# Energy-aware optimization of scalable load balancing strategies

*by*

**Rajita Chandak**
B01090236

Faculty Advisor: Dr. Kavita Ramanan
Post-doctoral Advisor: Dr. Debankur Mukherjee
Second Faculty Reader: Dr. Matt Harrison

Department of Applied Mathematics

Brown University

April 2019

# Acknowledgements

*I would like to thank Dr. Kavita Ramanan for her support throughout this process. I would also like to thank Dr. Debankur Mukherjee for his guidance and mentorship.*

# Abstract

*Queuing systems are one of the most prominent technological evolution in today's digital world. Data centers and cloud networks operated by large companies like Microsoft, Google and Amazon rely on large-scale queuing systems for their operations. The goal of minimizing energy consumption while optimizing performance has been the focus of recent literature around the subject. In this thesis, we consider the TABS scheme and investigate through simulations and a theoretical proof the implications of various parameters of the scheme on a variant of the decentralized parallel queuing system. Simulation results explore how changing the assumption of exponential service time distributions affects various performance metrics. Our theoretical result builds on previous results in order to identify patterns in fluctuations around mean behaviour of the system over long time intervals.*

# Contents

# 1 Introduction

## 1.1 Background and Motivation

As we move towards an increasingly digitized world, concerns of energy consumption have become more prominent. Modern day data centers and cloud networks are just one such technological evolution that present financial and environmental challenges. It is estimated that US data centers consume about 70 million MegaWatt hours annually, the equivalent of 6 million homes [7]. A crucial consideration for data centers is to achieve operational efficiency in terms of user-perceived performance while managing energy consumption and efficient use of servers throughout an extended period of time. A big challenge with this set up is that client demand is uncertain and often changes drastically with time [2]. Thus we may dynamically scale the amount of resources that are actively utilized with the actual observed demand conditions in order to minimize energy consumption and operation costs, while achieving targeted performance goals. This methodology of scaling resources is referred to as service elasticity. Achieving ideal service elasticity is challenging because by turning servers off during periods of low demand, data centers incur additional additional setup costs in terms of the time required to turn on servers, producing a time lag in the system. Typically the setup time for servers is orders of magnitude higher than the time required to service tasks [5]. This time lag can be counteracted by maintaining a significant number of idle servers on in case of sudden demand spikes. Another scheme to balance energy consumption with data center performance is auto-scaling. Auto-scaling adjusts capacity as a response to changing demand. It is currently a widely employed practice in industry by companies like Facebook, Google and Microsoft [8]. Many auto-scaling approaches use historical information to forecast load predictions and manage capacity, information that may not always be available in data centers. These schemes often look at centralized queueing systems.

In practice, data centers do not maintain a centralized queue. Instead, they distribute demand across parallel servers, each with its own queue. Operating within a parallel queuing network system provides less global information. Since parallel queuing systems allow for the chance that some servers remain idle while tasks are waiting at other servers, expected performance is not as easily analyzable as for the centralized queuing system and thus requires that algorithms to manage energy consumption operate independent of historical load prediction or global queuing information.

## 1.2 Literature Review

Load-balancing schemes for large-scale systems found in cloud networks and data centers have been studied extensively in the past. In particular, stability and energy consumption of systems with centralized queues is well understood [4, 10]. Many papers consider the idea of putting servers to sleep in the centralized system because servers in sleep mode consume significantly less energy than idle servers. It is estimated that idle servers consume 60-70% of the energy used during peak demand [2, 5]. However, as the number of servers becomes large,

maintaining a centralized queue becomes complicated to handle and presents inefficiency in operation speed [14]. Stankovic [14] shows in his paper that three of the decentralized algorithms he tested for queuing systems show stability and improved performance with only a modest increase in costs. A decentralized system maintains servers with individual, parallel queues. This is an example of a load balancing algorithm that is used to improve performance. That is, the system distributes demand (the network load) efficiently across servers. Such systems have been the focus of more recent research on cloud networks [12, 11]. Furthermore, in typical data center architectures and cloud environments no centralized queue is maintained, and load balancing algorithms immediately distribute incoming tasks among parallel queues [1, 12].

It is estimated that servers account for about 45% of a data center's amortized costs and utilities (electrical) make up about 15% of operational costs [2]. Thus, improving performance and lowering energy consumption of servers is of interest to companies that manage these centers. Since data centers, operating with tens of thousands of servers, face large amortized costs, operational efficiency is another prime concern for businesses. Unfortunately data centers tend to operate at low operational efficiency-usually less than 10%, either due to server memory constraints or safeguarding against unexpected demand shifts [2, 8]. That is, data centers tend to use significantly fewer number of servers than available and as a result a large portion of energy costs are due to idle servers (otherwise referred to as servers in standby) maintained through the day. One main reason for this, as stated by Greenberg et al. in [2] is uncertainty in expected demand. Since the demand for data centers can spike unexpectedly, businesses tend to err on the side of caution by operating with more than the necessary number of servers. This allows data centers to maintain higher capacity in case of a spike in demand and reduce the risk of system failure during times when profits can be maximized. There are many ways that costs could be reduced, as explored in [2]. One of the most impactful ones is reduced power consumption. Turning idle servers off to reduce energy consumption has been one of the most popular suggestions in literature. However, this comes at a cost. Data suggests the average setup time (the amount of time required for a server to turn on, warm up and be prepared to service new tasks) for a server is about 200 seconds whereas the time required to service a task is typically less than 1 second [4], meaning unexpected increases in demand could lead to system failure despite data centers having the capacity to keep up with the demand. Additionally, data centers could implement auto-scaling techniques. Auto-scaling refers to an automatic system of adjusting capacity of data centers to meet demand while maintaining performance goals. In much of the literature relevant to this thesis [12, 11, 4], auto-scaling is implemented in the form of turning servers off during periods of low demand. Thus, investigation into decentralized queuing systems presents great potential for reduced energy consumption, lower costs and higher operational efficiency, if managed carefully [2].

A particularly relevant idealized model for centralized load management is considered by Gandhi et al. in [4]. Specifically, they consider a variant of a standard queueing system, referred to as M/M/N/setup/delayedoff. In this scheme, servers can be turned on and off as needed. More specifically, in the proposed algorithm when a server finishes servicing a task, and finds no additional tasks in its queue, it waits for an exponential distributed amount of time with expectation $\mu$. If a task arrives during this time then it is immediately

assigned to the server (or one of the other servers on standby at random), otherwise the server is turned off. When a task arrives, if there is no server in standby, then the task waits in the centralized queue. The server takes an exponentially distributed amount of time, with expectation $\nu$ to setup and become idle once the time available to service tasks in the queue has elapsed. . Gandhi et al.[4] present in their paper a closed form solution for the M/M/N/setup/delayedoff system, through what they term as a Recursive Renewal Reward method. The methodology combines features of Renewal Reward theory and analysis of high demand intervals to find closed form solutions for the system's power consumption. The method introduce in this paper is robust in its application to Markov Chains. The idea of turning idle servers off is adapted to a decentralized system and introduced by Mukherjee et. al. in [12] as the TABS scheme.

The TABS scheme was first proposed by Mukherjee et. al. [12] as a scheme that aims to increase efficiency and reduce energy consumption on average for large scale distributed systems with parallel queues. The main goal of the paper was to dynamically scale the number of servers that are actively utilized with the actual observed task load conditions to curtail cost and energy consumption, while still satisfying the target performance criteria. By looking at a joint auto-scaling and load balancing scheme (in the form of turning servers off when idle for an exponentially distributed time and turning them back on as needed) which does not require any knowledge of the global queue length or explicit information of system parameters the scheme is still able to achieve near-optimal service elasticity. The TABS scheme allows for servers to turn themselves off after staying idle for an exponentially distributed amount of time (detailed in Section 2.1 of this thesis). This scheme is shown to reduce average energy consumption that would have otherwise persisted were servers to remain in standby mode for long periods of time. If all servers are busy when a new task arrives, the task joins the queue of a randomly selected busy server and the dispatcher selects one of the off servers to begin setup. Mukherjee et. al. [12] also considers how not operating at full server capacity may affect average wait time of the tasks in the system. The results show that the TABS scheme provides similar waiting times on average when compared to the M/M/N/setup/delayedoff schem despite the former being a centralized scheme. Mukherjee et al. prove that both the waiting time of tasks and the relative energy portion consumed by idle servers vanish in the limit as the standby time $\mu$ tends to infinity.

While [12] assumes finite buffer capacity, the question of system stability for an infinite buffer capacity system under the TABS algorithm is addressed in [11] by Mukherjee and Stolyar through a proof using induction and weak-monotonicity. Mukherjee and Stolyar [11] examine the asymptotic behaviour of the system as the number of servers becomes large and use mean-field analysis to prove stability and convergence of the TABS scheme (in Sections 2 and 3 of [11]). Important parameters in both [12] and [11] are the expected setup time, denoted by $\nu$, and the expected standby time, represented by $\mu$. From the results of [2], we may expect that the average amount of time for which servers remain idle before turning off or the expected amount of time it takes for servers to turn on and warm up before servicing tasks may affect the performance of the system. However, results of both papers [11] and [12] show that as the number of servers increases towards infinity, system performance metrics like the average waiting time and energy wastage do not depend on $\nu$ and $\mu$ values by virtue of the fluid-limit scaling techniques used. The parameters $\nu$ and $\mu$ are scaled according to

the number of servers and thus the steady-state of the queuing system is independent of their values.

In this thesis, we look at finer details of the system and their dependence on $\nu$ and $\mu$. Specifically, we will look at a diffusion scaling of the system. By looking at the behaviour of the TABS scheme in this asymptotic regime, we try to understand the steady state of the system under a different scaling method. The goal of looking at a diffusive scaling for the system is to eventually understand how $\nu$ and $\mu$ affect performance metrics like average task wait time and energy wastage. This paper consists of two different analyses of the system to better understand it's behaviour in the limit. First, we simulate the system with a large number of servers (ranging from 100-1000 servers) under different $\nu$ and $\mu$ values, with varying service time distributions. The simulation results show how the energy consumption and average task wait time varies with the service time distribution parameters and $\nu$ and $\mu$ in the long run. In this these, we also derive a proof for the limiting behaviour of this system. The proof extends ideas presented by Eschenfeldt and Gamarnik in [3] done on a similar queuing system. By understanding the steady state convergence of the diffusion scaling of the system under the TABS scheme, we aim to provide finer guidelines for building a more efficient system.

## 1.3 Key Contributions

In this thesis, we provide provide a more refined study of the performance of the TABS algorithm. The first half of this thesis presents a simulation-based analysis of the effect of varying service time distributions on the stability and behaviour of the system. By looking at the effect of several distributions centered around expectation 1 with different variances on the system's energy consumption and average task wait time, this paper explores robustness of the system with respect to the service time distributions.

In the second half of this thesis, by looking at the diffusion scaling of the TABS system, we are able to understand the effect of changing expected standby time and setup time on the long term behaviour of the system. The limit theorem proof provides a finer scale analysis of the system than previously examined in literature pertaining to the TABS scheme. The proof sets up a martingale representation of the system, with appropriate scaling after which the limit of the system of equations converges to a known limit, proving stability and convergence of the system with respect to varying setup and standby time time for servers.

## 1.4 Outline of Paper

The remainder of the paper is organized as follows. In section 2 we describe the system set up, the TABS Algorithm, and some key results from previous literature the provide support for following sections. In section 3 we present the simulations of the system, comparing various different system states. We also discuss the consequences of observed simulation results in this section. In Section 4 we present the limit theorem proof of convergence. Finally, in Section 5 we provide some concluding remarks and potential extensions of the work presented in this thesis.

# 2 Details of Model and Algorithm

## 2.1 TABS Algorithm

The queuing system we are looking at consists of 2 key actors. The first is the dispatcher that manages all incoming tasks. The second actor is the collection of servers. Servers operate based on task assignment from the dispatcher. The TABS algorithm uses the idea of tokens that are accessible to the dispatcher in order to assign new tasks to a server and turn servers on. Busy servers are identified by 'yellow' tokens, idle servers are identified by 'green' tokens, off servers are identified by 'red' tokens and servers in setup mode are identified by 'orange' tokens. Thus, in this system the only global knowledge available is the number of servers that are busy, in standby, in setup mode or off. No information about each server's queue length is stored. The scheme is reproduced below is outlined in Section 2 of [12]:

- When a server becomes idle, it sends a green token to the dispatcher, waits for an Exp($\mu$) time ( $\mu$ denotes expected standby time), and turns itself off by sending a red token to the dispatcher (this replaces the initial green token corresponding to the server) if it does not receive a task in the mean time.
- When a task arrives, the dispatcher selects a green token at random, if there are any, and assigns the task to that server (the corresponding green token is replaced by a yellow token). Otherwise, the task is assigned to an arbitrary busy server, and if at that arrival epoch there is a red token at the dispatcher, the dispatcher selects one at random, and the setup procedure of the corresponding server is initiated (setup time distributed as Exp($\nu$)), replacing its red token by an orange token. If all servers are off when the task arrives, it is thrown away by the dispatcher. We consider this to mean the system has lost a task.
- After the server in setup mode is ready to service tasks, it replaces the orange token with a green token to the dispatcher and waits for an Exp($\mu$) time for a possible assignment of a task, and turns itself off by replacing the green token with a red one if no task is assigned in the interim period.

