Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices

Gregory Hackmann, Mart Haitjema,
Christopher Gill, and Gruia-Catalin Roman

Dept. of Computer Science and Engineering, Washington University in St. Louis

Abstract. The Business Process Execution Language (BPEL) has become the dominant means for expressing traditional business processes as workflows. The widespread deployment of mobile devices like PDAs and mobile phones has created a vast computational and communication resource for these workflows to exploit. However, BPEL so far has been deployed only on relatively heavyweight server platforms such as Apache Tomcat, leaving the potential created by these lower-end devices untapped. This paper presents Sliver, a BPEL workflow process execution engine that supports a wide variety of devices ranging from mobile phones to desktop PCs. We discuss the design decisions that allow Sliver to operate within the limited resources of a mobile phone or PDA. We also evaluate the performance of a prototype implementation of Sliver.

1 Introduction

In today's world, there is an ever-growing need for collaboration among teams of people on complex tasks. The *workflow* model offers a powerful representation of groupware activities. This model is defined informally as "the operational aspect of a work procedure: how tasks are structured, who performs them, what their relative order is, how they are synchronized, how information flows to support the tasks and how tasks are tracked" [1]. In other words, workflow systems coordinate and monitor the performance of tasks by multiple active agents (people or software services) towards the realization of a common goal.

Many traditional business processes — such as loan approval, insurance claim processing, and expense authorization — can be modeled naturally as workflows. This has motivated the development of Web standards, such as the Business Process Execution Language [2] (BPEL), which describe these processes using a common language. Each task in a BPEL process is represented as a service that is invoked using the Simple Object Access Protocol [3] (SOAP). A centralized BPEL server composes these services into complex processes by performing an ordered series of invocations according to the user's specifications. Because BPEL builds on top of standards like XML and SOAP that are already widely deployed, it has been accepted readily in business settings.

The ubiquity of inexpensive mobile and embedded computing devices, like PDAs and mobile phones, offers a new and expanding platform for the deployment and execution of collaborative applications. In 2004, over 267 million Java-capable mobile phones were deployed worldwide, and Sun estimates that up to

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 503–508, 2006.

1.5 billion will be deployed by the end of 2007 [4]. Though each device is individually far less powerful than a standalone server, their aggregate computation and communication potential is remarkable, and has yet to be fully realized.

Many collaborative applications, such as those described in [5], could incorporate such devices advantageously. These applications can benefit greatly from Web standards like BPEL and SOAP. By defining a common language for inter-service interactions and data flow, these standards encourage the composition of simple services into powerful distributed applications.

Unfortunately, these applications pose several important challenges that the current state-of-the-art in SOAP and BPEL systems cannot meet. First, typical mobile devices feature severely constrained hardware requiring a very lightweight software infrastructure. Second, in the absence of a stable Internet connection, it may be impossible, impractical, or too expensive for mobile devices to contact centralized servers. Finally, wireless network links among mobile devices may be disrupted frequently and unpredictably. These challenges necessitate a lightweight, decentralized Web service middleware system which can perform on-the-fly replanning, reallocation, and/or reconfiguration in the face of network failure. Addressing these issues is a significant software engineering challenge.

In this paper, we describe Sliver, our first milestone in this long-term effort. Sliver supports the execution of SOAP services and BPEL processes on mobile devices like mobile phones and PDAs. Because Sliver builds on existing Web standards, it can be used in conjunction with a wide array of existing development tools. We emphasize that Sliver is not intended to *replace* existing SOAP and BPEL middleware: rather, it extends the Web services paradigm to new devices which did not previously support it. In Section 2, we discuss the fundamental characteristics of mobile devices that compel a new kind of middleware. Section 3 provides an overview of Sliver's architecture. The resulting prototype implementation is evaluated in Section 4. Finally, we give concluding remarks in Section 5.

