased. This decrease in total energy can be explained as the second server under the TAGS scheme receives longer jobs, which takes longer to be processed. So, if the second server is fast, the processing time will be reduced, leading to reduce total energy consumption. Moreover, producing higher throughput reduce the energy consumption per job. However, suppose the second server is less efficient in performance, and the configuration allows more jobs into the first server to be timed out and transferred to the second server. In that case, the total energy consumption will be increased due to being busy processing more jobs. Moreover, the throughput will be reduced because more jobs will be dropped from the second queue as the queue will be full. As a result of the increase in total energy consumption and the decrease in throughput, the energy per job will be increased.

7

Conclusion and future work

Contents

7.1 Conclusion

This thesis considered the trade-off between performance and energy consumption by examining the impact on performance and energy consumption of two main concepts. (1) single queue multi-servers with dynamically powered on or off servers in a homogeneous environment, (2) individual queue server pairs, and how work is distributed amongst queues in both a homogeneous and heterogeneous environment.PEPA was used as the modelling language in this thesis, in contrast to previous literature studies that used simulation, mathematical models or experiments in a real environment. Furthermore, unlike previous research in the literature, we considered the impact of heterogeneity on energy and performance in the individual queue server pairs experiments, where the focus was on the TAGS scheduling algorithm. This thesis contributes by developing PEPA models for some job scheduling algorithms and a dynamic servers allocation policy in queueing-type systems. In particular, this thesis demonstrates that PEPA can be used to compare performance and energy consumption for queuing type systems. The PEPA models can be generalised to any context with a comparable structural model.
Apart from PEPA models, the energy model proposed in this thesis depends on the processor P-state value, making it a generic model in the context of CPU energy consumption. Therefore, it applies to any discrete-state model of any type. This energy model does not need to be a PEPA model or a Markov chain. Furthermore, the energy model can be used in real-world experiment scenarios or simulation experiments, and it is not necessary to use formal modelling in these cases.

In Chapter 3 a policy to limit power consumption by servers in data centres was discussed and modelled in PEPA. The numerical experiments were carried out under different scenarios. There was more than one combination of servers in each scenario to determine a better one under each operating condition. The conducted experiments showed that no one combination of servers under this policy could perform well in all situations.

In chapter 5, energy consumption by the TAGS policy was studied, and compared with that of the shortest queue strategy and random allocation. The focus on these policies was in the case of the high variability in workload for a homogeneous environment. The analysis of energy consumption under the exponential and hyper-exponential distribution led to the conclusion that the TAGS mechanism consumes more energy than the other two policies regarding total energy consumption. The energy consumed per job followed the same trend under the exponential distribution. In contrast, TAGS consumed less energy per job under the hyper-exponential service demand. This finding led to further investigation of these policies in a heterogeneous environment in chapter 6.

Chapter 6 studied the TAGS algorithm's performance and energy consumption in a heterogeneous environment and compared the performance and energy results with the shortest queue and weighted random results. The analysis of the result showed that the TAGS algorithm is sensitive to the time out value. While in terms of performance, TAGS was observed to perform well and increase the system throughput in all scenarios when the time out is well-tuned and not very long nor very short. The very long time out causes a low throughput. In the same fashion, if the time out is too short, the throughput is also decreased. In terms of the total energy consumption, the TAGS algorithm consumes more energy than the shortest queue and the weighted random in all servers combinations. While in terms of energy per job, when servers are identical, or server2 is faster, it was observed that the shortest queue is the best strategy as long as the incoming jobs rate does not exceed 70% of the system capacity (arrival rate $\lambda < 15$). Furthermore, when the incoming task rate is more than 70% of the system capacity, the TAGS was observed to be the best. Consequently, the energy per job can be used to identify the best time out value for TAGS that produces the highest possible throughput with minimal impact on energy consumption.

7.2 Limitations

In Chapter 3 the model in PEPA has some limitations. Firstly, the model depends only on the arrival period's length to switch on or off more servers, indicating dependence on only one factor for decision-making. Secondly, we assumed that all servers in any state consume the same amount of energy. Finally, the experiment configurations in some scenarios changed only one factor and fixed the others, which may not be the situation in practice.

For TAGS experiments in Chapters 5 and Chapter 6 the processor performance states (P-states) value was relied upon to calculate energy consumption. The main downside of this is that energy consumption by other server components, such as hard disk and memory, was neglected. Moreover, in a heterogeneous environment, the first and second node variation was in performance only. In contrast, it was assumed there was no difference in energy consumption between nodes regardless of the performance difference. However, when there is a performance difference, usually there is a difference in energy consumption between servers.

7.3 Future work

For the dynamic server allocation approach in Chapter 3, the future work in this direction could look at an extended model to consider more servers with extended queue size. Also, each server state could be assigned different energy costs to represent the actual cost of energy consumption as close as possible. Finally, changing more than one factor would be useful instead of changing only one factor, as was done in some scenarios. Another scenario that could be studied is when the job size is variable. In addition, it will be valuable to compare the high/low policy with other existing policies to find out which policy is more useful under different scenarios. There are many other options for managing servers that remain to be explored using the approach presented in Chapter 3 and further investigation about them could open new possibilities.

For TAGS, it is worth looking at the possibility of improving the throughput without increasing energy consumption. So, it is worthwhile evaluating a system working with a hybrid scheduling approach. The idea could be a system working under the shortest queue strategy up to a level when both queues reach a waiting job threshold. Then, the system switches to using TAGS as a scheduling approach by identifying the node that is processing the current job for a long time, and its queue is longer than the other node's queue. Then, classify this other node as the subsequent node that will receive the long jobs.

Another option is to investigate the benefit of changing the TAGS algorithm behaviour. Thus, a threshold T is introduced to the first queue instead of killing the job when the timeout is triggered. So, as long as the number of waiting jobs in the queue is still less than T, the first server keeps processing the job even when the time out is triggered. Meanwhile, the second node could be switched off or moved to a low power consumption state and only switched back when the threshold triggered.

This approach could be helpful to reduce energy consumption by TAGS when the arrival rate is low.

Bibliography

[1] AMD. Power and colling in the data center. 2005.

[2] Victor Avelar, Dan Azevedo, Alan French, and Emerson Network Power. Pue: a comprehensive examination of the metric. *White paper*, 49, 2012.

[3] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[4] David J Brown and Charles Reams. Toward energy-efficient computing. *Communications of the ACM*, 53(3):50–58, 2010.

[5] Richard Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431. *Lawrence Berkeley National Laboratory.*

[6] P Buchanan, Vincent Yampolsky, and Bill Buchanan. Comparison of the power consumption and carbon footprint of a cloud infrastructure against standard desktops. *Computer*, 55:77, 2011.

[7] J. Chen and C. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*, pages 28–38, 2007.

[8] David Cole. Data center energy efficiency–looking beyond pue. *No Limits Software, White Paper*, 4, 2011.

[9] John Conti, Paul Holtberg, Jim Diefenderfer, Angelina LaRose, James T Turnure, and Lynn Westfall. International energy outlook 2013 with projections to 2040. *USDOE Energy Information Administration EIA, Washington, US*, 2013.

[10] Intel Corporation. Enhanced intel speedstep technology for the intel pentium m processor. Technical report, Intel Corporation., 2004.

[11] Susanna Donatelli, Marina Ribaudo, and Jane Hillston. A comparison of performance evaluation process algebra and generalized stochastic petri nets. In *Proceedings 6th International Workshop on Petri Nets and Performance Models*, pages 158–168. IEEE, 1995.

[12] Abdullah Elewi, Mohamed Shalan, Medhat Awadalla, and Elsayed M. Saad. Energy-efficient task allocation techniques for asymmetric multiprocessor embedded systems. *ACM Trans. Embed. Comput. Syst.*, 13(2s), January 2014.

[13] Enerdata. Global Energy Statistical Yearbook 2014.

[14] Enerdata. 2010 to 2015 government policy: greenhouse gas emissions.

[15] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. *SIGARCH Comput. Archit. News*, 35(2):13–23, June 2007.

[16] Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbitter. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications.* Springer, 2014.

[17] Frederick Eugene Fowle. *Smithsonian physical tables*, volume 2539. Smithsonian institution, 1921.

[18] Anshul Gandhi, Varun Gupta, Mor Harchol-Balter, and Michael A Kozuch. Optimality analysis of energy-performance trade-off for server farm management. *Performance Evaluation*, 67(11):1155–1171, 2010.

[19] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. Optimal power allocation in server farms. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '09, page 157–168, New York, NY, USA, 2009. Association for Computing Machinery.

[20] Mahdi Ghamkhari and Hamed Mohsenian-Rad. Energy and performance management of green data centers: A profit maximization approach. *IEEE Transactions on Smart Grid*, 4(2):1017–1025, 2013.

