WEB SERVICES DEPLOYMENT ON P2P NETWORKS, BASED ON JXTA

Marco Pereira, Marco Fernandes, Joaquim Arnaldo Martins and Joaquim Sousa Pinto
IEETA - Instituto de Engenharia Electrónica e Telemática de Aveiro, Universidade de Aveiro, Aveiro, Portugal

Keywords: Peer-to-peer, Digital libraries, Service oriented architecture.

Abstract: Digital libraries require several different services. A common way to provide services to Digital Libraries is with the use of Web Services but Web Services are usually based on centralized architectures. In our work we propose that Peer-to-peer networks can be used to relieve the weight of centralized components on the Digital Library infrastructure. One of the challenges of this approach is to develop a method that allows the integration of existing web services into this new reality. In this paper we propose the use of a proxy that allows exposing Web Services that are only accessible trough the peer-to-peer network to the outside world. This proxy also manages interactions with existing external Web Services making them appear as part of the network itself, thus enabling us to reap the benefits of both architectures.

1 INTRODUCTION

Digital libraries require several support services. Such services are often created as web services that live in a central server and are accessed when needed. A failure in this central server can be catastrophic to the digital library, and to counter this menace services are often replicated. In this scenario service replication happens within the same organisation with multiple copies of each service being offered to counter possible failures. Service discovery and selection is critical in this types of environments.

In our previous paper (Pereira et al., 2009) we introduced a peer-to-peer (P2P) network based on the JXTA ((JXTA), 2009) framework that can be used to support Digital Libraries. Our P2P approach can be used in two complementary scenarios: to create a distributed digital repository and to provide support services. The distributed repository scenario can be seen as an application of traditional file-sharing paradigm. Peers store content in a replicated fashion to create a robust repository. The system automatically taps into this redundancy to provide multi-source file transfers that increase overall speed of transfer. One of the problems that we face in the traditional file sharing paradigm is that querying capacity is limited, often relying on carefully constructed file names. To overcome this limitation we used an Apache Lucene ((ApacheLucene), 2009) based indexing system that allowed us to conduct rich queries across the network.

Providing support services came as a natural extension of the distributed digital repository. Digital libraries need support services, for example to convert images between formats. Support services are also required if instead of using the P2P digital repository as a front line storage space we wish to use it as a digital preservation mechanism. By giving each peer a set of services we create an environment where our repository can be adapted to serve multiple purposes. When providing support services, tasks that can be run in parallel at multiple nodes (such as converting a set of files to a different format) can be done faster than if they were performed in sequence in a single node, even when counting with network transfer overhead. On the other hand if we wish to configure our digital repository to act as a distributed preservation system we will need to provide services that, for example, preserve the integrity of the stored content and provide audit trails. By distributing the services throughout the nodes we were also able to reduce the dependency on centralised services, since if one node is unreachable there are others that can perform the same task.

To provide services to the P2P network we initially used the JXTA-SOAP (Amoretti, 2009) library. These services use the JXTA protocol for transport instead of the traditional HTTP. The choice of the web services paradigm was based on its ubiquity and simplicity. By providing a known paradigm we wanted to shield developers of the complexity of the underlying

Pereira M., Fernandes M., Arnaldo Martins J. and Sousa Pinto J. (2010).
WEB SERVICES DEPLOYMENT ON P2P NETWORKS, BASED ON JXTA.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 423-427
DOI: 10.5220/0002872704230427

P2P network. This objective was partially achieved: while we were able to create web services they possessed a number of restrictions and requirements that can be seen as breaking the web services approach. We were also disregarding existing web services: in a scenario where only one peer has access to an existing web service it was not possible to expose that service to the network without implementing a service specific proxy. Service specific proxies were also needed to expose the services that existed inside the P2P network as true web services. Given the perceived limitations of our initial approach it was decided that a complementary solution was needed. Instead of creating services that were limited in scope to the P2P network we now aim more at using its infrastructure for transport only. With this approach services will live outside the P2P network, and will be true web services, developed with any of the standard methods. The P2P network will be an alternative access method. This approach will enable us to use within the network existing web services, and the creation and deployment of semantically equal services across different peers, providing web service replication.

2 SERVICE DISCOVERY ON JXTA BASED PEER-TO-PEER NETWORK

Resource discovery in JXTA is based on the concept of advertisement ((JXTA), 2007). An advertisement is a small XML document with information about a particular resource. Advertisements possess a pre-determined lifetime, and if they are not renewed within that lifetime they expire. JXTA uses advertisements to describe its own resources (such as peers or communication channels) and applications are encouraged to used them as well. Being such a central concept in JXTA, the first step to locate a resource in the network is always to find a corresponding advertisement by querying the network (or the local cache).

