Optimal Resource Allocation in Synchronized Multi-Tier Internet Services

C. Verhoef, S. Bhulai, R.D. van der Mei

Abstract

Modern Internet systems have evolved from simple monolithic systems to complex multi-tiered architectures. For these systems, providing good response time performance is crucial for the commercial success. In practice, the response-time performance of multi-tiered systems is often degraded by severe synchronization problems, causing jobs to wait for responses from different tiers. Synchronization between different tiers is a complicating factor in the optimal control and analysis of the performance. In this paper, we study a generic multi-tier model with synchronization. The system is able to share processing capacity between arriving jobs that need to be sent to other tiers and the responses that have arrived after processing from these tiers. We provide structural results on the optimal resource allocation policy and provide a full characterization of the policy in the framework of Markov decision theory. We also highlight important effects of synchronization in the model and discuss their implications for practice. We validate our expressions through extensive experimentations for a wide range of resource configurations.

Keywords: Markov decision processes, multi-tier Internet services, optimal control, queueing networks, resource allocation, synchronization.

1 Introduction

The growing popularity of the Internet has driven the need for several businesses to open their business processes to their Web clients. Many businesses, such as retailers, auctioneers, and banks, offer significant portions of their services through the Internet. However, the computer systems that are engineered to host these applications are highly complex. Modern Internet applications have evolved from simple monolithic systems to complex multi-tiered systems. These applications do not simply deliver pre-written content but dynamically generate content on the fly using multi-tiered applications. Sometimes, these responses are generated by tens or hundreds of such multi-tiered applications. For instance, a single request to amazon.com (a major online retailer) is served by hundreds of applications operating in parallel [21]. Such elementary software applications, designed to be composed with each other, are commonly referred to as services.

As shown in Figure 1, a service generates a response to each request by executing some application-specific business logic, which in turn typically issues queries to its data store and/or

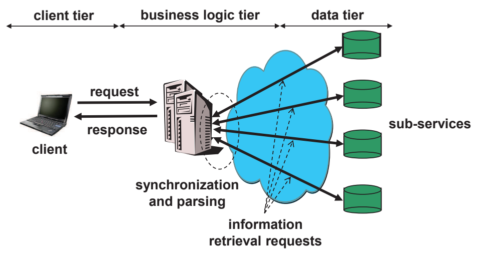

Figure 1: Application Model of an Internet Service.

requests to other services. A service is exposed through well-defined client interfaces accessible over the network. Examples of services include those for processing orders and maintaining shopping carts. For online services, providing a good client experience is one of the primary concerns. Human computer interaction studies have shown that users prefer response times of less than a second for most tasks, and that human productivity improves more than linearly as computer system response times fall in the sub-second range [17]. Hence, for such online services, the design of such systems that provide low response times to their clients is a crucial business requirement.

To provide good client response times, the administrators of these systems must have the ability to control its performance. In particular, one has control over the processing resources to handle jobs at different tiers. The optimal allocation of these processing resources is not trivial as different tiers are dependent on each other due to synchronization, i.e., a job cannot be further processed because it awaits answers from different tiers. For such systems, synchronization between different tiers is a complicating factor in the optimal control and analysis of the performance.

In the past few years, several groups have looked at modeling Internet applications. Most of them focus on modeling single-tier applications such as Web servers [13, 8, 10, 1] and databases [5]. However, very few have studied models for multi-tier applications, like those given in Figure 1, which are more commonplace in the Web. Applying simple single-tier models to multi-tiered systems leads to poor solutions as different tiers have different characteristics. Moreover, the single-tier models do not capture the dependence between different tiers due to synchronization.

In this paper we study a generic multi-tier model with synchronization in a methological framework. We provide structural results on the optimal resource allocation policy and provide a full characterization of the policy in the framework of Markov decision theory. A remarkable result is that the optimal allocation policy does not depend on the state of the sub-services in the system. We also highlight specific effects of synchronization in the model and discuss their implications for practice. We validate our expressions through extensive experimentations for a wide range of resource configurations. In contrast to most previous works on modeling Internet systems, we do not restrict ourselves to multi-tier systems with independent tiers.

The rest of the paper is organized as follows. Section 2 presents the related work and Section 3 presents the system model. Section 4 presents the structural results for the optimal resource allocation policy. Section 5 highlights important effects of synchronization in the model and discusses its implications for practice. Section 6 concludes the paper.

2 Related Work

2.1 Modeling Internet Systems

Various groups in the past have studied the problem of modeling Internet systems. Typical models include modeling Web servers, database servers, and application servers [13, 8, 5, 10]. For example, in [13], the authors use a queueing model for predicting the performance of Web servers by explicitly modeling the CPU, memory, and disk bandwidth in addition to using the distribution of file popularity. Bennani and Menasce [2] present an algorithm for allocating resources to a single-tiered application by using simple analytical models. Villela et al. [20] use an M/G/1/PS queueing model for business logic tiers and to provision the capacity of application servers. A G/G/1 based queueing model for modeling replicated Web servers is proposed in [19], which is to perform admission control during overloads. In contrast to these queueing based approaches, a feedback control based model was proposed in [1]. In this work, the authors demonstrate that by determining the right handles for sensors and actuators, the Web servers can provide stable performance and be resilient to unpredictable traffic. A novel approach to modeling and predicting the performance of a database is proposed in [5]. In this work, the authors employ machine learning and use an K-nearest neighbor algorithm to predict the performance of database servers during different workloads. However, the algorithm requires substantial input during the training period to perform effective prediction.

All the aforementioned research efforts have been applied only to single-tiered applications (Web servers, databases, or batch applications) and do not study complex synchronized multi-tiered applications which is the focus of this paper. Some recent works have focused on modeling multi-tier systems. In [10], the authors model a multi-tiered application as a single queue to predict the performance of a 3-tiered Web site. As mentioned, Urgaonkar et al. [18] model multi-tier applications as a network of queues and assume the request flows between queues to be independent. This assumption enables them to assume a product-form network so that they can apply a mean value analysis (MVA) to obtain the mean response time to process a request in the system. Although this approach can be very effective, MVA approaches can be limiting in nature as they do not allow us to get results for the synchronized system which is of crucial importance in large scale enterprises [21].

2.2 Performance Analysis

There are few works that study the performance of Internet systems in the context of multi-tier applications. Although the response time was made explicit for the first time in [12], a lot of research on response times has already been done in other systems. Results for the business logic, modeled as a processor sharing (PS) node, are given in [6], where the Laplace-Stieltjes Transform

(LST) is obtained for the M/M/1/PS node. In [14] an integral representation for this distribution is derived, and in [15] the distribution is derived for the more general M/G/1/PS system (in case a representation of the service times is given). The individual services, behind the business logic, are usually modeled as first-come-first-served (FCFS) queueing systems for which results are given in [7].

The first important results for calculating response times in a queueing network are given in [9], in which product-form networks are introduced. A multi-tier system modeled as a queueing network is of product-form when the following three conditions are met. First, the arrival process is a Poisson process and the arrival rate is independent of the number of requests in the network. Second, the duration of the services (behind the business logic) should be exponentially distributed, and is not allowed to depend on the number of requests present at that service. Finally, the sequence in which the services are visited is not allowed to depend on the state of the system except for the state of the node at which the request resides. Multi-tier systems that satisfy these properties fall within the class of so-called Jackson networks and have nice properties.

In [3], the authors give an overview of results on response times in queueing networks. In particular, they give expressions for the LST of the joint probability distribution for nodes which are traversed by requests in a product-form network according to a pre-defined path. Response times in a two-node network with feedback (such as at the business logic) were studied in [4]. The authors propose some solid approximations for the response times with iterative requests. They show that the approximations perform very well for the first moment of the response times. In [11], a single PS node is studied with several multi-server FCFS nodes. The authors derive exact results for the mean response time but do not allow for synchronization.

3 Problem Formulation

Consider a service system at which jobs arrive according to a Poisson process with rate λ. Upon arrival the job receives service, which has an exponentially distributed duration with parameter μ_{a}, from a server that works in a PS manner. After finishing its service, the job forks into n subjobs, and each subjob is sent to a backend server for further processing. The backend servers are modeled as single-server FIFO queues for which the service duration of backend server i is exponentially distributed with parameter μ_i for $i = 1, \ldots, n$. We assume that after *all* subjobs of the same job have finished their service, they join back to one job; we call this synchronization. Note that jobs that have not synchronized yet, do not block the backend servers from processing other jobs, neither do they use any processing capacity in the system. The unsynchronized jobs are stored in a temporary infinitely-sized buffer until synchronization occurs. After synchronization the job is then sent back to the first PS node for its final service, which has an exponentially distributed duration with parameter μ_{d}. The system is depicted in Figure 2. The load of the PS node is defined as $\rho_{\mathrm{ps}} = \lambda(\frac{1}{\mu_{\mathrm{a}}} + \frac{1}{\mu_{\mathrm{d}}})$. For backend server i the load is defined as $\rho_{\mathrm{i}} = \frac{\lambda}{\mu_{\mathrm{i}}}$. For the backend area the load is defined as the maximum load of all backend servers, thus $\rho_{\mathrm{sync}} = \max\{\rho_1, ..., \rho_{\mathrm{n}}\}$.