[21] Stephen Gilmore, Jane Hillston, Leıla Kloul, and Marina Ribaudo. Pepa nets: a structured performance modelling formalism. *Performance Evaluation*, 54(2):79–104, 2003.

[22] TG Grid. Green grid metrics: Describing datacenter power efficiency. *Online at http://www. thegreengrid. org/gg_content/Green_Grid_Metrics_WP. pdf*, 2007.

[23] Mor Harchol-Balter. Task assignment with unknown duration. *Journal of the ACM*, 49(2):260–288, March 2002.

[24] Jane Hillston. *A compositional approach to performance modelling*, volume 12. Cambridge University Press, 2005.

[25] Jane Hillston. Fluid flow approximation of pepa models. In *Second International Conference on the Quantitative Evaluation of Systems (QEST'05)*, pages 33–42, 2005.

[26] Jane Hillston. Tuning Systems: From Composition to Performance. *The Computer Journal*, 48(4):385–400, 01 2005.

[27] Jane Hillston and Stephen Gilmore. The pepa eclipse plug-in. *Laboratory for Foundations of Computer Science, The University of Edinburgh. Online at http://www.dcs.ed.ac.uk/pepa/downloads/*, Last accessed July 2021.

[28] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based optimization of power performance by using dynamic voltage scaling on a pc cluster. In *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, pages 8 pp.–, 2006.

[29] Ronald Lynn Jacobs. *Developing a Research Problem and Purpose Statement*, pages 125–142. Jossey-Bass, United States, March 2011.

[30] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. Applying queue theory for modeling of cloud computing: A systematic review. *Concurrency and Computation: Practice and Experience*, 31(17):e5186, 2019. e5186 CPE-18-0152.R1.

[31] Yichao Jin, Yonggang Wen, and Qinghua Chen. Energy efficiency and server virtualization in data centers: An empirical investigation. In *2012 Proceedings IEEE INFOCOM Workshops*, pages 133–138. IEEE, 2012.

[32] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 39–50, 2010.

[33] Milena Krasich. How to estimate and use mttf/mtbf would the real mtbf please stand up? In *2009 Annual Reliability and Maintainability Symposium*, pages 353–359, 2009.

[34] Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15, March 2009.

[35] Ricardo Lent. Analysis of an energy proportional data center. *Ad Hoc Networks*, 2015.

[36] Christopher Malone and Christian Belady. Metrics to characterize data center it equipment energy use, proceedings of digital power forum, richardson, tx. 09 2006.

[37] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V Vasilakos. Cloud computing: Survey on energy efficiency. *ACM Computing Surveys (CSUR)*, 47(2):33, 2015.

[38] Vimal Mathew. *Energy-Efficient Content Delivery Networks*. PhD thesis, University of Massachusetts Amherst, 2015.

[39] D Anthony Miles. A taxonomy of research gaps: Identifying and defining the seven research gaps. In *Doctoral Student Workshop: Finding Research Gaps-Research Methods and Strategies, Dallas, Texas*, 2017.

[40] Isi Mitrani. Service center trade-offs between customer impatience and power consumption. *Performance Evaluation*, 68(11):1222–1231, 2011.

[41] Isi Mitrani. Trading power consumption against performance by reserving blocks of servers. In *European Workshop on Performance Engineering*, pages 1–15. Springer, 2012.

[42] Thai Ha Nguyen, Matthew Forshaw, and Nigel Thomas. Operating Policies for Energy Efficient Dynamic Server Allocation. *Electronic Notes in Theoretical Computer Science*, 318:159 – 177, 2015.

[43] Choman Othman Abdullah and Nigel Thomas. Modelling unfairness in ieee 802.11g networks with variable frame length. In Sabine Wittevrongel and Tuan Phung-Duc, editors, *Analytical and Stochastic Modelling Techniques and Applications*, pages 223–238, Cham, 2016. Springer International Publishing.

[44] Tuan Phung-Duc. Impatient customers in power-saving data centers. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications*, pages 185–199. Springer, 2014.

[45] Tuan Phung-Duc. Controllable setup queue for energy-aware server. In *31st UK Performance Engineering Workshop 17 September 2015*, page 96, 2015.

[46] Emerson Network Power. Energy logic: Reducing data center energy consumption by creating savings that cascade across systems, a white paper from the experts in business-critical continuity, 2013. *RCI HI RCI*, 20:40–60.

[47] S Ricciardi, D Careglio, G Santos-Boada, J Sole-Pareta, U Fiore, and F Palmieri. Saving Energy in Data Center Infrastructures. *Data Compression, Communications and Processing (CCP), 2011 First International Conference on*, pages 265–270, 2011.

[48] Richard Sawyer. Calculating total power requirements for data centers. *White Paper, American Power Conversion 562*, 2004.

[49] Meenakshi Sharma and Pankaj Sharma. Performance evaluation of adaptive virtual machine load balancing algorithm. *Performance Evaluation*, 3(2), 2012.

[50] Joris Slegers, Isi Mitrani, and Nigel Thomas. Static and dynamic server allocation in systems with on/off sources. *Annals of Operations Research*, 170(1):251–263, September 2009.

[51] Joris Slegers, Isi Mitrani, and Nigel Thomas. Evaluating the optimal server allocation policy for clusters with on/off sources. *Performance Evaluation*, 66(8):453–467, 2009. Selected papers of the Fourth European Performance Engineering Workshop (EPEW) 2007 in Berlin.

[52] Joris Slegers, Nigel Thomas, and Isi Mitrani. Dynamic server allocation for power and performance. In *SPEC International Performance Evaluation Workshop*, pages 247–261. Springer, 2008.

[53] S. Srivastava and I. Banicescu. Pepa based performance modeling for robust resource allocations amid varying processor availability. In *2018 17th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 61–68, 2018.

[54] George Terzopoulos and Helen D. Karatza. Power-aware bag-of-tasks scheduling on heterogeneous platforms. *Cluster Computing*, 19(2):615–631, 2016.

[55] The Equipment Energy Efficiency E3 Program. Energy efficiency policy options for australian and new zealand data centres. 2014.

[56] Nigel Thomas. Comparing job allocation schemes where service demand is unknown. *Journal of Computer and System Sciences*, 74(6):1067–1081, 2008.

[57] Mirco Tribastone, Adam Duguid, and Stephen Gilmore. The pepa eclipse plugin. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):28–33, 2009.

[58] Tien Van Do. Comparison of allocation schemes for virtual machines in energy-aware server farms. *The Computer Journal*, page bxr007, 2011.

[59] Otto VanGeet, W Lintner, and B Tschudi. Femp best practices guide for energy-efficient data center design. *National Renewable Energy Laboratory*, 2011.

[60] Lizhe Wang and Samee U. Khan. Review of performance metrics for green data centers: a taxonomy study. *The Journal of Supercomputing*, 63(3):639–656, March 2013.

[61] Ghaith Warkozek, Elisabeth Drayer, Vincent Debusschere, and Seddik Bacha. A new approach to model energy consumption of servers in data centers. In *Industrial Technology (ICIT), 2012 IEEE International Conference on*, pages 211–216. IEEE, 2012.

[62] Wayne Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14(1):181–189, 1977.

[63] Xian Yang, Rui Han, Yike Guo, Jeremy Bradley, Benita Cox, Robert Dickinson, and Richard Kitney. Modelling and performance analysis of clinical pathways using the stochastic process algebra pepa. 13, 2012.

[64] Yishi Zhao and Nigel Thomas. Efficient solutions of a pepa model of a key distribution centre. *Performance Evaluation*, 67(8):740–756, 2010. Special Issue on Software and Performance.

A

Energy data in heterogeneous environment

In this Appendix, the full data of the total energy consumption and energy per job for all servers combinations in the heterogeneous environment presented. Moreover, the active energy consumption and idle energy consumption for both server1 and server2 presented.

