

Middleware Support for Data-flow Distribution in Web Services Composition

* Lucian-Mircea Patcas¹, John Murphy¹, and Gabriel-Miro Muntean²

¹ Performance Engineering Laboratory
Department of Computer Science
University College Dublin
Ireland

Lucian.Patcas@ucd.ie
J.Murphy@ucd.ie

² Performance Engineering Laboratory
School of Electronic Engineering
Dublin City University
Ireland

munteang@eeng.dcu.ie

Abstract. Composition of Web services helps to lower the time-to-market of service-based applications by reusing the functionality provided by services that are already deployed at geographically distributed locations. Due to the diversity of services, the client acceptance of new applications is determined more and more by non-functional aspects, such as Quality of Service, or cost. The throughput, response time, and communication cost of composite services depend on the characteristics of each component service involved, as well as on the manner these components are tied together. Research has shown that distributed data-flow models can offer better performance and lower communication costs in service composition than centralized models. However, an impediment towards data-flow distribution in Web services composition is that the component services cannot exchange data directly without central mediation. We propose therefore a non-intrusive approach for achieving data distribution among Web services that are engaged in composition, bringing minimal extensions to the underlying middleware and not tightening the coupling between them.

1 Introduction

The next expected step in large-scale software development is to achieve better intra- and inter-enterprise integration of applications, as fully integrated enterprise applications tend to be replaced by business networks in which each participant offers specialized services to the others [1]. This trend has led to the development of service-oriented technologies.

* **Acknowledgement** The support of the Advanced Technology Research Programme (ATRP) from the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged by the authors.

Web services are the practical implementation of the *service-oriented computing* [2] model. In the service-oriented approach, higher level services (*composite services*) are created by composing existing lower level services (*component services*) that were deployed at different physical locations and on heterogeneous platforms. Languages used for describing the composition of Web services, such as Business Process Execution Language (BPEL) for Web Services [3], employ centralized control-flow and centralized data-flow to tie services together. The centralized data-flow approaches generally perform poorer than decentralized ones in terms of throughput, response time, and communication cost because composite services are potential bottlenecks [4–6]. If component services were able to exchange data between them without all that data passing through a central coordinator, Web services composition would greatly benefit in terms of performance.

Our main objective in this paper is to propose a non-intrusive approach for achieving direct interaction between component Web services in order to support data-flow distribution in Web services composition. The approach aims at bringing minimal extensions to the existing infrastructure, while not tightening the coupling between the component services. In section 2 we present the benefits that the decentralized data-flow model can generally bring to service composition. Section 3 describes work related to current approaches for decentralizing the composition of Web services, and then our proposed approach is highlighted in section 4. Finally, conclusions and possible future work are presented in section 5.

2 Data-flow Distribution in Service Composition

The influence that different data-flow models have on performance of composite services is discussed in [5]. *Centralized control-flow centralized data-flow* and *centralized control-flow distributed data-flow* models are analyzed and compared, considering the aggregated cost and response time. Results show that the aggregated cost of the distributed data-flow model is no greater than the aggregated cost of the centralized data-flow model. Additionally, the response time of the distributed data-flow model is smaller than that of the centralized one when the following condition is met: the network bandwidth between component services is not lower than a network bandwidth of the communication channels between the composite service and the component services.

Figure 1 schematically presents both the centralized and the decentralized data-flow models in the context of centralized control-flow. In the centralized data-flow model (fig. 1(a)) all the data needed to be exchanged between component services passes through the central service. This way, the composite service can become a bottleneck when data-intensive services are composed. The distributed data-flow model (fig. 1(b)) relaxes the load of composite services, and uses resources of the networks that connect the component services by enabling them to exchange data directly. Therefore, data-flow distribution can improve

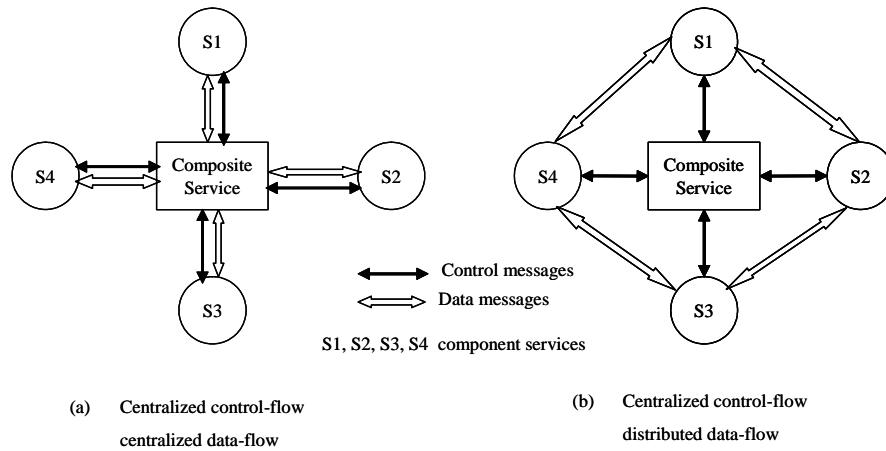


Fig. 1. Data-flow models in service composition

the overall performance of composite services in terms of throughput, response time and total communication cost.