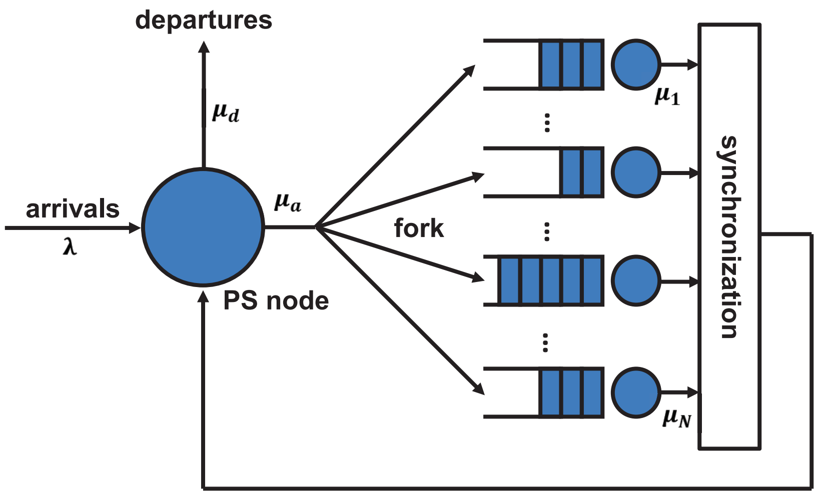

Figure 2: Model of the system

In order to describe the system mathematically, we define the state space of the system as $\mathcal{S} = \mathbb{N}_0^{n+2} = \{0, 1, \ldots\}^{n+2}$. An element $\vec{s} \in \mathcal{S}$ is given by $\vec{s} = (\vec{x}, \vec{y}) = (x_\mathrm{a}, x_\mathrm{d}, y_1, \ldots, y_n)$, where x_a and x_d represent the number of jobs in the PS node for arrival and departure, respectively, and y_i represents the number of subjobs at backend node i for $i = 1, \ldots, n$. Note that the backend node with the highest number of subjobs determines the number of forked jobs that have not synchronized yet, i.e., $\max\{\vec{y}\} = \max\{y_1, \ldots, y_n\}$ is the number of jobs that have forked and are being processed at the backend nodes but have not synchronized yet (synchronized jobs are counted in x_d). With this description we have full information on all jobs and subjobs in the system.

Based on the information $\vec{s}$ a controller can control the behavior of the PS node. Assuming that the capacity of the PS node is set to 1, the controller can assign a fraction $a \in \mathcal{A} = [0, 1]$ of the capacity to the x_a arriving jobs and a fraction of $1 - a$ to the x_d departing jobs. When $a = x_\mathrm{a}/(x_\mathrm{a} + x_\mathrm{d})$, then the PS node shares the capacity equally among all jobs, and thus is a pure processor-sharing node. However, when the control parameter a is chosen statically we get the discriminatory processor-sharing regime. In our setting the control parameter is allowed to be dynamically set and dependent not only on the jobs in the PS node, but also on the jobs in the backend nodes. Since we want to minimize response time [17] of the multi-tier application our goal is to minimize the expected sojourn time of jobs. Minimizing the expected sojourn time corresponds to minimizing the long-run average number of jobs in the system and therefore the aim of the controller is to minimize $x_\mathrm{a} + x_\mathrm{d} + \max\{\vec{y}\}$.

With this description, we can fully describe the control problem as a Markov decision problem. To this end we uniformize the system (see Section 11.5 of Puterman [16]) by defining $\gamma = \lambda + \max\{\mu_\mathrm{a}, \mu_\mathrm{d}\} + \sum_{i=1}^{n} \mu_i$. Without loss of generality, we assume that $\gamma = 1$, since we can always get this by scaling. Uniformizing is equivalent to adding dummy transitions (from a state to itself) such that the rate out of each state is equal to 1; then we can consider the rates to be transition probabilities. Let $\vec{1}$ denote the vector with a 1 at all entries. Then, the transition probabilities p, are given by $p\big((\vec{x},\vec{y}),a,(x_\mathrm{a}+1,x_\mathrm{d},\vec{y})\big)=\lambda$ for arrivals, $p\big((\vec{x},\vec{y}),a,(x_\mathrm{a}-1,x_\mathrm{d},\vec{y}+\vec{1})\big)=a\mu_\mathrm{a}$ if $x_\mathrm{a}>0$ for the forking of serviced arrivals, $p\big((\vec{x},\vec{y}),a,(x_\mathrm{a},x_\mathrm{d}+\max\{\vec{y}\}-\max\{\vec{y}-e_i\},\vec{y}-e_i)\big)=\mu_i$ for a service completion at backend node i for $i=1,\ldots,n$ (note that $\max\{\vec{y}\}-\max\{\vec{y}-e_i\}$ checks if synchronization has taken place), and $p\big((\vec{x},\vec{y}),a,(x_\mathrm{a},x_\mathrm{d}-1,\vec{y})\big)=(1-a)x_\mathrm{d}$ for the departures. Note that due to uniformization we have $p\big((\vec{x},\vec{y}),a,(\vec{x},\vec{y})\big)=1-\lambda-a\mu_\mathrm{a}-(1-a)\mu_\mathrm{d}-\sum_{i=1}^{n}\mu_i$. Define the cost function c as $c(\vec{x},\vec{y})=x_\mathrm{a}+x_\mathrm{d}+\max\{\vec{y}\}$, i.e., the number of jobs in the system. Then the tuple $(\mathcal{S},\mathcal{A},p,c)$ fully specifies the average-cost Markov decision problem.

Define a deterministic policy π as a function from $\mathcal{S}$ to $\mathcal{A}$, i.e., $\pi(\vec{s})\in\mathcal{A}$ for all $\vec{s}\in\mathcal{S}$. Let $u_t^\pi(\vec{s})$ denote the total expected costs up to time t when the system starts in state $\vec{s}$ under policy π. Note that for any stable and work-conserving policy, the Markov chain satisfies the unichain condition, so that the average expected costs $g(\pi)=\lim_{t\to\infty}u_t^\pi(\vec{s})/t$ is independent of the initial state $\vec{s}$ (see Proposition 8.2.1 of Puterman [16]). The goal is to find a policy π^* that minimizes the long-term average costs, thus $g=\min_\pi g(\pi)$. To determine this policy, we define $V(\vec{s})$ to be a real-valued function defined on the state space. This function will play the role of the relative value function, i.e., the asymptotic difference in total costs that results from starting the process in state $\vec{s}$ instead of some reference state. The long-term average optimal actions are a solution of the optimality equation (in vector notation) $g+V=TV$, where T is the dynamic programming operator acting on V defined as follows:

$$\begin{aligned}TV(\vec{x},\vec{y})&=\big(x_\mathrm{a}+x_\mathrm{d}+\max\{\vec{y}\}\big)+\lambda V(x_\mathrm{a}+1,x_\mathrm{d},\vec{y})\\&\quad+\min_{a\in[0,1]}\Big\{a\mu_\mathrm{a}\mathbb{1}_{\{x_\mathrm{a}>0\}}V(x_\mathrm{a}-1,x_\mathrm{d},\vec{y}+\vec{1})+a\mu_\mathrm{a}\mathbb{1}_{\{x_\mathrm{a}=0\}}V(x_\mathrm{a},x_\mathrm{d},\vec{y})\\&\quad+(1-a)\mu_\mathrm{d}V\big(x_\mathrm{a},(x_\mathrm{d}-1)^+,\vec{y}\big)+\big(1-\lambda-a\mu_\mathrm{a}-(1-a)\mu_\mathrm{d}-\sum_{i=1}^{n}\mu_i\big)V(x_\mathrm{a},x_\mathrm{d},\vec{y})\Big\}\\&\quad+\sum_{i=1}^{n}\mu_iV\big(x_\mathrm{a},x_\mathrm{d}+\max\{\vec{y}\}-\max\{(\vec{y}-e_i)^+\},(\vec{y}-e_i)^+\big).\end{aligned}$$

The first term in the expression $TV(x)$ models the direct costs that are occurred per time unit. The second term deals with the arriving jobs to the system. The terms between the brackets represent the control in which the tradeoff between services of newly arrived jobs and the departing jobs is made (the fourth term between the brackets is the uniformization constant). The last term models the departures of the subjobs in the backend nodes. The optimality equation $g+V=TV$ is usually hard to solve analytically in practice. Alternatively, the optimal actions can also be obtained by recursively defining $V_{l+1}=TV_l$ for arbitrary V_0. For $l\to\infty$, the maximizing actions converge to the optimal ones (for existence and convergence of solutions and optimal policies we refer to Chapter 8 of Puterman [16]). This procedure is also called value iteration. We shall use this in Section 5 for the illustration of our numerical experiments.