A.1 Total energy consumption

Table A.1: Total energy consumption for all combinations

Combination	**Arrival rate**	**TAGS**	**Shortest**	**Random**
S1 S2 Identical	5	123.3466754	115.3948424	113.7971632
	7	138.4718742	126.8568209	123.5679039
	9	153.1826067	137.7139113	132.4807403
	11	166.6668144	147.8562245	140.6300522
	13	178.3581929	157.2075605	148.0993625
	15	185.8435485	165.6883616	154.9596555
	17	190.7528244	173.1984271	161.2697226
	19	193.0821488	179.6565567	167.076584
S2 Faster by 10%	5	121.3369726	114.7352549	112.572252
	7	135.6947108	126.0001329	122.0016782
	9	151.0355433	136.7185973	130.6382015
	11	164.2325743	146.7805587	138.5646088
	13	175.660711	156.1054658	145.8555781
	15	184.0350522	164.6081708	152.5755136
	17	188.529631	172.1817483	158.7786567
	19	191.1538358	178.735209	164.5094697
S2 Faster by 20%	5	120.3876119	114.1742603	111.446842
	7	134.3610487	125.265107	120.5567738
	9	148.1123478	135.8573882	128.9320231
	11	161.166102	145.8423375	136.6453203
	13	173.0781593	155.136926	143.7635541
	15	181.4089889	163.6521626	150.3451984
	17	187.0050659	171.2760733	156.4401981
	19	189.7601614	177.9096842	162.0902838

Table A.1 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	118.9018413	113.6912926	110.4093556
	7	132.2982542	124.6275361	119.2197005
	9	145.5609044	135.1048735	127.347711
	11	159.011026	145.0167726	134.857358
	13	170.4635738	154.2790097	141.808655
	15	179.4372405	162.8000152	148.2547457
	17	184.615755	170.4640384	154.2415815
	19	187.7261459	177.1655774	159.8081514
S2 Faster by 40%	5	117.6278942	113.27113	109.4499217
	7	131.1272118	124.069229	117.9788672
	9	144.0454077	134.4416732	125.8727156
	11	156.5162545	144.2846915	133.187778
	13	168.1140563	153.5137483	139.9779842
	15	176.8293697	162.0356159	146.2916906
	17	183.0086146	169.7317346	152.171171
	19	185.7015755	176.4912526	157.6527548
S2 Slower by 10%	5	125.7997743	116.9883459	115.135287
	7	141.8170304	128.9725038	125.2712966
	9	157.0231683	140.2550233	134.4765267
	11	170.2669722	150.7189781	142.8587781
	13	180.8334467	160.2762688	150.5116314
	15	188.1264057	168.8239711	157.5133302
	17	192.0042135	176.248763	163.9273531
	19	194.8819429	182.4998831	169.8028253

Table A.1 continued from previous page

Combination	**Arrival rate**	**TAGS**	**Shortest**	**Random**
S2 Slower by20 %	5	127.4315248	118.8905718	116.6029622
	7	144.0841814	131.4608925	127.1305675
	9	159.7445784	143.2005021	136.6453196
	11	173.7255706	153.9821885	145.2706231
	13	183.1902786	163.6923737	153.1114989
	15	188.7223278	172.1997159	160.2541102
	17	193.7586729	179.4146815	166.7665228
	19	196.4809553	185.3673467	172.6990523
S2 Slower by 30%	5	131.1314093	121.2001181	118.2198361
	7	148.9385026	134.4274774	129.1679934
	9	164.5695234	146.6463388	139.0104053
	11	176.4494806	157.7060789	147.8886923
	13	184.8885731	167.448219	155.9208629
	15	190.9218559	175.7509842	163.2015403
	17	194.5496867	182.6300519	169.802831
	19	197.8068975	188.2250498	175.7745887
S2 Slower by 40%	5	133.8700926	124.0619436	120.0096934
	7	152.5218638	138.0185349	131.4102384
	9	167.9952999	150.7062968	141.5993276
	11	178.9717536	161.9173122	150.7399234
	13	185.9346496	171.4740654	158.9646882
	15	191.4899369	179.3939044	166.3769241
	17	195.6124423	185.8561197	173.0514262
	19	198.4134371	191.0695245	179.0348207

A.2 Energy per job

Table A.2: Energy per job in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	24.66960081	23.26579995	24.23262654
	7	19.78468102	18.51245145	19.58464037
	9	17.03662684	15.92934639	17.02582147
	11	15.21034852	14.32708374	15.40843088
	13	13.87406254	13.25058887	14.29503127
	15	12.71472842	12.49102703	13.48295047
	17	11.79776312	11.94085354	12.86559449
	19	11.05059962	11.53782354	12.3817098
S2 Faster by 10%	5	24.26761134	23.10698473	23.85786187
	7	19.38741667	18.34098903	19.20274484
	9	16.79546751	15.74709295	16.64019492
	11	14.97992645	14.13584278	15.0204626
	13	13.64997247	13.05178289	13.9053403
	15	12.56894806	12.28566436	13.09178801
	17	11.63115994	11.55272912	12.47298575
	19	10.90121553	11.32130504	11.98749631
S2 Faster by 20%	5	24.07768392	22.97293383	23.51831008
	7	19.19634685	18.19547327	18.85633273
	9	16.46807304	15.5916351	16.29019525
	11	14.69470334	13.9719959	14.66823662
	13	13.43671345	12.88081611	13.55151362
	15	12.37040625	12.10851904	12.73664084
	17	11.50950226	11.54699353	12.11659701
	19	10.78364491	11.13375983	11.62979382

Table A.2 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	23.78049667	22.85833879	23.20933571
	7	18.90133344	18.07049957	18.54075127
	9	16.18308677	15.45753526	15.97115489
	11	14.49339114	13.83011149	14.34706755
	13	13.2240794	12.73227851	13.2288415
	15	12.21819348	11.95419902	12.41276372
	17	11.33653835	11.38755579	11.79162904
	19	10.63190054	10.96977892	11.30372021
S2 Faster by 40%	5	23.52569039	22.75930125	22.92707397
	7	18.73384182	17.96206129	18.25212104
	9	16.01331152	15.34072866	15.67918295
	11	14.2622554	13.70609994	14.05305122
	13	13.0326685	12.60207192	12.93340226
	15	12.02536534	11.81859942	12.11621012
	17	11.2142832	11.24719675	11.49409862
	19	10.48335869	10.82521748	11.00523689
S2 Slower by 10%	5	25.16034005	23.62016148	24.6482212
	7	20.26372024	18.88035223	20.00765704
	9	17.46806519	16.30918809	17.45273172
	11	15.54828375	14.71776741	15.83782248
	13	14.08454263	13.65236738	14.72630184
	15	12.89513807	12.90576209	13.91589542
	17	11.90668309	12.37072605	13.30027162
	19	11.19452498	11.98331307	12.81840921

Table A.2 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	25.4868324	24.04641406	25.11151407
	7	20.58887931	19.31884834	20.47868835
	9	17.77623246	16.75868903	17.92784467
	11	15.87935333	15.17786857	16.31559023
	13	14.28979336	14.12468719	15.20615476
	15	12.96206799	13.39310158	14.39770403
	17	12.04706247	12.87412237	13.78422419
	19	11.32826672	12.50094043	13.30499956
S2 Slower by 30%	5	26.22712718	24.56859849	25.63099756
	7	21.28540103	19.85004279	21.00625844
	9	18.32297154	17.29861652	18.45971919
	11	16.14523593	15.72783088	16.8503415
	13	14.44606703	14.6882434	15.74327241
	15	13.14207327	13.97209365	14.93718108
	17	12.12977256	13.46627186	14.32645783
	19	11.44693715	13.10239657	13.85080265
S2 Slower by 40%	5	26.7753772	25.2227189	26.21728664
	7	21.80145519	20.50617921	21.60102394
	9	18.71818616	17.95894271	19.05906129
	11	16.40108049	16.39717659	17.45285487
	13	14.5557116	15.37024608	16.34855091
	15	13.20879511	14.66383812	15.54542139
	17	12.2275577	14.16252014	14.93839793
	19	11.52246547	13.80012793	14.46775822

A.3 Server 1 active energy consumption

Table A.3: Server 1 active energy consumption in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	47.47023761	46.41730753	45.42060477
	7	55.38405832	53.56779548	51.51602391
	9	63.27886885	60.34092750	57.07624451
	11	71.08917045	66.66814993	62.16014486
	13	78.20713018	72.50192594	66.81983020
	15	84.33237492	77.79262388	71.09958360
	17	89.60479552	82.47773406	75.03608189
	19	93.03707833	86.50660025	78.65865824
S2 Faster by 10%	5	47.47023761	46.26589187	44.65645111
	7	55.38405832	53.31982731	50.53894326
	9	62.92752081	60.00195320	55.92678754
	11	70.68199983	66.25205650	60.87163010
	13	77.77798224	72.02783024	65.42005844
	15	84.33237492	77.28404813	69.61225069
	17	89.17715864	81.96183950	73.48204509
	19	92.61007030	86.01070172	77.05717905
S2 Faster by 20%	5	47.27083461	46.13735981	43.95435715
	7	55.10592478	53.10749536	49.63753065
	9	62.92752081	59.70927092	54.86237746
	11	70.68199983	65.88994499	59.67426999
	13	77.77798224	71.61215167	64.11493618
	15	83.89189119	76.83495515	68.22085455
	17	89.17715864	81.50321018	72.02318294
	19	92.61007030	85.56714913	75.54795463