3 Related Work

The area of Web services composition benefits from the work done on workflow systems [7]. However, little work has been done in what concerns the distribution of control and data-flow in the composition of Web services. In this section we present the results we have found on this aspect.

Chafle et al [4] and Nanda et al [6, 8] present a method for decentralizing the execution of composite Web services, along with related issues. In [6] the authors propose an algorithm for partitioning centralized BPEL descriptions into smaller parts that are executed by distributed BPEL engines. The partitioning policy uses program dependence graphs to detect data dependencies, and tries to minimize communication costs and maximize the throughput of composite services by reducing the amount of data-flow between different partitions. The model used is distributed control-flow distributed data-flow. Experimental results obtained highlight a better throughput of the decentralized composition against the centralized one. A drawback of their approach is the necessity for the code of each partition to be deployed in a BPEL engine that runs on the same or on a topologically-close application server as the service it invokes. Therefore, dynamic composition of Web services is hard to achieve using this method. Moreover, it lowers the maintainability and evolvability of composite services, as every change in their workflow description must be followed by code repartitioning, and redeployment of the newly obtained partitions.

4 Middleware Support for Data-flow Distribution in Web Services Composition

The primary goal in the Web Services field is interoperability achievement among distributed and heterogeneous systems, while still preserving the loose coupling between them. Thus, the component services have no knowledge of each other and, consequently, they cannot exchange data directly without central mediation. In most cases, they are intended only to process requests and to respond accordingly to the clients that invoke them, therefore being passive in what concerns the control-flow required by the service composition logic. This is a major problem that stands in the way of achieving data-flow distribution among component Web services.

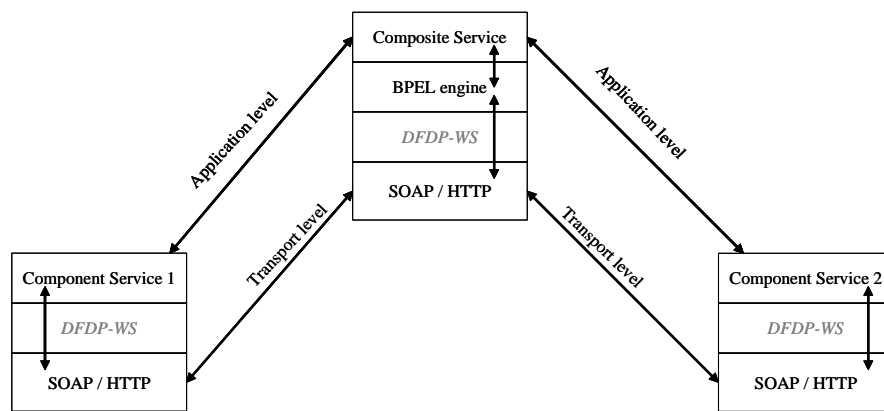


Fig. 2. Middleware configuration for centralized data-flow

For tackling this problem, we propose a minimal extension to the run-time infrastructure. The extension consists of the integration of a protocol that we call *Data-Flow Distribution Protocol for Web Services* (DFDP-WS) into the Web services stack. This protocol should be able to encapsulate both control-flow information and data in the messages it carries. DFDP-WS is to be interposed between the application level protocols and the transport level protocols. Depending on the specific needs encountered at run time, it can be enabled or disabled. When centralized data flow is desired, DFDP-WS is not used or plays neutral roles (fig. 2). When the data flow is distributed (fig. 3), DFDP-WS is activated and assures the infrastructure for component services to exchange data directly one with another.