4 Structural Results

In the previous section, we described the model and a solution technique to obtain the optimal policy. However, the optimal policy also possesses structural properties, which give insight into the system behaviour. In fact, in this section, we fully characterize the optimal policy as a function of the state. In order to derive these results, we reformulate the dynamic programming backward recursion equations as follows.

$$\begin{aligned}V_{n+1}(\vec{x},\vec{y}) = \big(x_\mathrm{a} + x_\mathrm{d} + \max\{\vec{y}\}\big) + \lambda V_n(x_\mathrm{a}+1, x_\mathrm{d}, \vec{y}) + \min_{a\in[0,1]} T_n^a(\vec{x},\vec{y}) \\ + \sum_{i=1}^{n} \mu_i V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}, (\vec{y}-e_i)^+\big),\end{aligned}$$

with

$$\begin{aligned}T_n^a(\vec{x},\vec{y}) &= a\mu_\mathrm{a}\mathbb{1}_{\{x_\mathrm{a}>0\}} V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) + a\mu_\mathrm{a}\mathbb{1}_{\{x_\mathrm{a}=0\}} V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \\ &\quad + (1-a)\mu_\mathrm{d} V_n(x_\mathrm{a}, (x_\mathrm{d}-1)^+, \vec{y}) + \big(1 - \lambda - a\mu_\mathrm{a} - (1-a)\mu_\mathrm{d} - \sum_{i=1}^{n}\mu_i\big) V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}).\end{aligned}$$

We first turn our attention to the stage a job find itself in. It is intuitive to conjecture that the further a job is in its processing, the better the system is. The stages of a job can be split up into four parts (from last stage to first stage); last service by the processor-sharing node, service by the backend nodes, first service by the processor-sharing node, and arrival of a new job. In order to show that the first in the list is the best state of the system and the last in the list is the least profitable, we need to show that for arbitrary $(\vec{x},\vec{y}) \in \mathcal{S}$ we have the following properties.

1. $V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) > V(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})$ for $x_\mathrm{d} > 0$,
2. $V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \geq V\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}, (\vec{y}-e_i)^+\big)$ for $i = 1, \ldots, N$,
3. $V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \geq V(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})$ for $x_\mathrm{a} > 0$,
4. $V(x_\mathrm{a}+1, x_\mathrm{d}, \vec{y}) > V(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$.

Note that combining the four properties above is the formalized statement of 'the further a job is in its processing, the better the state of the system is'. In the next four lemmas, we prove the properties that are listed above. We first start with the departure of a job in the last stage: the last service by the processor-sharing node.

Lemma 4.1: $V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) > V(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})$ for all $(\vec{x},\vec{y}) \in \mathcal{S}$ with $x_\mathrm{d} > 0$.

Proof: The proof is by induction on n in V_n. Define $V_0(\vec{x},\vec{y}) = x_\mathrm{a} + x_\mathrm{d} + \max\{\vec{y}\}$. Then, clearly the statement holds. Now, assume that $V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) > V_n(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})$ for some $n \in \mathbb{N}$. Now, we prove that $V_{n+1}(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$ satisfies the increasingness property as well. Then, for $(\vec{x},\vec{y}) \in \mathcal{S}$ with

$x_\mathrm{d} > 0$

$$\begin{aligned}
V_{n+1}(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_{n+1}(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y}) &= 1 + \lambda\big[V_n(x_\mathrm{a} + 1, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a} + 1, x_\mathrm{d} - 1, \vec{y})\big] \\
&\quad + \min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y}) \\
&\quad + \sum_{i=1}^{n} \mu_i \big[V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y} - e_i)^+\}, (\vec{y} - e_i)^+\big) \\
&\quad - V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y} - e_i)^+\} - 1, (\vec{y} - e_i)^+\big)\big] \\
&\quad > \min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y}).
\end{aligned}$$

The equality follows by expanding V_{n+1} into V_n. The inequality follows by using the induction hypothesis. Let $a^* \in \arg\min_{a\in[0,1]} T_n^a V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$. Then, for $x_\mathrm{a} > 0$

$$\begin{aligned}
\min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y}) &\geq T_n^{a^*}(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - T_n^{a^*}(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y}) \\
&= a^* \mu_a \big[V_n(x_\mathrm{a} - 1, x_\mathrm{d}, \vec{y} + \vec{1}) - V_n(x_\mathrm{a} - 1, x_\mathrm{d} - 1, \vec{y} + \vec{1})\big] \\
&\quad + (1 - a^*)\mu_d \big[V_n(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y}) - V_n\big(x_\mathrm{a}, (x_\mathrm{d} - 2)^+, \vec{y}\big)\big] \\
&\quad + \big(1 - \lambda - a^*\mu_a - (1 - a^*)\mu_d - \sum_{i=1}^{n} \mu_i\big)\big[V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y})\big] \\
&\geq 0.
\end{aligned}$$

The first inequality follows from taking a potentially suboptimal action in $\min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y})$. The equality follows from the definition of T_n^a. The second inequality follows from the induction hypothesis. When $x_\mathrm{a} = 0$, we get $a^*\mu_a\big[V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d} - 1, \vec{y})\big]$ for which the induction hypothesis applies as well. □

The lemma shows that a departure of the system leads to a better state of the system. This is quite intuitive since we have a job fewer in the system. We now turn our attention to the services at the backend nodes. Here, the number of jobs in the system remains the same, however, a job is closer to departure from the system. The next lemma shows that this leads to a better state as well.

Lemma 4.2: $V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \geq V\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y} - e_i)^+\}, (\vec{y} - e_i)^+\big)$ for all $(\vec{x}, \vec{y}) \in \mathcal{S}$ and $i = 1, \ldots, N$.

Proof: The proof is by induction on n in V_n. Define $V_0(\vec{x}, \vec{y}) = 0$. Then, clearly the statement holds. Now, assume that $V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \geq V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y} - e_i)^+\}, (\vec{y} - e_i)^+\big)$ for all i for some $n \in \mathbb{N}$. Now, we prove that $V_{n+1}(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$ satisfies the property as well. Let i such that $y_i > 0$, then

$$\begin{aligned}
&V_{n+1}(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_{n+1}(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i) \\
&\quad= \lambda\big[V_n(x_\mathrm{a}+1, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}+1, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i)\big] \\
&\quad+ \min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i) \\
&\quad+ \sum_{j=1}^n \mu_j \big[V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y} - e_j)^+\}, (\vec{y} - e_j)^+\big) \\
&\quad- V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y} - e_i\} - \max\{(\vec{y} - e_i - e_j)^+\}, (\vec{y} - e_i - e_j)^+\big)\big] \\
&\quad\geq \min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i).
\end{aligned}$$

The equality follows from expanding V_{n+1} into V_n. The inequality follows by using the induction hypothesis. Let $a^* \in \arg\min_{a\in[0,1]} T_n^a V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$. Then, for $x_\mathrm{a} > 0$

$$\begin{aligned}
&\min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i) \\
&\quad\geq T_n^{a^*}(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - T_n^{b^*}(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i) \\
&\quad= a^*\mu_a\big[V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) - V_n(x_\mathrm{a}-1, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i + \vec{1})\big] \\
&\quad+ (1-a^*)\mu_d\big[V_n\big(x_\mathrm{a}, (x_\mathrm{d}-1)^+, \vec{y}\big) - V_n\big(x_\mathrm{a}, (x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\} - 1)^+, \vec{y} - e_i\big)\big] \\
&\quad+ \big(1 - \lambda - a^*\mu_a - (1-a^*)\mu_d - \sum_{i=1}^n \mu_i\big)\big[V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \\
&\quad- V_n(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i)\big].
\end{aligned}$$

The first inequality follows from taking a potentially suboptimal action in $\min_{b\in[0,1]} T_n^b(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y}-e_i\}, \vec{y}-e_i)$. The equality follows from the definition of T_n^a. The second inequality follows from the induction hypothesis. Note that we used that $V_n(x_\mathrm{a} - 1, x_\mathrm{d}, \vec{y} + \vec{1}) \geq V_n(x_\mathrm{a} - 1, x_\mathrm{d} + \max\{\vec{y}+\vec{1}\} - \max\{\vec{y}+\vec{1}-e_i\}, \vec{y}-e_i+\vec{1}) = V_n(x_\mathrm{a}-1, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y}-e_i\}, \vec{y}-e_i+\vec{1})$. When $x_\mathrm{a} = 0$, we get $a^*\mu_a\big[V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{\vec{y} - e_i\}, \vec{y} - e_i)\big]$ for which the induction hypothesis directly applies. □

The proof of the lemma is somewhat more complicated, because of the max operator to count the number of jobs finished in the backend nodes. Note that the number of queues N does not play an essential role in the proof. A similar remark holds when a job is finished processing the first time at the processor-sharing node. The following lemma formalizes this.

