An MDP-based Admission Control for Service-oriented Systems*

Marco Abundo, Valeria Cardellini, Francesco Lo Presti
Università di Roma "Tor Vergata"
Dipartimento di Informatica, Sistemi e Produzione
marco.abundo@gmail.com, cardellini@ing.uniroma2.it, lopresti@info.uniroma2.it

Università di Roma "Tor Vergata"
Dipartimento di Informatica, Sistemi e Produzione
Technical Report RR-11.86

February 3, 2011

Abstract

In the service computing paradigm, a service broker can build new applications by composing network-accessible services offered by loosely coupled independent providers. In this paper, we address the problem of providing a service broker, which offers to prospective users a composite service with a range of different Quality of Service (QoS) classes, with a forward-looking admission control policy based on Markov Decision Processes (MDP). This mechanism allows the broker to decide whether to accept or reject a new potential user in such a way to maximize its gain while guaranteeing non-functional QoS requirements to its already admitted users. We model the broker using a continuous-time MDP and consider various techniques suitable to solve both infinite-horizon and finite-horizon MDPs. To assess the effectiveness of the MDP-based admission control, we present experimental results where we compare the optimal decisions obtained by the analytical solution of the MDP with other admission control policies.

1 Introduction

In the Service Oriented Architecture (SOA) paradigm, the design of complex software is facilitated by the possibility to build new applications by composing network-accessible loosely-coupled services. The so built composite service is offered by a *service broker* to a range of different classes of users characterized by diverse Quality of Service (QoS) requirements. The broker and its users generally engage in a negotiation process, which culminates in the definition of a *Service Level Agreement* (SLA) about their respective duties and QoS expectations.

In the upcoming Internet service marketplace, multiple service providers may offer similar competing services corresponding to a functional description but at differentiated levels of QoS and cost. Therefore, in undertaking the management of the SOA-based system that offers the composite service, the broker has to meet both functional requirements concerning the overall logic to be implemented and non-functional requirements concerning the QoS levels that should be guaranteed. Hence, the service broker has to select at runtime the best set of component services implementing the needed functionalities in order to maximize some utility goal (e.g., its revenue) while guaranteeing the QoS levels to the composite service users. However, the latter is a challenging task because of the highly variable nature of the SOA environment. Recently, a significant number of research efforts have been devoted to service selection issues, e.g., [2, 7, 8]. The common aim of these works is to identify for each abstract functionality in the composite service a pool (eventually a singleton) of corresponding concrete services, selecting them from a set of candidates.

However, the candidate concrete services that the service broker can use to provide the functionalities of its composite service are in a limited number. Furthermore, the service broker contracts a SLA with each concrete service provider; the set of these SLAs defines the constraints within which the broker should try to meet the QoS objectives

agreed with its users and possibly to earn some revenue. Therefore, the service broker needs to apply some admission control mechanism on the users requesting to establish a SLA for the composite service, so that it can admit only those users for which the SOA system holds sufficient resources without incurring the risk of overcommitment, and, at the same time, it can exploit the available resources in a cost-effective way.

In this paper, we consider a service broker that manages a composite service offering differentiated QoS service classes to its prospective users and propose admission control policies to determine the admissibility of a user once it requests to establish a SLA for using the composite service. We formulate the admission control policies for the broker using Markov Decision Processes (MDPs), which are a powerful tool that allows to define an optimal policy with the best actions to be taken. The decision to accept another user of the composite service may influence both the QoS levels perceived by that user as well as that of ongoing users already in the SOA system; moreover, this decision changes the state of the system and therefore has an impact on whether future users will be accepted.

Specifically, the MDP-based admission control policies we present in this paper are tailored to the service broker we proposed in [8, 5] and that is named MOSES, which stands for *MOdel-based SElf-adaptation of SOA systems*. We model the MOSES system as a continuous-time MDP, whose solution allows to define an optimal admission control policy, where acceptance or rejection decisions are not made myopically, but they rather forecast rewards and costs associated with the future system states. The admission control policies aims at maximizing the service broker reward, while guaranteeing non-functional QoS requirements to its users. Therefore, differently from our previous work [8, 5], the broker does not carry on an altruistic strategy, but it rather meets its targets in a selfish mood.

We consider admission control policies that apply infinite-horizon and finite-horizon decisions and analyze their performance through simulation experiments. We also compare the MDP-based admission control policies to a myopic admission control strategy, where the service broker takes admission decisions only on the basis of the requesting user and the admitted users that are already using the SOA system.

A considerable number of research efforts have focused on the application of MDP-based models and stochastic programming to SOA systems and, more generally, to software systems [3, 4, 6, 9, 11, 12, 15, 17].

Some of these works have proposed self-healing approaches in order to support the reconfiguration of running services, e.g., [4, 9], considering also proactive solutions for SOA systems [15]. Some recent approaches [11, 12, 17] have used MDPs to model service composition with the aim to create automatically an abstract workflow of the service composition that satisfies functional and non-functional requirements, and also to allow the composite service to adapt dynamically to a varying environment [17].

