

ESB Federation for Large-Scale SOA

Françoise Baude, Imen
Filali, Fabrice Huet,
Virginie Legrand, Elton
Mathias, Philippe Merle^{*}
, Cristian Ruz
INRIA Sophia-Antipolis
Méditerranée
I3S CNRS - UNS
2004 rte des Lucioles, B.P. 93,
FR-06902 Sophia-Antipolis,
France
first.last@inria.fr

Reto Krummenacher,
Elena Simperl
Semantic Technology Institute,
University of Innsbruck
Technikerstr. 21a
A-6020 Innsbruck, Austria
first.last@sti-innsbruck.at

Christophe Hammerling,
Jean-Pierre Lorre
EBM WebSourcing
4, Rue Amélie
FR-31000 Toulouse, France
first.last@
ebmwebsourcing.com

ABSTRACT

Embracing service-oriented architectures in the context of large systems, such as the Web, rises a set of new and challenging issues: increased size and load in terms of users and services, distribution, and dynamicity. A top-down federation of service infrastructure support that we name “service cloud” and that is capable of growing to the scale of the Internet, is seen as a promising response to such new challenges. In this paper, we define the service cloud concept, its promises and the requirements in terms of architecture and the corresponding middleware. We present some preliminary proofs of concept through the integration of a JBI-compliant enterprise service bus, extended to our needs, and a scalable semantic space infrastructure, both relying on an established grid middleware environment. The new approach offers service consumers and providers a fully transparent, distributed and federated means to access, compose and deploy services on the Internet. Technically, our contribution advances core service bus technology towards the service cloud by scaling the registries and message routers to the level of federations via a hierarchical approach, and by incorporating the communication and coordination facilities offered by a global semantic space.¹.

Categories and Subject Descriptors

C.2.4 [Distributed Systems]: Distributed applications;
D.2.11 [Software Engineering]: Software Architectures

^{*}INRIA Lille - Nord Europe

¹The presented work is funded by the EU FP7 NESSI strategic integrated project SOA4All; <http://www.soa4all.eu>.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'10 March 22-26, 2010, Sierre, Switzerland.

Copyright 2010 ACM 978-1-60558-638-0/10/03 ...\$10.00.

Keywords

Service Oriented Architecture, ESB, JBI, Federation, semantic storage, grid and cloud computing

1. INTRODUCTION

Scale-out, outsourcing, Software as a Service are now popular terms reflecting a shift in the way enterprises reorganize their IT services while re-focusing on their primary business. This for instance, translates in enterprises offering a handful of high-value added services running in-house, but relying on a large and open bunch of out-sourced and shared utility services, coupled along service-oriented architectures. Besides, Web 2.0, mash-ups, social networking are rising paradigms that also let users reuse, remix and enrich existing resources and components to new and potentially higher-level applications, further exposed as new services populating the Internet of Services arena. Both cases need very similar technological solutions in 1) outsourced, large-scale and flexible computing or storage support grounding in grid and cloud computing, and 2) integrated multi-facets solutions to discover, compose, deploy, run, monitor, publish, and annotate services that ground in Enterprise Service Bus, SOA and Semantic Web technologies.

The EU-funded project SOA4All was launched in 2008 and aims at contributing to a complete, large-scale solution that addresses these novel needs. One expected result of the project is an even more ambitious paradigm often acknowledged as the Service Web that we establish through the service cloud. A service cloud must be able to span the whole Internet, to allow end-users to invoke and coordinate external services, potentially executed anywhere on the globe. In the remainder of the paper, we thus use the service cloud as motivator for the design of a space where end-user can compose, use and execute services at Internet-scale.

The contributions focus on the infrastructure underlying the service cloud, and are thus strongly middleware oriented. The paper presents the concept of a *federation of ESBs* as a means to realize the service cloud. From a technical point of view, this translates into the identification of what are the requirements for such a middleware, concretized by a prototype. The federation integrates a global information space containing semantically annotated RDF information

about single or compound services. It also requires a solution to have the ESBs part of the federation be effectively inter-connected. To fulfill those two requirements, we rely on grid and cloud technology. Indeed, the information space and the routing layer have to be deployable anywhere on the Internet, i.e., this requires to harness machines from computing grids and/or computing clouds. The prototype we describe has been obtained by relying on an Enterprise Service Bus (ESB), namely the open source ObjectWeb 2 PETALS ESB,² that in itself offers already a distributed ESB (DSB for short). Being distributed, PETALS facilitated the task of promoting it as a federated ESB. However, the technical extensions that we present here after are applicable whenever other ESBs than PETALS have to be inter-connected by the federation. Both the overlay network supporting the information space, and the routing layer for inter-ESB messaging are based upon the open source OW2 ProActive Grid technology for achieving full portability, distribution and scalability of the federation.³ The combination of PETALS plus ProActive yields our federated ESB, which acts as the core infrastructure of the service cloud.

