Web Service and Business Process Execution on Peer-to-Peer Environments

Marco Pereira, Marco Fernandes and Joaquim Arnaldo Martins
DETI - Department of Electronics, Telecommunications and Informatics
IEETA - Institute of Electronics and Telematics Engineering of Aveiro
University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
{marcopereira, marcopsf, jam}@ua.pt

***Abstract*—Service oriented environments and peer-to-peer networks are on the forefront of research. This paper addresses the issues that arise when attempting to integrate these technologies, while at the same time makes explicit the benefits that can be gained from this integration. We propose the creation of a proxy for web services that allows the deployment of multiple instances of the same traditional web service in a peer-to-peer network. This proxy handles service discovery in the peer-to-peer network and can be used by existing clients with no modifications, thus offering a transparent way access to resource replication and decentralisation benefits that are traditionally associated with peer-to-peer networks. We then proceed to adapt a business process execution engine to be peer-to-peer aware, allowing the implementation of process partition and delegation techniques that can result in reductions in the network traffic required to execute a business process, as well as in a more efficient distribution of the service load through available peers.**

Keywords-Peer-to-Peer; Web Services; Service Oriented Architecture; Business Process Management.

I. INTRODUCTION

Access to computational resources is a key requirement of most modern organisations. This requirement arises from the need to produce text documents, to process employees' salaries, or to provide complex services. The typical response to this requirement leads to the proliferation of hardware throughout organisations, and even among individual users. While some of it is specialised hardware (such as dedicated servers) most takes the form of personal computers that are mainly used to perform simple tasks, wasting potential computational resources in the form of processing cycles and storage space. Using peer-to-peer (P2P) technology it is possible to tap into these otherwise wasted computational resources [1]. A possible use for these "*recovered*" computational resources is service deployment, particularly web services. If web services are themselves seen as resources it is possible to apply to them the same philosophy that is applied to files in traditional P2P networks, where availability and resilience is improved by the existence of multiple copies distributed throughout the network. By having multiple copies of a web service present in a P2P network one can avoid what can be seen as a potential centralised point of failure and provide an alternative way to implement a Service Oriented Architecture (SOA) [2]. While SOA is one of the most popular research topics, and has been a driving force in the software industry [3] to fully explore its advantages it is not possible to ignore the importance of service orchestration, that can be used to create composite services from the individual services available, thus creating a business process. The usage of SOA, business process orchestration and web services can bring numerous advantages for organisations [4] such as higher automation and process integration. Unfortunately the tools that perform service orchestration are not expecting the services to be available from multiple providers as it would be the case if the services were available as resources in a P2P network and do not take advantage of this fact. In this work we describe how to use a P2P network in combination with orchestration tools in order to reap benefits from making web services behaviour more akin to the one exhibited by files in file-centric P2P networks.

This work focuses on the development of integration strategies that explore the synergies that exist between web services, business process execution and P2P environments. We believe that better integration between these technologies will lead to improved performance and added robustness when executing business processes or individual web services. These improvements are achieved by using process delegation to reduce the overall network traffic generated by the execution of a business process and by allowing the replication of individual web services through multiple peers to ensure that a service can be executed even if some of the service providers become unreachable.

While designing a P2P based service-oriented environment we have established a few pre-requisites. First unless absolutely required existing standards should be used. Using standards compliant approach will enable us to accommodate already existing services, clients and business processes within the P2P environment, and does not impose any additional burden to developers. Second we make no particular assumptions about the underlying network. This means that our environment should transparently accommodate different topologies and be able to execute the services and business processes in a non-optimised fashion if the available peers do not offer specific capabilities.

The structure of this work is as follows: in Section II we review existing related work while in Section III and Section IV we describe how web services and business process execution can be made P2P-aware while the benefits and caveats that

 ISBN: 978-1-61208-173-1

can follow from having P2P-aware business process execution environment are analysed in Section V with a case study. Finally in Section VI we present the conclusion of this work and explain our plans to further improve the presented work.