Some works have proposed MDP-based admission control in service-oriented systems [3, 6] and are therefore most closely related to ours. In [3] Bannazadeh and Leon-Garcia have proposed an admission control for service-oriented systems which uses an online optimization approach for maximizing the system revenue, while in [6] Bichler and Setzer have applied an MDP-based formulation to tackle admission control for media on demand services. However, both these works do not consider composite services organized according to some business logic, while our approach is able to manage the admission control for a composite service whose workflow entails the composition patterns typical of orchestration languages such as BPEL [14], which is the de-facto standard for service workflows specification languages. To the best of our knowledge, the approach we propose in this paper is the first admission control policy based on MDPs for QoS-aware composite services.

Finally, MDPs have been extensively used to design call admission control at session level in wired and wireless networks, (e.g., [10, 13, 18] to name a few) and more generally to control communication systems (see [1] for a survey).

The rest of the paper is organized as follows. In Section 2 we present the SOA system managed by the service broker. In Section 3 we describe how we have modeled the SOA system with a continuous-time MDP. Then, in Section 4 we sketch out the implementation of our admission policies and present the simulation experiments to assess the effectiveness of the proposed MDP-based approach. Finally, we draw some conclusions and give hints for future work in Section 5.

2 MOSES System

MOSES, which stands for *MOdel-based SElf-adaptation of SOA systems*, is a QoS-driven runtime adaptation framework for SOA-based systems, designed as a service broker. In this section, we provide an overview on the MOSES system for which we propose in this paper MDP-based admission control policies. A detailed description of the MOSES methodology, architecture and implementing prototype can be found in [8] and [5], respectively.

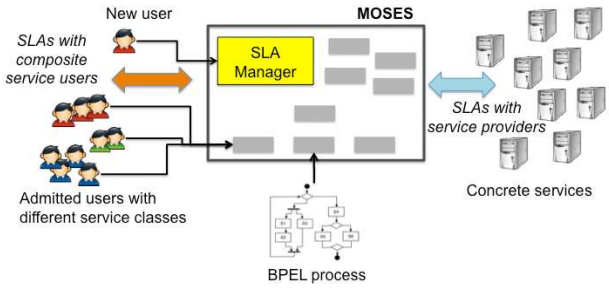

Figure 1: MOSES and its operating environment.

MOSES acts as a third-party intermediary between service users and providers, performing a role of provider towards the users and being in turn a requestor to the providers of the concrete services. It advertises and offers the composite service with a range of service classes which imply different QoS levels and monetary prices. Figure 2 shows a high-level view of the MOSES environment, where we have highlighted the MOSES component on which we focus in this paper, i.e., the *SLA Manager*.

The workflow that defines the composition logic of the service managed by MOSES can include all the different types of BPEL structured activities: `sequence`, `switch`, `while`, `pick`, and `flow` [14]. Figure 2 shows an example of BPEL workflow, described as a UML2 activity diagram, that can be managed by MOSES. This example encompasses all the BPEL structured activities mentioned above, except for the `pick` construct. The figure also shows the functionalities (named *tasks* and represented by $S_1, \ldots, S_6$) needed to compose the new added value service.

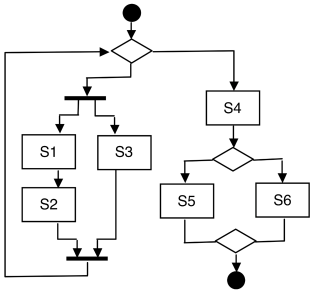

Figure 2: A MOSES-compliant workflow.

MOSES performs a two-fold role of service provider towards its users, and of service user with respect to the providers of the concrete services it uses to implement the composite service it is managing. Hence, it is involved in two types of SLAs, corresponding to these two roles. In general, a SLA may include a large set of parameters, referring to different kinds of functional and non-functional attributes of the service, and different ways of measuring them. MOSES presently considers the average value of the following attributes:

- *response time*: the interval of time elapsed from the service invocation to its completion;
- *reliability*: the probability that the service completes its task when invoked;
- *cost*: the price charged for the service invocation.

Other attributes, like reputation or availability, could be easily added.

Our general model for the SLA between the provider and the user of a service consists of a tuple $\langle T, C, R, L \rangle$, where: T is the upper bound on the average service response time, C is the service cost per invocation, R is the lower bound on the service reliability. The provider guarantees that thresholds T and R will hold on average provided that the request rate generated by the user does not exceed the load threshold L.

In the case of the SLAs between the composite service users and MOSES (acting the provider role), we assume that MOSES offers a set K of service classes. Hence, the SLA for each user u of a class $k \in K$ is defined as a tuple $\langle T^k_{\max}, C^k, R^k_{\min}, L^k_u, P^k_\tau, P^k_\rho \rangle$. The two additional parameters P^k_τ and P^k_ρ represent the penalty rates MOSES will refund its users with for possible violations of the service class response time and reliability, respectively. All these coexisting SLAs (for each u and k) define the QoS objectives that MOSES must meet. We observe that MOSES considers SLAs stating conditions that should hold globally for a *flow* of requests generated by a user.