Table A.3 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	47.27083461	46.02690631	43.30715233
	7	55.10592478	52.92366402	48.80343752
	9	62.92752081	59.45405435	53.87405182
	11	70.28140146	65.57201824	58.55890324
	13	77.34271151	71.24479618	62.89543090
	15	83.89189119	76.43555224	66.91678818
	17	88.72797565	81.09284215	70.65164178
	19	92.16212415	85.16804164	74.12431464
S2 Faster by 40%	5	47.27083461	45.93098067	42.70860376
	7	54.83755587	52.76297795	48.02933625
	9	62.58650370	59.22957927	52.95386719
	11	70.28140146	65.29070619	57.51732584
	13	77.34271151	70.91786130	61.75335609
	15	83.43481577	76.07808051	65.69212468
	17	88.72797565	80.72352651	69.36000399
	19	91.69452731	84.80699912	72.77965782
S2 Slower by 10%	5	47.47023761	46.73698182	46.25538541
	7	55.38405832	54.07690027	52.57867544
	9	63.27886885	61.02715982	58.32130414
	11	71.08917045	67.50869404	63.55052237
	13	78.20713018	73.46764074	68.32470993
	15	84.75614915	78.84540851	72.69267858
	17	89.60479552	83.57079361	76.69402905
	19	93.44201998	87.59505669	80.35940783

Table A.3 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	47.68354473	47.11855128	47.17100152
	7	55.68074940	54.67618315	53.73859032
	9	63.65016269	61.82583458	59.67431572
	11	71.08917045	68.47747757	65.05516308
	13	78.63369433	74.56758623	69.94664928
	15	84.33237492	80.02390936	74.40252678
	17	90.00984062	84.77009410	78.46525804
	19	93.82395351	88.76917719	82.16623219
S2 Slower by 30%	5	47.68354473	47.58195420	48.17965233
	7	55.68074940	55.39227412	55.00959311
	9	63.65016269	62.76800120	61.14972173
	11	71.51130386	69.60572354	66.68838725
	13	78.63369433	75.82384578	71.69920842
	15	85.16399444	81.33628696	76.24121530
	17	90.00984062	86.07600170	80.35939121
	19	94.18215214	90.02898363	84.08483694
S2 Slower by 40%	5	47.92171771	48.15677811	49.29625525
	7	56.01030923	56.26381164	56.40841823
	9	64.05653184	63.89701010	62.76482162
	11	71.51130386	70.92898205	68.46712856
	13	78.63369433	77.25126103	73.59809976
	15	85.16399444	82.78352422	78.22217951
	17	90.39166227	87.48881090	82.38602846
	19	94.18215214	91.37775471	86.11873391

A.4 Server 2 active energy consumption

Table A.4: Server 2 active energy consumption in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	55.28578604	46.41730753	45.42060477
	7	66.24349813	53.56779548	51.51602391
	9	76.70309547	60.34092750	57.07624451
	11	85.71688131	66.66814993	62.16014486
	13	93.18611714	72.50192594	66.81983020
	15	96.40026256	77.79262388	71.09958360
	17	97.25308779	82.47773406	75.03608189
	19	96.72707558	86.50660025	78.65865824
S2 Faster by 10%	5	52.77830345	45.74576363	44.65645113
	7	62.77846400	52.74688407	50.53894327
	9	74.37557773	59.43806016	55.92678755
	11	83.08687926	65.74214780	60.87163012
	13	90.24964854	71.60095102	65.42005847
	15	94.14382300	76.95345824	69.61225071
	17	94.90687236	81.72513045	73.48204511
	19	94.74815023	85.85294412	77.05717908
S2 Faster by 20%	5	51.79320032	45.17434939	43.95438432
	7	61.39260292	52.04213287	49.63756559
	9	70.72834112	58.65622199	54.86241880
	11	79.26087781	64.93365183	59.67431657
	13	87.02742918	70.80819386	64.11498703
	15	91.30779840	76.20975099	68.22090885
	17	93.00469045	81.05375972	72.02323997
	19	93.00927905	85.26649916	75.54801374

Table A.4 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	49.93942177	44.68220971	43.30712955
	7	58.81887852	51.43047446	48.80340811
	9	67.54493527	57.97253489	53.87401691
	11	76.97261317	64.22153098	58.55886379
	13	84.20051233	70.10513731	62.89538771
	15	88.84767111	75.54593970	66.91674192
	17	90.47275831	80.45096136	70.65159305
	19	90.91940790	84.73719330	74.12426399
S2 Faster by 40%	5	48.34993293	44.25390350	42.70860375
	7	57.62615155	50.89456730	48.02933622
	9	65.99508497	57.36954273	52.95386716
	11	73.85991607	63.58943400	57.51732581
	13	81.26904701	69.47726451	61.75335606
	15	86.05093673	74.94967937	65.69212464
	17	88.46754808	79.90659021	69.36000395
	19	88.86097206	84.25688870	72.77965777
S2 Slwer by 10%	5	58.34648866	48.08582883	46.25538544
	7	70.41721012	55.69840332	52.57867548
	9	81.49491913	62.82521069	58.32130418
	11	90.20875585	69.39942992	63.55052242
	13	96.27446212	75.36500244	68.32470998
	15	98.82478242	80.65210237	72.69267864
	17	98.81443121	85.19054279	76.69402911
	19	98.56771586	88.96572876	80.35940789

Table A.4 continued from previous page

Combination	**Arrival rate**	**TAGS**	**Shortest**	**Random**
S2 Slower by20%	5	60.16909747	50.07764437	47.17097042
	7	72.94921663	58.20385378	53.73855102
	9	84.51909668	65.70157528	59.67426999
	11	94.52400849	72.50211562	65.05511233
	13	98.78848942	78.52729089	69.94659470
	15	99.99208159	83.68547856	74.40246936
	17	100.59840451	87.94132164	78.46519872
	19	100.18085136	91.36930894	82.16617187
S2 Slower by 30%	5	64.78539992	52.49583520	48.17967421
	7	79.00589617	61.18913588	55.00962061
	9	90.53912383	69.05873855	61.14975356
	11	97.50046553	76.02012385	66.68842241
	13	100.90743160	81.95715529	71.69924605
	15	101.90478750	86.80397664	76.24125468
	17	101.58534310	90.64719386	80.35943165
	19	101.47701516	93.67502504	84.08487773
S2 Slower by 40%	5	67.96424985	55.49167733	49.29625525
	7	83.14725405	64.79811864	56.40841823
	9	94.40705569	72.99529156	62.76482162
	11	100.64747590	79.95117165	68.46712856
	13	102.21260891	85.55274133	73.59809976
	15	102.61357545	89.90196812	78.22217951
	17	102.52950908	93.25951149	82.38602846
	19	102.23378755	95.87527154	86.11873391

A.5 Server 1 and 2 total active energy consumption

Table A.5: Total active energy consumption in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	102.75602365	92.83461507	90.84120953
	7	121.62755644	107.13559097	103.03204783
	9	139.98196432	120.68185499	114.15248902
	11	156.80605176	133.33629987	124.32028972
	13	171.39324733	145.00385188	133.63966039
	15	180.73263748	155.58524775	142.19916720
	17	186.85788331	164.95546812	150.07216378
	19	189.76415392	173.01320049	157.31731649
S2 Faster by 10%	5	100.24854106	92.01165550	89.31290224
	7	118.16252232	106.06671137	101.07788653
	9	137.30309855	119.44001336	111.85357509
	11	153.76887909	131.99420429	121.74326021
	13	168.02763078	143.62878125	130.84011691
	15	178.47619792	154.23750638	139.22450140
	17	184.08403100	163.68696995	146.96409020
	19	187.35822053	171.86364584	154.11435813
S2 Faster by 20%	5	99.06403493	91.31170919	87.90874147
	7	116.49852769	105.14962823	99.27509624
	9	133.65586193	118.36549291	109.72479627
	11	149.94287764	130.82359682	119.34858656
	13	164.80541142	142.42034553	128.22992321
	15	175.19968959	153.04470614	136.44176340
	17	182.18184909	162.55696990	144.04642292
	19	185.61934935	170.83364829	151.09596837