II. Related Work

Exposing services as part of a P2P network can be seen as one of the achievements of the JXTA framework [5]. The JXTA framework provides the necessary protocols to create a P2P overlay network, establish connections between peers, and to discover resources in the network. JXTA is able to create unstructured P2P overlays that can be configured to form either a pure P2P system that resorts to flooding (using multicast where available) to perform network queries or an hybrid P2P system where queries can be directed to infrastructure peers (in addition to any local caching performed by client nodes). It should be noted that the use of an hybrid topology is mandatory in case the actual overlay needs to be extended past any type of network boundaries (such as NATs or firewalls). In JXTA every resource (be it a communication channel, a peer or a service) is represented by an advertisement [6]. An advertisement is a small XML document with information about a particular resource that possess a pre-determined lifetime that will expire if not explicitly renewed during that lifetime. The first step to locate a resource in a JXTA based P2P network is always to discover a corresponding advertisement by querying the network or the local cache. Of particular relevance is a family of advertisements, *Module Advertisements* that can be used to represent and discover services. This family possesses two advertisement types, *Module Specification Advertisement* and *Module Implementation Advertisement*, representing respectively the expected behaviour and protocol of a given service and a concrete implementation of the corresponding service. It should be noted that although the publication of a *Module Specification Advertisement* is optional, its publication is considered a good practice and has an advantage over the publication of a *Module Implementation Advertisement* alone. The advantage is that a *Module Specification Advertisement* is allowed to carry within it a *Pipe Advertisement*, which can be used to locate a peer that is able to execute the service. As it would be expected, services constructed in this way are deeply intertwined with the P2P network and difficult to expose to the outside, thus creating an impedance mismatch when trying to use them in SOA.

Another project, also based on JXTA took a different approach. Instead of creating pure JXTA services, JXTA-SOAP [7] allowed developers to create web services, that can then be deployed in the P2P network. Services created with JXTA-SOAP can be reached within the P2P network by discovering a *Module Specification Advertisement* that contains the WSDL of the web service and a *Pipe Advertisement* to contact the peer to execute the service. JXTA is used as a transport protocol instead of HTTP (handled automatically by the JXTA-SOAP library). Developing a service using the JXTA-SOAP approach requires the service to be developed in Java, and the implementation of a specific interface. It also requires the service to be deployed using the first generation of the Apache Axis platform [8]. As with native JXTA services, services developed with JXTA-SOAP are also difficult to expose to the outside.

The default approach to web service discovery is to rely on UDDI (Universal Description, Discovery and Integration protocol). UDDI provides a centralised broker that can be queried by a client to discover a provider of a given service, yet this centralised approach creates a single point of failure. To tackle this problem it was proposed in [9] that UDDI brokers could be federated using a P2P approach, where each UDDI broker acts as a super-peer for a group of peers that have shared interests (in this case they either require or provide similar services). Replacing UDDI with a completely decentralised P2P approach was proposed in [10]. In this scenario peers publish a semantic description of each provided service (based on OWL-S). When queried, the network can return the description to a particular service (or a semantically equivalent one that is currently available) by automatically producing a service composition described using Business Process Execution Language (BPEL). The resulting service composition can be later used in any application as a regular web service.

The issue of web service replication is approached in [11]. This work assumes an ad-hoc network scenario (similar to P2P networks) where frequent node disconnections and failures make traditional static binding unreliable. To increase the reliability and availability of services in those types of networks, it introduces an active monitoring scheme, based on a global view of the network that can be used to determine if a service is still available. To cope with expected service failure it allows dynamic deployment of replicas of web services (the web services must be Java based). To invoke a service a node must at first discover an available instance of the service (it is stated that this a responsibility of the client, not of the system and the suggested means to achieve this are described in [12][13]). After discovering this initial instance, it passes it to a tool called "*WSDL-finder*" that must be called every time before the service is actually invoked in order to discover and invoke the service from an available replica.

An alternative to the use of P2P networks for service replication can be found in SmartWS [14]. It relies on client side "*smart proxies*" that intercept the original web service call and redirects it to a service provider that at the moment offers optimal performance (based on a series of tests). All the service providers must be known before generating the client side "*smart proxy*", which means that any new provider that appear after the proxy creation will not be taken into consideration.