This paper is organized as follows: in Section 2 we define the challenging concept of service cloud, present existing solutions that could support this paradigm, and briefly describe some of the very few related works. In Section 3 we explain the requirements of bringing an Enterprise Service Bus to the service cloud, and present our architecture design and prototype for the service cloud in Section 4. Finally, we conclude and point to some further work.

2. MOTIVATION AND RELATED WORK

2.1 The Challenge to Take Up: The Service Cloud

The trends in service computing point far beyond the recent changes around the concept of “Software as a Service”. In fact, we are entering the era of “Everything as a Service” (XaaS) by binding humans, objects, and resources such as storage or computing devices as services to a global service delivery infrastructure. Information and computation of any kind will be exposed and made available through services and disappear behind the services interfaces. The services themselves disappear in the Web that becomes the platform, as public, open and distributed alternative to private legacy systems.

As a consequence, services turn into utilities. Their functionality and the offered quality of service become the decisive characteristics, rather than the endpoint location or provider. Through the cloud, distributed functionalities are then provided by the community, and no longer by individual nor dedicated providers [14]. In such scenarios, average users become prosumers (both providers and consumers [15]) of services in the cloud, leading to thousands of new services and business profiles created almost on-the-fly. Consequently, the number of services providing, respectively consuming entities reaches dimensions that will soon be comparable to the number of users and contributors to the World Wide Web. This number is only exceeded by the amount of data and computing resources that are exposed through service interfaces.

²<http://petals.ow2.org>

³<http://proactive.inria.fr>

Another force into action translates the notion of a service ecosystem: major actors in today IT, like Internet service providers (ISPs) for instance, continuously enlarge their service offerings, aiming to build communities of users that identify themselves as being member of the community. All services shared by a community are living and are being composed in the associated marketplace that the actor delimits. However, partnerships between such actors may arise as coalitions, translating directly in the need to interconnect their existing service marketplaces into peered or, more hierarchically structured federations, thus yielding to an eventually unique, however not flat global service cloud.

In order to manage this wealth of resources, there is a significant need for automation. Without automation, it would no longer be possible to offer the required functionalities of a service delivery platform, such as discovery, adaptation, negotiation and execution. Automation in turn relies heavily on metadata in order to create more abstracted views onto real-life objects. Semantic technologies are a key component for such work and were successfully applied in many recent projects on software and services to lift services and their descriptions to a level of abstraction that deals with machine-understandable conceptualizations, and that decreases the dependency upon human users⁴. Thanks to semantics, it is now possible to facilitate and automate the management of services’ entire life-cycles in terms of creation, discovery, composition, configuration and invocation.

The service cloud embodies both, a market place for the offering of services at Internet-scale, and an infrastructure and middleware – also referred to a global service delivery platform – that supports the provisioning, consumption and monitoring of the available services. Such a middleware is needed as a scalable layer for the communication and service coordination in the cloud, and the storage and manipulation of service annotations. While the notion of service cloud is still a rather abstract concept, we present our on-going work on bringing traditional service bus technology to Internet-scale. Service buses deliver the core functionality for service computing in corporate settings, but do not suffice in terms of dynamics, openness and scalability, required for Service-Oriented Architecture on the Web. Finally, one may also argue why not relying simply on the Web, as the underlying infrastructure? The Web can enable the interconnection of services and creation of Service-Oriented Architectures but relying on ESB-based technology ensures service prosumers to get more control over the service usage (e.g., enforce policies for using the service, controlling its access, monitor its performances). The architecture that we present in Section 4 still grounds in ESB technology that is enhanced with state-of-the-art technologies in distribution and semantics to provide the core infrastructure services such as storage, monitoring, deployment and communication for a virtually global Internet-scale service ecosystem.

2.2 Related Work

In this section we discuss, although only a few exist, some related works that pertain to provide end-users the notion of a service cloud, thanks to a supporting underlying mid-

⁴E.g., ASG, <http://asg-platform.org>; DIP, <http://dip.semanticweb.org>; INFRAWEB, <http://www.infraweb-eu.org>; SHAPE, <http://www.shape-project.eu>; STASIS, <http://www.stasis-project.net> or TripCom, <http://www.tripcom.org>.