2 Problem Statement

Today, developers can choose from a wide variety of support platforms for SOAP services and BPEL processes. Unfortunately, there are several practical issues that prevent existing SOAP and BPEL middleware packages from being deployed on mobile devices. The first issue is the combined footprint of the middleware and its support layers. For example, the open-source ActiveBPEL [6] engine depends on the Java Standard Edition 1.4.2 runtime and Apache Tomcat [7] application server, with a total footprint of 92 MB of disk space and 22 MB of RAM. While this requirement is reasonable for desktop computers and servers, only a handful of the highest-end mobile phones and PDAs can support systems with such large footprints.

The second issue is that these middleware frameworks and their support layers are often designed with Java 2 Standard Edition (J2SE) in mind. Generally, J2SE runtimes are not available for mobile devices. These devices support a more limited Java runtime, such as one based on the Mobile Information Device Profile (MIDP)

standard. Such runtimes support only a fraction of the features provided by a full J2SE runtime. Among other features, MIDP 2.0 does not offer most of J2SE's abstract data structures; its support for runtime reflection is minimal; and it features a unified API for file and network I/O that is incompatible with J2SE's I/O APIs.

Finally, existing BPEL systems typically use HTTP for all communication between hosts. However, this protocol is not a reasonable choice for many mobile devices. Because of network restrictions, many mobile devices (such as most mobile phones) cannot accept incoming TCP/IP sockets, and hence cannot serve HTTP requests. Incoming requests are often restricted to less-conventional transports, such as SMS messages, which current systems do not support.

Thus, if a SOAP or BPEL execution engine is to be deployed on mobile devices, it must embody three major traits: (1) it must have a suitably small storage and memory footprint, including all the libraries on which it depends; (2) it must depend only on the Java APIs that are available on all devices; and (3) it must support a wide variety of communication media and protocols flexibly. In the next section, we discuss how these traits influenced our design and implementation of Sliver.

3 Design and Implementation

Sliver exhibits several architectural decisions which fit the traits described above. Sliver uses a pluggable component architecture, as shown in Figure 1. This architecture provides a clean separation between communication and processing. Communication components can therefore be interchanged without affecting the processing components, and vice versa. In place of a heavyweight, general-purpose XML parser, Sliver uses a series of hand-written parsers developed using the lightweight kXML [8] and kSOAP [9] packages. These packages are designed with mobile devices in mind: they have a small combined footprint (47 KB of storage space) and operate on most available Java runtimes.

Excluding the communication components, Sliver is implemented using the features that J2SE, Java Foundation Profile, and MIDP 2.0 have in common. Sliver can be deployed on devices which support any of these standards, which includes most mobile phones and PDAs sold today. Sliver's streamlined API allows users to deploy an embedded SOAP or BPEL server in under 15 lines of Java code. Further information on Sliver's architecture and implementation, including sample code, can be found in [5].

4 Evaluation

Sliver currently supports BPEL's core feature set and has a total code base of 114 KB including all dependencies (excluding an optional HTTP library). Sliver supports all of the basic and structured activity constructs in BPEL, with the exception of the *compensate* activity, and supports basic data queries and transformations expressed using the XPath language [10]. Sliver also supports the use of BPEL Scopes and allows for local variables and fault handlers to be defined within them. However, Sliver does not currently support some of BPEL's

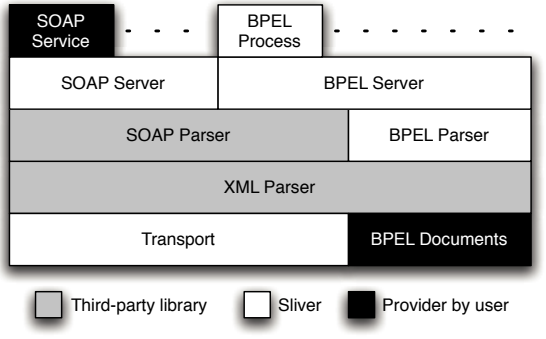

Fig. 1. The architecture of the Sliver execution engine

most advanced features, including *Serializable Scopes* and *Event Handlers*. In future work, we will extend Sliver to support these features.