On the business process side there have been several proposals that attempt to leverage the existence of multiple distinct providers, particularly of the orchestration engine itself. The proposed techniques can be applied to business process described using BPEL, and mainly deal on how to divide the business process in order to distribute the execution of a single business process by multiple BPEL engines in order to improve throughput. To achieve this goal it was proposed

 ISBN: 978-1-61208-173-1

in [15] the partition of the BPEL instruction sequence into a set of distributed processes (that can be reordered, but whose final output will always be the same as the original sequence). BPEL activities are divided into fixed (receive, reply invoke) and portable, where each fixed activity is aggregated with a process service (receive/reply pair with the entry point) and the portables can be moved. This approach allows the automatic extraction of parallelism from the flow activity and results in partitions containing one fixed activity and zero or more portable ones. According to the authors this approach leads to a projected throughput increase of 30% under normal system load and by a factor of two under high load, yet it assumes that every service-providing node has BPEL runtime capabilities.

Another approach can be found in [16] where it is proposed to decentralise the flow control and dynamically select the role that a given node should take. After executing an activity each node transfer all the generated state information to the following node (thus the participating nodes can be seen as stateless). It should be noted that this approach allows the dynamic discovery of business partners, yet it still requires the presence of a BPEL engine in every node and only considers simple flows without any type of synchronisation, restrictions or error handling. Alternative approaches to business process partitioning can be found in [17][18]. These works propose extensions to the existing BPEL standard in order to make the data flow (expressed in the form of shared process wide variables) as explicit as its control flow. Business processes using the proposed extensions can be partitioned taking into account both control requirements and shared data requirements. It should be noted that the partitioning process takes place before deploying the business process for execution.

III. Web Services in P2P Environments

Our starting point is a previously developed P2P framework designed to support digital libraries [19]. The P2P component is based on a JXTA unstructured hybrid overlay comprised of both infrastructure peers and client peers. In this overlay infrastructure peers are responsible for gathering advertisements that are sent periodically by client peers and other infrastructure peers. Client peers can contact infrastructure peers in order to discover and receive new advertisements that are stored in a local cache. When client peers need to locate a resource they first issue a query to their local advertisement cache and only in the event of not having a matching advertisement they issue the query to infrastructure peers. It was decided that the services should be standard web services, given the ubiquity and consequent familiarity of that technology. This decision lead us to use JXTA-SOAP to provide the bridge between web services and the P2P network, in an attempt to avoid exposing the details of the P2P network directly to service developers. While this initial approach allowed us to take advantage of JXTA built-in resource discovery mechanism (advertisements), it also possessed a number of shortcomings. Its use of JXTA-SOAP created a technological limitation to service developers by requiring that services to be deployed using Apache Axis. Additional requirements of the creation of services using JXTA-SOAP include the need to implement a specific interface and the creation of service descriptor (one for each service), which requires details from the P2P network itself, thus breaking the illusion that developers are creating standard web services. Furthermore the services created were only available within the P2P network, and making them available to the outside required a manually created per service proxy.

Given the limitations of our initial approach the need for a complementary solution became clear. Instead of creating our services inside the P2P network we decided that it would be better if we created regular web services. This allows developers to design new services without having to worry about implementing specific interfaces to allow the network to be service aware while also freeing them from having to create services using a particular technology or application server. In order to make the network aware of the existence of these external web services (some whose access might only be possible from the *localhost*) the local P2P client can be configured to fetch the WSDL service descriptions from either a set of addresses or a system folder. The same local client can be deployed and configured to run in tandem with already existing web services, exposing them to the P2P network. Services are then described using a specially crafted advertisement (based on JXTA *Module Specification Advertisement*) that uses the information available from the service WSDL. This advertisement will carry three pieces of information that allows the identification of both service and providing peer: its namespace, methods and address. Two distinct peers can deploy a copy of the service in their own application server and automatically generate advertisements for each service from its corresponding WSDL. Two advertisements will describe the same service if they possess the same namespace and method collection. With this strategy web services' clients are created in the traditional way, and if nothing is done we could run the risk of creating bindings that use the same service-providing peer, ignoring any replica present in the P2P network. To avoid this we chose to create a transparent web services proxy. Each peer can be configured to provide a small HTTP server whose main task is to capture SOAP messages. These messages have the required information (namespace and method) to locate a service in the network by searching for its advertisement, generating a list of potential providers. Another task of the HTTP server is to publish a modified version of the original WSDL of each service. This modified WSDL is identical to the original except that the <*soap:address*> of the binding will point to an address configured in the web service proxy instead of pointing to the service location directly, thus ensuring that clients generated from this modified WSDL version will be transparently using the P2P network. Assuming that previously deployed public services remain public, clients generated from the original WSDL will not be affected, ensuring that legacy applications will continue to work while still offering a clear upgrade path. As was said before each peer can be configured to act as a proxy regardless of the existence of other peers that are performing the same task. This enables us to provide multiple entry points into the P2P network or even to apply