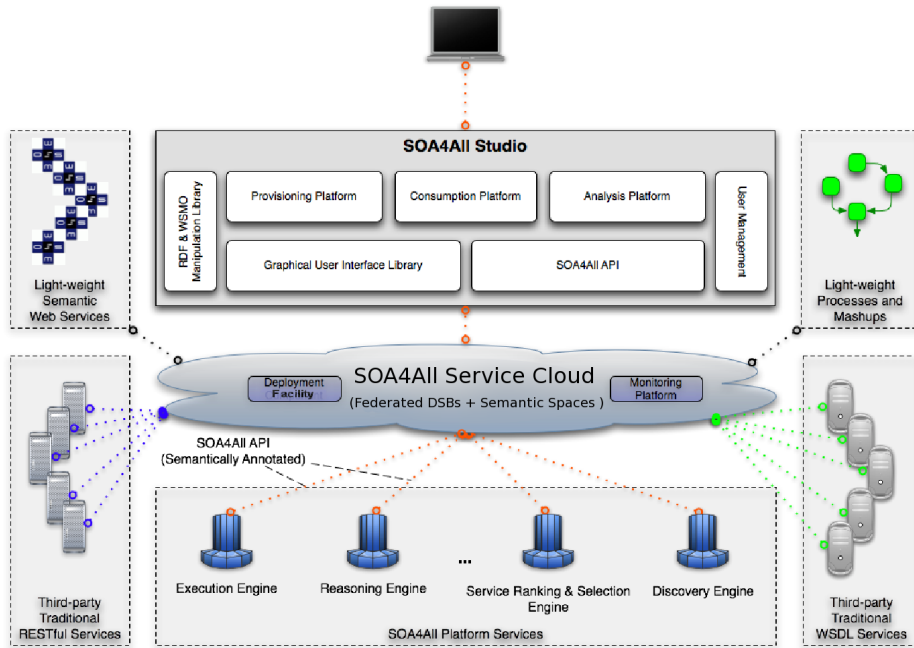


Figure 1: SOA4All overall architecture.

dleware that integrates services at large-scale.

The federated ESB from the EXDI (ESCB XML Data Integration Solution) [17] project supports simple, flexible, secure and scalable data exchanges within ESCB (European System of Central Banks), a world-wide federation of national banks such as the Banque de France or the European Central Bank. This federated bus serves as a basis for a transparent way to perform data exchanges (that are modelled as re-usable business process and stored within the so-called *EXDI registry*) between banks. This business-based architecture can be compared to a service cloud, specialized in the bank domain, and meet the challenges we have to address in SOA4All, i.e., scalability, availability, or trust to rename a few. However, this is a very particular business domain and the architecture seems to be very tied to bank activities and is not general enough to be opened to the general, non-technical Internet users as targeted by SOA4All; meeting the “for All” of SOA4All.

As stated in [16], where authors attempt to give a clear definition of cloud computing, the concept of cloud can be divided into several categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In this paper, we talk about an architecture that is able to deliver services to the user without having the user to care about how services are delivered. Thus, we position our service delivery cloud in the SaaS concept. There are numerous SaaS platforms such as Google Apps, force.com or the Microsoft Azure platform [5]⁵. One of the more suited for general purpose computing and composition is Microsoft Azure [1]. This brand new platform provides means to run Windows applications and storing their data in the cloud. However, the cloud infrastructure (including cloud technologies such as service execution or data storage) is localised

on Microsoft servers (all located in Microsoft data centers) even if it can be accessed via the Internet. In particular, the architecture relies on the so-called *Microsoft .NET Services* component that offers an ESB-like distributed infrastructure to cloud-based and local applications. The distribution of services is based on a service bus, allowing .NET applications to be exposed as Web services in order to have them inter-operate. In the upcoming release of Azure, Microsoft foresees to add a federation feature between private clouds to share data between organizations [7]. Those private clouds are comparable to the ESBs that take part in our proposed federation, however, the SOA4All federation is designed in a way that makes it totally open and agnostic with respect to the effective locations it is deployed on, and with respect to the technology to let services interact to get a large-scale SOA thanks to an ESB-based approach.

All the quoted platforms are enterprise-centric even if they propose a facility for outsourcing to companies or to interconnect companies’ service systems; consequently we argue that they follow a bottom-up approach. On the contrary, the goal we pursue for the service cloud is to install an open third-party solution for federating different ESBs in a totally-transparent manner to the end-users (prosumers). We seek to interconnect many services, existing SOAs and corresponding infrastructures into something that would approximate a truly enterprise-spanning SOA, but in a global and top-down way. From the end-user point of view, such a service cloud would ensure maximised flexibility in the action of composing services out of existing ones. Corresponding tools, such as the SOA4All Studio, key part of the SOA4All overall architecture as shown in Figure 1, are needed in order to make the infrastructure accessible to users. However, this is not the matter of this paper, as the focus is on the associated middleware aspects.

⁵<http://www.microsoft.com/azure/windowsazure.mspix>

3. REQUIREMENTS FOR AN ESB FEDERATION

In this section, we introduce ESB architectures and technical elements and show why ESB and DSB technologies are a good basis for enabling a large-scale service infrastructure.

3.1 ESBs and DSBs

An Enterprise Service Bus (ESB) is a sophisticated infrastructure to host SOA applications, which provides an interconnection between heterogeneous service providers and service consumers [13].

