Predicting Timing Failures in Web Services

Nuno Laranjeiro, Marco Vieira, and Henrique Madeira

CISUC, Department of Informatics Engineering,
University of Coimbra, Portugal
{cnl,mvieira,henrique}@dei.uc.pt

Abstract. Web services are increasingly being used in business critical environments, enabling uniform access to services provided by distinct parties. In these environments, an operation that does not execute on due time may be completely useless, which may result in service abandonment, and reputation or monetary losses. However, existing web services environments do not provide mechanisms to detect or predict timing violations. This paper proposes a web services programming model that transparently allows temporal failure detection and uses historical data for temporal failure prediction. This enables providers to easily deploy time-aware web services and consumers to express their timeliness requirements. Timing failures detection and prediction can be used by client applications to select alternative services in runtime and by application servers to optimize the resources allocated to each service.

Keywords: web services, timing failures, detection, prediction.

1 Introduction

Web services provide a simple interface between a provider and a consumer and are increasingly becoming a strategic vehicle for data exchange and content distribution [1]. Compositions, which are based on a collection of web services working together to achieve an objective, are particularly important. These compositions are normally defined at programming time as "business processes" that describe the sequencing and coordination of calls to the component web services. Calls between web service consumers and providers consist of messages that follow the SOAP protocol, which, along with WSDL and UDDI, form the core of the web services technology [1].

Developing web services able to deal with timeliness requirements is a difficult task as existing web services technology, programming models, and development tools do not provide easy support for assuring timeliness properties during web services execution. Although some transactional models provide basic support for detecting the cases when operations take longer than the expected/desired time [2], this usually requires a high development effort. In fact, developers have to select the most adequate middleware (including a transaction manager that must fit the deployment environment requirements), produce additional code to manage the transactions, specify their properties, and implement the basic support for timing requirements. Transactions are actually well suited for supporting typical transactional behavior, but they

L. Chen et al. (Eds.): DASFAA 2009 Workshops, LNCS 5667, pp. 182–196, 2009.

are inadequate for deploying simple time-aware services. Indeed, transactions provide poor support for timing failures detection and no support for prediction.

Despite the lack of mechanisms and tools for building time-aware web services, the number of real applications that have to support this kind of requirements is quickly increasing. Typically, developers deal with these by implementing ad-hoc solutions to support timing failures (this is, obviously, expensive and prone to fail). The concept of time has been, in fact, completely absent from the standard web services programming environment. Important features such as timing failure detection and forecasting have been overlooked, although these are particularly important if we consider that services are typically deployed over wide-area or open environments that exhibit poor baseline synchrony and reliability properties. In these environments it is normal for services to exhibit high or highly variable execution times. High execution times are usually associated with the serialization process involved in each invocation, coupled with a high amount of protocol information that has to be transmitted per each payload byte (e.g., the SOAP protocol requires a large amount of data to encapsulate the useful data to be transmitted). This serialization process is particularly important since a given web service can also behave as a client of another service, thus duplicating the end-to-end serialization effort. Variable execution times are essentially related to the use of unreliable, sometimes slow, transport channels (i.e., the internet) for client-server and inter web services communication. These characteristics make it difficult for developers to deal with timeliness requirements.

Two outcomes are possible when considering timing requirements during a web service execution: either the server is able to produce an answer on due time, or not. The problem is that, in both cases the client application has to wait for the execution to complete or for the deadline to be violated (in this case a timing failure detection mechanism must be implemented). However, in many situations it is possible to predict in advance the occurrence of timing failures. In fact, execution history can typically be used to confidently forecast if a timely response will be possible to obtain. Note that, this is of utmost importance for client applications, that can retry or use alternative services, but also for servers, that can use this information to conveniently manage the resources allocated to each operation (e.g., an operation that is predictably useless can be canceled or proceed executing under a degraded mode).

This paper proposes a new programming model for web services deployment that allows online detection and prediction of timing failures (wsTFDP: Web Services Timing Failures Detection and Prediction). By using this model clients are able to express their timeliness requirements for each service invocation by defining a timeout value and an associated confidence value for prediction. When timing requirements are not possible to satisfy (e.g., because the deadline was exceeded or because it will predictably be exceeded) the server responds with a well-known and consistent exceptional behavior. A simple and ready to use programming interface implementing the proposed model is also provided.

The structure of the paper is as follows. The following section presents background and related work. Section 3 presents a high level view of the timing failures detection and prediction mechanism. Sections 4 and 5 detail the design of the detection and prediction components. Section 6 shows how the mechanism can be used in practice, and Section 7 presents some experimental results. Section 8 concludes the paper.