JXTA provides an advertisement family (Module advertisements) specially crafted to describe service implementations. From this family JXTA-SOAP uses the *ModuleSpecAdvertisement* to describe its services. This advertisement type provides a field (the name field) that can be used in queries. We choose to place the web service namespace in this field. In XML namespaces are used to differentiate between elements with the same name. We will take advantage of this concept and extend it: in order to fully identify a web service request we will need both the namespace and the method to be invoked. This is done because in the same namespace can reside several services that do not possess the same methods. In the non queryable field (description) we can place several XML elements that describe the methods available in the service. This implies that to discover a service we must search by its namespace and then refine the results by analysing the description field from the matching advertisements. An important detail about *ModuleSpecAdvertisements* is that they carry within them a *PipeAdvertisement*. This *PipeAdvertisement* is used to communicate with the service provider without having to do any extra queries to the network.

This service discovery architecture type allows that several peers provide access to the same service. We can explore this feature in order to provide web service replication: each peer can be running a local copy of the web service instead of just providing a gateway to the centralised version of the web service. Each time we need to access a service a list of peers that offer that service is generated. If the service fails (because the peer is no longer available or due to a network error) the next peer on the list can be contacted in order to execute the request. A side effect of this replication will be the need for advanced peer selection policies. It should also be noted that different service types can have different requirements. Some might require readily available storage space where others will require quick response time with both cases requiring a distinct approach to peer selection.

3 JXTA-SOAP BASED WEB SERVICES

In our P2P network we started to provide services with the use JXTA-SOAP. JXTA-SOAP can be seen as an extension to Apache Axis and as such the standard procedures to create Axis services are applyable, with some extra steps. JXTA-SOAP deals both with Axis service lifecycle management and JXTA advertisement creation. In order to create a service with this library it is necessary to create a service descriptor with information relevant to the P2P network. We also felt the necessity of creating a standard interface that each service must implement to allow access to configuration settings. Both the service descriptor and the interface requirement can be seen as points where the illusion of being creating a “regular” web service is broken. Creating a client requires the creation of a service Call provided by JXTA-SOAP. This service call is constructed from a *ModuleSpecAdvertisement*, and it must be explicitly retreived from the P2P network by the creator of the web service client. As an

aditional requirement we choose to force the implementation of a standardised interface to implement a plug-in architecture for clients. Such requirements also contribute to break the web services facade we were striving to maintain.

We must admit that such facade breaking compromises are to be expected when dealing with a library that creates services within the P2P network. Initially we were willing to cope with the requirements of the JXTA-SOAP library (and our own) but we soon found out the limitations of this strategy: No support for attachments and no support for existing services. Without support for the use of attachments it becomes difficult to handle large sets of binary data. Image format conversion services that we used as example in our previous work suffered from a maximum image size limit. This limitation can be circumvented by the use of an external transmission mechanism, although this is not a desired solution. The lack of support for using existing web services is the major flaw of this approach. JXTA-SOAP main focus is the creation and deployement of services within the P2P network, and as such access to external services would have to be provided with the use of a service specific proxy, that would then be represented as a JXTA-SOAP based service. This is a combersome approach that defeats one of the goals that made us choose the web services paradigm (developer familiarity). As such it is clear that we needed a complementary approach to web services.

4 PROVIDING ACCESS TO EXTERNAL WEB SERVICES

Instead of creating our services inside the P2P network we decided that it would be better if we created regular web services. Developers should be able to design new services exactly as they have done before without having to worry about implementing specific interfaces to allow the network to be service-aware. Instead we will take onto our own hands the task of creating an advertisement (also based on *ModuleSpecAdvertisement*) that describes the external web service. When we wish to add a service to the network we need three vital pieces of information: the service namespace, methods and address. From that information we can compose an advertisement that will identify the service. Two distinct peers can now deploy their copy of the service in their own application server, and when notified the network will generate advertisements for each service. If they share the same namespace and method collection they will be recognised as the same service, even though the advertisement will contain a different random generated ID and the peer specific *PipeAdvertisement*.