To meet these objectives, we assume that MOSES (acting the user role) has already identified for each task $S_i \in \mathcal{F}$ in the composite service a pool of corresponding concrete services implementing it. The SLA contracted between MOSES and the provider of the concrete service $i.j \in I_i$ is defined as a tuple $\langle t_{ij}, c_{ij}, r_{ij}, l_{ij} \rangle$. These SLAs define the constraints within which MOSES should try to meet its QoS objectives.

New users requesting the composite service managed by MOSES are subject to an accept/deny decision, with which MOSES determines whether or not it is convenient to admit the user in the system according to the user SLA and the system state (present or even future, on the basis of the adopted admission control policy). We will present in Section 3 the MDP-based formulation of the admission control carried out by the SLA Manager component.

Once a user requesting a SLA has been admitted by the SLA Manager, it starts generating requests to the composite service managed by MOSES until its contract ends. Each user request involves the invocations of the tasks according to the logic specified by the composite service workflow. For each task invocation, MOSES binds dynamically the task of the abstract composition to an actual implementation (i.e., concrete service), selecting it from the pool of network accessible service providers that offer it. We model this selection by associating with each task S_i a vector $\boldsymbol{x}_i = (\boldsymbol{x}^1_i, ..., \boldsymbol{x}^{|K|}_i)$, where $\boldsymbol{x}^k_i = [x^k_{ij}]$ and $i.j \in I_i$. Each entry x^k_{ij} of $\boldsymbol{x}^k_i$ denotes the probability that the class-k request will be bound to concrete service $i.j$.

The service selection is driven by the solution of a suitable optimization problem. We assume that the broker wants to maximize its profit. We can formulate the service selection problem as a LP maximization problem which takes the following form:

Problem **MAXRW:**

$$\textbf{max } C(\Lambda, \boldsymbol{x}) = \sum_{k \in K} \Lambda^k \Big[C^k - \Big(C^k(\Lambda, \boldsymbol{x}) + P^k_T \tau^k + P^k_R \rho^k \Big) \Big]$$

$$\textbf{subject to: } T^k(\Lambda, \boldsymbol{x}) \leq T^k_{\max} + \tau^k, \quad k \in K \tag{1}$$

$$R^k(\Lambda, \boldsymbol{x}) \leq R^k_{\min} - \rho^k, \quad k \in K \tag{2}$$

$$C^k(\Lambda, \boldsymbol{x}) \leq C^k, \quad k \in K \tag{3}$$

$$l_{ij}(\Lambda, \boldsymbol{x}) \leq l_{ij}, \quad j \in I_i,\ i \in \mathcal{F} \tag{4}$$

$$x^k_{ij} \geq 0, j \in I_i, \sum_{j \in I_i} x^k_{ij} = 1, \quad i \in \mathcal{F}, k \in K$$

$$\tau^k \geq 0, \rho^k \geq 0, \quad k \in K \tag{5}$$

where: $\Lambda = (\Lambda^k)_{k \in K}$ and $\Lambda^k = \sum_u L^k_u$ is the aggregate class-k users service request rate; $T^k(\Lambda, \boldsymbol{x})$, $R^k(\Lambda, \boldsymbol{x})$, and $C^k(\Lambda, \boldsymbol{x})$ the class-k response time, reliability and implementation cost, respectively, under the service selection strategy $\boldsymbol{x}$. Inequalities (1) and (2) represent the service constraints on class-k response time and reliability. In these equations τ^k and ρ^k represent the amount of violation with respect to the agreed upon response time $T^k_{\max}$ and reliability $R^k_{\min}$ for which the broker pays a penalty proportional to. Inequalities (3) are the cost constraints which ensure to the broker that that class-k service implementation cost $C^k(\Lambda, \boldsymbol{x})$ is covered by the service class user per invocation cost C^k, *i.e.,* the implementation is cost-effective. Inequalities (4) require that each concrete service $i.j$ load under policy $\boldsymbol{x}$, $l_{ij}(\Lambda, \boldsymbol{x})$ does not exceed the maximum load l_{ij} the broker has agreed with its service providers. We omit the details which can be found in [8][1].

The objective function $C(\Lambda, \boldsymbol{x})$ is the broker per unit of time reward, which is the sum over all service classes of the service class service invocation rate Λ^k times the per invocation reward that is C^k minus the cost $C^k(\boldsymbol{x})$ (which is increased by the penalty $P^k_T \tau^k + P^k_R \rho^k$ for service violation).

[1]The formulation of the service selection optimization problem we consider in this paper differs from [8] in that in the former no violation of the SLAs with the users is allowed.

Since the proposed optimization problem is a Linear Programming problem it can be efficiently solved via standard techniques. We will denote by $\boldsymbol{x}^*(\Lambda)$ the optimal service selection policy.

3 An MDP Formulation for MOSES Admission Control

In this section we formulate the MOSES admission control problem as a Continuous-time Markov Decision Process (CTMDP). We first present our broker model and define the user state space model. Then, we define the broker actions/decisions and present the state transition dynamics. Finally, we present our performance criterion and how to compute the optimal policy.