Lemma 4.3: $V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \geq V(x_\mathrm{a} - 1, x_\mathrm{d}, \vec{y} + \vec{1})$ for all $(\vec{x}, \vec{y}) \in \mathcal{S}$ with $x_\mathrm{a} > 0$.

Proof: The proof is by induction on n in V_n. Define $V_0(\vec{x}, \vec{y}) = 0$. Then, clearly the statement holds. Now, assume that $V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \geq V_n(x_\mathrm{a} - 1, x_\mathrm{d}, \vec{y} + \vec{1})$ for some $n \in \mathbb{N}$. Now, we prove that

$V_{n+1}(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$ satisfies the property as well. Then, for $(\vec{x}, \vec{y}) \in \mathcal{S}$ with $x_\mathrm{a} > 0$

$$\begin{aligned}
V_{n+1}(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_{n+1}(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) &= \lambda\big[V_n(x_\mathrm{a}+1, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}+\vec{1})\big] \\
&+ \min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) \\
&+ \sum_{i=1}^n \mu_i \big[V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}, (\vec{y}-e_i)^+\big) \\
&- V_n\big(x_\mathrm{a}-1, x_\mathrm{d} + \max\{\vec{y}+\vec{1}\} - \max\{\vec{y}+\vec{1}-e_i\}, \vec{y}+\vec{1}-e_i\big)\big] \\
&\geq \min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) \\
&+ \sum_{i=1}^n \mu_i \big[V_n\big(x_\mathrm{a}, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}, (\vec{y}-e_i)^+\big) \\
&- V_n\big(x_\mathrm{a}-1, x_\mathrm{d} + \max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}, \vec{y}+\vec{1}-e_i\big)\big] \\
&\geq \min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}).
\end{aligned}$$

The equality follows by expanding V_{n+1} into V_n. The first inequality follows by using the induction hypothesis on the terms with the arrivals. Note that we also have used the fact that $\max\{\vec{y}+\vec{1}\} - \max\{\vec{y}+\vec{1}-e_i\}$ is equal to $\max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}$ for all i. The second inequality then follows by the induction hypothesis again. Let $a^* \in \arg\min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$. Then, for $x_\mathrm{a} > 1$

$$\begin{aligned}
\min_{a\in[0,1]} T_n^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) &\geq T_n^{a^*}(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - T_n^{a^*}(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) \\
&= a^*\mu_a\big[V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) - V_n(x_\mathrm{a}-2, x_\mathrm{d}, \vec{y}+\vec{2})\big] \\
&+ (1-a^*)\mu_d\big[V_n\big(x_\mathrm{a}, (x_\mathrm{d}-1)^+, \vec{y}\big) - V_n\big(x_\mathrm{a}-1, (x_\mathrm{d}-1)^+, \vec{y}+\vec{1}\big)\big] \\
&+ \big(1 - \lambda - a^*\mu_a - (1-a^*)\mu_d - \sum_{i=1}^n \mu_i\big)\big[V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})\big] \\
&\geq 0.
\end{aligned}$$

The first inequality follows from taking a potentially suboptimal action in $\min_{b\in[0,1]} T_n^b(x_\mathrm{a} - 1, x_\mathrm{d}, \vec{y}+\vec{1})$. The equality follows from the definition of T_n^a. The second inequality follows from the induction hypothesis. When $x_\mathrm{a} = 1$, we get $a^*\mu_a\big[V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) - V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})\big] = 0$. $\square$

The lemma shows that when a job forks after finishing its processing at the first time it visits the processor-sharing node, the state of the system improves. It remains to show that an arrival of a job leads to a less good state of the system. The following lemma formalizes this statement.

Lemma 4.4: $V(x_\mathrm{a}+1, x_\mathrm{d}, \vec{y}) > V(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$ for all $(\vec{x}, \vec{y}) \in \mathcal{S}$.

Proof: The proof is by induction on n in V_n. Define $V_0(\vec{x}, \vec{y}) = x_\mathrm{a} + x_\mathrm{d} + \max\{\vec{y}\}$. Then, clearly the statement holds. Now, assume that $V_n(x_\mathrm{a}+1, x_\mathrm{d}, \vec{y}) \geq V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$ for some $n \in \mathbb{N}$. Now, we prove that $V_{n+1}(\vec{x}, \vec{y})$ satisfies the increasingness property as well. Then,

$$\begin{aligned}
V_{n+1}(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y}) - V_{n+1}(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}) &= 1 + \lambda\big[V_n(x_{\mathrm{a}}+2, x_{\mathrm{d}}, \vec{y}) - V_n(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y})\big] \\
&+ \min_{a\in[0,1]} T_n^a(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}) \\
&+ \sum_{i=1}^{n} \mu_i \big[V_n\big(x_{\mathrm{a}}+1, x_{\mathrm{d}} + \max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}, (\vec{y}-e_i)^+\big) \\
&- V_n\big(x_{\mathrm{a}}, x_{\mathrm{d}} + \max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}, (\vec{y}-e_i)^+\big)\big] \\
&> \min_{a\in[0,1]} T_n^a(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}).
\end{aligned}$$

The equality follows by expanding V_{n+1} into V_n. The inequality follows by using the induction hypothesis. Let $a^* \in \arg\min_{a\in[0,1]} T_n^a(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y})$. Then, for $x_{\mathrm{a}} > 0$

$$\begin{aligned}
\min_{a\in[0,1]} T_n^a(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y}) - \min_{b\in[0,1]} T_n^b(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}) &\geq T_n^{a^*}(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y}) - T_n^{a^*}(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}) \\
&= a^*\mu_a\big[V_n(\vec{x}, \vec{y}+\vec{1}) - V_n(x_{\mathrm{a}}-1, x_{\mathrm{d}}, \vec{y}+\vec{1})\big] \\
&+ (1-a^*)\mu_d\big[V_n\big(x_{\mathrm{a}}+1, (x_{\mathrm{d}}-1)^+, \vec{y}\big) - V_n\big(x_{\mathrm{a}}, (x_{\mathrm{d}}-1)^+, \vec{y}\big)\big] \\
&+ \big(1 - \lambda - a^*\mu_a - (1-a^*)\mu_d - \sum_{i=1}^{n}\mu_i\big)\big[V_n(x_{\mathrm{a}}+1, x_{\mathrm{d}}, \vec{y}) - V_n(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y})\big] \\
&\geq 0.
\end{aligned}$$

The first inequality follows from taking a potentially suboptimal action in $\min_{b\in[0,1]} T_n^b(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y})$. The equality follows from the definition of T_n^a. The second inequality follows from the induction hypothesis. When $x_{\mathrm{a}} = 0$, we get $a^*\mu_a\big[V_n(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}+\vec{1}) - V_n(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y})\big]$ as first term in the equality. By applying Lemma 4.2 to $V_n(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}+\vec{1})$ for all i we derive that $V_n(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}+\vec{1}) \geq V_n(x_{\mathrm{a}}, x_{\mathrm{d}}+1, \vec{y})$. By applying Lemma 4.1 subsequently, we have $V_n(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}+\vec{1}) \geq V_n(x_{\mathrm{a}}, x_{\mathrm{d}}+1, \vec{y}) > V_n(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y})$, which finishes the proof. □

Lemmas 4.1–4.4 show that it is better to have a job further in its processing stages. This observation also suggests that the resources should be allocated such that the better stages are reached as quickly as possible. Before we turn our attention to this question, we first observe that the optimal policy allocates *all* resources to either the incoming or the outgoing jobs at the processor node.

Lemma 4.5: The optimal policy assigns all resources to either the incoming or the outgoing jobs, i.e., $\arg\min_{a\in[0,1]} T^a(x_{\mathrm{a}}, x_{\mathrm{d}}, \vec{y}) \in \{0, 1\}$ for all $(\vec{x}, \vec{y}) \in \mathcal{S}$.

Proof: Note that

$$\min_{a\in[0,1]} T^a(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) = \min_{a\in[0,1]} \Big\{ a\mu_\mathrm{a} \mathbb{1}_{\{x_\mathrm{a}>0\}} V(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) + a\mu_\mathrm{a} \mathbb{1}_{\{x_\mathrm{a}=0\}} V(x_\mathrm{a}, x_\mathrm{d}, \vec{y})$$

$$+ (1-a)\mu_\mathrm{d} V\big(x_\mathrm{a}, (x_\mathrm{d}-1)^+, \vec{y}\big) + \big(1-\lambda - a\mu_\mathrm{a} - (1-a)\mu_\mathrm{d} - \sum_{i=1}^{n} \mu_i\big) V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \Big\}$$

$$= \min_{a\in[0,1]} \Big\{ a\big[\mu_\mathrm{a}\big(\mathbb{1}_{\{x_\mathrm{a}>0\}} V(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) + \mathbb{1}_{\{x_\mathrm{a}=0\}} V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V(x_\mathrm{a}, x_\mathrm{d}, \vec{y})\big)$$

$$+ \mu_\mathrm{d}\big(V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V\big(x_\mathrm{a}, (x_\mathrm{d}-1)^+, \vec{y}\big)\big)\big] \Big\}.$$

Now it is clear that the sign of the terms between the square brackets determines if $a^* = 0$ or $a^* = 1$. In particular, if $x_\mathrm{a} > 0$ and $x_\mathrm{d} = 0$, we have that $V(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1}) - V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) \le 0$ (see Lemma 4.3). Hence, we have $a^* = 1$. On the other, if $x_\mathrm{a} = 0$ and $x_\mathrm{d} > 0$, we have that $V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y}) > 0$ (see Lemma 4.1). Hence, $a^* = 0$. □

We are now ready to prove our main theorem based on the insights obtained so far. We have already seen that based on Lemmas 4.1–4.4, the further a job is in the stage of service in the system, the better the state of the system is. Hence, based on these insights, one could conjecture that assigning all resources to the job furthest in the system might be optimal. The next theorem shows that this is indeed true and fully characterizes the optimal policy.