A key assumption in literature for previous results underlying data center simulation as well as the TABS algorithm is that tasks arrive with an exponential distribution with some expectation $\lambda < 1$ and tasks are serviced independently by servers with an exponential distribution with expectation 1. Previous literature alludes to additional constraints that can be imposed on the system such as buffer capacity for server queue lengths. In this case, the dispatcher will all be able to access information of whether each busy server has queue length equal to the buffer capacity, B. When the dispatcher receives a new task, the task will be assigned to a random server that has not reached capacity according to the same algorithm outlined above. In the event that all servers have queue length B, the dispatcher will throw the incoming task away.

## 2.2 Notation and Previous Results

**Notation:** For a system with $n$ servers, $Q_i^n(t)$ denotes the number of servers with queue length greater than or equal to i at time t, including the task being serviced. $\Delta_0^n(t)$ denotes the number of off servers at time t and $\Delta_1^n(t)$ denotes the number of servers in setup mode at time t. Thus, $(Q_i^n(t), \Delta_0^n(t), \Delta_1^n(t))$ provides a Markovian representation of the system since the servers are independent and identically distributed. Furthermore, for any fixed n, this process is an irreducible countable-state Markov chain [12]. Thus, we know the existence of a unique stationary distribution. Let $\mu$ denote the average standby time for an idle-on server before it turns off and let $\nu$ denote the average setup time required for an off server to turn on and be ready to start servicing tasks.

Previous convergence results for the TABS algorithm from [11] look at fluid-scaled limiting behaviour of the system, that is, $(Q_i^n(t)/n, \Delta_0^n(t)/n, \Delta_1^n(t)/n)$ and its convergence to a limit. Specifically, for fixed $\mu, \nu > 0$ and $\lambda < 1$, the system converges to the following limit

$$\left(\frac{Q_1^n(t)}{n}, \frac{\Delta_0^n(t)}{n}, \frac{\Delta_1^n(t)}{n}\right) \Rightarrow (\lambda, 1-\lambda, 0)$$

$$Q_i^n(t) \Rightarrow 0 \text{ for any } i > 1, \tag{2.1}$$

From these results we see that the limiting behaviour of the system under fluid scaling is independent of $\mu$ and $\nu$. A finer analysis of the system should provide further insight into the extent to which $\mu$ and $\nu$ and the specific form of the service time distribution affect the system in the long run.

## 2.3 Detailed Outline of Goals

In this section, we return to the goals outlined in Section 1.3 in the context of the notation and known results established in the previous sections. The first mathematical result we present in Section 3 through simulations is the behaviour of $\left(\frac{Q_1^n(t)}{n}, \frac{\Delta_0^n(t)}{n}, \frac{\Delta_1^n(t)}{n}\right)$ in the limit given alternative service time distributions. We maintain the constraints established on the exogenous parameters introduced in [11] and [12]. That is, we set $\mu, \nu > 0$ and $\lambda < 1$, while also constraining our various service time distributions to have expectation 1. We explore the effects of generalized Pareto, Folded Normal and Hyperexponential distributions on the system's performance in comparison with the Exponential distribution. The performance metrics used to understand the behaviour of the TABS scheme under these service time distributions will be the same as those in [11]-average task wait time and average energy consumption.

In Section 4, we explore the behaviour of the TABS system under diffusion scaling and over long time intervals. That is, we center each of the system variables around their fluid limit expectations and then scale by $\sqrt{n}$. This allows for us to understand the behaviour of the system on a finer scale and thus understand the impact of various $\mu$ and $\nu$ values. In

mathematical notation, we are looking at the long run behaviour of

$$\left(\frac{Q_1^n(t)-\lambda n}{\sqrt{n}}, \frac{\Delta_0^n(t)-(1-\lambda)n}{\sqrt{n}}, \frac{\Delta_1^n(t)}{\sqrt{n}}\right) \text{ and}$$

$$\frac{Q_i^n(t)}{\sqrt{n}}, \text{ for all } i \geq 2.$$

# 3 Simulation Results

In this section we will explore the effects of alternate service time distributions on the queuing system under the TABS algorithm. We can simulate various features such as queue lengths, fraction of busy, standby and off servers, energy consumption and average task wait time to see how it compares to the TABS results from [11]. All graphs that follow were produced using MATLAB.

We will start by first adjusting the service time distribution to be a generalized Pareto distribution with expectation (approximately) 1 to test how this affects long term behaviour of the system. The pdf of the generalized Pareto distribution is given by $f$

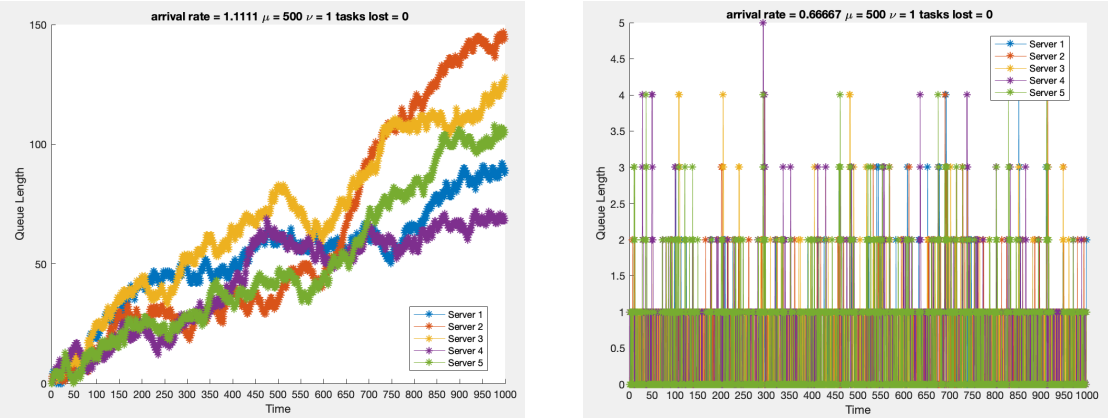
$$f(x|k,\sigma,\theta) = \frac{1}{\sigma}\left(1+k\frac{(x-\theta)^{-1-\frac{1}{k}}}{\sigma}\right)$$

For the simulations in this thesis we set the parameters to be $k=-0.6, \sigma=0.7, \theta=0.56$. The expectation of the Generalized Pareto distribution is 1 and the variance is 0.0870. The following sections will show similar simulations for Folded Normal and Hyperexponential service time distributions. We will also explore the effect of service time distribution variance on system performance.

## 3.1 Queue Lengths with Varying Number of Servers

**Queues with varying arrival rates:** Before looking at the performance metrics shown in [12], we will look at the behaviour of the basic system with varying parameters. First, we start with a small number of servers and observe how the queue lengths vary over time based on the arrival rate. For this, we set the standby time to be large enough such that the likelihood of servers turning off during the simulation is very low ($\mu = 500s$). We start the system with all servers idle and on.

Figure 1 shows how a 5 server system evolves over 1000 seconds.

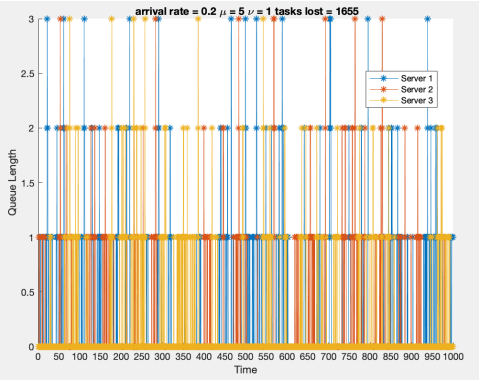



(a) Arrival Rate: 1.11 (b) Arrival Rate: 0.67

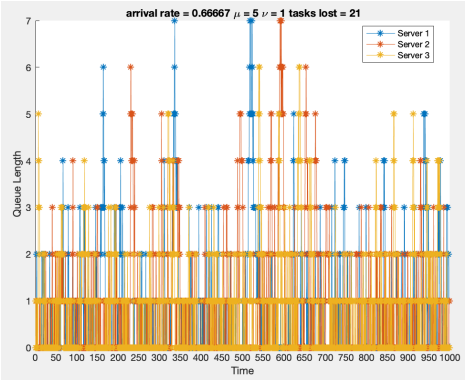
Figure 1: Queue lengths, multiple servers

As we can see in Figure 1(a), with a high arrival rate, the queue lengths for all servers are growing to infinity. This tells us that when more tasks arrive than are serviced in a given time period, on average, the queue lengths will tend to infinity. This fits into our model's expectations. Since the service time takes 1 second on average for any given server, if tasks arrive faster than any single server can process them, the queue lengths will grow rapidly and tend to infinity. In Figure 1(b) where the arrival rate is smaller than the expected servicing time, however, we see the system is stable. Servers have queue lengths mostly between 1-3 tasks. The queue lengths do not grow rapidly because the servers are, on average, able to service tasks faster than they arrive. Thus, we can note in systems with arrival rates greater than service rate, the system will be unstable.

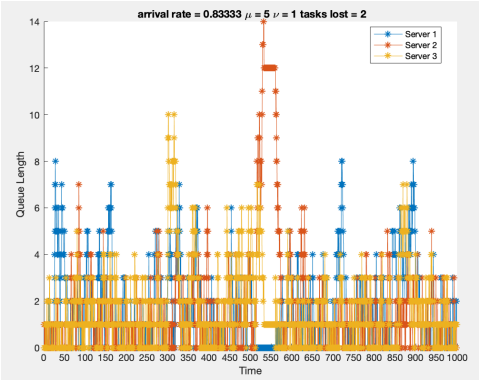
**Queues for reduced standby time:** We can now reduce the expected standby time to see how this affects the system. The simulations below use an average standby time of 5 seconds. The average setup time is kept constant at 1s to minimize the effect this will have on the system. All other parameters of the system are unchanged from previous simulations. Thus, the resulting change in queuing behaviour in this new system should reflect the effect of changing the expected standby time for servers. We set up the simulation such that if all servers are off, the task is thrown away by the dispatcher. As mentioned in Section 2.1, in this case, we consider the task to be 'lost' by the system. For the sake of visual representation, we maintain a small number of servers in the system.




(a) Arrival Rate=0.2




(b) Arrival Rate=0.67



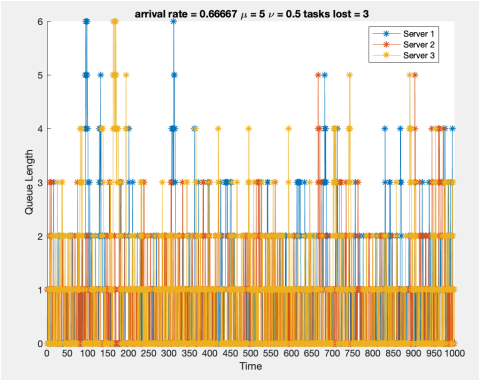

(c) Arrival Rate=0.83

Figure 2: Queue lengths, 3 servers, varying arrival rates

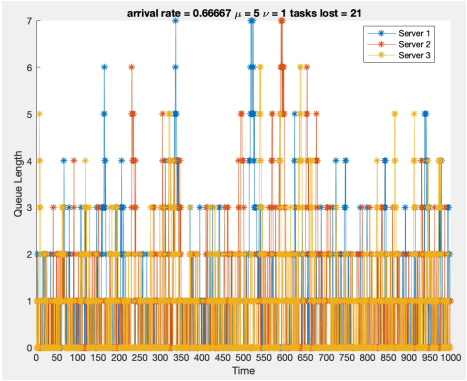
Note that there are a large number of tasks lost in these simulations. This happens because once a server becomes idle, it will turn off on average after 5 seconds if it does not receive a new task to service. If all servers are off, the system looses the task. Since tasks are lost in the system and servers have an infinite buffer capacity, we know that there are many instances in these simulations where all servers have turned off. We also notice that as the arrival rate increases, the number of tasks lost decreases. This implies that the average standby time must change based on the arrival rate in order to minimize the number of tasks that are lost. If the arrival rate is very small, and we do not want to loose any tasks, servers should have a larger expected standby time. A system with small average standby time for the servers will loose fewer tasks if the arrival rate is relatively high. The smaller the inter-arrival time between new tasks, the smaller chance that an idle server will turn off before the arrival of the next task and thus the greater the chance there will be at least one server available to service tasks.

**Varying expected setup time:** Finally, we can observe how increasing the average server setup time will affect queue lengths and number of tasks lost by the system. Once again, all other parameters of the system are left unchanged. In the simulations shown in Figure 3, the arrival rate is fixed at 0.67 and the average standby time is constant at 5s. As expected, the longer it takes for servers to set up, the more tasks are likely to be lost given all other

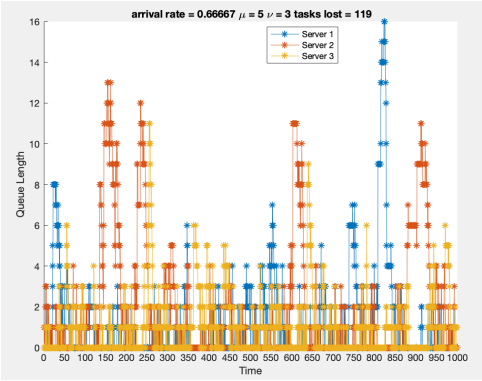
parameters of the system remain unchanged.




