Revenue Optimization of Service Compositions using Conditional Request Retries

Miroslav Živković, Performance of Networks and Systems, TNO, Delft, The Netherlands & Design and Analysis of Communication Systems, University of Twente, Enschede, The Netherlands

Hans van den Berg, Performance of Networks and Systems, TNO, Delft, The Netherlands & Design and Analysis of Communication Systems, University of Twente, Enschede, The Netherlands

ABSTRACT

Due to the inherent stochastic nature of services execution environment within service oriented systems, a runtime adaptation of the given composition may be required. The authors investigate a runtime service adaptation mechanism based on conditional retries for orchestrated web services. The conditional retry may be issued while a concrete service within composition is executed. The retry could either invoke the same concrete service or a functionally equivalent web service that implements the same task. The authors use dynamic programming to determine the optimal time instances at which the current request should be terminated before request replication. The calculation takes into account different QoS parameters like services' response-time distributions and cost-related parameters, and the solution optimizes the expected revenue of the composite service provider. The authors illustrate the benefits of their approach by numerical calculations, and discuss the impact of considered QoS parameters to the solution at hand.

Keywords: Composite Service Provider (CSP), Dynamic Programming, Heavy Tailed Distributions, Optimal Retry Policies, Service Oriented Architecture

INTRODUCTION

Within Service Oriented Architecture, service orchestration refers to the compositions where one central controller “orchestrates” (steers) the execution logic, according to a pre-defined workflow. Due to the inherent stochastic nature of web services execution environment, QoS provisioning is a difficult task and Service Level Agreements (SLAs) violations could occur relatively often, leading to providers’ (financial) losses and customer dissatisfaction. The issue of QoS provisioning is a challenging one, and is usually addressed by adaptation of the composition itself at runtime. One way to perform the adaptation of the web service composition at runtime may be by service rebinding/substitution, (Ardagna, Comuzzi et al., 2007; DOI: 10.4018/jwsr.2013040101)
Leitner, Hummer et al., 2011). In such a case, the adaptation is performed by exchanging a service within the composition by another (in ideal case functionally equivalent) one, that is usually faster but more expensive than the initial service selected. This implies that service adaptation usually brings additional costs for the composite service provider. When actual service adaptation costs are not carefully taken into account, one may end up with the situation when it is economically better to pay the penalty for the SLA violation without adaptation, than otherwise, (Leitner, 2011).

This paper addresses the problem of runtime adaptation of the orchestrated service composition that is based on conditional retry mechanisms. The goal of such adaptation is to maximize the revenues of the composite service provider (CSP), while meeting the end-to-end deadline specified in the SLA that the CSP has with its clients, even when some of the services used for the composition becomes transiently less responsive (its performability worsens). Motivated by practical applications, an important feature of our model is that it takes into account multiple QoS service parameters, like (heavy-tailed) response time performance and service costs. We use a dynamic programming approach (described in Bellman (2003)) in order to derive the optimal selection policy that will be used at runtime. This policy is determined before the actual deployment of a composite service, and allows optimal service selection at runtime. The policy also specifies the (optimal) moments when retry should be attempted, and which service should be used for a retry. As the policy is determined before runtime, our solution does not require lengthy (and time consuming) re-calculation of the service composition (i.e. adaptation) at runtime. At the same time, the calculated policy takes into account actual time to deadline, current performability of the services, execution costs and expected revenue of the composite service provider.

To illustrate our approach, we observe the orchestrated composite web service which sequential workflow is depicted at Figure 1. This example workflow consists of four abstract services (tasks), and each task maps to a number of concrete services (alternatives), which are deployed by (independent) third-party service providers. When the client’s request is served within the agreed end---to---end deadline, the composite service provider is rewarded by the client. Otherwise, when the deadline is not met, the provider pays a penalty to the client. The orchestrator performs run-time service selection for a given task at decision points illustrated at Figure 1 by A, B, C and D, respectively. The selection or retry decisions at each of these points are taken based on a pre-calculated policy, and takes into account which of the considered services are available at the decision moment, the execution costs, and the remaining time to meet the end---to---end deadline.

When remaining time-to-deadline at decision point C has certain value \( \delta_3 \), the decision policy may specify that the third task is executed by concrete service \( CS^3 \). Besides, the policy specifies that, for \( \delta_3 \), when no response is generated by \( CS^3 \) within the specified period \( \theta \), the orchestrator should make a retry. In order to perform the retry the orchestrator terminates the original request, makes service substitution, and initiates a new service invocation for the same task. The service selected for a retry is also specified in the policy, and is, in given example \( CS^2 \). In other words, when “timer” \( \theta \) expires, the orchestrator “concludes” that the time instance has been reached at which particular service \( CS^3 \) is considered practically (transiently) unavailable. If the response from \( CS^3 \) is generated within the time \( t \leq \theta \), the orchestrator executes the next task within the workflow, and invokes the next concrete service specified by the policy for the given time-to-deadline \( \delta_4 = \delta_3 - t \), e.g. \( CS^4 \). The policy for task 4 and given \( \delta_4 \) indicates that when timer \( \theta' \) expires, \( CS^4 \) should be substituted by \( CS^4 \). Further, Figure 1 shows examples of two requests, noted as request 1 and request 2. We see that, in case of request 1, in total of two retries have been made, for tasks 3 and 4,