Theorem 4.1: The optimal policy in state $(\vec{x}, \vec{y}) \in \mathcal{S}$ is given by $a^* = \mathbb{1}_{\{x_\mathrm{d}=0\}}$, i.e., assign all resources to the outgoing jobs if any ($a^* = 0$ when $x_\mathrm{d} > 0$), and assign all resources to arriving jobs otherwise ($a^* = 1$ when $x_\mathrm{d} = 0$).

Proof: In the proof of Lemma 4.5, we have already seen that if $x_\mathrm{a} > 0$ and $x_\mathrm{d} = 0$, we have that $a^* = 1$. In addition, it was also shown that if $x_\mathrm{a} = 0$ and $x_\mathrm{d} > 0$, we have that $a^* = 0$. Clearly, when $x_\mathrm{a} = 0$ and $x_\mathrm{d} = 0$, then $a^* = 1$ is optimal (as is any other value for a^*). Hence, we need to show that for $x_\mathrm{a} > 0$ and $x_\mathrm{d} > 0$ we have $a^* = 0$, which translates to (cf. see the proof of Lemma 4.5) showing that

$$\mu_\mathrm{d}\big[V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})\big] > \mu_\mathrm{a}\big[V(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})\big]$$

for all $(\vec{x}, \vec{y}) \in \mathcal{S}$ with $x_\mathrm{a} > 0$ and $x_\mathrm{d} > 0$. The proof is by induction on n in V_n. Define $V_0(\vec{x}, \vec{y}) = 0$. Then, clearly the statement holds. Now, assume that the statement holds for some $n \in \mathbb{N}$. Now, we prove that the statement also holds for $n+1$. Thus,

$$
\begin{aligned}
&\mu_{\mathrm{d}}\big[V_{n+1}(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - V_{n+1}(x_{\mathrm{a}},x_{\mathrm{d}}-1,\vec{y})\big] - \mu_{\mathrm{a}}\big[V_{n+1}(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - V_{n+1}(x_{\mathrm{a}}-1,x_{\mathrm{d}},\vec{y}+\vec{1})\big] \\
&\quad= \mu_{\mathrm{d}}\big[(x_{\mathrm{a}}+x_{\mathrm{d}}+\max\{\vec{y}\}) - (x_{\mathrm{a}}+x_{\mathrm{d}}+1+\max\{\vec{y}\})\big] \\
&\quad+ \mu_{\mathrm{d}}\lambda\big[V_n(x_{\mathrm{a}}+1,x_{\mathrm{d}},\vec{y}) - V_n(x_{\mathrm{a}}+1,x_{\mathrm{d}}-1,\vec{y})\big] + \mu_{\mathrm{d}}\big[T_n^*(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - T_n^*(x_{\mathrm{a}},x_{\mathrm{d}}-1,\vec{y})\big] \\
&\quad+ \mu_{\mathrm{d}}\sum_{i=1}^{n}\mu_i\big[V_n\big(x_{\mathrm{a}},x_{\mathrm{d}}+\max\{\vec{y}\}-\max\{(\vec{y}-e_i)^+\},(\vec{y}-e_i)^+\big) \\
&\quad- V_n\big(x_{\mathrm{a}},x_{\mathrm{d}}+\max\{\vec{y}\}-\max\{(\vec{y}-e_i)^+\}-1,(\vec{y}-e_i)^+\big)\big] \\
&\quad- \mu_{\mathrm{a}}\big[(x_{\mathrm{a}}+x_{\mathrm{d}}+\max\{\vec{y}\}) - (x_{\mathrm{a}}-1+x_{\mathrm{d}}+\max\{\vec{y}+\vec{1}\})\big] \\
&\quad- \mu_{\mathrm{a}}\lambda\big[V_n(x_{\mathrm{a}}+1,x_{\mathrm{d}},\vec{y}) - V_n(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}+\vec{1})\big] - \mu_{\mathrm{a}}\big[T_n^*(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - T_n^*(x_{\mathrm{a}}-1,x_{\mathrm{d}},\vec{y}+\vec{1})\big] \\
&\quad- \mu_{\mathrm{a}}\sum_{i=1}^{n}\mu_i\big[V_n\big(x_{\mathrm{a}},x_{\mathrm{d}}+\max\{\vec{y}\}-\max\{(\vec{y}-e_i)^+\},(\vec{y}-e_i)^+\big) \\
&\quad- V_n(x_{\mathrm{a}}-1,x_{\mathrm{d}}+\max\{\vec{y}+1\}-\max\{\vec{y}+\vec{1}-e_i\},\vec{y}+\vec{1}-e_i)\big].
\end{aligned}
$$

The equality follows from expanding V_{n+1} into V_n. By rearranging terms, we find

$$
\begin{aligned}
&\mu_{\mathrm{d}}\big[V_{n+1}(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - V_{n+1}(x_{\mathrm{a}},x_{\mathrm{d}}-1,\vec{y})\big] - \mu_{\mathrm{a}}\big[V_{n+1}(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - V_{n+1}(x_{\mathrm{a}}-1,x_{\mathrm{d}},\vec{y}+\vec{1})\big] \\
&\quad= \mu_{\mathrm{d}} + \lambda\big[\mu_{\mathrm{d}}\big(V_n(x_{\mathrm{a}}+1,x_{\mathrm{d}},\vec{y}) - V_n(x_{\mathrm{a}}+1,x_{\mathrm{d}}-1,\vec{y})\big) \\
&\quad- \mu_{\mathrm{a}}\big(V_n(x_{\mathrm{a}}+1,x_{\mathrm{d}},\vec{y}) - V_n(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}+\vec{1})\big)\big] \\
&\quad+ \mu_{\mathrm{d}}\big[T_n^*(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - T_n^*(x_{\mathrm{a}},x_{\mathrm{d}}-1,\vec{y})\big] - \mu_{\mathrm{a}}\big[T_n^*(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - T_n^*(x_{\mathrm{a}}-1,x_{\mathrm{d}},\vec{y}+\vec{1})\big] \\
&\quad+ \sum_{i=1}^{n}\mu_i\big[\mu_{\mathrm{d}}\big(V_n\big(x_{\mathrm{a}},x_{\mathrm{d}}+\max\{\vec{y}\}-\max\{(\vec{y}-e_i)^+\},(\vec{y}-e_i)^+\big) \\
&\quad- V_n\big(x_{\mathrm{a}},x_{\mathrm{d}}+\max\{\vec{y}\}-\max\{(\vec{y}-e_i)^+\}-1,(\vec{y}-e_i)^+\big)\big) \\
&\quad- \mu_{\mathrm{a}}\big(V_n\big(x_{\mathrm{a}},x_{\mathrm{d}}+\max\{\vec{y}\}-\max\{(\vec{y}-e_i)^+\},(\vec{y}-e_i)^+\big) \\
&\quad- V_n(x_{\mathrm{a}}-1,x_{\mathrm{d}}+\max\{\vec{y}+\vec{1}\}-\max\{\vec{y}+\vec{1}-e_i\},\vec{y}+\vec{1}-e_i)\big)\big] \\
&\quad> \mu_{\mathrm{d}}\big[T_n^*(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - T_n^*(x_{\mathrm{a}},x_{\mathrm{d}}-1,\vec{y})\big] - \mu_{\mathrm{a}}\big[T_n^*(x_{\mathrm{a}},x_{\mathrm{d}},\vec{y}) - T_n^*(x_{\mathrm{a}}-1,x_{\mathrm{d}},\vec{y}+\vec{1})\big].
\end{aligned}
$$