Both the inner architecture of the ESB and the applications it supports can be mapped onto the traditional publish-find-bind SOA pattern by offering a service registry to publish and to find the services. It must be clear, however, that two kinds of registries are usually part of the picture. Services can be published in open Internet-scale registries like UDDI-compliant servers to play a role of external services. It might also be possible that some applications wrapped as services (as those requested by the bus to perform correctly) must be only publicized internally, i.e., only accessible from within the ESB and consequently stored in some internal or private registry within the bus. To hide heterogeneity of protocols for accessing services (external or not), the ESB acts as a broker between clients and server applications/services, meaning that its duty is to bring client requests to the appropriate service provider. To this aim, each service involved in an SOA application deployed on the ESB is accessed through a software artifact (known as a Binding Component (BC) in case of JBI-compliant buses) that is capable of translating the request's form into one understandable by the targeted service. As a consequence, deploying a SOA application onto an ESB triggers the creation of such a component for each involved type of service. Its corresponding internal address (aka service end-point) is registered in the internal registry of the bus, making the service endpoint reachable by all the bus service consumers.

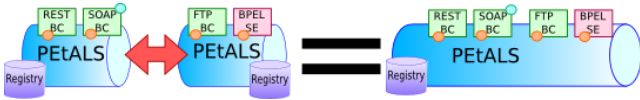


Figure 2: Distributed PEtALS ESB architecture.

In a distributed ESB architecture, some software artifacts can be hosted on different nodes (named containers) and service engines may be either replicated or distributed over containers. Since such a DSB gives a unified vision of the bus, it must be able to look for any binding component or service engine address, hosted or not on the same container. This requires the use of a technical registry that can be either distributed or replicated. Figure 2 presents the unified and transparent view of a DSB, illustrated on the PEtALS case (right side) from a client point of view.

3.2 Federations: Beyond ESB and DSB

According to the SOA4All vision, the service cloud infrastructure has to cope with billions of external services and orchestration enactments of thousands of compound services involving subsets of those billions above. Moreover, millions of users are concurrently accessing and deploying services

through the SOA4All Studio with a few thousands of them acting as service composers. However the selection of services to be composed confines itself more or less, to *service marketplaces*, delimited e.g. along geographical provenance, topics, mercantile business, etc. Additionally, alliances and partnerships may be set up to enlarge, increase the respective marketplaces influences without necessarily merging them. Consequently, federating ESBs must enable to build recursively hierarchical federations, in order to be able to reflect the corresponding service marketplaces alliances. In short, the primary goal of the SOA4All service cloud is to ensure that SOA4All scales to the dimensions of the Internet by enabling appropriate distribution techniques without altering the communication and interaction patterns of the ESB core.

The baseline for the design of the federated bus infrastructure is given by the DSB, as it already addresses service location agnosticism. So far, the DSB mainly lacks Internet-wide inter-connection mechanisms for its containers, and a global and shared store of metadata about federated services. Additionally, the resulting federation should be deployable on third-party computing resources, featuring also elasticity to cope with increasing or decreasing needs for nodes in the federation or the global information space. ESB scalability may be achieved in different ways: Many available tools are based on Hub and spoke architecture: this means that ESB is centralised (like classical EAI solution) and provide connectors (SOAP, RMI, etc.) for distributed services. The main drawback of this architecture is that it doesn't ensure end-to-end QoS. PEtALS extends the specification in order to provide a multi-nodes architecture that addresses scalability, providing many JBI compliant nodes connected together. This leads to a moderate scalable architecture. The SOA4All DSB federation extends this concept to the federated architecture that allows to scale at a very high level.

While state-of-the-art ESB technology mostly relies on client-server communication, in SOA4All we aim at providing services with more powerful communication patterns (e.g., publish-subscribe, event-driven, and semantic-oriented) that allow further decoupling of the communicating entities in terms of time, processing flow and data schema. This is an important principle relying on the fact that the SOA4All service cloud must allow any type of communication efficiently and transparently over the Internet by means of sharing or exchanging any type of data in between any type and number of distributed actors. Such needs have to be addressed at the level of the federated bus, and quite naturally leads to the integration of a global information space in form of semantic spaces to deliver the desired data sharing and event propagation middleware [10].

4. DESIGN AND IMPLEMENTATION

4.1 Service Cloud Overview

The SOA4All service cloud infrastructure consists of the merger of service buses which can be abstracted as a single Internet-wide ESB, and a P2P-based global semantic space (Figure 3). As the focal point, the service cloud enables and realizes the overall operational semantics of the SOA support, and serves as a broker that connects the SOA4All Studio, platform services and business services alike. Platform services yield the minimal necessary functionality of a

service delivery platform, such as service discovery, ranking and selection, composition and invocation.