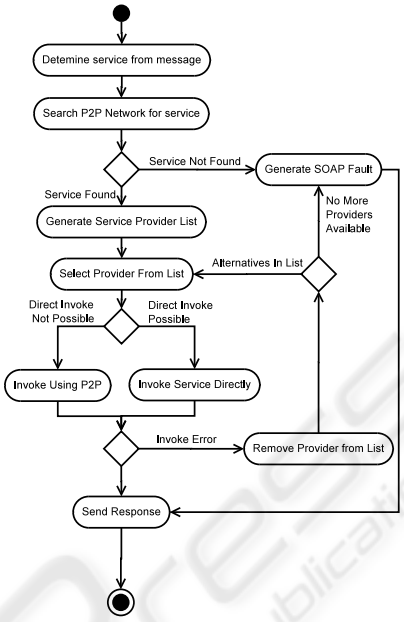

Figure 1: Decision model.

To maintain the web services paradigm, web services clients must be created in the traditional way, thus preserving compatibility with standard tools provided to generate web services clients from the WSDL of the service. To support this goal a transparent web services proxy was created. Each peer can be configured to provide a small HTTP server whose sole responsability is to capture SOAP messages. SOAP messages have the necessary information to allow a service to be located in the network: namespace and method. With this information a peer can search the network for a corresponding service and generate a list of potential service providers. This strategy will transform every peer that provides the service proxy into a gateway to the network, but will require service requests to be directed to the proxy. As was stated before most client generating tools only require a service WSDL to generate a client. Accordingly to the W3C, WSDL is “an XML-based language for describing web services and how to access them”. This description points out what we need to know in order to divert a web service to our P2P network: we need to know how to access it. The key element here is the <soap:address> element of a binding. By altering the attribute location of this element we can divert any request to our web-services proxy.

As for the actual invocation two strategies can be applied: deferred contact and direct contact. The decision to use one or another contact method is illustrated in Figure 1. The crucial part of this model is

how to determine if a given service is directly accessible or not. To do this we need to resolve the service address. If we succeed it indicates that a direct connection is available. Failure to resolve the address has two possible meanings: either the service is only available through the P2P network or the provider is no longer available. Since both failure types are indistinguishable we assume that the service will be available from the same provider through the use of the P2P network (deferred contact strategy), dealing with the possibility of the provider no longer being available as part of that strategy. When confronted with a invoke error we choose another provider from the potential service providers list, removing the one that failed from it.

4.1 Direct Contact

When a service is directly accessible from the web services proxy there is no need to take the extra step of sending the SOAP message trough the P2P network so that the peer can locally invoke the service. After choosing a peer we will have access to the address where the service resides. With this information and with the use of the SOAP with Attachments API for Java (SAAJ) ((SAAJ), 2009) we can send the SOAP message directly to the intended service via HTTP protocol. From the same library we can then obtain the raw SOAP reply that the service sent thus allowing us to redirect it back to the original calling peer. This process is illustrated in Figure 2 by the solid lines (that represent operations common to both scenarios) and by the dotted line.

4.2 Deferred Contact

Although a service is listed on the P2P network, it may not be accessible directly by conventional means. This applies to machines that only allow access to the services for example from the localhost. To overcome this limitation it is possible to transfer the SOAP message to the peer that is actually running the service via JXTA-based transport. The message must be segmented into chunks that offer guaranties that will not be too large to be transported by JXTA. When in possession of the complete SOAP message then the peer can locally invoke the service, and return the generated SOAP message back to the calling peer. To finalise the process the message is returned to the original caller via HTTP response. This process is illustrated in Figure 2 by the solid lines (that represent operations common to both scenarios) and by the dashed lines.

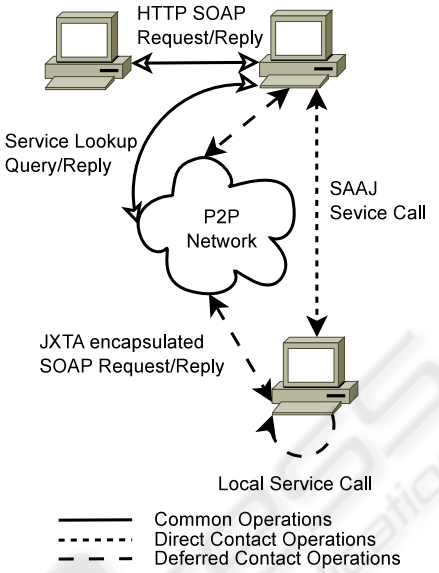

Figure 2: Direct contact and deferred contact.

5 RELATED WORK

Pure JXTA services are the native approach to services deployment on JXTA based networks, thus consciously exposing the underlying P2P structure. This makes the approach suitable for internal network services but difficult to expose to the outside.