 ISBN: 978-1-61208-173-1

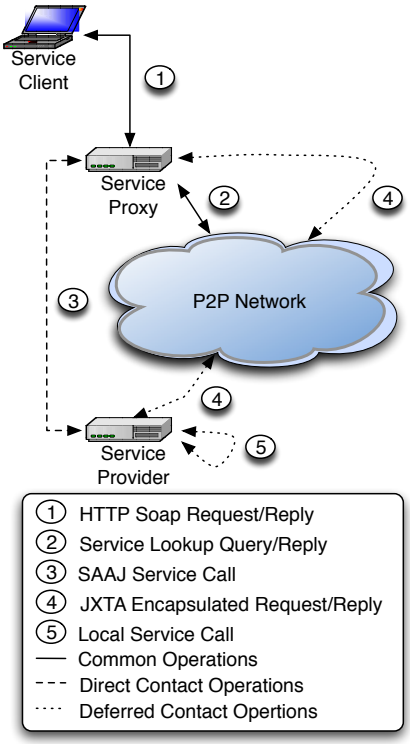

Figure 1: Direct contact and deferred contact.

load balance techniques between entry points.

It should also be noted that peers that act as service proxies have two distinct service invocation methods that they can use: direct contact and deferred contact (as seen in Figure 1). Direct contact can be used when the service is directly accessible using the standard HTTP protocol, while deferred contact first transfers the raw SOAP message that the service proxy received to the peer that is actually going to perform the service using the P2P network. The decision to use one or another method (illustrated in Figure 2) depends of whether a service can be reached using the standard HTTP protocol or not, and determining this requires resolving the service address. A successful resolve indicates that the service is directly accessible and is available, while a failure can indicate that the service provider is no longer available, or that the service is only accessible from the P2P network. Since both types of failures are indistinguishable we make the assumption that the requested service will be available from the same provider through the use of the P2P network (thus adopting a deferred contact strategy). If this leads to a new failure while contacting the peer, a new one can be selected from those that offer the same service.

A disadvantage of this approach when compared with both native JXTA services and JXTA-SOAP based services is that it will require manual service replication. JXTA services were designed to be portable across machines that are running the same JXTA version (JXSE or JXTA-C), with their runtime requirements being described in *Module Advertisements*. A similar philosophy can be applied to JXTA-SOAP services,