Table A.5 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	97.21025638	90.70911603	86.61428189
	7	113.92480330	104.35413848	97.60684563
	9	130.47245608	117.42658924	107.74806873
	11	147.25401463	129.79354922	117.11776703
	13	161.54322384	141.34993349	125.79081861
	15	172.73956230	151.98149194	133.83353010
	17	179.20073395	161.54380351	141.30323483
	19	183.08153205	169.90523493	148.24857863
S2 Faster by 40%	5	95.62076754	90.18488417	85.41720751
	7	112.46370741	103.65754526	96.05867247
	9	128.58158866	116.59912201	105.90773435
	11	144.14131753	128.88014020	115.03465166
	13	158.61175852	140.39512581	123.50671215
	15	169.48575249	151.02775988	131.38424933
	17	177.19552373	160.63011672	138.72000794
	19	180.55549937	169.06388782	145.55931559
S2 Slower by 10%	5	105.81672627	94.82281065	92.51077085
	7	125.80126844	109.77530359	105.15735092
	9	144.77378798	123.85237051	116.64260832
	11	161.29792631	136.90812396	127.10104479
	13	174.48159230	148.83264318	136.64941991
	15	183.58093157	159.49751088	145.38535721
	17	188.41922674	168.76133640	153.38805816
	19	192.00973584	176.56078545	160.71881572

Table A.5 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	107.85264220	97.19619565	94.34197195
	7	128.62996603	112.88003693	107.47714134
	9	148.16925937	127.52740985	119.34858571
	11	165.61317894	140.97959319	130.11027541
	13	177.42218375	153.09487711	139.89324398
	15	184.32445651	163.70938792	148.80499614
	17	190.60824513	172.71141574	156.93045676
	19	194.00480487	180.13848613	164.33240406
S2 Slower by 30%	5	112.46894466	100.07778941	96.35932654
	7	134.68664558	116.58141000	110.01921372
	9	154.18928652	131.82673975	122.29947529
	11	169.01176939	145.62584739	133.37680966
	13	179.54112593	157.78100107	143.39845446
	15	187.06878193	168.14026361	152.48246998
	17	191.59518373	176.72319556	160.71882286
	19	195.65916730	183.70400867	168.16971468
S2 Slower by 40%	5	115.88596755	103.64845543	98.59251050
	7	139.15756328	121.06193028	112.81683647
	9	158.46358753	136.89230166	125.52964325
	11	172.15877976	150.88015371	136.93425712
	13	180.84630324	162.80400235	147.19619951
	15	187.77756988	172.68549234	156.44435902
	17	192.92117135	180.74832239	164.77205692
	19	196.41593969	187.25302625	172.23746783

A.6 Server 1 Idle energy consumption

Table A.6: Server 1 Idle energy consumption in heterogeneous environment

Combination	**Arrival rate**	**TAGS**	**Shortest**	**Random**
S1 S2 Identical	5	11.07108835	11.28011364	11.47797682
	7	9.50005460	9.86061498	10.26792802
	9	7.93279470	8.51602816	9.16412563
	11	6.38231132	7.25996229	8.15488125
	13	4.96927008	6.10185430	7.22985107
	15	3.75330037	5.05155693	6.38024412
	17	2.70663138	4.12147947	5.59877940
	19	2.02526237	3.32167809	4.87963375
S2 Faster by 10%	5	11.07108835	11.31017234	11.62967487
	7	9.50005460	9.90984106	10.46189586
	9	8.00254351	8.58332057	9.39231322
	11	6.46314191	7.34256422	8.41067428
	13	5.05446356	6.19597070	7.50773059
	15	3.75330037	5.15251824	6.67550608
	17	2.79152489	4.22389369	5.90728327
	19	2.11003104	3.42012274	5.19755577
S2 Faster by 20%	5	11.11067338	11.33568823	11.76905297
	7	9.55526904	9.95199272	10.64084225
	9	8.00254351	8.64142319	9.60361750
	11	6.46314191	7.41444978	8.64837150
	13	5.05446356	6.27849027	7.76682050
	15	3.84074421	5.24167116	6.95172289
	17	2.79152489	4.31493975	6.19689327
	19	2.11003104	3.50817579	5.49716359

Table A.6 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	11.11067338	11.35761521	11.89753461
	7	9.55526904	9.98848650	10.80642453
	9	8.00254351	8.69208820	9.79981766
	11	6.54266779	7.47756388	8.86979158
	13	5.14087252	6.35141685	8.00891393
	15	3.84074421	5.32095972	7.21060319
	17	2.88069568	4.39640508	6.46916852
	19	2.19895630	3.58740571	5.77978137
S2 Faster by 40%	5	11.11067338	11.37665815	12.01635711
	7	9.60854502	10.02038553	10.96009736
	9	8.07024146	8.73665049	9.98249063
	11	6.54266779	7.53340931	9.07656316
	13	5.14087252	6.41631922	8.23563603
	15	3.93148178	5.39192420	7.45372062
	17	2.88069568	4.46972078	6.72558153
	19	2.29178256	3.65907905	6.04671960
S2 Slower by 10%	5	11.07108835	11.21665263	11.31225806
	7	9.50005460	9.75954865	10.05697285
	9	7.93279470	8.37979887	8.91695921
	11	6.38231132	7.09309938	7.87886666
	13	4.96927008	5.91014279	6.93110576
	15	3.66917367	4.84256051	6.06398652
	17	2.70663138	3.90448777	5.26964749
	19	1.94487428	3.10560018	4.54200480

Table A.6 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	11.02874311	11.14090432	11.13049202
	7	9.44115615	9.64058037	9.82670917
	9	7.85908628	8.22124777	8.64836242
	11	6.38231132	6.90077867	7.58016879
	13	4.88458953	5.69178410	6.60912203
	15	3.75330037	4.60860719	5.72455133
	17	2.62622275	3.66640535	4.91802712
	19	1.86905370	2.87251644	4.18331812
S2 Slower by 30%	5	11.02874311	11.04891063	10.93025695
	7	9.44115615	9.49842361	9.57439258
	9	7.85908628	8.03421099	8.35546816
	11	6.29851035	6.67680184	7.25594483
	13	4.88458953	5.44239430	6.26120796
	15	3.58820915	4.34807697	5.35953905
	17	2.62622275	3.40715954	4.54200810
	19	1.79794492	2.62242252	3.80244105
S2 Slower by 40%	5	10.98146155	10.93479789	10.70859147
	7	9.37573268	9.32540796	9.29670099
	9	7.77841480	7.81008270	8.03484219
	11	6.29851035	6.41411156	6.90283313
	13	4.88458953	5.15902706	5.88424436
	15	3.58820915	4.06077470	4.96628252
	17	2.55042438	3.12669186	4.13968466
	19	1.79794492	2.35466754	3.39867643

A.7 Server 2 Idle energy consumption

Table A.7: Server 2 Idle energy consumption in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	9.51956339	11.28011364	11.47797682
	7	7.34426321	9.86061498	10.26792802
	9	5.26784764	8.51602816	9.16412563
	11	3.47845130	7.25996229	8.15488125
	13	1.99567553	6.10185430	7.22985107
	15	1.35761066	5.05155693	6.38024412
	17	1.18830973	4.12147947	5.59877940
	19	1.29273248	3.32167809	4.87963375
S2 Faster by 10%	5	10.01734315	11.41342702	11.62967487
	7	8.03213393	10.02358045	10.46189586
	9	5.72990119	8.69526334	9.39231322
	11	4.00055335	7.44379014	8.41067428
	13	2.57861662	6.28071380	7.50773059
	15	1.80555393	5.21814619	6.67550608
	17	1.65407510	4.27088464	5.90728326
	19	1.68558426	3.45144042	5.19755577
S2 Faster by 20%	5	10.21290359	11.52686288	11.76904758
	7	8.30725194	10.16348606	10.64083531
	9	6.45394234	8.85047208	9.60360929
	11	4.76008249	7.60429093	8.64836225
	13	3.21828431	6.43809016	7.76681041
	15	2.36855511	5.36578527	6.95171211
	17	2.03169194	4.40416363	6.19688195
	19	2.03078102	3.56786015	5.49715185

Table A.7 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	10.58091151	11.62456133	11.89753913
	7	8.81818187	10.28491110	10.80643036
	9	7.08590486	8.98619609	9.79982459
	11	5.21434360	7.74565955	8.86979941
	13	3.77947744	6.57765935	8.00892250
	15	2.85693400	5.49756359	7.21061238
	17	2.53432538	4.52382980	6.46917819
	19	2.44565751	3.67293675	5.77979143
S2 Faster by 40%	5	10.89645323	11.70958771	12.01635712
	7	9.05495933	10.39129818	10.96009736
	9	7.39357763	9.10590073	9.98249064
	11	5.83226918	7.87114201	9.07656317
	13	4.36142529	6.70230323	8.23563604
	15	3.41213538	5.61593184	7.45372062
	17	2.93239517	4.63189713	6.72558153
	19	2.85429353	3.76828573	6.04671961
S2 Slower by 10%	5	8.91195964	10.94888258	11.31225805
	7	6.51570736	9.43765154	10.05697284
	9	4.31658567	8.02285389	8.91695920
	11	2.58673454	6.71775478	7.87886664
	13	1.38258428	5.53348284	6.93110575
	15	0.87630047	4.48389974	6.06398651
	17	0.87835537	3.58293883	5.26964748
	19	0.92733274	2.83349747	4.54200479