To meet the scalability requirements of SOA4All, the distributed cloud infrastructure is grounded in well-established Grid technology, ProActive, (briefly described below) that implements the inter-buses messaging layer and that delivers the networking support for the semantic spaces. To illustrate the new elements that constitute the federation, we prototyped the required extensions in the open source ESB PEtALS. We continue this section with a presentation of the technical baseline of ProActive and PEtALS before elaborating on the federated DSB approach of SOA4All.

4.2 GCM/ProActive Grid Middleware

ProActive hosted by the OW2 consortium is a 100% pure Java solution. More precisely, it is an asynchronous active-object based middleware offering the notion of asynchronous calls with futures (a future is a promise to get back a response) among distributed objects, extended with the possibility to transparently handle groups of objects and security (e.g., authentication, encryption) for inter-object communications [2]. The transport layer communication protocol used by ProActive remote method invocation can be chosen at will (e.g., RMI, RMI over SSH, or HTTP, amongst others). ProActive also implements a component-oriented programming model, a grid extension of the Fractal model [4] called Grid Component Model (GCM) [3] that is used within PEtALS too. Active objects and component interfaces can be exposed as Web services if needed. ProActive runtimes, which act as containers for active objects, are grid-aware in the sense that they can be started remotely from any machine, using any remote access protocol, taking charge, if needed, of all the required file transfers (e.g., Java version, ProActive bundles) at the remote places using any file transport protocol. It is thus possible to gather machines from grids (e.g., Grid'5000⁶ in France), private clusters, clouds (e.g., Amazon EC2) or desktop networked machines to form an overlay network of ProActive runtimes. This overlay network can also be constituted along peer-to-peer principles. When new runtimes are started, they can automatically join an unstructured peer-to-peer overlay of ProActive runtimes. This overlay network provides the fundament for any distributed application, and thus in particular of our service cloud infrastructure too, as detailed here after.

4.3 PEtALS Service Bus Architecture

PEtALS is built on top of agile technologies such as JBI and the Fractal Software Component Framework⁷. Fractal is a modular and extensible component model that can be used with various programming languages to design, implement, deploy and reconfigure various systems and applications, from operating systems to middleware platforms and to graphical user interfaces. From the PEtALS's point of view, all the container services, such as service registry, message router, message transporter or discovery, are implemented as Fractal components. This is a major feature which allows core developers to specialize a PEtALS distribution by choosing the software components to be used for specific needs. The distributed behavior of PEtALS is mainly provided by two software components which are the *technical registry* and the *message transporter*.

⁶<http://www.grid5000.org/>

⁷available at <http://fractal.ow2.org>

The technical registry stores all non-functional artifacts (JBI services, endpoints, interfaces, container location) which provide all necessary metadata to route messages to the desired endpoint. The registry entries are replicated among all the bus nodes using a distributed hash table over a multicast channel. This is equivalent to data flooding between registries; when an entry is added to the registry, data is sent to all the network registries. In this way, all the registries have a complete view of the services hosted by all the containers. This approach is the base of the unified service bus, but as it currently relies on multicast techniques, it is not scalable. In the Internet-wide scenario of SOA4All, thousands of services references would need to be replicated to all the registries at the level of a single DSB. The current release of PEtALS is thus as-is not exploitable for the service cloud.

The message transporter constitutes the core mechanism to exchange messages between containers in a transparent way at the service consumer and provider levels. In a standard JBI implementation the *normalized message router* gets the local endpoint reference from the local registry and sends the message to local JBI endpoint. In the PEtALS approach, once the endpoint is retrieved from the registry, the message and the endpoint reference are sent to the transport layer which is in charge of delivering the message to the JBI endpoint, independently of the container's location (local or remote).

The available PEtALS DSB architecture constitutes a solid starting point for materializing the federation approach of SOA4All. As stated above, PEtALS natively assumes that service artefacts are not all co-located, and naturally integrates mechanisms in response to distribution concerns.

4.4 Complete Federation Architecture

In the remainder of this section, we present the SOA4All Distributed Service Bus, as core infrastructural enabler of our service cloud. First we present how the merger of PEtALS and ProActive provides scalable message routings and technical registries, and finally introduce the fully P2P-based semantic space implementation on top of ProActive that yields SOA4All's shared semantic memory and coordination platform.

4.4.1 Federation Topology and Message Routing

The most crucial element to federate PEtALS-based DSBs is to provide a solution for message routing without posing a constraint on having point-to-point connections among all the different containers, and without using the original JMS broker which would be difficult to deploy at Internet-scale. The chosen solution is a multi-level, hierarchical organisation of bus nodes. This flat-to-hierarchical interconnection of PEtALS containers requires that end-point references of deployed artifacts are generalized to comply to a multi-level numbering identification schema. For our prototype it is sufficient to add one level only, as we build federations of enterprise service buses; the first level is the enterprise level, and the second one the inter-enterprise level given by the Internet. Recursively building hierarchical federations of such federations would of course be possible and allows for adding as many new levels as desired in the identification schema.