3.1 Model

We consider a broker that has a fixed set of candidate concrete services (and associated SLAs) with which offers the composite service to prospective users. Prospective users contact the broker to establish a SLA for a given class of service k and for a given period of length. We model the arrival process for service class k and contract duration of expected length $1/\mu_d$ as a Poisson process with rate λ_d^k. We assume that the contract durations are exponentially distributed with finite mean $1/\mu_d > 0$ $d \in D = \{1, \ldots, d_{\max}\}$ (which we assume for the sake of simplicity to not depend on the service class k). Upon a user arrival, the broker has to decide whether to admit a user or not. If a user is admitted, the user will generate a flow of requests at rate L^k for the duration of the contract. When a user contract expires, the user simply leaves the system. The broker set of actions is then just the pair $\mathcal{A} = \{a_a, a_r\}$, with a_a denoting the accepting decision and a_r the refusal decision.

System State

We model the state of our system as in [18]. The state s consists of the following two components:

- the broker users matrix $n = (n_d^k)_{k \in K, d \in D}$, where n_d^k denotes the number of users for each service class k and expected contract duration $1/\mu_d$ before the last random event occurred;
- the last random event ω.

n takes values in the set $\mathcal{N}$ of all possible broker user matrices for which the optimization problem **MAXRW** introduced in Section 2 has a feasible solution. ω represents the last random event, *i.e.*, a user arrival or departure, occurred in the system. We will denote it by a matrix $\omega = (\omega_d^k)_{k \in K, d \in D}$, where $\omega_d^k = 1$ if a new user makes an admission request for service class k and for a contract duration with mean $1/\mu_d$, $\omega_d^k = -1$ if an existing user of class k and contract duration of mean $1/\mu_d$ terminates his contract, and $\omega_d^k = 0$ otherwise. We will denote by Ω the set of all possible events.

The state space $\mathcal{S}$ consists of all possible user configuration-next event combinations, *i.e.*,

$$\mathcal{S} = \left\{ s = (n, \omega) | n \in \mathcal{N}, \omega \in \Omega, \omega_d^k \geq 0 \text{ if } n_d^k = 0 \right\}$$

It is important to observe that, following [18], there is a subtle relationship between a state $s = (n, \omega)$ value and the associated user configuration n. Indeed, if the current state is $s = (n, \omega)$ it means that the user configuration *was* n before the last occurred event ω. The actual current user configuration is instead n', which depends on both the event ω and the decision a taken by the broker as discussed below.

Actions

The broker avails itself of the possibility to accept or refuse a new user. For each state $s = (n, \omega)$, the set of available broker actions/decisions $A(s)$ depends on the event ω. If ω denotes an arrival, the broker has to determine whether to accept it or not; thus $A(s) = \{a_a, a_r\}$. If, instead, ω denotes a contract termination, there is no decision to take and $A(s) = \emptyset$.

Table 1: System transitions.

Event ω	Decision	Next state $s' = (n', \omega')$
arrival	admitted ($a = a_a$)	$(n + \omega, \omega')$
	refused ($a = a_r$)	(n, ω')
departure	-	$(n + \omega, \omega')$

Transitions

System transitions are caused by users arrivals or departures. Given the current state $s = (n, \omega)$, the new state $s' = (n', \omega')$ is determined as follows:

- ω' is the event occurred;
- n' is the user configuration *after* the event ω (the previous event) and the decision $a \in A(s)$ taken by the broker upon ω. n' differs from n upon a user departure or a user arrival provided it is accepted. In compact form we can write $n' = n + \omega \mathbf{1}_{\{a \neq a_r\}}$, where $\mathbf{1}_{\{.\}}$ is the indicator function.

Observe that while the system is in state s the actual user configuration is n', which will characterize the next state s'. Table 1 summarizes all the possible transitions.

The associated transition rates are then readily obtained:

$$q_{ss'} = \begin{cases} \lambda_d^k & \omega_d'^k = 1 \\ \mu_d n_d'^k & \omega_d'^k = -1 \end{cases} \tag{6}$$

3.2 Optimal Policy

An admission control policy π for the service broker is a function $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which defines for each state $s \in \mathcal{S}$ whether the broker should admit or refuse a new user. We are interested in determining the admission control policy which maximizes the broker discounted expected reward/profit with discounting rate $\alpha > 0$. For a given policy π let $v_\alpha^\pi(s)$ be the expected infinite-horizon discounted reward given s as initial state, defined as:

$$v_\alpha^\pi(s) = E_s^\pi \left\{ \sum_{i=1}^{\infty} \int_{\sigma_n}^{\sigma_{n+1}} e^{-\alpha u} c(s_i, a_i) du \right\} \tag{7}$$

where $\sigma_1, \sigma_2, \ldots$ represents the time of the successive system decision epochs which, in our model, coincide with user arrivals and departures. $c(s_i, a_i)$ is the broker reward between decision epochs i and $i + 1$, that is MOSES reward under the optimal service selection strategy x^* between the two decision epochs. To compute its value, let us denote by $\Lambda^k(s, a)$ the aggregate class-k users service request rate when the state is s and the broker action was a and let $\Lambda(s, a) = (\Lambda^k(s, a))_{k \in K}$. Then, $\Lambda^k(s, a) = n'^k L^k$ where $n' = n + \omega \mathbf{1}_{\{a \neq a_r\}}$ is the next state configuration given the actual state is $s = (n, \omega)$ and decision a was taken and $n'^k = \sum_{d \in D} n_d'^k$ is the number of user in service class k. We thus have