Table A.7 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	8.55013946	10.55347187	11.13049820
	7	6.01305917	8.94027519	9.82671698
	9	3.71623279	7.45184451	8.64837150
	11	1.73008037	6.10181664	7.58017887
	13	0.88350528	4.90571250	6.60913287
	15	0.64457089	3.88172077	5.72456273
	17	0.52420503	3.03686045	4.91803890
	19	0.60709674	2.35634418	4.18333009
S2 Slower by 30%	5	7.63372156	10.07341810	10.93025260
	7	4.81070087	8.34764375	9.57438712
	9	2.52115064	6.78538806	8.35546184
	11	1.13920086	5.40342962	7.25593785
	13	0.46285768	4.22482359	6.26120049
	15	0.26486484	3.26264360	5.35953123
	17	0.32828022	2.49969683	4.54200007
	19	0.34978524	1.89861864	3.80243295
S2 Slower by 40%	5	7.00266348	9.47869031	10.70859147
	7	3.98856788	7.63119667	9.29670099
	9	1.75329757	6.00391248	8.03484219
	11	0.51446349	4.62304692	6.90283313
	13	0.20375683	3.51103600	5.88424436
	15	0.12415787	2.64763735	4.96628252
	17	0.14084653	1.98110545	4.13968466
	19	0.19955250	1.46183069	3.39867643

A.8 Total Idle consumption

Table A.8: Total Idle energy consumption for all combinations

Combination	**Arrival rate**	**TAGS**	**Shortest**	**Random**
S1 S2 Identical	5	20.59065174	22.56022729	22.95595364
	7	16.84431780	19.72122996	20.53585604
	9	13.20064234	17.03205631	18.32825125
	11	9.86076262	14.51992459	16.30976250
	13	6.96494561	12.20370860	14.45970215
	15	5.11091104	10.10311387	12.76048825
	17	3.89494111	8.24295895	11.19755880
	19	3.31799486	6.64335618	9.75926751
S2 Faster by 10%	5	21.08843150	22.72359936	23.25934974
	7	17.53218853	19.93342151	20.92379172
	9	13.73244470	17.27858390	18.78462644
	11	10.46369526	14.78635436	16.82134856
	13	7.63308018	12.47668451	15.01546117
	15	5.55885430	10.37066443	13.35101216
	17	4.44559999	8.49477833	11.81456653
	19	3.79561530	6.87156316	10.39511154
S2 Faster by 20%	5	21.32357697	22.86255111	23.53810055
	7	17.86252097	20.11547878	21.28167756
	9	14.45648585	17.49189527	19.20722679
	11	11.22322440	15.01874070	17.29673374
	13	8.27274787	12.71658043	15.53363091
	15	6.20929931	10.60745643	13.90343501
	17	4.82321683	8.71910338	12.39377523
	19	4.14081207	7.07603594	10.99431544

Table A.8 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	21.69158489	22.98217654	23.79507374
	7	18.37345091	20.27339760	21.61285489
	9	15.08844837	17.67828430	19.59964226
	11	11.75701139	15.22322342	17.73959098
	13	8.92034995	12.92907620	16.01783643
	15	6.69767821	10.81852331	14.42121557
	17	5.41502105	8.92023488	12.93834671
	19	4.64461381	7.26034246	11.55957280
S2 Faster by 40%	5	22.00712661	23.08624586	24.03271423
	7	18.66350435	20.41168371	21.92019472
	9	15.46381908	17.84255122	19.96498127
	11	12.37493696	15.40455131	18.15312633
	13	9.50229780	13.11862246	16.47127207
	15	7.34361716	11.00785603	14.90744124
	17	5.81309084	9.10161791	13.45116306
	19	5.14607609	7.42736478	12.09343920
S2 Slower by 10%	5	19.98304799	22.16553521	22.62451611
	7	16.01576196	19.19720019	20.11394569
	9	12.24938036	16.40265276	17.83391841
	11	8.96904586	13.81085416	15.75773330
	13	6.35185436	11.44362563	13.86221151
	15	4.54547415	9.32646025	12.12797303
	17	3.58498675	7.48742660	10.53929497
	19	2.87220702	5.93909765	9.08400958

Table A.8 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	19.57888257	21.69437619	22.26099022
	7	15.45421532	18.58085556	19.65342615
	9	11.57531908	15.67309227	17.29673391
	11	8.11239169	13.00259532	15.16034766
	13	5.76809481	10.59749660	13.21825490
	15	4.39787126	8.49032797	11.44911405
	17	3.15042778	6.70326580	9.83606602
	19	2.47615044	5.22886062	8.36664821
S2 Slower by 30%	5	18.66246467	21.12232873	21.86050955
	7	14.25185702	17.84606736	19.14877970
	9	10.38023693	14.81959904	16.71093000
	11	7.43771121	12.08023146	14.51188268
	13	5.34744721	9.66721789	12.52240846
	15	3.85307399	7.61072056	10.71907028
	17	2.95450297	5.90685636	9.08400817
	19	2.14773015	4.52104116	7.60487401
S2 Slower by 40%	5	17.98412502	20.41348820	21.41718293
	7	13.36430057	16.95660463	18.59340197
	9	9.53171237	13.81399517	16.06968437
	11	6.81297384	11.03715848	13.80566627
	13	5.08834637	8.67006306	11.76848872
	15	3.71236702	6.70841206	9.93256504
	17	2.69127091	5.10779730	8.27936932
	19	1.99749741	3.81649823	6.79735287

B

Performance data in heterogeneous environment

In this Appendix, the performance metrics data for all servers combinations presented. The data includes throughput, job loss, average response time and utilisation for server1 and server2.

B.1 Throughput

Table B.1: Throughput for all server combinations in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	4.99994614	4.95984847	4.69603091
	7	6.99894399	6.85251336	6.30942930
	9	8.99136948	8.64529579	7.78116583
	11	10.95746190	10.32005027	9.12682500
	13	12.85551311	11.86419427	10.36019857
	15	14.61639937	13.26459076	11.49300784
	17	16.16855860	14.50469403	12.53496080
	19	17.47254949	15.57109588	13.49382166
S2 Faster by 10%	5	4.99995533	4.96539277	4.71845518
	7	6.99911252	6.86986578	6.35334580
	9	8.99263704	8.68214836	7.85076149
	11	10.96351006	10.38357323	9.22505601
	13	12.86894250	11.96047062	10.48917718
	15	14.64204095	13.39839393	11.65429149
	17	16.20901371	14.90398904	12.72980343
	19	17.53509371	15.78750934	13.72342192
S2 Faster by 20%	5	4.99996645	4.96994686	4.73872662
	7	6.99930303	6.88441049	6.39343692
	9	8.99390884	8.71347920	7.91470090
	11	10.96763223	10.43818926	9.31572920
	13	12.88098908	12.04402925	10.60867134
	15	14.66475597	13.51545651	11.80414838
	17	16.24788472	14.83295828	12.91123225
	19	17.59703356	15.97929961	13.93750279

Table B.1 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	4.99997300	4.97373381	4.75710968
	7	6.99941380	6.89673994	6.43014400
	9	8.99463165	8.74038915	7.97360691
	11	10.97127818	10.48558233	9.39964613
	13	12.89039249	12.11715637	10.71965788
	15	14.68606965	13.61864688	11.94373381
	17	16.28502011	14.96932631	13.08059989
	19	17.65687566	16.15033254	14.13765985
S2 Faster by 40%	5	4.99997629	4.97691598	4.77382861
	7	6.99948324	6.90729349	6.46384422
	9	8.99535412	8.76370844	8.02801498
	11	10.97415872	10.52704213	9.47749894
	13	12.89943471	12.18162770	10.82298234
	15	14.70469833	13.71022151	12.07404701
	17	16.31924317	15.09102566	13.23906955
	19	17.71393892	16.30371426	14.32524864
S2 Slower by 10%	5	4.99992345	4.95290203	4.67113980
	7	6.99856831	6.83104331	6.26116773
	9	8.98915630	8.59975509	7.70518501
	11	10.95085315	10.24061421	9.02010224
	13	12.83914227	11.73981510	10.22059938
	15	14.58894078	13.08128648	11.31895042
	17	16.12575156	14.24724485	12.32511319
	19	17.40868354	15.22950140	13.24679393

