An MDP-based Admission Control for a QoS-aware Service-oriented System

Marco Abundo, Valeria Cardellini, Francesco Lo Presti
DISP, Università di Roma "Tor Vergata"
Via del Politecnico 1, 00133 Roma, Italy
marco.abundo@gmail.com, cardellini@ing.uniroma2.it, lopresti@info.uniroma2.it

Abstract

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 (MDPs).

Index Terms

Admission control, Markov Decision Process, Quality of Service, Service Oriented Architecture.

1. Introduction

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 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. Finally, in Section 4 we summarize our experimental results.

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 (please refer to [1] and [2] for details in the MOSES methodology).

978-1-4577-0103-0/11/$26.00

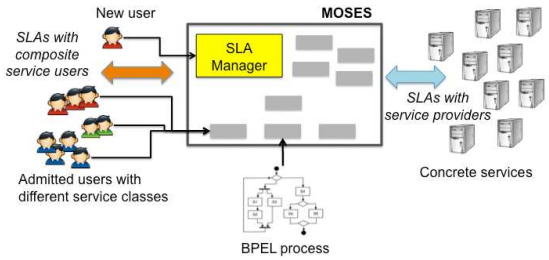

Figure 1. MOSES and its operating environment.

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 offers the composite service with a range of service classes with different QoS levels and monetary prices. Figure 1 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` [3].

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 manages. Hence, it is involved in two types of SLAs, corresponding to these two roles. MOSES presently considers SLAs based on 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.

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 T and R will hold provided that the request rate does not exceed 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. To meet these QoS objectives, we assume that MOSES (acting the user role) has 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. 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 $i.j$.

MOSES determines the service selection strategy $\boldsymbol{x}$ by solving the following cost minimization 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 $\boldsymbol{x}$. Details can be found in [1]. The objective function $C(\Lambda, \boldsymbol{x})$ is the broker per unit of time reward. 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).

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^k_d. 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.

We model the state of our system as in [4]. The state s consists of the following two components:

- the broker users matrix $n = (n^k_d)_{k \in K, d \in D}$, where n^k_d 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 $\omega = (\omega^k_d)_{k \in K, d \in D}$, where $\omega^k_d = 1$ if a new

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	-	$(n + \omega, \omega')$

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. We will denote by $\mathcal{S}$ the state space.

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 $A(s) = \emptyset$.

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.

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:

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

3.2. Optimal Policy

An admission control policy π for the service broker is a function $\pi : \mathcal{S} \to \mathcal{A}$ 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 $v_\alpha^\pi(s)$ with discounting rate $\alpha > 0$ [5]. The optimal policy π^* satisfies the optimality equation (see 11.5.4 in [5]):

$$v_\alpha^{\pi^*}(s) = \sup_{a \in A(s)} \left\{ \frac{c(s,a)}{\alpha + \beta(s,a)} + \sum_{s' \in \mathcal{S}} \frac{q_{ss'}}{\alpha + \beta(s,a)} v_\alpha^{\pi^*}(s') \right\}, \forall s \in \mathcal{S} \quad (7)$$

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 (7), 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 (7) via standard techniques, *e.g.*, value iteration, LP formulation [5].

4. Experimental Results

We conducted a thorough set of simulations experiments to assess the effectiveness of the proposed solution (see [6] for the details). We compared our proposed MDP-based admission control policy against a *blind* policy which simply accepts a new user if the **MAXRW** problem has a solution and considered as main metric the average reward per second of the service broker. The results show that the MDP-based admission control always guarantees higher average rewards than the *blind* policy with an improvement that ranges from 30% to 200% depending on the scenario.

Finally, we also considered finite horizon policies which are computationally more efficient than the infinite horizon counterpart and that allow us to trade-off optimality with computational complexity/time-horizon length. Our findings have shown that even with the simple 1-step horizon policy it is possible to achieve better results with respect to the *blind* policy, quite close to the infinite horizon optimum, but at a fraction of the computational cost.

References

[1] 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*, 2007, pp. 743–750.

[2] 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*, ser. LNCS, vol. 6470. Springer, Dec. 2010.

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

[4] C. Wu and D. Bertsekas, "Admission control for wireless networks," Lab. for Information and Decision Systems, MIT, Tech. Rep. LIDS-P- 2466, 1999.

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

[6] M. Abundo, V. Cardellini, and F. Lo Presti, "An MDP-based Admission Control for Service-oriented Systems," DISP, Univ. of Roma "Tor Vergata", Tech. Rep. RR-11.86, Feb. 2011, http://www.ce.uniroma2.it/publications/RR-11.86.pdf.