$$c(s, a) = C(\Lambda(s, a), x^*(\Lambda(s, a)) \tag{8}$$

The optimal policy π^* satisfies the optimality equation (see 11.5.4 in [16]):

$$v_\alpha^{\pi^*}(s) = \sup_{a \in A(s)} \left\{ \frac{c(s, a)}{\alpha + \beta(s, a)} + \sum_{s' \in \mathcal{S}} \frac{q_{ss'}}{\alpha + \beta(s, a)} v_\alpha^{\pi^*}(s') \right\}, \forall s \in \mathcal{S} \tag{9}$$

where $\beta(s, a)$ is the rate out of state s if action a is chosen, *i.e.*,

$$\beta(s, a) = \sum_{k \in K} \sum_{d \in D} (\lambda_d^k + n_d'^k \mu_d).$$

In (9), the first term $\frac{c(s,a)}{\alpha + \beta(s,a)}$ represents the expected total discounted reward between the first two decision epochs given the system initially occupied state s and taken decision a. The second term represents the expected discounted reward after the second decision epoch under the optimal policy.

The optimal policy π^* can be obtained by solving the optimality equation (9) via standard techniques, *e.g.*, value iteration, LP formulation [16].

A potential limitation of the infinite-horizon approach we presented above arises from the curse of dimensionality which gives rise to state explosion. As shown in the next section, in our setting, even for small problem instances, we incurred high computational costs because of the large state space. As a consequence, this approach might not be feasible for online operation where a new policy must be recomputed as user statistics or the set of concrete services varies over time unless we resort to heuristics. In alternative, we also consider finite horizon policies which not only are amenable to efficient implementations, and allow to trade-off complexity vs horizon length, but also take into account the fact that in a time varying system it might not be appropriate to consider a stationary, infinite horizon policy.

In a finite-horizon setting, our aim is to optimize the expected N step finite-horizon discounted reward given s as initial state, $v_\alpha^\pi(s)$ defined as:

$$v_\alpha^{\pi_N}(s) = E_s^\pi \left\{ \sum_{i=1}^{N} \int_{\sigma_n}^{\sigma_{n+1}} e^{-\alpha u} c(s_i, a_i) du \right\} \tag{10}$$

for a suitable N which defines the number of decision epochs over which the reward is computed.

For finite horizon problem, the optimal policy π_N^* satisfies the following optimality equation:

$$v_{i,\alpha}^{\pi_N^*}(s) = \sup_{a \in A(s)} \left\{ \frac{c(s,a)}{\alpha + \beta(s,a)} + \sum_{s' \in \mathcal{S}} \frac{q_{ss'}}{\alpha + \beta(s,a)} v_{i+1,\alpha}^{\pi_N^*}(s') \right\}, \forall s \in \mathcal{S} \tag{11}$$

where $v_{i,\alpha}^{\pi_N}(s)$ is the expected discounted reward under policy π from decision epoch i up to N and $v_\alpha^{\pi_N}(s) = v_{1,\alpha}^{\pi_N}(s)$. The optimal policy π_N^* can be computed directly from (11) via backward induction by exploiting the recursive nature of the optimality equation [16]. In the special case of $N = 1$, a 1-step horizon policy, which we will consider in the next section, (11) reduces to:

$$v_{1,\alpha}^{\pi_1^*}(s) = \sup_{a \in A(s)} \left\{ \frac{c(s,a)}{\alpha + \beta(s,a)} \right\}, \forall s \in \mathcal{S} \tag{12}$$

4 Experimental Analysis

In this section, we present the experimental analysis we have conducted through simulation to assess the effectiveness of the MDP-based admission control for MOSES. We first describe the simulation model and then present the simulation results.

4.1 Simulation Model

Following the broker model in Section 3, we consider an open system model, where new users belonging to a given service class $k \in K$ and expected contract duration $1/\mu_d$ arrive according to a Poisson process of rate λ_d^k. We also assume exponential distributed contract duration. Once a user is admitted, it starts generating requests to the composite service according to an exponential inter-arrival time with rate L_u^k until its contract expires.

The discrete-event simulator has been implemented in C language using the CSIM 20 tool. Multiple independent random number streams have been used for each stochastic model component. The experiments involved a minimum of 10,000 completed requests to the composite service; for each measured mean value the 95% confidence interval has been obtained using the run length control provided by CSIM. As regards the admission control policies, they have been implemented in MATLAB.

4.2 Experimental Results

We illustrate the dynamic behavior of our admission control policies assuming that MOSES provides the composite service whose workflow is shown in Figure 2. For the sake of simplicity, we assume that two candidate concrete services (with their respective SLAs) have been identified for each task, except for S_2 for which four concrete services have been identified. The respective SLAs differ in terms of cost c, reliability r, and response time t (being the latter

measured in sec.); the corresponding values are reported in Table 2 (where $i.j$ denotes the concrete service). For all concrete services, $l_{ij} = 10$ invocations per second.