The inequality follows by using the induction hypothesis. Note that we also have used the fact that $\max\{\vec{y}+\vec{1}\} - \max\{\vec{y}+\vec{1}-e_i\}$ is equal to $\max\{\vec{y}\} - \max\{(\vec{y}-e_i)^+\}$ for all i. First, consider the case $x_\mathrm{d} \geq 2$, then

$$\begin{aligned}
&\mu_\mathrm{d}\big[T_n^*(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - T_n^*(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})\big] - \mu_\mathrm{a}\big[T_n^*(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - T_n^*(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})\big] \\
&\quad= \mu_\mathrm{d}\mu_\mathrm{d}\big[V_n(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d}-2, \vec{y})\big] \\
&\quad- \mu_\mathrm{d}\mu_\mathrm{a}\big[V_n(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y}) - V_n(x_\mathrm{a}-1, x_\mathrm{d}-1, \vec{y}+\vec{1})\big] \\
&\quad+ \mu_\mathrm{d}\big(1-\lambda-\mu_\mathrm{d}-\sum_{i=1}^n \mu_i\big)\big[V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})\big] \\
&\quad- \mu_\mathrm{a}\big(1-\lambda-\mu_\mathrm{d}-\sum_{i=1}^n \mu_i\big)\big[V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})\big] \\
&\quad= \mu_\mathrm{d}\big[\mu_\mathrm{d}\big(V_n(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d}-2, \vec{y})\big) \\
&\quad- \mu_\mathrm{a}\big(V_n(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y}) - V_n(x_\mathrm{a}-1, x_\mathrm{d}-1, \vec{y}+\vec{1})\big)\big] \\
&\quad+ \big(1-\lambda-\mu_\mathrm{d}-\sum_{i=1}^n \mu_i\big)\big[\mu_\mathrm{d}\big(V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})\big) \\
&\quad- \mu_\mathrm{a}\big(V_n(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - V_n(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})\big)\big] \\
&\quad\geq 0.
\end{aligned}$$

The first equality follows by expanding $T_n^*(\vec{x}, \vec{y})$ by taking the optimal action $a^* = 0$ in the expansion. The second equality follows by rearranging the terms so that the induction hypothesis can be applied, which yields the inequality. Now, suppose that $x_\mathrm{d} = 1$. Recall that the uniformization constant γ was defined as $\gamma = \lambda + \mu_\mathrm{a} + \mu_\mathrm{d} + \sum_{i=1}^n \mu_i = 1$. Then,

$$\begin{aligned}
&\mu_\mathrm{d}\big[T_n^*(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - T_n^*(x_\mathrm{a}, x_\mathrm{d}-1, \vec{y})\big] - \mu_\mathrm{a}\big[T_n^*(x_\mathrm{a}, x_\mathrm{d}, \vec{y}) - T_n^*(x_\mathrm{a}-1, x_\mathrm{d}, \vec{y}+\vec{1})\big] \\
&\quad= \mu_\mathrm{d}\big[\mu_\mathrm{d} V_n(x_\mathrm{a}, 0, \vec{y}) + \big(1-\lambda-\mu_\mathrm{d}-\sum_{i=1}^n \mu_i\big) V_n(x_\mathrm{a}, 1, \vec{y})\big] \\
&\quad- \mu_\mathrm{d}\big[\mu_\mathrm{a} V_n(x_\mathrm{a}-1, 0, \vec{y}+\vec{1}) + \big(1-\lambda-\mu_\mathrm{a}-\sum_{i=1}^n \mu_i\big) V_n(x_\mathrm{a}, 0, \vec{y})\big] \\
&\quad- \mu_\mathrm{a}\big[\mu_\mathrm{a} V_n(x_\mathrm{a}-1, 1, \vec{y}+\vec{1}) + \big(1-\lambda-\mu_\mathrm{a}-\sum_{i=1}^n \mu_i\big) V_n(x_\mathrm{a}, 1, \vec{y})\big] \\
&\quad+ \mu_\mathrm{a}\big[\mu_\mathrm{d} V_n(x_\mathrm{a}-1, 0, \vec{y}+\vec{1}) + \big(1-\lambda-\mu_\mathrm{d}-\sum_{i=1}^n \mu_i\big) V_n(x_\mathrm{a}-1, 1, \vec{y}+\vec{1})\big] \\
&\quad= \mu_\mathrm{d}\mu_\mathrm{d} V_n(x_\mathrm{a}, 0, \vec{y}) + \mu_\mathrm{d}\mu_\mathrm{a} V_n(x_\mathrm{a}, 1, \vec{y}) \\
&\quad- \mu_\mathrm{d}\mu_\mathrm{a} V_n(x_\mathrm{a}-1, 0, \vec{y}+\vec{1}) - \mu_\mathrm{d}\mu_\mathrm{d} V_n(x_\mathrm{a}, 0, \vec{y}) \\
&\quad- \mu_\mathrm{a}\mu_\mathrm{a} V_n(x_\mathrm{a}-1, 1, \vec{y}+\vec{1}) - \mu_\mathrm{a}\mu_\mathrm{d} V_n(x_\mathrm{a}, 1, \vec{y}) \\
&\quad+ \mu_\mathrm{a}\mu_\mathrm{d} V_n(x_\mathrm{a}-1, 0, \vec{y}+\vec{1}) + \mu_\mathrm{a}\mu_\mathrm{a} V_n(x_\mathrm{a}-1, 1, \vec{y}+\vec{1}) \\
&\quad= 0.
\end{aligned}$$

The first equality follows from taking the optimal actions in the first, second and fourth term, and a suboptimal action in the third term. By rewriting all uniformization constants, we obtain the

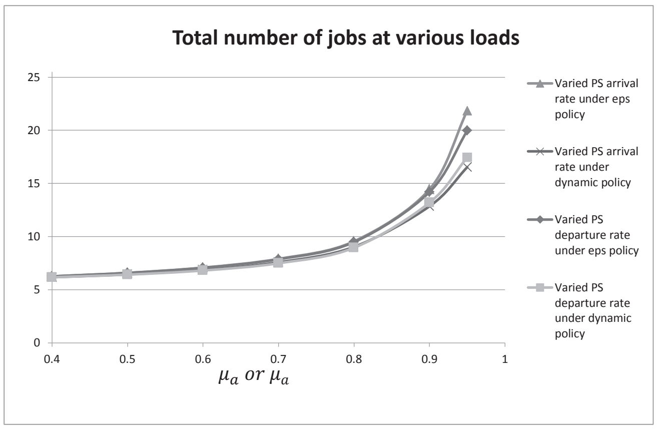

Figure 3: Expected total number of jobs for different values of ρ_{ps}.

second equality. □

The previous theorem shows that the policy has a remarkably simple structure: only serve departing jobs first, and if none are available serve the newly arrived jobs first. This policy is does not take the states of the backend nodes into account. This is quite surprising, because one could expect that jobs in the backend nodes provide information to the decision maker. However, the theorem clearly shows that this intuition does not hold.

5 Numerical experiments

In the previous section, we derived structural properties of the optimal policy. The main purpose of this section is to get insight in the performance gain of the optimal policy. We numerically validate the structural results obtained in Section 4 with a broad range of parameter settings. In the following experiments we study the model as described in Section 3 with $n = 2$, thus we investigate the system with two backend servers in the synchronizing area. We focus on the expected total number of jobs in the system (L_{tot}). For computational tractability, we truncate the state-space to 50 jobs for each dimension. This truncation at 50 jobs is high enough to obtain results that are accurate to 10^{-6}.

In our first experiment, we study the performance of the system under different workload utilizations. We do this, by changing the parameter μ_{a} of the PS-node while keeping μ_{d} fixed, and in a different experiment, we change μ_{d} while keeping μ_{a} fixed. For both the optimal dynamic policy and under the pure processor-sharing (pps) regime (i.e., the allocation fraction $a = x_{\text{a}}/(x_{\text{a}}+x_{\text{d}})$), we calculate the expected total number of jobs L_{tot} in the system. For this purpose, let $\lambda = 1$ and $\mu_1 = \mu_2 = \frac{5}{4}$, so that $\rho_{\text{sync}} = 0.8$ in all cases. Figure 3 shows the results for these experiments for load values ρ_{ps} that range from 0.1 to 0.95. The figure depicts four curves; the two lower curves are for the optimal policy and two upper curves are for the processor-sharing regime. For each policy, one curve depicts L_{tot} when μ_{a} is varied (note that μ_{d} is kept constant). For the other

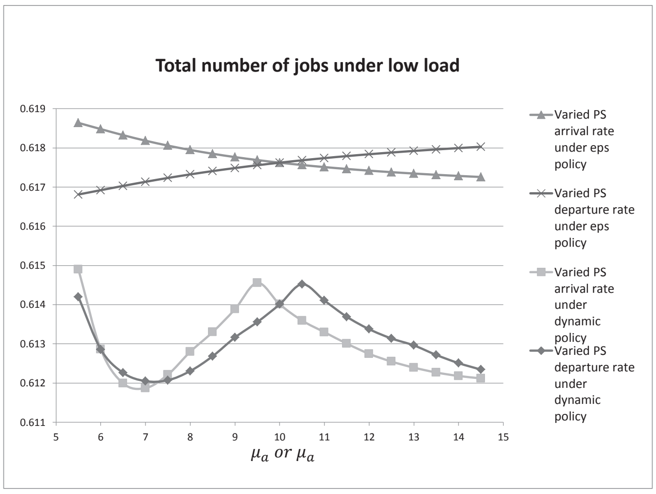

Figure 4: Expected total number of jobs, $\rho = 0.2$.