(a) $\nu = 2$s




(b) $\nu$= 1s



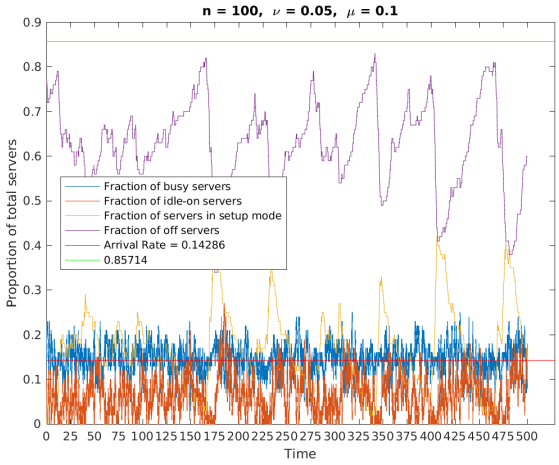

(c) $\nu = 0.3$s

Figure 3: Queue lengths, 3 servers, varying average setup times

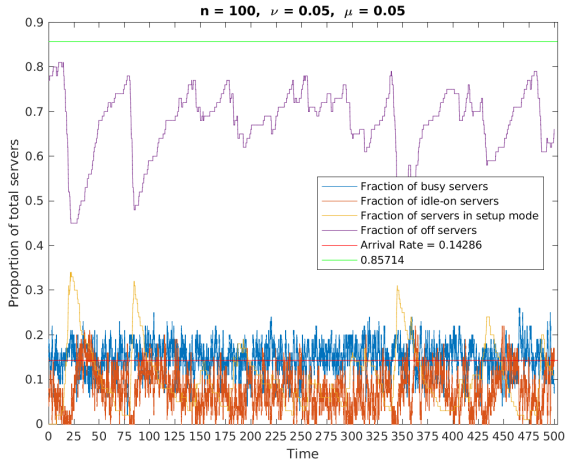
## 3.2 Fraction of Active Servers

Now that we understand the evolution of the system over time and how each parameter affects system behaviour (in terms of number of tasks lost in the system), We can look at systems with a large number of servers, starting with 100 servers. Plotting queue lengths for every server, as was done earlier, would be inefficient and not very informative. Instead, we look at the proportion of servers that are busy, standby, off or in setup mode. This gives a better picture of how the resources, in terms of capacity, are being used in the system.

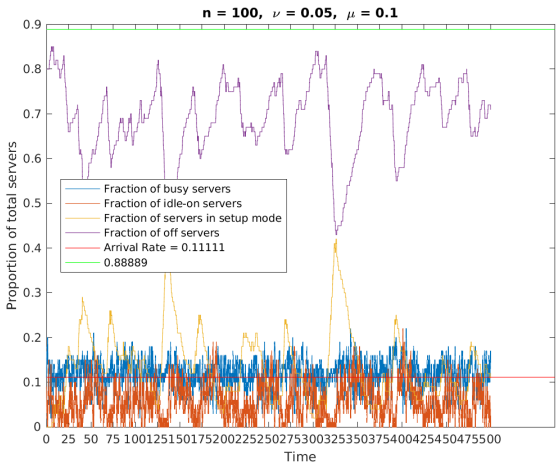
From the results of [11], we expect that with a large number of servers, the system to converge to the limit given in Equation 2.1. That is, with a large number of servers, in a stable system, the fraction of busy servers will converge to $\lambda$, the fraction of idle-off servers converges to $1 - \lambda$ and the fraction of idle servers and servers in setup mode converges to 0, where $\lambda$ is the arrival rate for tasks in the system.




(a) $\lambda$=0.14, $\nu$=20s, $\mu$=10s




(b) $\lambda$=0.11, $\nu$=20s, $\mu$=20s




(c) $\lambda$=0.14, $\nu$=20s, $\mu$=10s

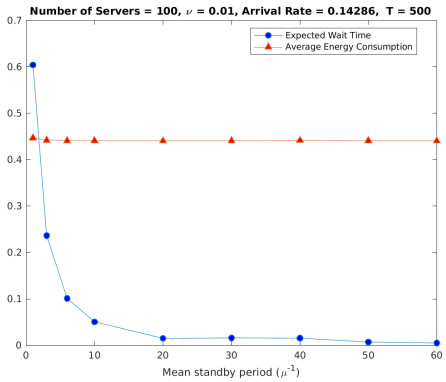
Figure 4: Server Proportions

The graphs in Figure 4 show that the simulations with a generalized Pareto distribution match the theoretical result from [11] for varying arrival rates, setup time and standby times. Thus, the fact that the graphs in Figure 4 show similar patterns to theoretical expectations for exponential distribution, provides a starting point to analyse the TABS scheme under various other distributions and test for robustness of the results in literature [11, 12].

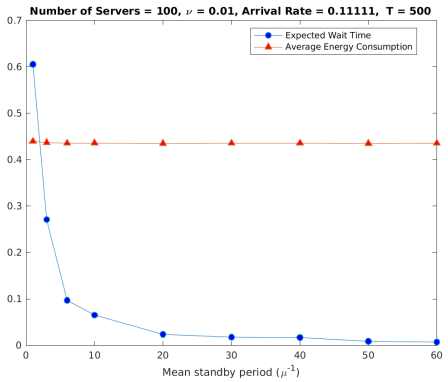
## 3.3 Energy and Wait-Time Graphs

The efficiency of the queuing system can be determined in two different ways. The first is to look at the average energy consumption of the system for varying standby times. This concerns the operational costs of data centers and aims to minimize them. The second is to

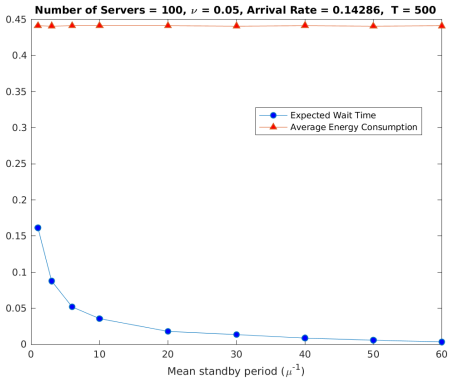
look at efficiency in terms of how fast tasks can be serviced. This concerns user experience and affects total business engagement by data centers. Data centers want to service tasks as soon as possible—delayed servicing can lead to loss in number of customers and total revenue. The goal is that both the average energy consumption and the average wait time for a new task in the system be as low as possible for quality performance and high operational efficiency. The values in following graphs are obtained using the methodology of calculating average wait time and energy consumption from [12].Average wait time is determined by observing the average of number of servers with queue lengths greater than 1, proportional to arrival rate. Average energy consumption is determined by taking the weighted average of the number of busy servers, servers in setup mode and idle-on servers. The graphs in Figure 5 show average energy consumption and wait-times corresponding to the system evolution plotted in Figure 4.




(a) Arrival Rate=0.14, setup time=10s




(b) Arrival Rate=0.11, setup time=10s




(c) Arrival Rate=0.14, setup time=20s

Figure 5: Average Energy Consumption and Wait-time

From Figure 4 we see that as the expected standby time increases, the average wait time for a task approaches zero. Also note that as $\mu$ increases, average energy consumption levels out. We also observe that as setup time increases, the limit that average energy consumption approaches, increases. These results match theoretical predictions and simulation results from [11]. This provides the first step in establishing the hypothesis that TABS convergence results may be independent of the service time distribution given that the expectation of the

distribution is 1.

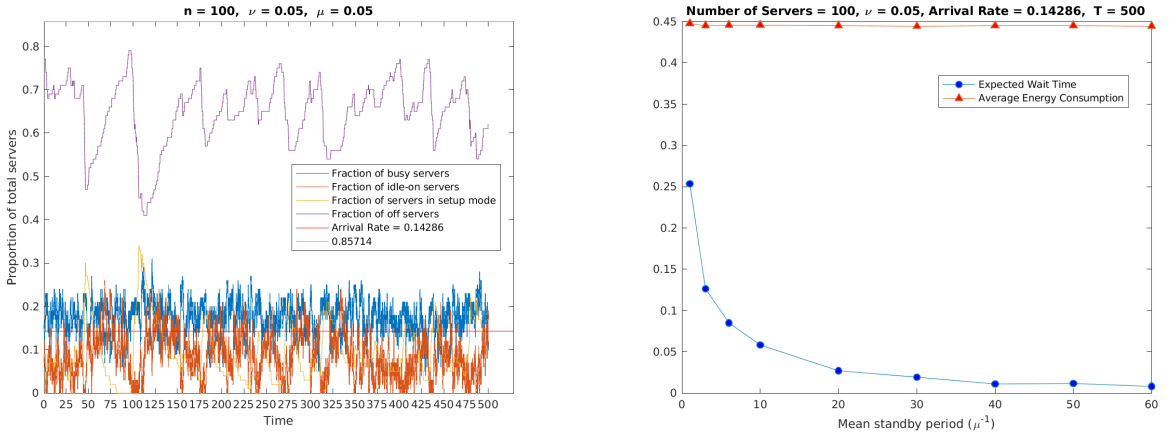
In the next sections we simulate the behaviour of the TABS scheme under the Folded Normal and Hyperexponential distributions with expectation 1.

## 3.4 Folded Normal Service Time Distribution

Just as was done with the generalized Pareto distribution, we simulate the TABS scheme with a folded normal distribution with expectation 1. The service time samples for each server are taken from the Folded Normal pdf, given by

$$f(x) = \left| \frac{1}{\sigma\sqrt{2\pi}} exp(\frac{-(x-\alpha)^2}{2\sigma^2}) \right|$$

Where $\alpha$ denotes the expectation of the distribution. The mean and variance of this pdf are set to equal 1 for all of the results presented below. We find the proportion of servers plots and average energy consumption and wait time. One such simulation result is shown in Figure 6. The parameters for this simulated run are $\lambda = 0.14$, $\nu = 20s$, $\mu = 20s$, $\sigma^2 = 1$. The folded normal distribution is simulated first from the normal distribution with expectation 1 and variance 1 and then the absolute value of the sample is taken to be the service time for a given server when it begins servicing a task.




(a) Server Proportions

(b) Average energy consumption and wait time

Figure 6: Simulation with folded normal service distribution

As we see from Figure 6, the proportion of active servers behaves similarly to the graphs from the generalized Pareto distribution and exponential service distribution. Furthermore, the average energy consumption approaches a limit of about 0.45, which is very similar to the limit observed with the generalized Pareto distribution. The average wait time also vanishes as the expected standby time increases, aligning with the results from [12].

## 3.5 Hyperexponential Service Time Distribution

Finally, we simulate the system under TABS scheme with a Hyperexponential service time distribution. The Hyperexponential distribution is defined as the weighted average of $n$ independent exponential distributions with expectation $\lambda_i \ \forall i = 1, ..., n$. The probability density function of a Hyperexponential distribution with $n$ independent exponential random variables $(X_1, X_2, ..., X_n)$, where each $X_i$ has the pdf of an exponential distribution with mean from the corresponding entry in the mean vector $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$. The probability of each $X_i$ is determined by the probability vector $p = (p_1, p_2, ..., p_n)$ $s.t.$ $\sum_{i=1}^{n} p_i = 1$ is given by

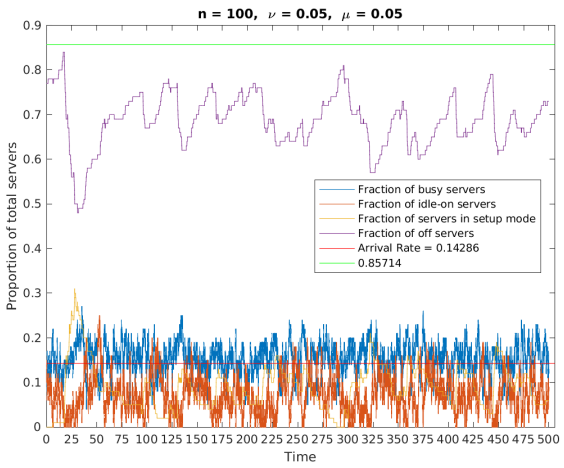
$$f_Y(y) = \sum_{i=1}^{n} p_i f_{X_i}(x)$$

To simulate from the Hyperexponential with expected value of 1 we use the following pseudocode:

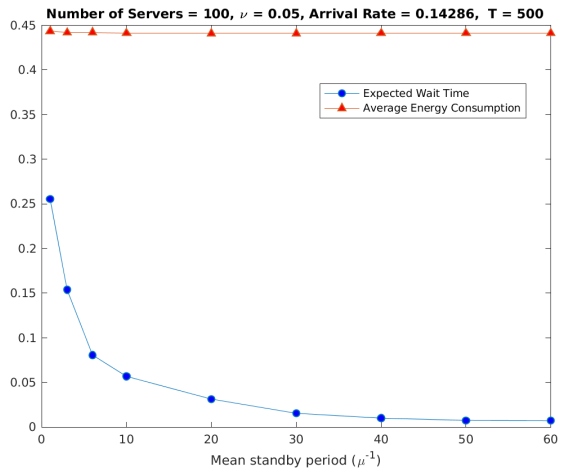
- Set $n = 3$, that is, $(X_1, X_2, X_3)$ with a mean vector of $(1.2, .9, 1)$. Assign probabilities of $(0.1, 0.3, 0.6)$ to for the respective $X_i$. (*This distribution has expectation 1 and variance approximately equal to 1*)
- Generate a sample, U, from the uniform distribution on $[0, 1]$
- Given, U sample for the Hyperexponential from $f_Y(y|U = u)$

$$f_Y(y|U = u) = \begin{cases} X_1 \sim \dfrac{1}{1.2} exp(1.2) & \text{if } u \leq 0.1 \\ X_2 \sim \dfrac{1}{0.9} exp(0.9) & \text{if } 0.1 \leq u \leq 0.4 \\ X_3 \sim exp(1) & \text{if } 0.4 \leq u \end{cases}$$

The final sample generated from the exponential random variable is the service time required to service the next task for the server that just started processing a new task. Figure 7, below, shows a sample evolution of the fraction of active servers and the corresponding expected wait time and average energy consumption. We notice that the graphs appear similar to those seen in previous sections as well as the previous TABS results. The convergence limits for average energy consumption and average wait time appear to be identical to the limits observed with generalized Pareto, Folded Normal and exponential service time distributions.