Table 2: Concrete service SLA parameters.

$i.j$	c_{ij}	r_{ij}	t_{ij}	$i.j$	c_{ij}	r_{ij}	t_{ij}
1.1	6	0.995	2	3.2	1.8	0.995	2
1.2	3	0.99	4	4.1	1	0.995	0.5
2.1	4.5	0.99	1	4.2	0.8	0.99	1
2.2	4	0.99	2	5.1	2	0.99	2
2.3	2	0.95	4	5.2	1.4	0.95	4
2.4	1	0.95	5	6.1	0.5	0.99	1.8
3.1	2	0.995	1	6.2	0.4	0.95	4

On the user side, we assume a scenario with four classes (*i.e.*, $1 \leq k \leq 4$) of the composite service managed by MOSES. The SLAs negotiated by the users are characterized by a wide range of QoS requirements as listed in Table 3, with users in class 1 having the most stringent performance requirements and highest cost paid to the broker, and users in class 4 the least stringent performance requirements and lowest cost. The penalty rates P_τ^k and P_ρ^k are equal to the reciprocal of the corresponding SLA parameter. Furthermore, for each service class we consider two possible contract

Table 3: Class SLA parameters.

Class k	C^k	$R^k_{\min}$	$T^k_{\max}$
1	25	0.95	7
2	18	0.9	11
3	15	0.9	15
4	12	0.85	18

durations (*i.e.*, $d_{\max} = 2$), which can be either *short* or *long*. Therefore, the system state $s = (n, \omega)$ is characterized by a 4×2 broker users matrix n, as defined in Section 3.1.

We compare the results of the following admission control policies for MOSES. Under the **infinite horizon** policy, the admission control decisions are based on the optimal policy π^*, which is obtained by solving the optimality equation (9) via the value iteration method setting the discount rate $\alpha = -\ln(0.9) = 0.1054$ and the parameter $\epsilon = 0.01$.

With the **1-step horizon** policy, the admission control decisions are based on the optimal policy π^*_N with a local 1-step reasoning, *i.e.*, $N = 1$. In this case, as explained in Section 3.2, we obtain π^*_N by solving (12) setting the discount exponent $\alpha = -\ln(0.9) = 0.1054$.

Finally, with the **blind** policy, no reasoning about future rewards is considered, because MOSES accepts a new contract request only if the service selection optimization problem **MAXRW** described in Section 2 can be solved given the SLA requested by the new users and the SLAs agreed by MOSES with its currently admitted users. Specifically, the user request rate L^k_u is added to the aggregate flow Λ^k of class-k requests currently served by MOSES, and the so obtained instance of the LP optimization problem is solved. If a solution exists, the user is admitted; otherwise, its SLA request is rejected, because MOSES does not currently hold sufficient resources to manage it and the already admitted users with their SLAs.

We consider three different scenarios, where we vary the arrival rate of the contract requests. On the other hand, in all scenarios the amount of request generated by an admitted user is $L^k_u = 1$ req/sec and the contract duration is fixed to $(1/\mu_d)_{d \in D} = (50, 200)$, where the first component corresponds to short contracts and the latter to longer contracts. In the following, we will denote short and long contracts with *s* and *l*, respectively.

In the *first scenario*, we set the matrix $(\lambda^k_d)_{k \in K, d \in D} = \begin{pmatrix} 0.02 & 0.02 \\ 0.02 & 0.02 \\ 0.02 & 0.02 \\ 0.02 & 0.02 \end{pmatrix}$, that is all the contract requests arrive at the same rate, irrespectively of the service class.

In the *second scenario*, $(\lambda^k_d)_{k \in K, d \in D} = \begin{pmatrix} 0.02 & 0.02 \\ 0.02 & 0.02 \\ 0.04 & 0.04 \\ 0.08 & 0.08 \end{pmatrix}$, that is contract requests for service classes 3 and 4, which are less profitable for the broker as their SLAs have lower C^k (see Table 3), arrive at a double (class 3) or quadruple

(class 4) rate with respect to requests for service classes 1 and 2.

In the *third scenario*, $(\lambda_d^k)_{k \in K, d \in D} = \begin{pmatrix} 0.08 & 0.08 \\ 0.04 & 0.04 \\ 0.02 & 0.02 \\ 0.02 & 0.02 \end{pmatrix}$, that is contract requests for service classes 1 and 2, which are more profitable for the broker as their SLAs have higher C^k (see Table 3), arrive at a quadruple (class 1) or double (class 2) rate with respect to requests for service classes 3 and 4.

To compare the performance of the different admission control policies, we consider as main metrics the average reward per second of the service broker over the simulation period and the percentage of rejected contract requests. Furthermore, for the MDP-based admission control policies we analyze also the mean execution time. For space reason, we do not show the QoS satisfaction levels achieved by the users for the response time, reliability, and cost SLA parameters. Anyway, we found that once a contract request has been accepted, the QoS levels specified in the SLAs are quite largely met by MOSES for each flow of service class, independently on the applied admission policy.