A JXTA based P2P web services composition platform is introduced in (Zhengdong et al., 2009). Semantic web services descriptions based on OWL-S are distributed trough the peers. When the need arises to perform a given task these descriptions are looked up and composed into a new service described in BPEL4WS format. This service must then be manually integrated in the application that requires it.

An alternative approach to service replication can be found on the form of SmartWS system (Jr. et al., 2007). This system experiments with several server selection policies in order to provide optimal performance. This system relies on the creation of smart proxies that intercept web service calls and select a service provider. Unlike our approach this is done on the client side, and at the time of the creation of the smart proxy it must know all the available service providers thus ignoring new providers that might appear after its creation.

6 FUTURE WORK

An interesting path of research would be the inclusion of semantic aspects into the service discovery

mechanism. Traditional WSDL only provide a syntactic description of the service. If a semantic description was used (with the introduction where possible of WSDL-S or OWL-S for example) it would be possible to construct a semantic discovery mechanism that would allow more refined queries. The ultimate goal of this improved discovery mechanism would be to create a scenario where it would be possible to swap a faulty service for a semantically equivalent service (or a combination of services) that reside in a different namespace all in an automated way (a similar approach can be seen in PANIC (Hunter and Choudhury, 2006)). The semantic description could also be used to aid peer selection process to choose between peers offering the exact same service by applying an appropriate peer selection algorithm based on the service requirements. For example if the requested service was a storage service it would benefit to be performed by a peer that has more available storage space instead of one that offers more processing power, leading to a more rational use of the available hardware resources. It should be noted that having advanced peer/service selection algorithms is of the utmost importance to achieve performance gains and is an important research topic (Xhafa et al., 2009), (Mendonça and Silva, 2005), (Dykes et al., 2000).

A critical component to the performance of the system is the list of available providers. At this time the list is generated dynamically each time a service is requested. A possible improvement to the system could be the introduction of a caching mechanism. This would increase system performance for frequently requested services. In this scenario peers should not be permanently removed from the provider list, instead they should be temporarily blacklisted and moved to the bottom of the provider list. The time that a peer is blacklisted should increase when we experience repeated failures, thus implying some sort of failure counter. It is important to note that the counter must be eventually be decreased in order to prevent lasting impacts that a past transient network failure might have provoked. The caching mechanism would also need to be refreshed periodically to allow newly discovered peers to join already created lists. Careful tuning of when to perform a refresh and how to increase and decrease the failure counter will be the key to further performance gains.

ACKNOWLEDGEMENTS

This work was funded in part by the Portuguese Foundation for Science and Technology grants SFRH/BD/23976/2005 and GRID/GRI/81872/2006.

REFERENCES

Amoretti, M. (2009). Jxta-soap project.

(ApacheLucene) (2009). Apache lucene. http://lucene.apache.org/java/docs/index.html.

Dykes, S., Robbins, K., and Jeffery, C. (2000). An empirical evaluation of client-side server selection algorithms. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1361–1370 vol.3.

Hunter, J. and Choudhury, S. (2006). Panic: an integrated approach to the preservation of composite digital objects using semantic web services. *Int. J. Digit. Libr.*, 6(2):174–183.

Jr., J. G. R., do Carmo, G. T., Valente, M. T., and Mendonça, N. C. (2007). Smart proxies for accessing replicated web services. *IEEE Distributed Systems Online*, 8(12).

(JXTA) (2007). Jxta v2.0 protocol specification. https://jxta-spec.dev.java.net/.

(JXTA) (2009). Jxta homepage. https://jxta.dev.java.net/.

Mendonça, N. C. and Silva, J. A. F. (2005). An empirical evaluation of client-side server selection policies for accessing replicated web services. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1704–1708, New York, NY, USA. ACM.

Pereira, M., Fernandes, M., Martins, J. A., and Pinto, J. S. (2009). Service oriented p2p networks for digital libraries, based on jxta. In Shishkov, B., Cordeiro, J., and Ranchordas, A., editors, *ICSOFT (2)*, pages 141–146. INSTICC Press.

(SAAJ) (2009). The soap with attachments api for java. https://saaj.dev.java.net/.

Xhafa, F., Barolli, L., Daradoumis, T., Fernández, R., and Caballé, S. (2009). Jxta-overlay: An interface for efficient peer selection in p2p jxta-based systems. *Comput. Stand. Interfaces*, 31(5):886–893.

Zhengdong, Z., Yahong, H., Ronggui, L., Weiguo, W., and Zengzhi, L. (2009). A p2p-based semantic web services composition architecture. In *e-Business Engineering, 2009. ICEBE '09. IEEE International Conference on*, pages 403 –408.