The component services do not need to implement extra functional features for using our approach. The constraint however is the need for each of the component services to be deployed into containers that implement our proposed middleware support. This brings only a minimal extension to the existing infrastructure, as DFDP-WS is a lightweight protocol. The method presented in [6],

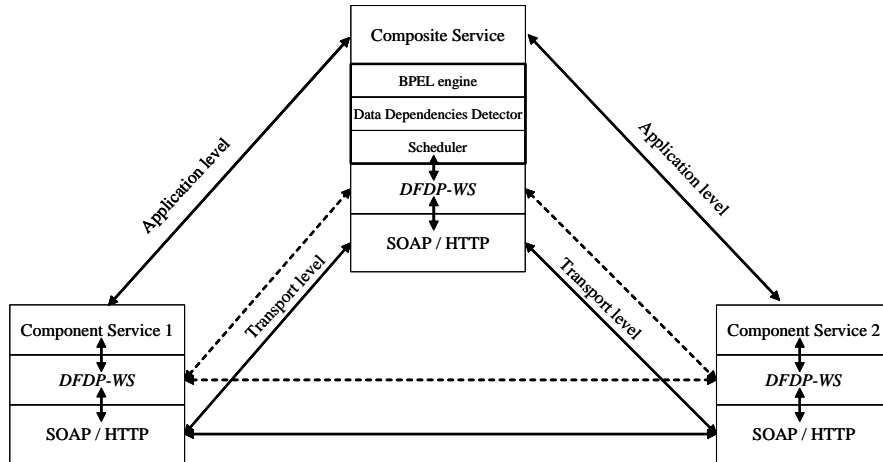


Fig. 3. Middleware support for decentralized data-flow

and described in section 3, requires the component services to have BPEL engines around. BPEL engines are expected to be more complex than DFDP-WS. Nevertheless, as the direct interactions between the component services are carried out by the middleware, their loose coupling is preserved at the application level.

We do not focus on detection of data-flow dependencies in the centralized BPEL descriptions. However, a detection algorithm similar to the one described in [6] can be moved from the application level to middleware. Data-flow dependencies can be detected statically or at run-time, and the corresponding service invocation sequences can be generated by a scheduler (fig. 3). Both the data dependencies detector and the scheduler can be implemented as extensions to the BPEL engines. The optimization process is therefore transparent to the application level.

Web Services Coordination (WS-C) [9] is an extensible framework for providing protocols that enable the service-based applications to coordinate their distributed component services. As WS-C allows for the publication of new protocols, the extension we propose can fit non-intrusively into the existing frameworks, provided that these frameworks offer WS-C support.

5 Conclusion and Future Work

Because non-functional aspects of (composite) Web services, such as Quality of Service and cost, play a key role in their selection by clients, we attempt to propose a way to improve their performance in terms of throughput, response time, and aggregated cost. The key points we address are: i) the need for a middleware support for data-flow distribution in Web Services composition, and ii) the way we can integrate this support into the existing infrastructure with mini-

mal extensions and non-intrusiveness. The middleware enhancement we propose consists of a lightweight protocol (DFDP-WS) that is capable of transporting both control and data messages. DFDP-WS brings minimal extensions to the middleware and still preserves the loose coupling between the component services. The WS-C framework might be a solution for integrating the proposed protocol into the run-time infrastructure non-intrusively.

As future work we intend to develop a specification for the DFDP-WS protocol and to investigate the feasibility of its integration into Web application containers, by implementing a proof-of-concept. We also intend to compare the performance-related results obtained when using DFDP-WS with those obtained when the centralized composition was utilized.

References

1. Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S.: The next step in web services. *Communications of the ACM* **46** (2003) 29–34
2. Huns, M.N., Singh, M.P.: Service-oriented computing: Key concepts and principles. *IEEE Internet Computing* **9** (2005) 75–81
3. Business Process Execution Language for Web Services. Technical specification, IBM–DeveloperWorks
<http://www-128.ibm.com/developerworks/library/specification/ws-bpel>.
4. Chafle, G.B., Chandra, S., Mann, V., Nanda, M.G.: Decentralized orchestration of composite web services. In: *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, New York, NY, USA, ACM Press (2004) 134–143
5. Liu, D., Law, K.H., Wiederhold, G.: Analysis of integration models for service composition. In: *WOSP '02: Proceedings of the third international workshop on Software and Performance*, New York, NY, USA, ACM Press (2002) 158–165
6. Nanda, M.G., Chandra, S., Sarkar, V.: Decentralizing execution of composite web services. In: *OOPSLA '04: Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications*, New York, NY, USA, ACM Press (2004) 170–187
7. van der Aalst, W.M., Dumas, M., ter Hofstede, A.H.: Web service composition languages: Old wine in new bottles? In: *Euromicro Conference, 2003. Proceedings. 29th.* (2003) 298–305
8. Nanda, M.G., Karnik, N.: Synchronization analysis for decentralizing composite web services. In: *Proceedings of the ACM Symposium on Applied Computing (SAC)*. (2003)
9. Web Services Coordination. Technical specification, IBM–DeveloperWorks
<http://www-128.ibm.com/developerworks/library/specification/ws-tx>.