Table 4 shows the average reward per second earned by the service broker for the various admission control policies and under the different considered scenarios. We can anticipate that, as expected, the infinite horizon policy maximizes

Table 4: Average reward per second.

Admission policy	Scenario 1	Scenario 2	Scenario 3
Blind	40.536	25.012	58.801
1-step horizon	59.607	63.865	75.751
Infinite horizon	66.737	65.553	76.116

the broker reward, achieving a largely significant improvement over the blind policy under all scenarios. Anyway, the 1-step horizon policy also allows to obtain a much better reward with respect to the blind policy and very close to that achieved using the infinite horizon policy, notwithstanding that the first evaluates the optimal policy by taking into account only one step in the future.

Let us now analyze the performance metrics separately for each scenario. From Table 4 we can see that in the first scenario the 1-step horizon policy let the broker earn 47% more than the blind policy, while the improvement achieved by infinite horizon policy over blind is even higher, being equal to almost 65%.

Figure 3 shows the percentage of rejected SLA contracts for all the service classes, distinguishing further between short and long contract durations, achieved by the different admission control policies (for each policy, the first four bars regard the short-term contracts for the various service classes, while the latter four the long-term ones). While the

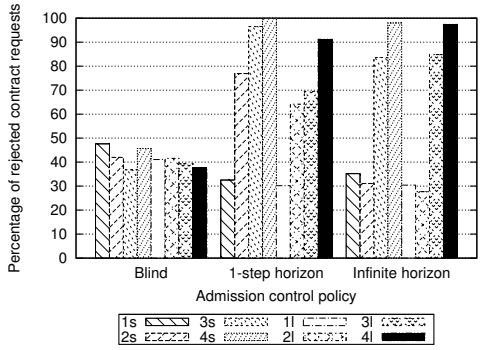

Figure 3: Rejected contract requests under scenario 1.

blind policy is not able to differentiate among the service classes from the admission control point of view, because it rejects in the same way the classes that pay most and least the service broker, the MDP-based policies tend to accept the more profitable classes 1 and 2, which pay more for the composite service, and to reject the less profitable ones. We also observe that the infinite horizon policy does not differentiate between short-term and long-term contracts, probably because in this scenario contract durations are not essential in the decisions taken by the optimal policy in the long distance. On the contrary, fixed the service class, the 1-step horizon policy tends to accept more the long-term contracts than the short-term ones, except for class 1 for which the two types of contract are accepted equally, being the latter the most profitable class for which the policy attempts to maximize its acceptance.

For the second scenario, which is characterized by a higher contract request arrival rate for classes 3 and 4, Figure 4 shows that, as expected, all the admission control policies reject a higher percentage of contract requests for these classes with respect to the first scenario. However, MDP-based admission control, independently on the horizon

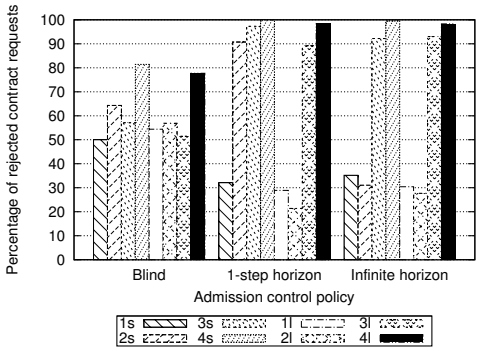

Figure 4: Rejected contract requests under scenario 2.

width, prefers clearly service classes 1 and 2 with respect to 3 and 4, since the former ones let the system achieve higher rewards while the latter, that could use the limited system resources with a low revenue for the broker, incur in a very high refusal percentage (almost total for class 4, which is the least advantageous one). We also observe that the infinite horizon policy now slightly differentiates within classes 1 and 2 according to the contract duration: long-term contracts are preferred to short-term ones (a reduction in the rejection decisions equal to 16% and 11% for long-term classes 1 and 2, respectively). This behavior is much more evident for the 1-step horizon policy, especially for class 2. Analyzing the average reward reported in Table 4, we can see that under the second scenario the MDP-based policies allow the broker to more than double its revenue: the 1-step horizon and infinite horizon policies let the broker earn 155% and 162% respectively more than the blind policy. The type of MOSES rewards lets the 1-step horizon policy behavior be enough accurate; therefore, the reward achieved by this policy is only slightly less than that obtained by a strategy taking a wider (infinite) horizon into account.

In the third scenario, where classes 1 and 2 arrive more frequently than classes 3 and 4 (with a quadruple and double rate, respectively), the MDP-based admission control policies still allow to achieve a good improvement in the reward gained by the broker, as reported in Table 4 (29% and 29.5% for 1-step horizon and infinite horizon policies, respectively, when compared to the blind one). Figure 5 shows the corresponding rejection percentage. We observe that the infinite horizon policy does not differentiate in a significant way between long-term and short-term contract requests, while the 1-step horizon admission control refuses twice short-term requests with respect to long-term requests for class 2.