(a) Server Proportions



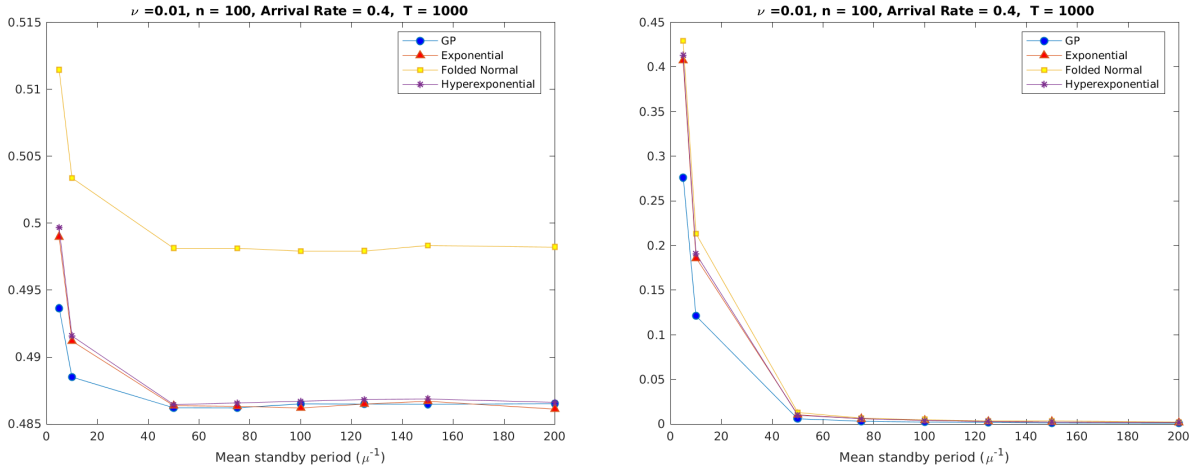

(b) Average energy consumption and wait time

Figure 7: Simulation with Hyperexponential service distribution

## 3.6 Comparing Distributions

So far, we have analysed the affects of various service time distribution on system behaviour individually. These results provide evidence that long term convergence behaviour of TABS remains consistent regardless of the service time distribution. To best understand how the system depends on the various service time distributions, we can plot the average energy consumption and average wait time from simulations with all of the service time distributions together and compare the results.

Figure 8 shows the comparisons between the generalized Pareto, Folded Normal, Hyperexponential and Exponential distributions. The key difference in the plots is the variance of the each of the distributions. The variance for the generalized Pareto distribution is 0.087, for the rest of the distributions the variance is approximately 1. Another key change made in the following simulation plots is that the simulation was run 10 times and each point on the average wait time and energy consumption plots is the average of the 10 simulated runs. The reason for taking the average of 10 simulations is to remove any randomness that may show up in a single run and present the limiting behaviour of the system under this modified TABS algorithm.




(a) Average Energy Consumption (b) Average wait time

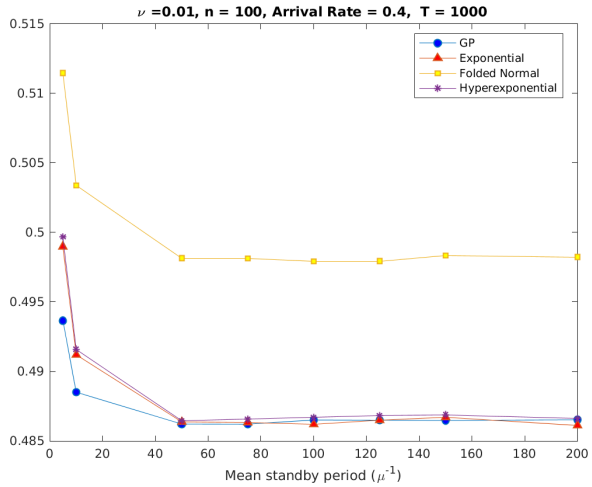
Figure 8: Comparing Multiple Service Time Distributions

We can see clearly in Figure 8(b) that average wait time for a task is approximately the same for each distribution at each $\mu$ value as well as in the limit $\mu \to \infty$. The key difference in results appears in 8(a) where the Folded Normal distribution has a strictly higher energy consumption than any other distribution for all $\mu$ values. Although, we do note that the general trend of energy consumption remains the same across all distributions. It is also worth noting that although the folded normal seems to consume more energy than any of the other distributions, the difference between the amount of the energy consumption is at most 0.02, which may arguably be a statistically insignificant difference.

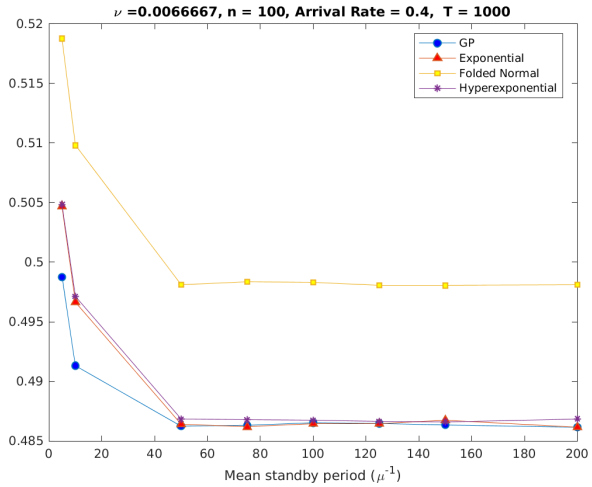
### 3.6.1 Comparing Different Setup Times

To ensure this pattern is not unique to the set of parameters chosen for the simulation presented in Figure 8, we run the simulation with various different setup times and compare the average wait time and energy consumption across the different service time distributions.

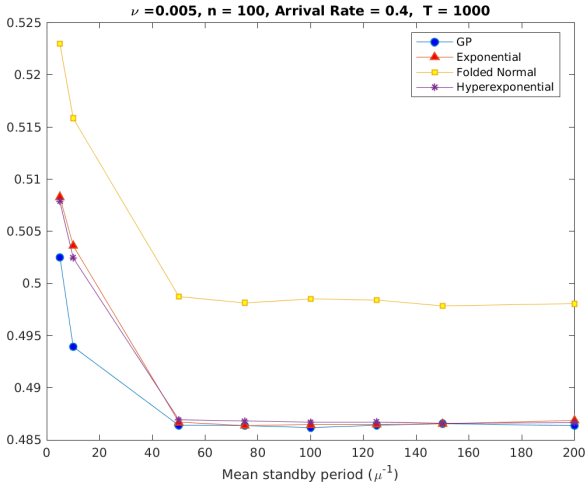
**Average Energy Consumption:** We notice identical behaviour in energy consumption across distributions and different average setup time in Figure 9. It is noteworthy that the convergence limits for each distribution remain unchanged with varying setup times.




(a) $\nu$=100s



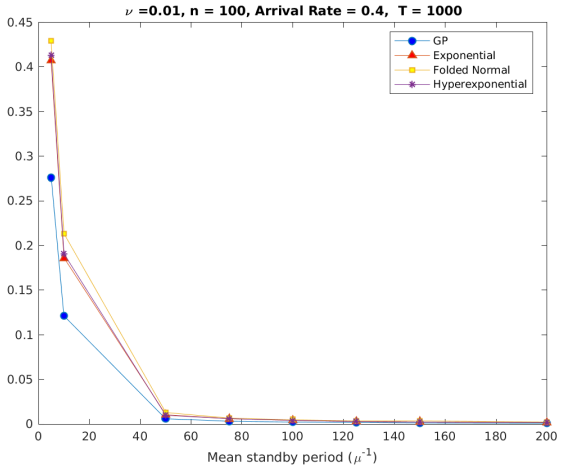

(b) $\nu$=150s



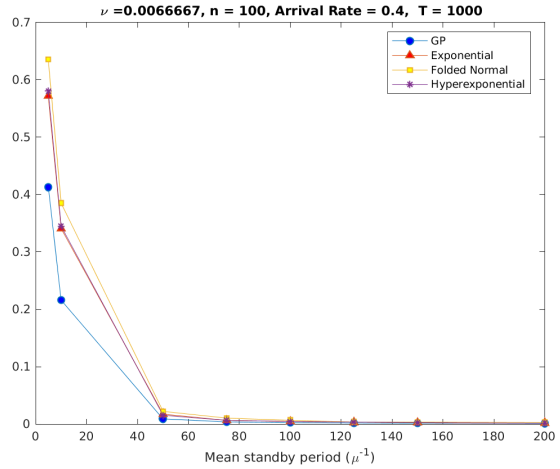

(c) $\nu$=200s

Figure 9: Average Energy consumption with varying setup times

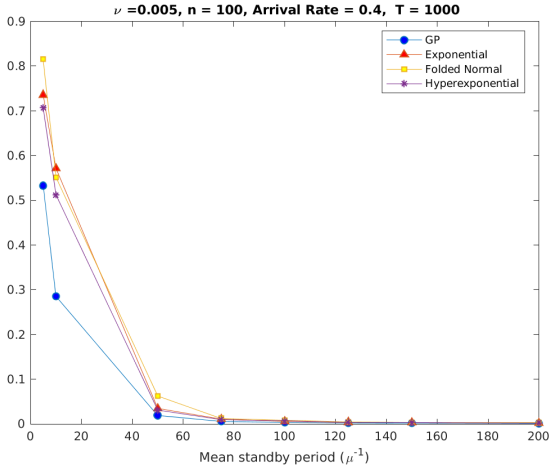
**Average Wait Time:** As with the average energy consumption trend, Figure 10 shows that the trend across varying distributions remains unchanged with respect to setup time. Furthermore, the limit of each distribution approaches zero as $\mu \to \infty$ for all setup times at about the same rate.




(a) $\nu$=100s




(b) $\nu$=150s

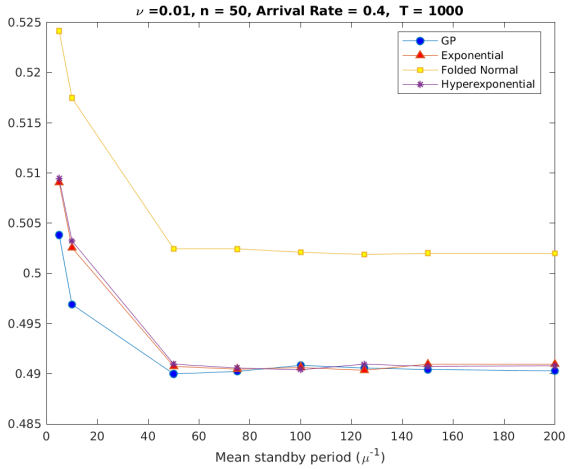



(c) $\nu$=200s

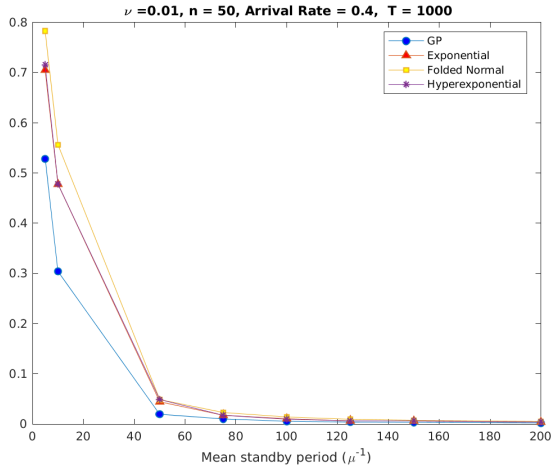
Figure 10: Average wait time with varying setup times

### 3.6.2 Comparing Number of Servers

We can also run the simulation with varying number of servers to see whether the observed patterns hold. The results in Figure 11 are from a system with 50 servers with $\nu = 100s$ and $\lambda = 0.4$. We note that the observed patterns remain constant within these graphs as well, further contributing to the hypothesis that TABS is consistent independent of the service time distribution.