Table B.1 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	4.99989652	4.94421212	4.64340628
	7	6.99815562	6.80479965	6.20794483
	9	8.98641367	8.54485108	7.62196026
	11	10.94034291	10.14517867	8.90379208
	13	12.81965903	11.58909727	10.06904778
	15	14.55958477	12.85734413	11.13053233
	17	16.08347873	13.93607086	12.09836117
	19	17.34430872	14.82827215	12.98001187
S2 Slower by 30%	5	4.99983885	4.93313113	4.61237749
	7	6.99721384	6.77215051	6.14902429
	9	8.98159576	8.47734492	7.53047237
	11	10.92888834	10.02719828	8.77659912
	13	12.79854045	11.40015279	9.90396779
	15	14.52752941	12.57871501	10.92585939
	17	16.03902181	13.56203512	11.85239457
	19	17.28033402	14.36569629	12.69057059
S2 Slower by 40%	5	4.99974628	4.91865862	4.57750244
	7	6.99594878	6.73058269	6.08351895
	9	8.97497752	8.39171321	7.42950167
	11	10.91219287	9.87470686	8.63697799
	13	12.77399929	11.15623423	9.72347269
	15	14.49715401	12.23376192	10.70263198
	17	15.99767075	13.12309658	11.58433636
	19	17.21970334	13.84548936	12.37474514

B.2 Job loss

Table B.2: Job loss for all server combinations in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	0.00005387	0.04015153	0.30396909
	7	0.00105645	0.14748664	0.69057070
	9	0.00863692	0.35470421	1.21883417
	11	0.04258772	0.67994973	1.87317500
	13	0.14478098	1.13580573	2.63980143
	15	0.38472991	1.73540924	3.50699216
	17	0.83432594	2.49530597	4.46503920
	19	1.53435114	3.42890412	5.50617834
S2 Faster by 10%	5	0.00004468	0.03460723	0.28154482
	7	0.00088770	0.13013422	0.64665420
	9	0.00736989	0.31785164	1.14923851
	11	0.03653921	0.61642677	1.77494399
	13	0.13134432	1.03952938	2.51082282
	15	0.35893923	1.60160607	3.34570851
	17	0.79395853	2.09601096	4.27019657
	19	1.47201087	3.21249066	5.27657808
S2 Faster by 20%	5	0.00003355	0.03005314	0.26127338
	7	0.00069729	0.11558951	0.60656308
	9	0.00609550	0.28652080	1.08529910
	11	0.03240144	0.56181074	1.68427080
	13	0.11923459	0.95597075	2.39132866
	15	0.33623368	1.48454349	3.19585162
	17	0.75484854	2.16704172	4.08876775
	19	1.40969152	3.02070039	5.06249721

Table B.2 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	0.00002700	0.02626619	0.24289032
	7	0.000586383	0.10326006	0.56985600
	9	0.00537111	0.25961085	1.02639309
	11	0.02875859	0.51441767	1.60035387
	13	0.10983720	0.88284363	2.28034212
	15	0.31478293	1.38135312	3.05626619
	17	0.71777679	2.03067369	3.91940011
	19	1.34996372	2.84966746	4.86234015
S2 Faster by 40%	5	0.00002371	0.02308402	0.22617139
	7	0.00051705	0.09270651	0.53615578
	9	0.00464943	0.23629156	0.97198502
	11	0.02586772	0.47295787	1.52250106
	13	0.10074759	0.81837230	2.17701766
	15	0.29617596	1.28977849	2.92595299
	17	0.57492694	1.90897434	3.76093045
	19	1.29296295	2.69628574	4.67475136
S2 Slower by 10%	5	0.00007657	0.04709797	0.32886020
	7	0.00143253	0.16895669	0.73883227
	9	0.01085482	0.40024491	1.29481499
	11	0.04922291	0.75938579	1.97989776
	13	0.16123482	1.26018490	2.77940062
	15	0.41219020	1.91871352	3.68104958
	17	0.87733693	2.75275515	4.67488681
	19	1.59793536	3.77049860	5.75320607

Table B.2 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	0.00010349	0.05578788	0.35659372
	7	0.00184520	0.19520035	0.79205517
	9	0.01359869	0.45514892	1.37803974
	11	0.05976949	0.85482133	2.09620792
	13	0.18074880	1.41090273	2.93095222
	15	0.41726903	2.14265587	3.86946767
	17	0.91946512	3.06392914	4.90163883
	19	1.66194197	4.17172785	6.01998813
S2 Slower by 30%	5	0.00016120	0.06686887	0.38762251
	7	0.00278804	0.22784949	0.85097571
	9	0.01842748	0.52265508	1.46952763
	11	0.07124627	0.97280172	2.22340088
	13	0.20196085	1.59984721	3.09603221
	15	0.47373198	2.42128499	4.07414061
	17	0.96403615	3.43796488	5.14760543
	19	1.72545398	4.63430371	6.30942941
S2 Slower by 40%	5	0.00025378	0.08134138	0.42249756
	7	0.00405383	0.26941731	0.91648105
	9	0.02505560	0.60828679	1.57049833
	11	0.08799515	1.12529314	2.36302201
	13	0.22657715	1.84376577	3.27652731
	15	0.50417861	2.76623808	4.29736802
	17	1.00513494	3.87690342	5.41566364
	19	1.78615632	5.15451064	6.62525486

B.3 Average response time

Table B.3: Response time for all server combinations in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	0.20326085	0.23898310	0.43930825
	7	0.25951324	0.30136246	0.47725241
	9	0.33571446	0.35710236	0.50316692
	11	0.42324798	0.40801711	0.52313921
	13	0.53074265	0.45636981	0.54000438
	15	0.59265748	0.50368738	0.55530736
	17	0.61431295	0.54993037	0.57000767
	19	0.60615562	0.59378680	0.58476877
S2 Faster by 10%	5	0.18434024	0.22474182	0.41279365
	7	0.22942398	0.28343615	0.44953501
	9	0.32161954	0.33656331	0.47452438
	11	0.39975323	0.38557157	0.49361189
	13	0.49282006	0.43246507	0.50953117
	15	0.55282017	0.47863051	0.52376459
	17	0.57738939	0.51608436	0.53722182
	19	0.57467078	0.56721373	0.55052288
S2 Faster by 20%	5	0.18159588	0.21296307	0.38885550
	7	0.22561885	0.26845926	0.42448901
	9	0.28378870	0.31925955	0.44865450
	11	0.35329384	0.36652839	0.46697739
	13	0.44633087	0.41206230	0.48209809
	15	0.51598659	0.45713406	0.49544467
	17	0.55048146	0.50163253	0.50788536
	19	0.55402450	0.54423789	0.52000410

Table B.3 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	0.16841806	0.20307098	0.36714779
	7	0.20573060	0.25577120	0.40175302
	9	0.25523370	0.30449494	0.42517666
	11	0.33930722	0.35018170	0.44283038
	13	0.42017975	0.39445845	0.45726792
	15	0.48264625	0.43850350	0.46986913
	17	0.51861404	0.48218343	0.48146718
	19	0.52576065	0.52418520	0.49261601
S2 Faster by 40%	5	0.15795896	0.19465398	0.34738314
	7	0.20317503	0.24489237	0.38102765
	9	0.25066206	0.29175707	0.40377680
	11	0.30796077	0.33600581	0.42083827
	13	0.38615386	0.37912461	0.43468433
	15	0.45404462	0.42221209	0.44665133
	17	0.49518197	0.46511771	0.45754218
	19	0.49948527	0.50653928	0.46788566
S2 Slower by 10%	5	0.22943103	0.26136047	0.46882047
	7	0.30268490	0.32939683	0.50808258
	9	0.39810826	0.38995401	0.53504912
	11	0.49510549	0.44552960	0.55605969
	13	0.59186710	0.49874386	0.57406184
	15	0.63751995	0.55080702	0.59067418
	17	0.64325533	0.60091903	0.60691717
	19	0.63923569	0.64735182	0.62350603

Table B.3 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	0.23879961	0.28919419	0.50184791
	7	0.31999133	0.36400347	0.54256992
	9	0.42973704	0.43044239	0.57075013
	11	0.58215051	0.49188423	0.59299852
	13	0.64418618	0.55096742	0.61238991
	15	0.67389537	0.60788917	0.63063192
	17	0.68291486	0.66103512	0.64882011
	19	0.67336651	0.70891565	0.66772954
S2 Slower by 30%	5	0.28601431	0.32469903	0.53903432
	7	0.40363161	0.40780272	0.58139455
	9	0.54269932	0.48169076	0.61099820
	11	0.64630240	0.55055867	0.63474876
	13	0.71308575	0.61593900	0.65586901
	15	0.72726326	0.67649608	0.67617669
	17	0.71064657	0.73084315	0.69685975
	19	0.70766315	0.77885240	0.71875677
S2 Slower by 40%	5	0.31375681	0.37145908	0.58118960
	7	0.45974404	0.46506587	0.62541673
	9	0.62381485	0.54872656	0.65672170
	11	0.75697080	0.62615870	0.68233229
	13	0.77379793	0.69612181	0.70564874
	15	0.76397105	0.75729018	0.72862777
	17	0.74798987	0.81066495	0.75256781
	19	0.72917189	0.85801639	0.77836068