To inter-connect PEtALS containers, the message transporters must be extended so that they can subcontract the message transport to a grid-aware routing layer. Through

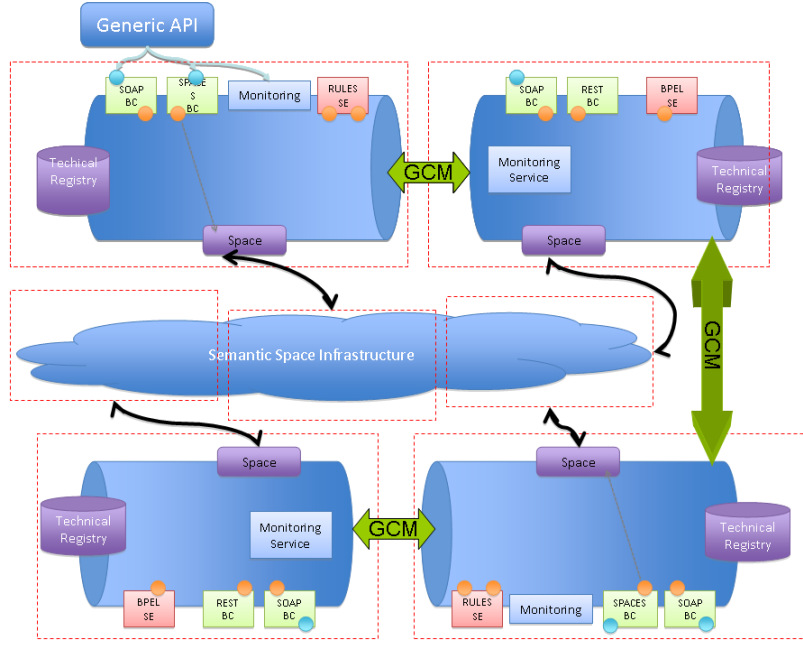


Figure 3: The technical architecture of the federated DSB.

ProActive such a hierarchical grid aware routing layer exists, and we rely on the available GCM/ProActive components [9]. This routing support was initially designed for message exchanges in a MPI-based application that was deployed on a hierarchically organized multi-cluster, inter-grids, inter-clouds computing infrastructure.⁸ Its architecture leverages the hierarchical GCM composition model: activities deployed on a same cluster of machines (i.e., intra-domain) can use native communication protocols such as an MPI implementation, or any built-in PETALS message transport protocol, as for example JMS, to exchange messages. In case the target destination hierarchical identifier shows that the recipient is hosted outside the domain, the messages are tunneled to their enclosing GCM component, which in turn is in charge of routing them according to the hierarchy. Each composite component is bound, thanks to the GCM collective interface bindings, to all its peers representing the other domains also involved in the same level of the federation. Interface bindings among the composite components concretize partnership agreements that materialize as direct communication paths. Such a path is in turn implemented as a ProActive communication channel to take advantage of all its grid-aware capabilities (Section 4.2); more details about the routing process can be found in [9]. Such a generic routing substrate is perfectly adequate to materialize a federation of ESBs along peering or more hierarchical alliances among partner ESBs. Additionally, the dynamic (re)configuration of the GCM bindings' capability allows to reflect on possible modifications of partnerships, according to some global SOA governance rules.

4.4.2 Distributed Registry

The second key issue, in getting a federation of DSBs, is

⁸Message Passing Interface (see <http://www.mpi-forum.org/docs/docs.html>).

to scale the PETALS technical registry at the level of the federation by agreeing on the partnerships and alliance relations. An alliance translates into mutual agreements to share end-point references for further use of containers or business external and internal services that are attached to the involved DSBs.

The principle elements of the technical registry still apply also in the federated scenario. However, there is a need for a inter-domain lookup protocol that we realize via a multi-level lookup (and associated registration) strategy. If the standard registry lookup of PETALS does not return any results, the lookup query is recursively propagated one level up according to the GCM-based routing organisation made of GCM composite components, and then translated into the forwarding of lookup queries down to each federated DSB. The lookup response makes its way back along the same route, and cancels any still ongoing search if any. In addition, the lookup strategy is capable of dynamically establishing shortcuts as optimizations for further use of the route [9]. We expect a performance benefit from using GCM collective interfaces and bindings within the architecture of the routing layer, as they naturally introduce parallelism in the lookup process. Indeed, [9] includes competitive performance evaluation results (compared with gridified versions of MPI implementations) of patterns that are very close to our lookup protocol.

For the sake of performance when managing replicated copies, we do not consider to implement a systematic replication of all the information stored in the respective technical registries of all federated buses, but rely on the lookup multi-level strategy explained above. It does however not ban the technical registries of individual DSBs from storing lookup responses for a specific predefined delay, acting as a cache. Besides, the technical registry copies of each DSB may still be managed using replication, applying the

already built-in protocol. In fact, it is not expected that a single DSB may be constructed of more than a few tens of containers, also in the Internet-scale scenario.