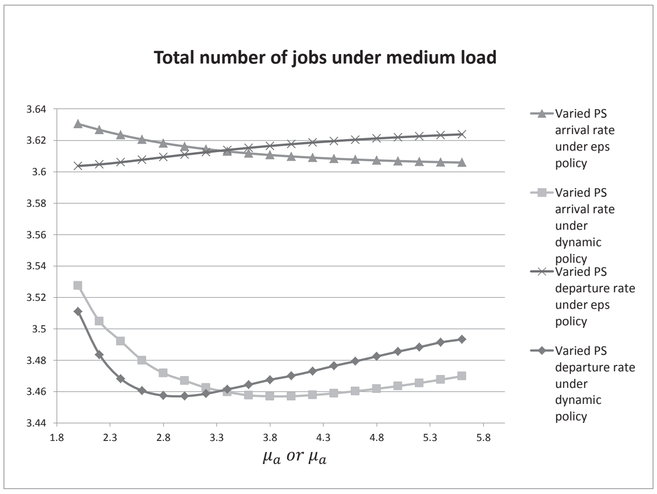

Figure 5: Expected total number of jobs, $\rho = 0.6$.

curve, the parameter μ_{d} is varied in a similar manner. The first observation is that the dynamic policy performs better in all cases, in particular, for higher load values. The second observation, is that the system performance seems to be asymmetric, i.e., the performance of the system for a fixed $(\mu_{\mathrm{a}}, \mu_{\mathrm{d}})$ is different than the performance for $(\mu_{\mathrm{d}}, \mu_{\mathrm{a}})$ even though the load is the same.

In order to understand the previous results more in-depth, we continue with a series of experiments in which we keep the load constant, thus we vary the parameters μ_{a} and μ_{d} such that the offered load to the system remains equal. We show that the asymmetric behavior becomes more apparent for both the optimal dynamic policy as well as the pure processor-sharing regime. The parameters for these experiments are $\lambda = 1$ and $\mu_1 = \mu_2 = \frac{10}{9}$, and done under low, medium, and high loads, i.e., $\rho = 0.2, 0.6$, and 0.9.

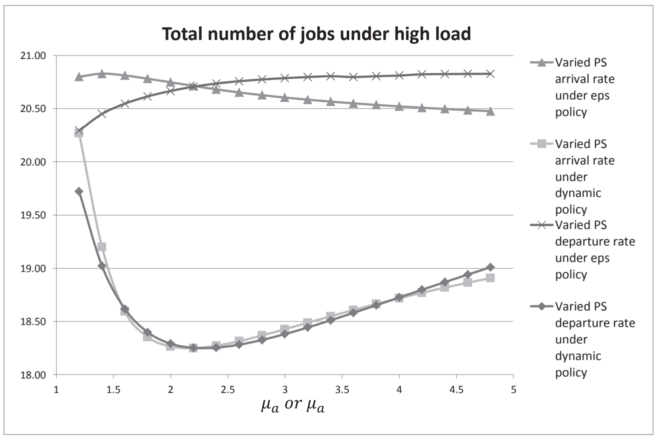

Figure 6: Expected total number of jobs, $\rho = 0.9$.

The results are shown in Figures 4–6. The figures depicts the four curves as in the previous graph; the two lower curves are for the optimal policy and two upper curves are for the processor-sharing regime. For each policy, one curve depicts L_{tot} when μ_{a} is varied (note that μ_{d} is then completely determined, since the load is kept constant). For the other curve, the parameter μ_{d} is varied in a similar manner. In case of symmetric behaviour, the two curves should be indistinguishable from each other. However, the results show that this indeed is not the case. Under the pure processor-sharing regime this is clear for all different values of the load. In case of the optimal policy, we see that the two curves become more similar as the load increases. Next, we observe that the dynamic policy has a better performance, as predicted by our theorem, than the pure processor-sharing regime, especially under high loads. Finally, we observe that the performance is not monotone under the dynamic policy at low loads. This can be seen by the 'W'-shape of L_{tot} in Figure 4. The effect diminishes as the load increases.

In order to gain more insight into the performance gains of the dynamic policy versus the pure processor-sharing regime, we calculate the gain explicitly for our experiments. Table 1 shows the percentage gain $\Delta = \big(L_{\text{tot}}(\text{PPS}) - L_{\text{tot}}(\text{dynamic})\big)/L_{\text{tot}}(PPS)$ for the three load cases. We can see that one can achieve gains of 1% for low loads and almost 12% for high loads. The results in the table also suggest that when $\mu_{\text{a}} << \mu_{\text{d}}$ or $\mu_{\text{d}} << \mu_{\text{a}}$, the pure processor-sharing regime will mimic the $0-1$ control of the optimal policy. Hence, the performance gains in this case as less than in other cases. We give an extremal case in which this effect becomes clear. Let $\lambda = 0.87$, $\mu_{\text{a}} = 10, \mu_{\text{d}} = 1.0$, and $\mu_1 = \mu_2 = 1.0$, yielding a very high load of $\rho = 0.957$. The optimal strategy gives $L_{\text{tot}} = 17.061$, where the pure processor-sharing regime gives $L_{\text{tot}} = 17.153$. This is merely a gain of only 0.54%, whereas one would expect much higher gains due to the high load.