(a) Average Energy Consumption



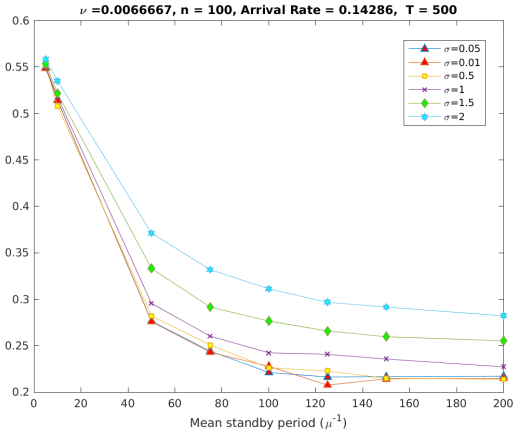

(b) Average wait time

Figure 11: Comparing Multiple Service Time Distributions

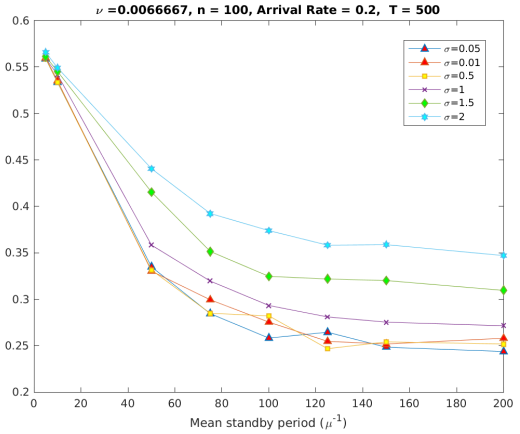
### 3.6.3 Comparing Variance Effect

Finally, we can take a closer look at the effect of changing the variance of the service time distribution. For simplicity of adjusting variance, we consider only the Folded Normal distribution. Each point on the graphs in Figures 12 and 13 is obtained by running the simulation 10 times and taking the average value to remove the randomness of any one simulation. We generate each average wait time and average energy consumption figure for various different arrival rates to observe the interaction between changing variance and arrival rates. The average setup time in all simulations is fixed at 150s.

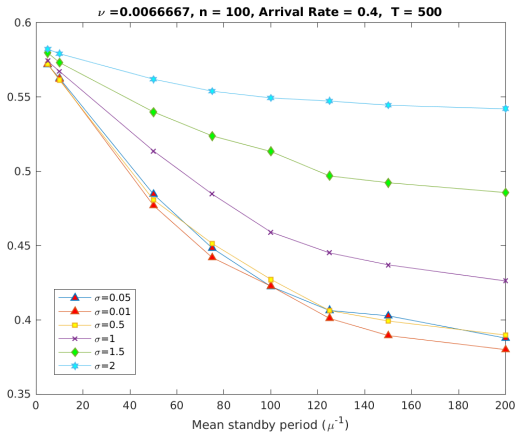
**Average Energy Consumption:** Figure 12 shows that as the variance of the distribution increases, the average energy consumed by the system increases across all $\mu$ values.




(a) $\lambda$=0.15




(b) $\lambda$=0.2

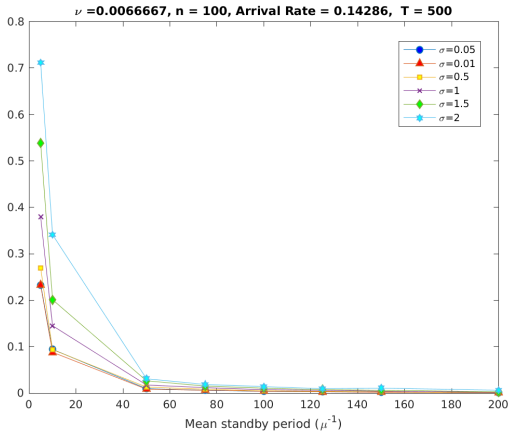



(c) $\lambda$=0.4

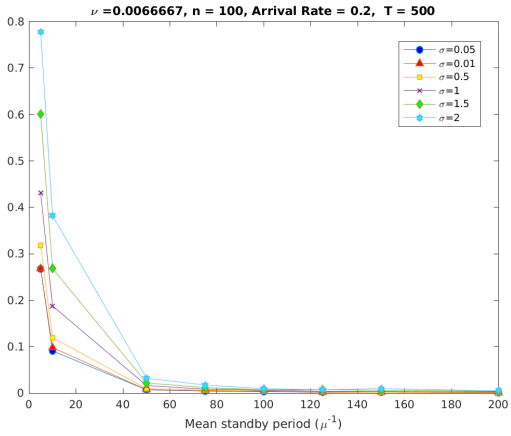
Figure 12: Average energy consumption with varying arrival rates

The general trend in Figure 12 is that energy consumption falls as $\mu$ increases holds across all combinations of distribution variance and arrival rates. Furthermore, the magnitude of average energy consumption does not change drastically across varying arrival rates.

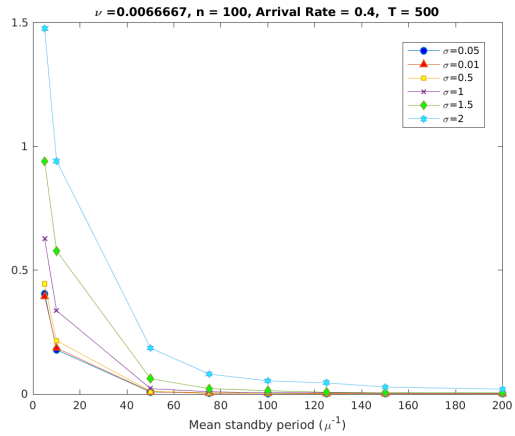
**Average Wait time:** In Figure 13 we see once again that as variance increases, the average task wait time increases. Upon close inspection of the graphs, we note the main difference in these trends from the energy consumption trends in Figure 12 is that the increase in average wait time seems to persist only for lower $\mu$ values. After about $\mu = 70$ for arrival rates $\lambda = 0.15, 0.2$ the difference in wait time across service distributions with different variances appear to vanish. For the graph with $\lambda = 0.4$ this difference seems negligible for $\mu \geq 100$ for all service distributions except the distribution with variance of 2. The distribution with variance of 2 in the system with $\lambda = 0.4$ seems to realize consistently higher wait times for all $\mu$ values. However, we maintain that the general trend across different variance values remains the same.




(a) $\lambda$=0.15




(b) $\lambda$=0.2




(c) $\lambda$=0.4

Figure 13: Average wait time with varying arrival rates

It is worth noting that this analysis looks only at first order behaviour of the system and that by conducting second order analysis on the average wait time and number of servers in setup mode, we may obtain a better understanding of the impact of different service time distributions on the performance of the TABS Scheme.

We conclude this section with the hypothesis that TABS behaviour is independent of the service time distribution and, to an extent, its variance conditioned upon the expectation of the distribution being 1. Each of the figures in this section provide an argument in favor of this hypothesis. Further sensitivity analysis will allow for a rigorous proof of this hypothesis.

# 4 Limit Theorem Proof

We now move to discuss the behaviour of the queuing system under TABS with respect to $\mu$ and $\nu$. As noted in Section 2, previous results for the TABS scheme have proven its convergence limit and stationary distribution under fluid-limit scaling [11, 12]. The limits proved to be independent of both $\mu$ and $\nu$ parameters [11]. From the discussion in Section 3.1, we are motivated to find the dependence of mean behaviour of the system on $\mu$ and $\nu$ values for long time intervals. This result will allow large-scale data centers to optimize $\mu$ and $\nu$ with respect to system performance. In this section we present a proof for the convergence of the system representation under TABS with diffusion scaling to a multi-dimensional Brownian motion that captures the effect of $\mu$ and $\nu$ on the system.

Eschenfeldt and Gamarnik [3] provide a proof for the diffusion scaling limit theorem for a queuing system with all servers always on. They consider a slightly modified version of the scheme. Specifically, they prevent any server's queue from being greater than 2. They also restrict the number of servers with queue length 2 to be no more that $B\sqrt{n}$. In the limit these restrictions prove to emulate the system without any restrictions on queue lengths. These

assumptions are helpful in simplifying the analysis and largely do not affect the validity and application of the result to the general model.

Our goal is to find a similar limiting behaviour for the TABS scheme. We will also assume that queue lengths cannot be greater than 2 and that a maximum of $B\sqrt{n}$ servers can have a queue length of 2. Additionally, we will assume that the service time distribution is exponential with expectation 1 as was done in the initial proposal of the TABS scheme [12, 11].

In the following analysis $Q_i^n(t)$ represents the number of queues with at least i customers at time $t \geq 0$. We know that $Q_1^n(t) \leq n$, $\forall t \geq 0$ We start by introducing a centered and scaled version of the system, defined by

$$X_1^n(t) = \frac{(Q_1^n(t) - \lambda n)}{\sqrt{n}} \tag{4.1}$$

$$X_2^n(t) = \frac{Q_2^n(t)}{\sqrt{n}} \tag{4.2}$$

$$\overline{\Delta_0^n}(t) = \frac{\Delta_0^n(t) - (1-\lambda)n}{\sqrt{n}} \tag{4.3}$$

$$\overline{\Delta_1^n}(t) = \frac{\Delta_1^n(t)}{\sqrt{n}} \tag{4.4}$$

In addition to the earlier bounds set on $Q_1^n(t)$ and $Q_2^n(t)$, we will require the following equation to hold for all $t \geq 0$. This is the reflected diffusion constraint of the system.

$$\Delta_0^n(t) + \Delta_1^n(t) + Q_1^n(t) \leq n \tag{4.5}$$

## 4.1 System Evolution

Given the scaled and centered representations of the 4 random variables that define the system, we can build equations to show how each variable evolves with time. Equations 4.6-4.9 show the evolution of the system with the random variables that model the system.

$$Q_1^n(t) = Q_1^n(0) + A_1(\lambda n t) - D_1\left(\int_0^t (Q_1^n(s) - Q_2^n(s))ds\right) - U_1^n(t) \tag{4.6}$$

$$Q_2^n(t) = Q_2^n(0) + U_1^n(t) - D_2\left(\int_0^t Q_2^n(s)ds\right) - U_2^n(t) \tag{4.7}$$

$$\begin{aligned}\Delta_0^n(t) = \Delta_0^n(0) &+ A_2\left(\int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s))ds\right) \\ &- D_3\left(\int_0^t \mathbb{1}\{Q_1^n(s) + \Delta_0^n(s) + \Delta_1^n(s) = n)\}\mathbb{1}\{\Delta_0^n(s) > 0\}dA(\lambda n s)\right)\end{aligned} \tag{4.8}$$

$$\Delta_1^n(t) = \Delta_1^n(0) + \int_0^t \mathbb{1}\{Q_1^n(s) + \Delta_0^n(s) + \Delta_1^n(s) = n)\}\mathbb{1}\{\Delta_0^n(s) > 0\} dA(\lambda n s)$$
$$-D_4\left(\int_0^t \nu \Delta_1^n(s) ds\right) \tag{4.9}$$

In the equations above, $A_i(\cdot)$ and $D_k(\cdot)$ represent point processes that account for increases and decreases, respectively, for each variable over time. $U_1^n(t)$ is the number of arrivals in [0,t] when every server has at least one task. $U_2^n(t)$ is the number of arrivals in [0,t] when every server has at least one task and *exactly* $B\sqrt{n}$ servers have two tasks. (Implicitly we require that $B \leq \sqrt{n}$).