In order to provide an adequate evaluation of Sliver, it is important not only to benchmark its performance against an existing BPEL engine, but also to examine to what extent the expressive power of BPEL is preserved by Sliver. A framework has been proposed which allows for the analysis of workflow languages in terms of a set of 20 commonly reoccurring workflow patterns [11]. A study of the BPEL language in terms of this framework shows that BPEL can support in full 16 of these 20 workflow patterns, and partially supports one other pattern [12]. Sliver currently supports all but 2 of these 17 patterns.

Our performance benchmark consists of 12 of the 20 patterns listed in [11]. The Multi-Merge, Discriminator, and Arbitrary Cycle patterns are excluded because BPEL does not support them. Sliver also does not presently support all of the BPEL features used by the one of the Multiple Instances patterns and the Interleaved Parallel Routing pattern. The Multiple Instances without Synchronization pattern is not a practical benchmark, since it creates child processes which may continue executing even after the parent process has completed. Finally, the Deferred Choice and Milestone patterns are non-deterministic and therefore do not make practical benchmarks.

In Figure 2, we compare Sliver's execution of these 12 patterns versus the combination of ActiveBPEL 2.0.1.1 and Apache Axis 1.4, popular open source engines for BPEL and SOAP respectively[1]. Our test platform for this comparison is a desktop computer equipped with a 3.2 GHz Pentium 4 CPU, 512 MB of RAM, Linux 2.6.16, and Sun Java 1.5.0_07. Both ActiveBPEL and Apache Axis are hosted on Apache Tomcat 5.5.15. Additionally, this figure shows Sliver's performance running these processes on a Dell Axim X30 PDA which is equipped with a 624 MHz XScale CPU, Windows Mobile 2003, and IBM WebSphere Micro

[1] We once again emphasize that Sliver is not intended to replace feature-rich SOAP and BPEL engines on capable hardware, but rather to support the execution of BPEL processes on resource-limited devices. Our comparison is only intended to provide a metric for acceptable performance.

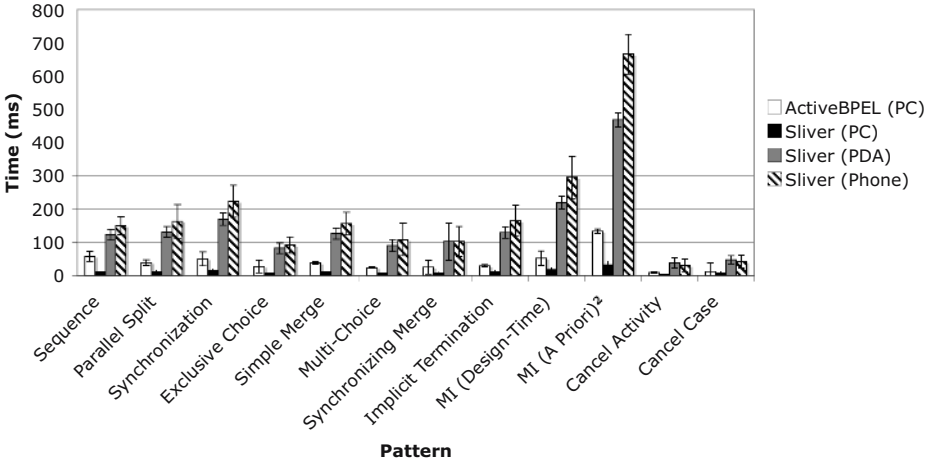

Fig. 2. The cost of executing BPEL patterns; results are the mean of 100 runs

Environment 5.7; and on a Nokia 6682 mobile phone which is equipped with a 220 MHz ARM9 CPU, Symbian OS 8.0a, and a bundled MIDP 2.0 runtime. To isolate the cost of process execution from network delays, the BPEL process and SOAP service are colocated.