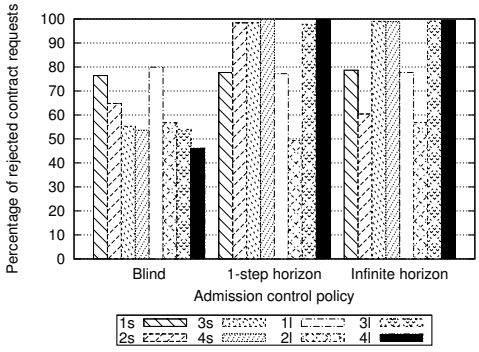

Figure 5: Rejected contract requests under scenario 3.

Under all the considered scenarios, the 1-step horizon policy allows the service broker to make a profit comparable, although slightly reduced, to the infinite horizon policy. However, a strong argument in favor of the 1-step horizon policy regards the execution time needed to achieve the optimal decision. We have measured the mean execution time

on a machine with Intel Core 2 Duo T7250 2 GHz and 2 GB RAM. The 1-step horizon policy requires only 0.0021 sec, while the infinite horizon one requires 233 sec. for the state space generation, 5502 sec. for the matrix generation, and 800 sec. for the value iteration method. This long execution time is also due to the computation of $c(s, a)$, which requires to solve the service selection optimization problem (see (8)). Therefore, the reduced computational cost of the 1-step horizon policy makes it amenable to take online admission control decisions and to integrate the MDP-based admission control with other adaptation triggers (e.g., changes in the set of concrete services) that are managed by MOSES.

5 Conclusions

In this paper, we have studied the admission control problem for a service broker, MOSES, which offers to prospective users a composite service with different QoS levels. We have formulated the admission control problem as a Markov Decision Process with the goal to maximize the broker discounted reward, while guaranteeing non-functional QoS requirements to its users. We have considered both infinite-horizon and the less computational demanding finite-horizon cost functions. We have compared the different solutions through simulation experiments. Our results show that the MDP-based policies guarantee much higher profit to the broker while guaranteeing the users QoS levels with respect to a simple myopic policy which accepts users as long as the broker has sufficient resources to serve them. In particular, the simple 1-step horizon policy achieves near to optimal performance at a fraction of the computational cost which makes it amenable to online implementation.

In our future work we plan to implement the MDP-based admission control in the existing MOSES prototype and run experiments in realistic scenarios.

References

[1] E. Altman. Applications of Markov decision processes in communication networks. In E. Feinberg and A. Shwartz, editors, *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer, 2002.

[2] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Softw. Eng.*, 33(6):369–384, 2007.

[3] H. Bannazadeh and A. Leon-Garcia. Online optimization in application admission control for service oriented systems. In *Proc. IEEE APSCC '08*, 2008.

[4] M. Beckmann and R. Subramanian. Optimal replacement policy for a redundant system. *OR Spectrum*, 6(1):47–51, 1984.

[5] A. Bellucci, V. Cardellini, V. Di Valerio, and S. Iannucci. A scalable and highly available brokering service for SLA-based composite services. In *Proc. ICSOC '10*, volume 6470 of *LNCS*. Springer, Dec. 2010.

[6] M. Bichler and T. Setzer. Admission control for media on demand services. *Service Oriented Computing and Applications*, 1(1):65–73, 2007.

[7] G. Canfora, M. Di Penta, R. Esposito, and M. Villani. A framework for QoS-aware binding and re-binding of composite web services. *J. Syst. Softw.*, 81, 2008.

[8] V. Cardellini, E. Casalicchio, V. Grassi, and F. Lo Presti. Flow-based service selection for web service composition supporting multiple QoS classes. In *Proc. IEEE ICWS '07*, pages 743–750, 2007.

[9] M. Chen and R. Feldman. Optimal replacement policies with minimal repair and age-dependent costs. *Eur. J. Oper. Res.*, 98(1):75–84, 1997.

[10] C. Comaniciu and H. Poor. Jointly optimal power and admission control for delay sensitive traffic in CDMA networks with LMMSE receivers. *IEEE Trans. Signal Process.*, 51(8):2031–2042, Aug. 2003.

[11] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition: using Markov decision processes. *Int'l J. Web Service Res.*, 2(1), 2005.

[12] A. Gao, D. Yang, S. Tang, and M. Zhang. Web service composition using Markov decision processes. In *Proc. WAIM '05*, volume 3739 of *LNCS*. Springer, 2005.

[13] E. Nordstrom and J. Carlstrom. Call admission control and routing for integrated CBR/VBR and ABR services: a Markov decision approach. In *Proc. IEEE 1999 ATM Workshop*, pages 71–76, 1999.

[14] OASIS. Web Services Business Process Execution Language Version 2.0, Jan. 2007.

[15] S. Pillai and N. Narendra. Optimal replacement policy of services based on Markov decision process. In *Proc. IEEE SCC '09*, pages 176–183, 2009.

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

[17] H. Wang, X. Zhou, W. Liu, W. Li, and A. Bouguettaya. Adaptive service composition based on reinforcement learning. In *Proc. ICSOC '10*, volume 6470 of *LNCS*, pages 92–107. Springer, Dec. 2010.

[18] C. Wu and D. Bertsekas. Admission control for wireless networks. Technical Report LIDS-P- 2466, Lab. for Information and Decision Systems, MIT, 1999.