Below we represent how $U_1^n(t)$ and $U_2^n(t)$ evolve with time. We also rewrite the evolution in terms of the scaled and centered random variables from 4.1-4.4. This will prove useful for calculations in the following section.

$$\begin{aligned} U_1^n(t) &= \int_0^t \mathbb{1}\{Q_1^n(s) - n + \Delta_0^n + \Delta_1^n = 0\} dA(\lambda n s) \\ &= \int_0^t \mathbb{1}\{Q_1^n(s) - \lambda n + \Delta_0^n - (1-\lambda)n + \Delta_1^n = 0\} dA(\lambda n s) \\ &= \int_0^t \mathbb{1}\left\{\frac{Q_1^n(s) - \lambda n + \Delta_0^n - (1-\lambda)n + \Delta_1^n}{\sqrt{n}} = 0\right\} dA(\lambda n s) \\ &= \int_0^t \mathbb{1}\left\{X_1^n(s) + \overline{\Delta_0^n}(s) + \overline{\Delta_1^n}(s) = 0\right\} dA(\lambda n s) \end{aligned} \tag{4.10}$$

$$\begin{aligned} U_2^n(t) &= \int_0^t \mathbb{1}\{Q_1^n(s) - n + \Delta_0^n + \Delta_1^n = 0, Q_2^n(s) - B\sqrt{n - \Delta_0^n - \Delta_1^n} = 0\} dA(\lambda n s) \\ &= \int_0^t \mathbb{1}\left\{Q_1^n(s) - \lambda n + \Delta_0^n - (1-\lambda)n + \Delta_1^n = 0, \frac{Q_2^n(s) - B\sqrt{n - \Delta_0^n - \Delta_1^n}}{\sqrt{n}} = 0\right\} dA(\lambda n s) \\ &= \int_0^t \mathbb{1}\left\{\frac{Q_1^n(s) - \lambda n + \Delta_0^n - (1-\lambda)n + \Delta_1^n}{\sqrt{n}} = 0, \frac{Q_2^n(s) - B\sqrt{n - \Delta_0^n - \Delta_1^n}}{\sqrt{n}} = 0\right\} dA(\lambda n s) \\ &= \int_0^t \mathbb{1}\left\{X_1^n(s) + \overline{\Delta_0^n}(s) + \overline{\Delta_1^n}(s) = 0, X_2^n(s) - B\sqrt{\frac{\lambda n + (1-\lambda)n - \Delta_0^n - \Delta_1^n}{n}} = 0\right\} dA(\lambda n s) \\ &= \int_0^t \mathbb{1}\left\{X_1^n + \overline{\Delta_0^n} + \overline{\Delta_1^n} = 0, X_2^n - B\sqrt{\lambda - (\overline{\Delta_0^n} - \overline{\Delta_1^n})\sqrt{n}} = 0\right\} dA(\lambda n s) \end{aligned} \tag{4.11}$$

## 4.2 Martingale Equations

Now we can rewrite the time changes of Poisson processes from 4.6-4.9 as time changes of scaled Poisson processes. To do this, we first define scaled martingales that will simplify the

scaled and centered representation of the system's evolution in the next section.

$$M_1^{n,1} = \frac{1}{\sqrt{n}} A_1(\lambda n t) - \lambda \sqrt{n} t \tag{4.12}$$

$$M_1^{n,2} = \frac{1}{\sqrt{n}} A_2 \left( \int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s)) ds \right) - \frac{1}{\sqrt{n}} \int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s)) ds \tag{4.13}$$

$$M_2^{n,1} = \frac{1}{\sqrt{n}} D_1 \left( \int_0^t (Q_1^n(s) - Q_2^n(s)) ds \right) - \frac{1}{\sqrt{n}} \int_0^t (Q_1^n(s) - Q_2^n(s)) ds \tag{4.14}$$

$$M_2^{n,2} = \frac{1}{\sqrt{n}} D_2 \left( \int_0^t Q_2^n(s) ds \right) - \frac{1}{\sqrt{n}} \int_0^t Q_2^n(s) ds \tag{4.15}$$

$$\begin{aligned} M_2^{n,3} = & \frac{1}{\sqrt{n}} D_3 \left( \int_0^t \mathbb{1}\{Q_1^n(s) + \Delta_0^n(s) + \Delta_1^n(s) = n)\} \mathbb{1}\{\Delta_0^n(s) > 0\} dA(\lambda n s) \right) \\ & - \frac{1}{\sqrt{n}} \int_0^t \mathbb{1}\{Q_1^n(s) + \Delta_0^n(s) + \Delta_1^n(s) = n)\} \mathbb{1}\{\Delta_0^n(s) > 0\} dA(\lambda n s) \end{aligned} \tag{4.16}$$

$$M_2^{n,4} = \frac{1}{\sqrt{n}} D_4 \left( \int_0^t \nu \Delta_1^n(s) ds \right) - \frac{1}{\sqrt{n}} \int_0^t \nu \Delta_1^n(s) ds \tag{4.17}$$

For simplicity of calculations in the following sections, we also introduce scaled versions of $U_1^n(t)$ and $U_2^n(t)$

$$V_1^n(t) = \frac{U_1^n(t)}{\sqrt{n}} \quad \text{and} \quad V_2^n(t) = \frac{U_2^n(t)}{\sqrt{n}} \tag{4.18}$$

## 4.3 Martingale Decomposition

Given the martingales from section 4.2, we can center, scale and simplify each of the evolution equations (4.6-4.9) to get an appropriate martingale decomposition to represent the entire system.

Starting with $Q_1^n(t)$, we subtract and add the expectation of each of the Poisson processes

$$\begin{aligned} Q_1^n(t) = & Q_1^n(0) + (A_1(\lambda n t) - \lambda n t) + \lambda n t \\ & - \left( D_1 \left( \int_0^t (Q_1^n(s) - Q_2^n(s)) ds \right) - \int_0^t (Q_1^n(s) - Q_2^n(s)) ds \right) \\ & - \int_0^t (Q_1^n(s) - Q_2^n(s)) ds - U_1^n(t) \end{aligned}$$

Next, we scale by $\frac{1}{\sqrt{n}}$ and rearrange the variables. Note that we now have martingales constructed in 4.12-4.17 appear on the RHS above. By substituting for these martingales, we get the following equation:

$$\frac{(Q_1^n(t)-\lambda n)}{\sqrt{n}} = \frac{Q_1^n(0)-\lambda n}{\sqrt{n}} + M_1^{n,1}(t) - M_2^{n,1}(t) - \frac{1}{\sqrt{n}}\left(\int_0^t (Q_1^n(s)-Q_2^n(s))ds + U_1^n(t)\right)$$

Finally, we substitute in the scaled representations to get a final martingale decomposition for the first random variable of the system.

$$\boxed{X_1^n(t) = X_1^n(0) + M_1^{n,1}(t) - M_2^{n,1}(t) - \int_0^t (X_1^n(s)-X_2^n(s))ds - V_1^n(t)}$$

We repeat this process for each of the other three evolution equations from Section 4.1. For each of the remaining three equations from 4.6-4.9, first we center, then we scale and finally we substitute the martingale representations.

$$\begin{aligned} Q_2^n(t) &= Q_2^n(0) + U_1^n(t) - D_2\left(\int_0^t Q_2^n(s)ds\right) - U_2^n(t) \\ &= Q_2^n(0) + U_1^n(t) - \left(D_2\left(\int_0^t Q_2^n(s)ds\right) - \int_0^t Q_2^n(s)ds\right) - \int_0^t Q_2^n(s)ds - U_2^n(t) \end{aligned}$$

$$\frac{Q_2^n(t)}{\sqrt{n}} = \frac{Q_2^n(0) + U_1^n(t) - \left(D_2\left(\int_0^t Q_2^n(s)ds\right) - \int_0^t Q_2^n(s)ds\right) - \int_0^t Q_2^n(s)ds - U_2^n(t)}{\sqrt{n}}$$

$$\boxed{X_2^n(t) = X_2^n(0) + V_1^n(t) - M_2^{n,2}(t) - \int_0^t X_2^n(s)ds - V_2^n(t)}$$

$$\begin{aligned} \Delta_0^n(t) &= \Delta_0^n(0) + A_2\left(\int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s))ds\right) \\ &\quad - D_3\left(\int_0^t \mathbb{1}\{Q_1^n(s)+\Delta_0^n(s)+\Delta_1^n(s)=n)\}\mathbb{1}\{\Delta_0^n(s)>0\}dA(\lambda n s)\right)^{1} \\ &= (\Delta_0^n(0) - (1-\lambda)n) + (1-\lambda)n \\ &\quad + \left(A_2\left(\int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s))ds\right) - \int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s))ds\right) \\ &\quad - \left(D_3\left(\int_0^t \mathbb{1}\{\Delta_0^n(s)>0\}dU_1^n(s)\right) - \int_0^t \mathbb{1}\{\Delta_0^n(s)>0\}dU_1^n(s)\right) \\ &\quad + \int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s))ds - \int_0^t \mathbb{1}\{\Delta_0^n(s)>0\}dU_1^n(s) \end{aligned}$$

[1]Note that by nature of the indicator function inside the integral in $M_2^{n,3}$, we can simplify the integral to

$$\int_0^t \mathbb{1}\{Q_1^n(s)+\Delta_0^n(s)+\Delta_1^n(s)=n)\}\mathbb{1}\{\Delta_0^n(s)>0\}dA(\lambda n s) = \int_0^t \mathbb{1}\{\Delta_0^n(s)>0\}dU_1^n(s)$$

$$\begin{aligned}\frac{\Delta_0^n(t)}{\sqrt{n}} = &\frac{(\Delta_0^n(0)-(1-\lambda)n)+(1-\lambda)n}{\sqrt{n}}\\ &\frac{+\left(A_2\left(\int_0^t \mu(n-Q_1^n(s)-\Delta_1^n(s))ds\right)-\int_0^t \mu(n-Q_1^n(s)-\Delta_1^n(s))ds\right)}{\sqrt{n}}\\ &\frac{-\left(D_3\left(\int_0^t \mathbb{1}\{\Delta_0^n(s)^n>0\}dU_1^n(s)\right)-\int_0^t \mathbb{1}\{\Delta_0^n(s)>0\}dU_1^n(s)\right)}{\sqrt{n}}\\ &\frac{+\int_0^t \mu(n-Q_1^n(s)-\Delta_1^n(s))ds-\int_0^t \mathbb{1}\{\Delta_0^n(s)>0\}dU_1^n(s)}{\sqrt{n}}\end{aligned}$$

$$\begin{aligned}\overline{\Delta_0^n}(t) &= \overline{\Delta_0^n}(0)+M_1^{n,2}(t)-M_2^{n,3}(t)+\int_0^t \frac{\mu(n-Q_1^n(s)-\Delta_1^n(s))}{\sqrt{n}}ds\\ &\quad-\int_0^t \frac{\mathbb{1}\{\Delta_0^n(s)>0\}}{\sqrt{n}}dU_1^n(s)\\ &= \overline{\Delta_0^n}(0)+M_1^{n,2}(t)-M_2^{n,3}(t)+\int_0^t \frac{\mu(\lambda n+(1-\lambda)n-Q_1^n(s)-\Delta_1^n(s))}{\sqrt{n}}ds\\ &\quad-\int_0^t \frac{\mathbb{1}\{\Delta_0^n(s)-(1-\lambda)n>-(1-\lambda)n\}}{\sqrt{n}}dU_1^n(s)\\ &= \overline{\Delta_0^n}(0)+M_1^{n,2}(t)-M_2^{n,3}(t)+\int_0^t \mu(\sqrt{n}(1-\lambda)-X_1^n(s)-\overline{\Delta_1^n}(s))ds\\ &\quad-\int_0^t \mathbb{1}\{-\overline{\Delta_0^n}(s)<\sqrt{n}(1-\lambda)\}dU_1^n(s)\end{aligned}$$

$$\boxed{\begin{aligned}\overline{\Delta_0^n}(t) = \overline{\Delta_0^n}(0)+M_1^{n,2}(t)-M_2^{n,3}(t)+\int_0^t \mu(\sqrt{n}(1-\lambda)-X_1^n(s)-\overline{\Delta_1^n}(s))ds\\ -\int_0^t \mathbb{1}\{-\overline{\Delta_0^n}(s)<\sqrt{n}(1-\lambda)\}dU_1^n(s)\end{aligned}}$$

$$\begin{aligned}\Delta_1^n(t) &= \Delta_1^n(0)+\int_0^t \mathbb{1}\{Q_1^n(s)+\Delta_0^n(s)+\Delta_1^n(s)=n)\}\mathbb{1}\{\Delta_0^n(s)>0\}dA(\lambda ns)\\ &\quad -D_4\left(\int_0^t \nu\Delta_1^n(s)ds\right)\\ &= \Delta_1^n(0)+\int_0^t \mathbb{1}\{\Delta_0^n(s)>0\}dU_1^n(s)\\ &\quad -\left(D_4\left(\int_0^t \nu\Delta_1^n(s)ds\right)-\int_0^t \nu\Delta_1^n(s)ds\right)-\int_0^t \nu\Delta_1^n(s)ds\end{aligned}$$

$$\frac{\Delta_1^n(t)}{\sqrt{n}} = \frac{\Delta_1^n(0) + \int_0^t \mathbb{1}\{\Delta_0^n(s) > 0\} dU_1^n(s)}{\sqrt{n}} \\ \frac{-\left(D_4\left(\int_0^t \nu\Delta_1^n(s)ds\right) - \int_0^t \nu\Delta_1^n(s)ds\right) - \int_0^t \nu\Delta_1^n(s)ds}{\sqrt{n}}$$

$$\overline{\Delta_1^n}(t) = \overline{\Delta_1^n}(0) - M_2^{n,4}(t) + \int_0^t \frac{\mathbb{1}\{\Delta_0^n(s) - (1-\lambda)n > -(1-\lambda)n\}}{\sqrt{n}} dU_1^n(s) - \int_0^t \frac{\nu\Delta_1^n(s)}{\sqrt{n}} ds$$
$$= \overline{\Delta_1^n}(0) - M_2^{n,4}(t) + \int_0^t \mathbb{1}\{-\overline{\Delta_0^n}(s) < \sqrt{n}(1-\lambda)\} dU_1^n(s) - \int_0^t \nu\overline{\Delta_1^n}(s) ds$$

$$\boxed{\overline{\Delta_1^n}(t) = \overline{\Delta_1^n}(0) - M_2^{n,4}(t) + \int_0^t \mathbb{1}\{-\overline{\Delta_0^n}(s) < \sqrt{n}(1-\lambda)\} dU_1^n(s) - \int_0^t \nu\overline{\Delta_1^n}(s) ds}$$

Thus, we have the final four martingale decompositions:

$$X_1^n(t) = X_1^n(0) + M_1^{n,1}(t) - M_2^{n,1}(t) - \int_0^t (X_1^n(s) - X_2^n(s)) ds - V_1^n(t) \tag{4.19}$$

$$X_2^n(t) = X_2^n(0) + V_1^n(t) - M_2^{n,2}(t) - \int_0^t X_2^n(s) ds - V_2^n(t) \tag{4.20}$$

$$\overline{\Delta_0^n}(t) = \overline{\Delta_0^n}(0) + M_1^{n,2}(t) - M_2^{n,3}(t) + \int_0^t \mu(\sqrt{n}(1-\lambda) - X_1^n(s) - \overline{\Delta_1^n}(s)) ds \\ - \int_0^t \mathbb{1}\{-\overline{\Delta_0^n}(s) < \sqrt{n}(1-\lambda)\} dU_1^n(s) \tag{4.21}$$

$$\overline{\Delta_1^n}(t) = \overline{\Delta_1^n}(0) - M_2^{n,4}(t) + \int_0^t \mathbb{1}\{-\overline{\Delta_0^n}(s) < \sqrt{n}(1-\lambda)\} dU_1^n(s) - \int_0^t \nu\overline{\Delta_1^n}(s) ds \tag{4.22}$$

## 4.4 Martingale Behaviour

### 4.4.1 Quadratic Variation

We now turn to inspect each of the martingale equations in further detail. The first step is to identify the square-integrable martingales and their quadratic variations and covariations with each martingale.