4.4.3 Semantic Spaces

As stated in the requirements section, there is an eminent need for complementary communication and coordination means – as alternatives to the established client-server model – that allow traditional service bus technology to scale up to Internet-scale. Semantic spaces are a novel type of communication platform that has recently gained momentum in the middleware community, as a response to the raising challenges of data sharing and service coordination in large-scale, distributed, open and highly dynamic Web environments [10]. Semantic spaces fuse tuple space computing [8] known from parallel processing, blackboard-style problem solving [6] known from artificial intelligence and semantic technologies to a distributed (semantic) data management platform. Such a platform offers a simple interface for persistently publishing and reading semantic data, and semantic pattern-matching as enabler for type-based retrieval and publish/subscribe mechanisms rather than direct addressing and message passing. A semantic patterns is any triple pattern or graph pattern, as they are supported by most RDF query languages (cf. for example SPARQL as defacto standard [11]). Consequently, semantic spaces bring more Web-style communication patterns to service computing, as alternative to the traditional message-driven approaches of XML-based Web services.

This is particularly interesting and promising in the context of many of our SOA4All large-scale data sharing scenarios for which the amount of semantic data that has to be processed is likely to exceed the numbers that we are currently aware of. Moreover, SOA4All (refer back to Figure 1) will exploit the semantic space add-on as distributed semantic repository for service and process descriptions, as collaboration infrastructure for sharing monitoring data and user profiles, and as asynchronous and event-driven communication platform for the realization of service mash-ups. Service mash-ups are data-centric specifications over collections of services whose executions is coordinated by shared access to data in the space. It is important to note that by offering the complementary communication facilities of semantic spaces to all platform services and business services that are bound and interacting over the service cloud, we realize the core infrastructure services of SOA4All in an integrated manner and do not require anyone to take care of additional access points.

The implementation of the semantic space infrastructure itself is based on well-established P2P overlays and deployed on ProActive nodes. In that way the space nodes and inter-bus routing nodes are co-existent and coordinated resources relying upon the same grid technology, and consequently, same grid management tools as the federated DSB. Each peer of the overlay is implemented using an Active object and thus benefits from its asynchronous communications. Although it is possible to store any kind of data in the exploited P2P overlay, the infrastructure is optimized for handling semantic data in form of RDF triples. Triples are indexed and distributed according to a CAN overlay [12]. CAN offers a structured peer-to-peer architecture which allows efficient storing/retrieving of data. Their design centers around a virtual d-dimensional Cartesian space. The entire

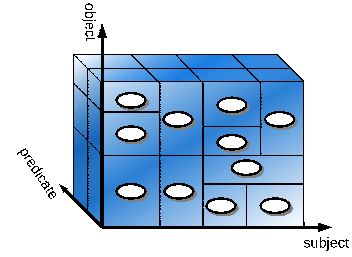


Figure 4: Three Dimensional CAN space. CAN axis represent subjects, predicates, objects of RDF triples. Space is partitioning into zones. Each zone is owned by a peer (ProActive Active Object).

coordinate space is dynamically partitioned among all peers in the system. The coordinates are used to store tuple values by mapping them to points in the virtual space. A three dimensional CAN overlay offers a natural solution to the storing of RDF triples. Axis of the overlay represent respectively subjects, predicates and objects of RDF statements and values are ordered lexicographically (see Figure 4). The number of dimensions can be increased to handle additional data like meta-information for each RDF triple.

In order to integrate the semantic space infrastructure with the federation of service buses, its interface must be exposed as a service that is offered via a JBI binding component. This component is the link between any service bus and the semantic space infrastructure. With this approach, manipulating data, expressing queries or being notified of new data is equivalent to performing: 1) a direct service invocation: writing, reading or querying data means sending the right message to the space service. Since the space service will be described using WSDL (according to the JBI specification), this becomes a standard operation at the level of the bus, and 2) listening to events: when new data is available in the space, the binding component is able to notify a set of services which have registered listeners. This provides a working solution for relying upon Event-Driven Architecture patterns for building SOAs inside a single distributed bus as much as at the level of federations.

4.4.4 Bringing All Together

The resulting service cloud architecture is shown on Figure 3. DSBs are interconnected and linked to the semantic space: red-dotted rectangles represent ProActive/GCM runtimes that host semantic space nodes and service bus nodes respectively. The large bi-directional arrows denote the GCM transporter extensions that are needed for the technical registry and message routing mechanisms between bus nodes. The smaller dark bi-directional arrows represent the links between the DSBs and the semantic space via standard JBI mechanisms, but implemented by ProActive based communication channels.