Where not noted otherwise, 105 runs of each benchmark were used to generate Figure 2. The first few runs of each benchmark have unusually high costs (often 5 to 10 times the mean) due to class loading, etc. For this reason, we discarded the first 5 runs of each benchmark and computed the mean of the remaining 100 runs. The error bars indicate the standard deviation.

These results demonstrate that it is feasible to deploy BPEL processes on limited hardware. Even on the resource-limited PDA and phone platforms, the cost of carrying out most processes is on the order of 100 ms. (The only exceptions are the Multiple Instances patterns, which contain loops that make them inherently slower than the other patterns.) As noted above, in order to isolate the costs of the BPEL engine, we evaluated processes which invoke a trivial SOAP service located on the same host. Realistically, the cost of executing non-trivial SOAP services (including network delays) is expected to dwarf the cost of supporting the BPEL process in Sliver.

5 Conclusion

In this paper, we have presented Sliver, a middleware engine that supports BPEL process execution on mobile devices. Our design flexibly supports many different communication protocols and media, while still maintaining a minimal footprint.

[2] Due to the complexity of the MI (A Priori) pattern, and very limited hardware resources, the Nokia 6682 is unable to perform 100 runs of this benchmark consecutively. 50 consecutive runs of this pattern were used on the Nokia platform.

Sliver uses a series of small, hand-written parsers in place of a heavyweight, fully-validating XML parser. These parsers keep Sliver's code size and runtime overhead suitably low for deployment on even the most resource-limited mobile devices. In its current implementation, which is available as open-source software at [13], Sliver can host many useful processes on hardware ranging from mobile phones to desktop computers. In future work, we plan to address the remaining BPEL compliance issues and consider ways to further modularize Sliver.

The development of middleware engines like Sliver is an important step toward the long-term goal of bringing groupware to mobile devices. Other important challenges — including task allocation, data distribution, and user interface design — still remain. Nevertheless, Sliver's runtime performance demonstrates that today's mobile devices are already capable of hosting sophisticated groupware applications, and that this ultimate goal is practical as well as desirable.

Acknowledgment. This research is supported by the NSF under grant number IIS-0534699. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily represent the views of the research sponsors.

References

1. Wikipedia: Workflow. *http://en.wikipedia.org/wiki/Workflow* (2006)
2. OASIS Open: OASIS web services business process execution language (WSBPEL) TC. *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel* (2006)
3. Box, D., et al.: Simple object access protocol (SOAP) 1.1. Technical Report 08 May 2000, W3C (2000)
4. Ortiz, C.E.: J2ME technology turns 5! *http://developers.sun.com/techtopics/mobility/j2me/articles/5anniversary.html* (2004)
5. Hackmann, G., Haitjema, M., Gill, C., Roman, G.C.: Sliver: A BPEL workflow process execution engine for mobile devices. Technical Report WUCSE-06-37, Washington University, Department of Computer Science and Engineering (2006)
6. ActiveBPEL LLC: ActiveBPEL engine. *http://www.activebpel.org/* (2006)
7. Apache Software Foundation: Apache tomcat. *http://tomcat.apache.org/* (2006)
8. Haustein, S.: kXML 2. *http://kxml.sourceforge.net/kxml2/* (2005)
9. Haustein, S., Seigel, J.: kSOAP 2. *http://ksoap.org/* (2006)
10. Clark, J., DeRose, S.: XML path language (XPath) version 1.0. Technical Report 16 November 1999, W3C (1999)
11. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
12. Wohed, P., et al.: Pattern based analysis of BPEL4WS. Technical Report FIT-TR-2002-04, Queensland University of Technology (2002)
13. Hackmann, G.: Sliver. *http://mobilab.wustl.edu/projects/sliver/* (2006)