**Definition 4.1.** A martingale, $M(t) \equiv \{M(t) \colon t \geq 0\}$(with respect to some filtration), is **square-integrable** if $\mathbb{E}[M(t)^2] \leq \infty \; \forall \; t \geq 0$. [13]

From Theorem 7.2 of [13], we can identify $M_1^{n,1}, M_1^{n,2}, M_2^{n,1}, M_2^{n,2}, M_2^{n,3}, M_2^{n,4}$ as square integrable with respect to an appropriate filtration. We also obtain the following predictable quadratic variations(PQV) for each of the martingales.

$$\langle M_1^{n,1} \rangle(t) = \lambda t \tag{4.23}$$

$$\langle M_1^{n,2} \rangle(t) = \frac{1}{n} \int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s)) ds \tag{4.24}$$

$$\langle M_2^{n,1} \rangle(t) = \frac{1}{n} \int_0^t (Q_1^n(s) - Q_2^n(s)) ds \tag{4.25}$$

$$\langle M_2^{n,2} \rangle(t) = \frac{1}{n} \int_0^t Q_2^n(s) ds \tag{4.26}$$

$$\langle M_2^{n,3} \rangle(t) = \frac{1}{n} \int_0^t \mathbb{1}\{\Delta_0^n(s) > 0\} dU_1^n(s) \tag{4.27}$$

$$\langle M_2^{n,4} \rangle(t) = \frac{1}{n} \int_0^t \nu \Delta_1^n(s) ds \tag{4.28}$$

### 4.4.2 Quadratic Covariation

Now, we find the quadratic covariation for each pair of martingales. From [13] we have the formula to derive the quadratic covariation of two square-integrable martingales as

$$\langle M_1, M_2 \rangle = \frac{1}{2}(\langle M_1 + M_2 \rangle - \langle M_1 \rangle - \langle M_2 \rangle)$$

To determine the quadratic covariation, we first need to identify the instantaneous rate change of each of the point processes that make up the each of the martingales. Once we have the functions, we can compute the variance of each pair to identify the limiting behaviour and thus the quadratic covariation between each pair of martingales. The instantaneous rate change of each point process below is identified by the taking the derivative of the rate function with respect to the time variable $t$.

$$A_1(\lambda n t) \Rightarrow \lambda n$$

$$A_2\left(\int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s)) ds\right) \Rightarrow \mu(n - Q_1^n(t) - \Delta_1^n(t))$$

$$D_1\left(\int_0^t (Q_1^n(s) - Q_2^n(s)) ds\right) \Rightarrow Q_1^n(t) - Q_2^n(t)$$

$$D_2\left(\int_0^t Q_2^n(s)ds\right) \Rightarrow Q_2^n(t)$$

$$D_3\left(\int_0^t \mathbb{1}\{\Delta_0^n(s) > 0\}dU_1^n(s)\right) \Rightarrow \mathbb{1}\{\Delta_0^n(t) > 0\}$$

$$D_4\left(\int_0^t \nu\Delta_1^n(s)ds\right) \Rightarrow \nu\Delta_1^n(t)$$

Given these functions, we can now determine the quadratic covariance between each pair of martingales. Note that for the $M_1^{n,1}$ the instantaneous rate change is a constant function, i.e. the function does not depend on t. This gives us the feature that the covariance of $M_1^{n,1}$ with respect to all other martingales will be 0. Additionally, for $M_2^{n,3}$ the instantaneous rate is an indicator function which is assumed to be 1 for the purpose of this system set up. This is because from previous results we know that $\Delta_0^n(t)$ hovers around $(1-\lambda)n$ and so the probability that $\Delta_0^n(t)$ reaches 0 is exponentially decreases as n increases. Thus the quadratic covariation of $M_2^{n,3}$ with all other martingales is 0. We calculate the quadratic covariation of the rest of the paired equations below:

$$\begin{aligned}\mathrm{C}ov(\mu(n - Q_1^n(t) - \Delta_1^n(t)), Q_1^n(t) - Q_2^n(t)) &= -\mu\mathrm{V}ar(Q_1^n(t)) + \mu\mathrm{C}ov(Q_1^n(t), Q_2^n(t)) \\ &\quad -\mu\mathrm{C}ov(\Delta_1^n(t), Q_1^n(t)) + \mu\mathrm{C}ov(\Delta_1^n(t), Q_2^n(t)) \\ &= \mu[-o(\sqrt{n}) + o(\sqrt{n}) + O(\sqrt{n} - O(\sqrt{n})]\end{aligned}$$

If we scale this covariance by $\sqrt{n}$, we can obtain the limit for the quadratic covariance between $M_1^{n,2}$ and $M_2^{n,1}$:

$$\frac{1}{\sqrt{n}}\mathrm{C}ov(\mu(n - Q_1^n(t) - \Delta_1^n(t)), Q_1^n(t) - Q_2^n(t)) \Rightarrow 0$$

$$\boxed{\langle M_1^{n,2}, M_2^{n,1}\rangle = 0} \tag{4.29}$$

$$\begin{aligned}\frac{1}{\sqrt{n}}\mathrm{C}ov(\mu(n - Q_1^n(t) - \Delta_1^n(t)), Q_2^n(t)) &= \frac{1}{\sqrt{n}}\mu\mathrm{C}ov(Q_1^n(t), Q_2^n(t)) - \mu\mathrm{C}ov(\Delta_1^n(t), Q_2^n(t)) \\ &= \frac{1}{\sqrt{n}}\mu(o(\sqrt{n}) - O(\sqrt{n})) \\ &\Rightarrow -a\mu \text{ for some constant } a > 0\end{aligned}$$

Thus, the quadratic covariation is given by:

$$\boxed{\langle M_1^{n,2}, M_2^{n,2}\rangle = \frac{1}{2}\left(-a\mu - \frac{1}{n}\int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s))ds - \frac{1}{n}\int_0^t Q_2^n(s)ds\right)} \tag{4.30}$$

Repeating this process for the rest of the pairs of martingales we get the following equations:

$$\begin{aligned}\frac{1}{\sqrt{n}}\mathrm{C}ov(\mu(n - Q_1^n(t) - \Delta_1^n(t)), \nu\Delta_1^n(t)) &= \frac{1}{\sqrt{n}}\mu\nu\mathrm{C}ov(Q_1^n(t), \Delta_1^n(t)) - \mu\nu\mathrm{V}ar(\Delta_1^n(t)) \\ &= \frac{1}{\sqrt{n}}\mu\nu(O(\sqrt{n}) - o(\sqrt{n})) \\ &\Rightarrow b\mu\nu \text{ for some constant } b > 0\end{aligned}$$

$$\boxed{\langle M_1^{n,2}, M_2^{n,4}\rangle = \frac{1}{2}\left(b\mu\nu - \frac{1}{n}\int_0^t \mu(n - Q_1^n(s) - \Delta_1^n(s))ds - \frac{1}{n}\int_0^t \nu\Delta_1^n(s)ds\right)} \tag{4.31}$$

$$\begin{aligned}\frac{1}{\sqrt{n}}\mathrm{C}ov(Q_1^n(t) - Q_2^n(t), Q_2^n(t)) &= \frac{1}{\sqrt{n}}\mathrm{C}ov(Q_1^n(t), Q_2^n(t)) - \mathrm{V}ar(Q_2^n(t)) \\ &= \frac{1}{\sqrt{n}}o(\sqrt{n}) - o(\sqrt{n}) \\ &\Rightarrow 0\end{aligned}$$

$$\boxed{\langle M_2^{n,1}, M_2^{n,2}\rangle = 0} \tag{4.32}$$

$$\begin{aligned}\frac{1}{\sqrt{n}}\mathrm{C}ov(Q_1^n(t) - Q_2^n(t), \nu\Delta_1^n(t)) &= \frac{1}{\sqrt{n}}\nu\mathrm{C}ov(Q_1^n(t), \Delta_1^n(t)) - \nu\mathrm{C}ov(Q_2^n(t), \Delta_1^n(t)) \\ &= \frac{1}{\sqrt{n}}\nu(O(\sqrt{n}) - O(\sqrt{n})) \\ &\Rightarrow 0\end{aligned} \tag{4.33}$$

$$\boxed{\langle M_2^{n,1}, M_2^{n,4}\rangle = 0} \tag{4.34}$$

$$\begin{aligned}\frac{1}{\sqrt{n}}\mathrm{C}ov(Q_2^n(t), \nu\Delta_1^n(t)) &= \frac{1}{\sqrt{n}}\nu\mathrm{C}ov(Q_2^n(t), \Delta_1^n(t)) \\ &= \frac{1}{\sqrt{n}}\nu O(\sqrt{n}) \\ &\Rightarrow c\nu \text{ for some constant } c > 0\end{aligned} \tag{4.35}$$

$$\boxed{\langle M_2^{n,2}, M_2^{n,4}\rangle = \frac{1}{2}\left(c\nu - \frac{1}{n}\int_0^t Q_2^n(s)ds - \frac{1}{n}\int_0^t \nu\Delta_1^n(s)ds\right)} \tag{4.36}$$

Furthermore, based on Equations 4.10 and 4.6-4.9, we can derive some additional system constraints. By Equation 4.10, we have:

$$\begin{aligned}0 &= \int_0^\infty \mathbb{1}\{Q_1^n(s) \le n - \Delta_0^n - \Delta_1^n\}dU_1^n(s) \\ &= \int_0^\infty \mathbb{1}\{Q_1^n(s) + \Delta_0^n + \Delta_1^n - \lambda n - (1-\lambda)n < 0\}dU_1^n(s) \\ &= \int_0^\infty \mathbb{1}\{X_1^n(s) + \overline{\Delta_0^n}(s) + \overline{\Delta_1^n}(s) < 0\}dV_1^n(s)\end{aligned} \tag{4.37}$$

Additionally, since $U_2^n(s)$ can increase only when $Q_1^n(s) = n - \Delta_0^n - \Delta_1^n$ and $Q_2^n(s) = B\sqrt{n}$, if $Q_2^n(s) < B\sqrt{n}$, $U_2^n(s)$ cannot increase. From this, we arrive at the following condition:

$$\begin{aligned} 0 &= \int_0^\infty \mathbb{1}\{Q_2^n(s) < B\sqrt{n}\} dU_2^n(t) \\ &= \int_0^\infty \mathbb{1}\{X_2^n(s) < B\} dV_2^n(t) \end{aligned} \tag{4.38}$$

The next step in our proof requires convergence of the martingales from equations 4.19-4.22.

## 4.5 Martingale Convergence

Given the martingales are square integrable and we have the predictable quadratic variations of each martingale, we can identify the limits of each PQV. We first define the time changes as follows.

$$\Phi_{A_1,n}(t) = \lambda t \tag{4.39}$$

$$\Phi_{A_2,n}(t) = \mu t - \frac{\mu}{n}\int_0^t Q_1^n(s)ds - \frac{1}{n}\int_0^t \mu\Delta_1^n(s)ds \tag{4.40}$$

$$\Phi_{D_1,n}(t) = \frac{1}{n}\int_0^t Q_1^n(s)ds - \frac{1}{n}\int_0^t Q_2^n(s)ds \tag{4.41}$$

$$\Phi_{D_2,n}(t) = \frac{1}{n}\int_0^t Q_2^n(s)ds \tag{4.42}$$

$$\Phi_{D_3,n}(t) = \frac{1}{n}\int_0^t \mathbb{1}\{\Delta_0^n(s) > 0\} dU_1^n(s) \tag{4.43}$$

$$\Phi_{D_4,n}(t) = \frac{\nu}{n}\int_0^t \Delta_1^n(s)ds \tag{4.44}$$

Now we can rewrite the martingales as

$$\begin{aligned} &M_1^{n,1} = M_{A_1,n} \circ \Phi_{A_1,n}, \quad M_1^{n,2} = M_{A_2,n} \circ \Phi_{A_2,n}, \quad M_2^{n,1} = M_{D_1,n} \circ \Phi_{D_1,n} \\ &M_2^{n,2} = M_{D_2,n} \circ \Phi_{D_2,n}, \quad M_2^{n,3} = M_{D_3,n} \circ \Phi_{D_3,n}, \quad M_2^{n,4} = M_{D_4,n} \circ \Phi_{D_4,n} \end{aligned}$$

Next, we define the limits of the time change equations (4.39-4.44). Before identifying the limits, we define the topological space over which these processes are defined. Let $D = D([0,\infty), \mathbb{R})$ be the space of right continuous functions with left limits mapping $[0,\infty)$ in $\mathbb{R}$. Where necessary, $D^k = D([0,\infty), \mathbb{R}^k)$ for $k \geq 0$ is the product space of these functions. Thus all the following functions and limits are with respect to the domain $D$.