$\rho = 0.2$				$\rho = 0.6$				$\rho = 0.9$			
$(\mu_a; \mu_d)$	L_{tot} pps	L_{tot} opt	Δ% gain	$(\mu_a; \mu_d)$	L_{tot} pps	L_{tot} opt	Δ% gain	$(\mu_a; \mu_d)$	L_{tot} pps	L_{tot} opt	Δ% gain
(55; 5.5)	0.6168	0.6142	0.4236	(10; 2)	3.6037	3.5110	2.5714	(1.2; 15)	20.7990	20.2703	2.5420
(30; 6)	0.6169	0.6129	0.6587	(6.875; 2.2)	3.6046	3.4835	3.3600	(1.4; 5.385)	20.8279	19.2007	7.8127
(21.667; 6.5)	0.6170	0.6123	0.7732	(5.455; 2.4)	3.6060	3.4682	3.8235	(1.6; 3.636)	20.8113	18.5939	10.6549
(17.5; 7)	0.6171	0.6120	0.8240	(4.643; 2.6)	3.6076	3.4605	4.0771	(1.8; 2.903)	20.7802	18.3537	11.6770
(15; 7.5)	0.6172	0.6121	0.8362	(4.118; 2.8)	3.6093	3.4574	4.2070	(2; 2.5)	20.7470	18.2655	11.9604
(13.333; 8)	0.6173	0.6123	0.8127	(3.75; 3)	3.6109	3.4571	4.2600	(2.2; 2.245)	20.7129	18.2496	11.8925
(12.143; 8.5)	0.6174	0.6127	0.7652	(3.478; 3.2)	3.6124	3.4586	4.2584	(2.4; 2.069)	20.6803	18.2719	11.6460
(11.25; 9)	0.6175	0.6132	0.7000	(3.269; 3.4)	3.6139	3.4614	4.2194	(2.6; 1.94)	20.6514	18.3148	11.3147
(10.556; 9.5)	0.6176	0.6136	0.6479	(3.103; 3.6)	3.6152	3.4643	4.1737	(2.8; 1.842)	20.6260	18.3685	10.9450
(10; 10)	0.6176	0.6140	0.5852	(2.969; 3.8)	3.6164	3.4675	4.1188	(3; 1.765)	20.6036	18.4275	10.5618
(9.545; 10.5)	0.6177	0.6145	0.5119	(2.857; 4)	3.6176	3.4700	4.0785	(3.2; 1.702)	20.5836	18.4882	10.1799
(9.167; 11)	0.6177	0.6141	0.5884	(2.763; 4.2)	3.6186	3.4730	4.0234	(3.4; 1.65)	20.5655	18.5485	9.8076
(8.846; 11.5)	0.6178	0.6137	0.6636	(2.683; 4.4)	3.6195	3.4764	3.9560	(3.6; 1.607)	20.5492	18.6074	9.4494
(8.571; 12)	0.6178	0.6134	0.7224	(2.614; 4.6)	3.6204	3.4793	3.8986	(3.8; 1.57)	20.5343	18.6641	9.1077
(8.333; 12.5)	0.6179	0.6131	0.7679	(2.553; 4.8)	3.6212	3.4824	3.8343	(4; 1.538)	20.5206	18.7182	8.7834
(8.125; 13)	0.6179	0.6130	0.8029	(2.5; 5)	3.6219	3.4854	3.7687	(4.2; 1.511)	20.5080	18.7696	8.4766
(7.941; 13.5)	0.6180	0.6127	0.8484	(2.453; 5.2)	3.6226	3.4883	3.7082	(4.4; 1.486)	20.4963	18.8183	8.1870
(7.778; 14)	0.6180	0.6125	0.8879	(2.411; 5.4)	3.6232	3.4913	3.6419	(4.6; 1.465)	20.4855	18.8643	7.9138
(7.632; 14.5)	0.6180	0.6123	0.9199	(2.373; 5.6)	3.6238	3.4933	3.6031	(4.8; 1.446)	20.4754	18.9078	7.6561
(5.5; 55)	0.6186	0.6149	0.6044	(2; 10)	3.6304	3.5274	2.8376	(15; 1.2)	20.2934	19.7228	2.8114
(6; 30)	0.6185	0.6129	0.9086	(2.2; 6.875)	3.6267	3.5047	3.3636	(5.385; 1.4)	20.4494	19.0214	6.9834
(6.5; 21.667)	0.6183	0.6120	1.0242	(2.4; 5.455)	3.6234	3.4921	3.6237	(3.636; 1.6)	20.5464	18.6179	9.3860
(7; 17.5)	0.6182	0.6119	1.0211	(2.6; 4.643)	3.6206	3.4799	3.8860	(2.903; 1.8)	20.6141	18.3986	10.7478
(7.5; 15)	0.6181	0.6122	0.9455	(2.8; 4.118)	3.6182	3.4717	4.0483	(2.5; 2)	20.6654	18.2915	11.4874
(8; 13.333)	0.6179	0.6128	0.8333	(3; 3.75)	3.6161	3.4669	4.1268	(2.245; 2.2)	20.7053	18.2521	11.8482
(8.5; 12.143)	0.6178	0.6133	0.7367	(3.2; 3.478)	3.6144	3.4624	4.2053	(2.069; 2.4)	20.7353	18.2541	11.9659
(9; 11.25)	0.6178	0.6139	0.6282	(3.4; 3.269)	3.6129	3.4597	4.2423	(1.94; 2.6)	20.7569	18.2820	11.9235
(9.5; 10.556)	0.6177	0.6146	0.5073	(3.6; 3.103)	3.6117	3.4576	4.2673	(1.842; 2.8)	20.7732	18.3263	11.7793
(10; 10)	0.6176	0.6140	0.5852	(3.8; 2.969)	3.6106	3.4570	4.2561	(1.765; 3)	20.7860	18.3814	11.5682
(10.5; 9.545)	0.6176	0.6136	0.6427	(4; 2.857)	3.6097	3.4570	4.2314	(1.702; 3.2)	20.7961	18.4436	11.3122
(11; 9.167)	0.6175	0.6133	0.6821	(4.2; 2.763)	3.6090	3.4578	4.1893	(1.65; 3.4)	20.8041	18.5108	11.0234
(11.5; 8.846)	0.6175	0.6130	0.7217	(4.4; 2.683)	3.6083	3.4588	4.1424	(1.607; 3.6)	20.7961	18.5809	10.6520
(12; 8.571)	0.6174	0.6127	0.7569	(4.6; 2.614)	3.6077	3.4602	4.0888	(1.57; 3.8)	20.8041	18.6527	10.3410
(12.5; 8.333)	0.6174	0.6125	0.7828	(4.8; 2.553)	3.6072	3.4618	4.0332	(1.538; 4)	20.8103	18.7253	10.0192
(13; 8.125)	0.6173	0.6124	0.8022	(5; 2.5)	3.6068	3.4635	3.9738	(1.511; 4.2)	20.8218	18.7980	9.7200
(13.5; 7.941)	0.6173	0.6123	0.8171	(5.2; 2.453)	3.6064	3.4654	3.9099	(1.486; 4.4)	20.8240	18.8699	9.3838
(14; 7.778)	0.6173	0.6122	0.8268	(5.4; 2.411)	3.6061	3.4676	3.8424	(1.465; 4.6)	20.8256	18.9407	9.0506
(14.5; 7.632)	0.6173	0.6121	0.8323	(5.6; 2.373)	3.6058	3.4698	3.7732	(1.446; 4.8)	20.8267	19.0097	8.7241

Table 1: Δ% gain for all parameters settings.

6 Conclusion

In this paper, we have explored the optimal dynamic policy for a generic multi-tier model with synchronization. We derived structural results on the optimal resource allocation policy and we fully characterized the policy in the framework of Markov decision theory. A remarkable conclusion is that for the optimal policy the information about the status of jobs in the backend nodes is not needed, only information about current jobs in the PS-node is of importance. We have furthermore shown that the gain of using the optimal policy compared to the standard pure processing-sharing regime is significant. Especially, in cases of high load utilization. Moreover, the optimal policy is easy to implement making it a very strong addition to traffic policing algorithms.

There are some interesting avenues for further research. First, in the paper it is assumed that the service times at each of the nodes are exponentially distributed. In practice, however, processing times in this type of settings are often far from exponential. Extension of the results toward phase-type service times is an interesting topic for further research. This becomes especially relevant when the system is subjected to jobs of multiple classes. Second, the assumptions of Poisson arrivals may be unrealistic, because the job arrival processes may be autocorrelated. It is a challenging topic for follow-up research to investigate the (near-)optimality of the control policies under non-Poisson arrival streams. Finally, our control policy does not depend on the information in the backend nodes. This is a remarkable result that significantly simplifies the structure of the optimal policy and makes in tractable enough for implementation. It remains to be investigated under which conditions such results hold in more general systems.

References

[1] Tarek F. Abdelzaher, Kang G. Shin, and Nina Bhatti. Performance guarantees for web server end-systems: A control-theoretical approach. *IEEE Trans. Parallel Distrib. Syst.*, 13(1):80–96, 2002.

[2] Mohamed N. Bennani and Daniel A. Menasce. Resource allocation for autonomic data centers using analytic performance models. In *ICAC '05: Proceedings of the Second International Conference on Automatic Computing*, pages 229–240, Washington, DC, USA, 2005. IEEE Computer Society.

[3] O.J. Boxma and H. Daduna. Sojourn times in queueing networks. *Stochastic Analysis of Computer and Communication Systems*, pages 401–450, 1990.

[4] O.J. Boxma, R.D. van der Mei, J.A.C. Resing, and K.M.C. van Wingerden. Sojourn time approximations in a two-node queueing network. In *Proceedings of the 19th International Teletraffic Congress - ITC 19*, pages 1121–1133, 2005.

[5] Jin Chen, Gokul Soundararajan, and Cristiana Amza. Autonomic provisioning of backend databases in dynamic content web servers. In *Proceedings of the 3rd IEEE International Conference on Autonomic Computing (ICAC 2006)*, 2006.

[6] E.G. Coffman, R.R. Muntz, and H. Trotter. Waiting time distributions for processor-sharing systems. *Journal of the ACM*, 17(1):123–130, 1970.

[7] R.B. Cooper. *Introduction to Queueing Theory.* North Holland, 1981.

[8] RP Doyle, JS Chase, OM Asad, W Jin, and A Vahdat. Web server software architectures. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, 2003.

[9] J.R. Jackson. Networks of waiting lines. *Operations Research*, 5:518–521, 1957.

[10] A. Kamra, V. Misra, and E. Nahum. Yaksha: A controller for managing the performance of 3-tiered websites. In *Proceedings of the 12th IWQoS*, 2004.

[11] R.D. van der Mei, B.M.M. Gijsen, P. Engelberts, J.L. van der Berg, and K.M.C. van Wingerden. Response times in queueing networks with feedback. *Performance Evaluation*, 64, 2006.

[12] R.D. van der Mei and H.B. Meeuwissen. Modelling end-to-end quality-of-service for transaction-based services in a multi-domain environment. In *Proceedings IEEE International Conference on Web Services ICWS*, Chicago, USA, 2006.

[13] Daniel A. Menasce. Web server software architectures. *IEEE Internet Computing*, 7(6):78–81, 2003.

[14] J.A. Morrison. Response-time distribution for a processor-sharing system. *SIAM Journal on Applied Mathematics*, 45(1):152–167, 1985.

[15] T.J. Ott. The sojourn time distribution in the M/G/1 queue with processor sharing. *Journal of Applied Probability*, 21:360–378, 1984.

[16] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, 1994.

[17] Ben Shneiderman. Response time and display rate in human performance with computers. *ACM Comput. Surv.*, 16(3):265–285, 1984.

[18] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. An analytical model for multi-tier internet services and its applications. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 291–302, 2005.

[19] Bhuvan Urgaonkar and Prashant Shenoy. Cataclysm: policing extreme overloads in internet applications. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 740–749, New York, NY, USA, 2005. ACM Press.

[20] Daniel Villela, Prashant Pradhan, and Dan Rubenstein. Provisioning servers in the application tier for e-commerce systems. In *Proceedings of the Twelfth IEEE International Workshop on Quality of Service (IWQoS 2004)*, Montreal, Canada, June 2004.

[21] Werner Vogels. Learning from the Amazon technology platform. *ACM Queue*, 4(4), 2006.