B.4 Utilisation Server 1

Table B.4: Server 1 Utilisation for all server combinations in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	0.26192744	0.24799242	0.23480155
	7	0.36666303	0.34262567	0.31547147
	9	0.47114702	0.43226479	0.38905829
	11	0.57451258	0.51600251	0.45634125
	13	0.66871533	0.59320971	0.51800993
	15	0.74977998	0.66322954	0.57465039
	17	0.81955791	0.72523470	0.62674804
	19	0.86498251	0.77855479	0.67469108
S2 Faster by 10%	5	0.26192744	0.24598851	0.22468834
	7	0.36666303	0.33934393	0.30254028
	9	0.46649710	0.42777863	0.37384579
	11	0.56912387	0.51049572	0.43928838
	13	0.66303576	0.58693529	0.49948463
	15	0.74977998	0.65649878	0.55496626
	17	0.81389834	0.71840709	0.60618112
	19	0.85933126	0.77199182	0.65349628
S2 Faster by 20%	5	0.25928844	0.24428745	0.21539647
	7	0.36298206	0.33653382	0.29061052
	9	0.46649710	0.42390512	0.35975883
	11	0.56912387	0.50570335	0.42344190
	13	0.66303576	0.58143398	0.48221197
	15	0.74395039	0.65055526	0.53655181
	17	0.81389834	0.71233735	0.58687378
	19	0.85933126	0.76612161	0.63352243

Table B.4 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	0.25928844	0.24282565	0.20683103
	7	0.36298206	0.33410090	0.27957170
	9	0.46649710	0.42052745	0.34667882
	11	0.56382215	0.50149574	0.40868056
	13	0.65727517	0.57657221	0.46607240
	15	0.74395039	0.64526935	0.51929312
	17	0.80795362	0.70690633	0.56872210
	19	0.85340291	0.76083962	0.61468124
S2 Faster by 40%	5	0.25928844	0.24155612	0.19890953
	7	0.35943033	0.33197430	0.26932684
	9	0.46198390	0.41755663	0.33450062
	11	0.56382215	0.49777271	0.39489579
	13	0.65727517	0.57224539	0.45095760
	15	0.73790121	0.64053839	0.50308529
	17	0.80795362	0.70201861	0.55162790
	19	0.84721450	0.75606140	0.59688536
S2 Slower by 10%	5	0.26192744	0.25222316	0.24584946
	7	0.36666303	0.34936342	0.32953514
	9	0.47114702	0.44134674	0.40553605
	11	0.57451258	0.52712671	0.47474222
	13	0.66871533	0.60599048	0.53792628
	15	0.75538842	0.67716263	0.59573423
	17	0.81955791	0.73970082	0.64869017
	19	0.87034171	0.79295999	0.69719968

Table B.4 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	0.26475046	0.25727305	0.25796720
	7	0.37058959	0.35729464	0.34488606
	9	0.47606091	0.45191682	0.42344251
	11	0.57451258	0.53994809	0.49465541
	13	0.67436070	0.62054773	0.55939186
	15	0.74977998	0.69275952	0.61836324
	17	0.82491848	0.75557298	0.67213153
	19	0.87539642	0.80849890	0.72111213
S2 Slower by 30%	5	0.26475046	0.26340596	0.27131620
	7	0.37058959	0.36677176	0.36170716
	9	0.47606091	0.46438593	0.44296879
	11	0.58009931	0.55487988	0.51627034
	13	0.67436070	0.63717371	0.58258614
	15	0.76078606	0.71012820	0.64269740
	17	0.82491848	0.77285603	0.69719946
	19	0.88013701	0.82517183	0.74650393
S2 Slower by 40%	5	0.26790256	0.27101347	0.28609390
	7	0.37495115	0.37830614	0.38021993
	9	0.48143901	0.47932782	0.46434385
	11	0.58009931	0.57239256	0.53981112
	13	0.67436070	0.65606486	0.60771704
	15	0.76078606	0.72928169	0.66891450
	17	0.82997171	0.79155388	0.72402102
	19	0.88013701	0.84302216	0.77342157

B.5 Utilisation Server 2

Table B.5: Server 1 Utilisation for all server combinations in heterogeneous environment

Combination	Arrival rate	TAGS	Shortest	Random
S1 S2 Identical	5	0.36536244	0.24799242	0.23480155
	7	0.51038245	0.34262567	0.31547147
	9	0.64881016	0.43226479	0.38905829
	11	0.76810325	0.51600251	0.45634125
	13	0.86695496	0.59320971	0.51800993
	15	0.90949262	0.66322954	0.57465039
	17	0.92077935	0.72523470	0.62674804
	19	0.91381783	0.77855479	0.67469108
S2 Faster by 10%	5	0.33217712	0.23910487	0.22468834
	7	0.46452440	0.33176130	0.30254028
	9	0.61800659	0.42031578	0.37384579
	11	0.73329644	0.50374732	0.43928838
	13	0.82809223	0.58128575	0.49948463
	15	0.87962974	0.65212359	0.55496626
	17	0.88972833	0.71527436	0.60618112
	19	0.88762772	0.76990397	0.65349628
S2 Faster by 20%	5	0.31913976	0.23154247	0.21539683
	7	0.44618320	0.32243426	0.29061098
	9	0.56973718	0.40996853	0.35975938
	11	0.68266117	0.49304727	0.42344252
	13	0.78544771	0.57079399	0.48221264
	15	0.84209633	0.64228098	0.53655253
	17	0.86455387	0.70638909	0.58687454
	19	0.86461460	0.76214266	0.63352321

Table B.5 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Faster by 30%	5	0.29460590	0.22502924	0.20683072
	7	0.41212121	0.31433926	0.27957131
	9	0.52760634	0.40092026	0.34667836
	11	0.65237709	0.48362270	0.40868004
	13	0.74803484	0.56148938	0.46607183
	15	0.80953773	0.63349576	0.51929251
	17	0.83104497	0.69841135	0.56872145
	19	0.83695617	0.75513755	0.61468057
S2 Faster by 40%	5	0.27356978	0.21936082	0.19890953
	7	0.39633604	0.30724679	0.26932684
	9	0.50709482	0.39293995	0.33450062
	11	0.61118205	0.47525720	0.39489579
	13	0.70923831	0.55317978	0.45095760
	15	0.77252431	0.62560454	0.50308529
	17	0.80450699	0.69120686	0.55162790
	19	0.80971376	0.74878095	0.59688536
S2 Slower by 10%	5	0.40586936	0.27007449	0.24584946
	7	0.56561951	0.37082323	0.32953514
	9	0.71222762	0.46514307	0.40553605
	11	0.82755103	0.55214968	0.47474222
	13	0.90782771	0.63110114	0.53792628
	15	0.94157997	0.70107335	0.59573423
	17	0.94144298	0.76113741	0.64869017
	19	0.93817782	0.81110017	0.69719968

Table B.5 continued from previous page

Combination	Arrival rate	TAGS	Shortest	Random
S2 Slower by20%	5	0.42999070	0.29643521	0.25796679
	7	0.59912939	0.40398165	0.34488553
	9	0.75225115	0.50321037	0.42344190
	11	0.88466131	0.59321222	0.49465474
	13	0.94109965	0.67295250	0.55939114
	15	0.95702861	0.74121862	0.61836248
	17	0.96505300	0.79754264	0.67213074
	19	0.95952688	0.84291039	0.72111133
S2 Slower by 30%	5	0.49108523	0.32843879	0.27131649
	7	0.67928661	0.44349042	0.36170753
	9	0.83192329	0.54764080	0.44296921
	11	0.92405328	0.63977136	0.51627081
	13	0.96914282	0.71834509	0.58258663
	15	0.98234234	0.78249043	0.64269792
	17	0.97811465	0.83335354	0.69720000
	19	0.97668098	0.87342542	0.74650447
S2 Slower by 40%	5	0.53315577	0.36808731	0.28609390
	7	0.73409547	0.49125356	0.38021993
	9	0.88311350	0.59973917	0.46434385
	11	0.96570243	0.69179687	0.53981112
	13	0.98641621	0.76593093	0.60771704
	15	0.99172281	0.82349084	0.66891450
	17	0.99061023	0.86792630	0.72402102
	19	0.98669650	0.90254462	0.77342157