Starting with $\Phi_{A_1,n}$, since $\lambda \in (0,1)$ is a constant,

$$\boxed{\Phi_{A_1,n} \Rightarrow \lambda t \text{ as } n \to \infty} \tag{4.45}$$

Next we look at $\Phi_{D_1,n}$. Note that in $\Phi_{D_1,n}$ the second term is exactly $\Phi_{D_2,n}$ so

$$\Phi_{D_1,n} \Rightarrow f - g \text{ as } n \to \infty$$

where g is the limit of $\Phi_{D_2,n}$ and f is the limit of the first term of $\Phi_{D_1,n}$. To find f and g, we need to show the fluid limits for $Q_1^n(s)$ and $Q_2^n(s)$. We find the fluid limit just as was done in Lemma 6 in [3], reproduced below.

**Lemma 4.2.** *Let $\Psi_i^n$ for $i = 1, 2$ be defined as*

$$\Psi_i^n(t) = \frac{Q_i^n(t)}{n},\ t \geq 0.$$

*Then,*

$$\Psi_1^n \Rightarrow \omega \text{ and } \Psi_2^n \Rightarrow 0 \text{ as } n \to \infty \tag{4.46}$$

*where $\omega(t) = \lambda$ for $t \geq 0$ since $\lambda$ is a constant in our system.*

The proof of this lemma is given in Section 5.1 of [3]. From this we have that f is the identity function in $D$. That is,

$$\frac{1}{n}\int_0^t Q_1^n(s)ds \Rightarrow \lambda t \text{ as } n \to \infty.$$

Similarly, $g(t) = \int_0^t 0ds = 0$, so $g = 0$ on $D$. Putting the two limits together, we have

$$\boxed{\Phi_{D_1,n} \Rightarrow \lambda t \text{ as } n \to \infty} \tag{4.47}$$

and

$$\boxed{\Phi_{D_2,n} \Rightarrow 0 \text{ as } n \to \infty} \tag{4.48}$$

Next we look at $\Phi_{D_4,n} = \frac{\nu}{n}\int_0^t \Delta_1^n(s)ds$. Since $\nu$ represents the average startup time for a server, we know $\nu$ is generally some fixed constant, c, independent of other parameters in the system. Thus, we only need to understand the convergence of the integral $\int_0^t \frac{\Delta_1^n(s)}{n}ds$. We find the fluid limit for $\Delta_1^n(s)$ just as was done for $Q_1^n(s)$ and $Q_1^n(s)$

**Lemma 4.3.** *Let $\Psi_3^n$ be defined as*

$$\Psi_3^n(t) = \frac{\Delta_1^n(t)}{n},\ t \geq 0.$$

*Then,*

$$\Psi_3^n \Rightarrow 0 \text{ as } n \to \infty \tag{4.49}$$

The proof of this lemma is identical to that of Lemma 4.2. Thus, we have that given $\nu \to c$,

$$\boxed{\Phi_{D_4,n} \Rightarrow 0} \tag{4.50}$$

For $\Phi_{A_2,n}$, note that the second term is $\mu$ times the first term of $\Phi_{D_1,n}$ and the second term is precisely $\Phi_{D_4,n}$. Thus, we can write the limit as

$$\Phi_{A_2,n} \Rightarrow k - \mu f - h$$

where k is the limit of $\overline{\Phi}_{A_2,n}$ with

$$\overline{\Phi}_{A_2,n} = \mu t$$

and h is the limit of $\Phi_{D_4,n}$. To find the limit of k we can use the fact that $\mu$ is a constant.

$$\overline{\Phi}_{A_2,n} \Rightarrow \mu t \text{ as } n \to \infty$$

From previous calculations, we have that $f \Rightarrow \lambda t$ as $n \to \infty$ and $\Phi_{D_4,n} \Rightarrow 0$. Putting all the limits together, we have

$$\boxed{\Phi_{A_2,n} \Rightarrow \mu(1-\lambda)t \text{ as } n \to \infty} \tag{4.51}$$

Finally, we look at $\Phi_{D_3,n}$

$$\Phi_{D_3,n}(t) = \frac{1}{n}\int_0^t \mathbb{1}\{\Delta_0^n(s) > 0\} dU_1^n(s) \tag{4.52}$$

As before, we assume that the indicator function $\mathbb{1}\{\Delta_0^n(s) > 0\}$ is equal to 1 for all time intervals. This assumption is made from earlier results with respect to the TABS scheme limiting behaviour $\Delta_0^n$ approaches $(1-\lambda)$. The probability that $\Delta_0^n$ will hit 0 is exponentially unlikely. Using this assumption we get the following convergence result:

$$\boxed{\Phi_{D_3,n}(t) = \frac{1}{n} \Rightarrow 0 \text{ as } n \to \infty} \tag{4.53}$$

## 4.6 Martingale Limit

We can now put together the martingales and show their convergence. That is,

$$(M_1^{n,1}, M_1^{n,2}, M_2^{n,1}, M_2^{n,2}, M_2^{n,3}, M_2^{n,4}) \Longrightarrow (W_1, W_2, W_3, W_4, W_5, W_6) \tag{4.54}$$

with a zero mean vector and $\Sigma$ covariance matrix.

$$\Sigma = \begin{bmatrix} \lambda t & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu t & 0 & f(t) & 0 & g(t) \\ 0 & 0 & \lambda t & 0 & 0 & 0 \\ 0 & f(t) & 0 & 0 & 0 & h(t) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & g(t) & 0 & h(t) & 0 & 0 \end{bmatrix} \tag{4.55}$$

where

$$f(t) = \frac{1}{2}\left(-a\mu - 1 - \lambda t\right)$$

$$g(t) = \frac{1}{2}\left(b\mu\nu - \mu - \mu\lambda t\right)$$
$$h(t) = \frac{c\nu}{2}$$

for fixed $a, b, c > 0$

Thus we have identified the diffusion limit for the TABS system with exponential service time distribution under diffusion scaling. The system converges to Brownian motion, as expected with a specified covariance matrix. This completes the proof of martingale convergence.

We now have a result that fits into the criteria of applying the Skorokhod map [9] to solve the stochastic differential equations governing the system and thus identify the dependence of mean behaviour of the system variables on $\mu$ and $\nu$. The final result can then be understood as a Taylor expansion of mean behaviour of each of the random variables of interest. For example, the mean behaviour for $Q_1^n(t)$ can be broken down into:

$$Q_1^n(t) = n \cdot \hat{Q}_1^n(t) + \sqrt{n} \cdot X_1^n(t) + o(\sqrt{n}) \tag{4.56}$$

Here we see that the first part of the approximation for mean behaviour of $Q_1^n(t)$ was the result obtained by Mukherjee et. al. [12]. The completion of this proof provides the second term in Equation 4.56 (in red).

# 5 Conclusion

## 5.1 Summary of Results

The TABS scheme proposed to optimize data center performance through load-balancing is known to be stable in the limit with respect to time and number of servers under fluid scaling. Previous results about the TABS scheme have assumed an exponential service time distribution. In this thesis, we investigated the behaviour of the TABS scheme under alternative service time distributions through simulations and presented a proof for TABS limit convergence under diffusion scaling conditional on certain system properties being satisfied.

The first section of this thesis presented simulations of the TABS scheme on large-scale data centers with varying service time distributions. We considered three different service time distributions- generalized Pareto, Folded Normal and Hyperexponential service time distributions with expectation $\lambda$ and variance 1. A comparison of performance metrics of average task wait time and average system energy consumption showed that varying service time distributions did not contribute to larges shifts in performance of the TABS scheme in comparison with the standard exponential distribution. We also simulated the TABS scheme under the Folded Normal service time distribution for a range of variance values. Within this range, the TABS scheme is consistent with respect to the service time distribution variance. We observed that higher variance typically led to higher average wait times for tasks in the system as as well as higher overall energy consumption of the data center. The

simulated data center performance allows us to hypothesize the robustness of the TABS scheme's performance under varying service time distribution parameters.

The second half of this thesis presented a convergence proof for the diffusion-scaled TABS system under specific assumptions. We modified the typical setup of the TABS scheme from literature to scale each variable of interest around its long-term expected value. This diffusion scaling of the system variables allowed for a closer look at the randomness of the variables in the limit. We provided a Martingale representation of the four fundamental equations that characterize the evolution of the system under the TABS scheme and use information about the predictable quadratic variance and covariance of the martingales in the equations to identify limit for the system of martingales. The proof showed convergence of the set of Martingales to a multidimensional Brownian motion for which we explicitly calculated the mean vector and covariance matrix.

## 5.2 Open Questions

The simulation results in this thesis appear to corroborate the hypothesis that the TABS scheme contributes to improved performance of large-scale data centers regardless of the service time distribution and to an extent, the variance of the distribution itself. One open question from the results of this thesis is whether the minor differences observed in simulation results with varying service time distributions is statistically significant. Future work could conduct a sensitivity analysis on the TABS scheme to investigate the robustness of the scheme to varying service time distributions. Furthermore, the results presented in this thesis looked at first order effects of changing service time distributions their variance. Second order analysis, done by looking at $\sqrt{n}$ scaled behaviour of number of servers in setup mode or average wait time in the system will provide a thorough analysis of how service time distributions and their variances affect the performance of the queuing system. Further work on the simulation results to take into account the varying tails of each of the distributions to observe their effect of the performance of the system. This could include analysis of TABS with additional service time distributions not considered in this thesis.

The proof of the system convergence can be completed by the application of the Skorokhod map [9]. Additionally, the proof of the TABS system convergence under diffusion scaling in this thesis uses some assumptions on typical server behaviour, we also imposed constraints on number of servers with queue lengths greater than or equal to 1. The proof also assumes that the likelihood of extreme events that may impact the overall performance of the system to be exponentially small. To completely justify the conclusions of this thesis, one would need to prove that these assumptions are indeed satisfied by the system.

# References

[1] Aghajani, Reza, Xingjie Li, and Kavita Ramanan. "The PDE method for the analysis of randomized load balancing networks." Proceedings of the ACM on Measurement and Analysis of Computing Systems 1, no. 2 (2017): 38.

[2] Barroso, Luiz Andr, and Urs Hlzle. "The case for energy-proportional computing." *Computer* 12 (2007): 33-37.

[3] Eschenfeldt, Patrick, and David Gamarnik. "Join the Shortest Queue with Many Servers. The Heavy-Traffic Asymptotics." *Mathematics of Operations Research* (2018).

[4] Gandhi, Anshul, et al. "Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward." ACM SIGMETRICS Performance Evaluation Review. Vol. 41. No. 1. ACM, 2013.

[5] Gandhi, Anshul, Mor Harchol-Balter, and Ivo Adan. "Server farms with setup costs." Performance Evaluation 67, no. 11 (2010): 1123-1138.

[6] Greenberg, Albert, et al. "The cost of a cloud: research problems in data center networks." *ACM SIGCOMM computer communication review* 39.1 (2008): 68-73.

[7] Koomey, Jonathan G. "Estimating total power consumption by servers in the US and the world." (2007).

[8] Koomey, Jonathan G. "Worldwide electricity used in data centers." Environmental research letters 3, no. 3 (2008): 034008.

[9] Kruk, Lukasz, John Lehoczky, Kavita Ramanan, and Steven Shreve. "An explicit formula for the Skorokhod map on [0, a]." The Annals of Probability 35, no. 5 (2007): 1740-1768.

[10] Liu, Zhenhua, et al. "Renewable and cooling aware workload management for sustainable data centers." *ACM SIGMETRICS Performance Evaluation Review*. Vol. 40. No. 1. ACM, 2012.

[11] Mukherjee, Debankur, and Alexander Stolyar. "Join-idle-queue with service elasticity: large-scale asymptotics of a non-monotone system." *arXiv preprint arXiv:1803.07689* (2018).

[12] Mukherjee, Debankur, et al. "Optimal service elasticity in large-scale distributed systems." *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1.1 (2017): 25.

[13] Pang, Guodong, Rishi Talreja, and Ward Whitt. "Martingale proofs of many-server heavy-traffic limits for Markovian queues." *Probability Surveys* 4 (2007): 193-267.

[14] Stankovic, John A. "Simulations of three adaptive, decentralized controlled, job scheduling algorithms." Computer Networks (1976) 8, no. 3 (1984): 199-217.