5. CONCLUSION

In this paper, we have presented the concept, architecture and prototype implementation of an Internet-wide service cloud based on the federation of distributed services buses and semantic spaces. Our proposal is based upon the PEtALS and ProActive/GCM technologies and delivers a

large-scale, open and distributed SOA environment.

The needed extensions to existing service bus realizations that were identified required the improvement of the *technical registry* and the capability to route messages within *federations*. Although, realized in the context of PEtALS, we believe that such extensions would be needed for any ESB/DSB willing to take part within such a federation. Validation this assumption by means of alternative open source ESBs on the market are planned, but have not yet started.

Additional effort to effectively finalize a working prototype of a federation, relying upon the use of PEtALS buses is currently undertaken. It pertains both at the algorithmic and experimental levels, and is required to definitively decide for one or another federated technical registries management strategy. Being able to rely on GCM-based structures to represent the topology of the federation is seen to be a key advantage in defining the most adequate strategies for lookup and update of the involved technical registries. An alternative to not rule out is to rely on the global semantic space to serve as the technical registry of the prototyped federation, taking advantage of its foreseen scalable, and performant peer-to-peer discovery and storage protocols. However, the aim of the semantic space is much more focused on sharing semantic artefacts than on being a registry of JBI end-point references. Consequently, a specific technical registry management protocol within the federated PEtALS DSBs may eventually suit better. This remains to be further investigated, also in terms of what and how registry information is formalized and published; a semantic space-based solution would require an RDFication of JBI end-point references.

In summary, the ESB federation exposes the traditional enterprise service bus functionality extended with scalable Web-style publishing and reading. The integrated support for semantics and event-based communication mechanisms serves as basis for shared data management and collaborative activities. In particular, the integrated support for semantics empowers direct links to reasoning or mediation techniques. By adding semantic spaces, SOA4All moreover realizes an integrated infrastructure that grants access to distributed semantic repositories for service descriptions, a storage infrastructure for sharing monitoring data and other collaborative tasks, and event-driven communication without requiring platform services and business services to take care of additional access points; i.e., the presented work is conceptualized to provide a multi-functional service cloud for SOA4All in a all-in-one solution.

6. REFERENCES

- [1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A Berkeley view of cloud computing. *University of California, Berkeley, Tech. Rep.*, 2009.
- [2] L. Baduel, F. Baude, D. Caromel, A. Contes, F. Huet, M. Morel, and R. Quilici. Programming, composing, deploying for the grid. *Grid Computing: Software Environments and Tools. Springer, Heidelberg*, 2005.
- [3] F. Baude, D. Caromel, C. Dalmaso, M. Danelutto, V. Getov, L. Henrio, and C. Pérez. Gcm: A grid extension to fractal for autonomous distributed components. *Annals of Telecommunications*, 64(1):5–24, 2009.
- [4] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J. Stefani. The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems. *Softw. Pract. Exper.*, 36(11-12):1257–1284, 2006.
- [5] David Chappel. Introducing the azure services platforms: an early look at windows azure, .net services, sql services, and live services, October 2008. <http://download.microsoft.com/download/e/4/3/e43bb484-3b52-4fa8-a9f9-ec60a32954bc/Azure\Services\Platform.pdf>.
- [6] R. Englemore. *Blackboard Systems*. Addison-Wesley Publishers, Nov. 1988.
- [7] John Fontana. Microsoft melding view of local, cloud-based virtual machines, April 2009. <http://www.networkworld.com/news/2009/042909-microsoft-virtual-machines.html>.
- [8] D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, Jan. 1985.
- [9] E. Mathias, V. Cavé, S. Lanteri, and F. Baude. Grid-enabling SPMD Applications through Hierarchical Partitioning and a Component-Based Runtime. In *15th Int. European Conf. on Parallel and Distributed Computing (Euro-Par 2009)*, volume 5704 of *LNCS*, pages 691–703, 2009.
- [10] L.J.B. Nixon, E. Simperl, R. Krummenacher, and F. Martin-Recuerda. Tuplespace-based computing for the Semantic Web: A survey of the state of the art. *Knowledge Engineering Review*, 23(1):181–212, March 2008.
- [11] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, January 2008.
- [12] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.
- [13] M.T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. The enterprise service bus: Making service-oriented architecture real. *IBM Systems Journal*, 44(4):781–797, 2005.
- [14] C. Schroth and T. Janner. Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. *IT Professional*, 9(3):36–41, 2007.
- [15] D. Tapscott. *The Digital Economy: Promise and Peril In The Age of Networked Intelligence*. McGraw-Hill, 1997.
- [16] Luis M. Vaquero, Luis Roderio-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, 2009.
- [17] David Wright. The federated enterprise service bus for european central banking, February 2008. http://www.softwareag.com/de/images/EuropeanCentralBank_CaseStudy_2008_tcm17-50665.pdf.