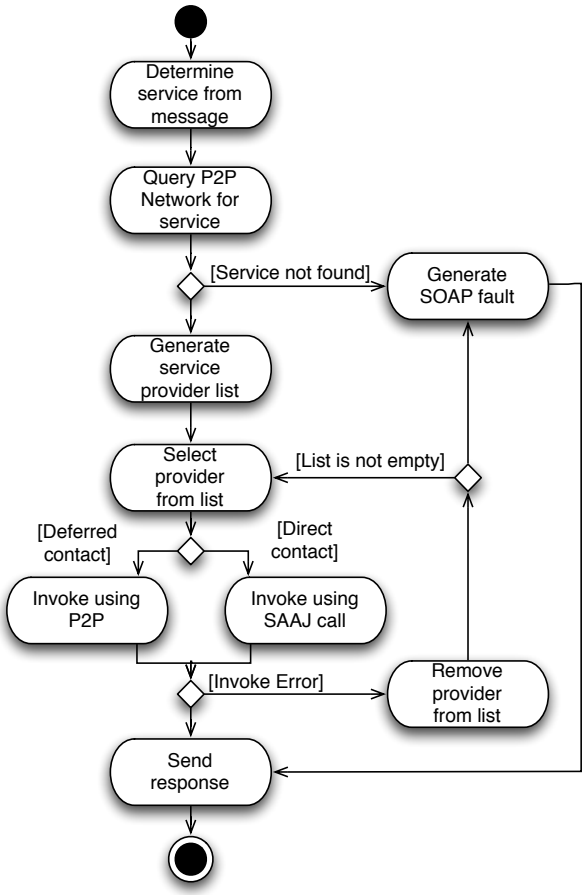

Figure 2: Decision model.

since their base requirements (an Axis application server and java based JXSE JXTA version) is known a priori, and their runtime dependencies can be bundled in a package. With our proposed approach the only pieces of information that we have about a service is its WSDL and providing peer. Since we have no information about the application server that they require, or about their runtime dependencies it is not yet possible to devise an appropriate and completely automated service migration/replication policy. On the other hand our strategy does not mandate the use of any specific technology for the creation of services, unlike previously referenced approaches [6][7][11].

IV. Business Process Execution In P2P Environments

While the strategy detailed in the previous section regarding web services deployment in P2P environments can be used transparently in the context of business process execution, we believe that a BPEL engine can benefit from the fact of being P2P-aware. The most obvious benefit is that it can use the service discovery mechanism directly, avoiding having to go through the web services proxy for each service that it intends to invoke from the P2P network. Having direct access to the discovery mechanism means that the BPEL engine can act as a simple load balance mechanism, exploring service

 ISBN: 978-1-61208-173-1

replication to avoid sending too many service requests to the same service-providing peer. This can help to alleviate the fact that the traditional approach to business process execution is a centralised one, in which service calls are dispatched to partner links (usually generated from a WSDL and thus bound to a single provider) and where state is centrally managed, by replacing those pre-bounded calls with dynamically discovered service providers. Going one step further, the ability to execute a BPEL process is in itself a service, which can be replicated and advertised by the P2P network. It should also be noted that the composition of multiple services executed by a BPEL engine is itself exposed as a web service described by a WSDL. This fact can be explored in a P2P environment in several ways: by replicating business processes (seen as regular web services) throughout the network; by allowing peers that provide an entry point into the P2P network to perform load balance between multiple composite service providing peers and more importantly by realising that the network may have multiple peers that can provide BPEL execution as a service. Being able to discover other available BPEL engines opens the door to distributing the orchestration process through multiple service providers (as opposed to execute the entire process in a single centralised provider). Distributing the orchestration through multiple peers has several advantages, particularly in high load scenarios or in scenarios where there is the need to transfer large amounts of data between service providers and consumers. Achieving this goal requires a careful partitioning process in order to reduce the number of messages and the amount of data transferred, thus increasing throughput.

Regarding the partitioning process, previous work assumes that every partner node will have BPEL capabilities, which in a P2P network designed to take advantage of already existing computational resources might not be the most convenient approach. It is possible to safely alleviate this assumption when using a BPEL engine that is P2P aware, since before executing a business process we can discover not only the required service providers but also any other available BPEL engines (since BPEL execution itself is a service). If no other engine is found then business process execution will proceed in the traditional centralised way, yet if one or more engines are found the BPEL process definition can be partitioned and parts of it delegated to other engines. If those engines are themselves P2P aware it is possible to continue the partition process. It should be noted that the absence of this "BPEL in every node" assumption means that some of the previously proposed partition mechanisms can not safely be applied to this scenario, yet some of the previously proposed design principles remain valid. When there is a parallel execution (a flow activity), an entire branch can be still be partitioned if the first invoke service activity exists at a BPEL-capable peer. Furthermore having access to the service discovery mechanism means that we can eventually use information about the services themselves both to decide what will be the more adequate service provider to use and to infer the best tasks to be delegated.

As was asserted before, process delegation has the potential to greatly reduce the amount of data that must be transferred through the network, mainly by eliminating round trips in the invocation calls. Since this effect can be seen more clearly when delegating services that require the transmission of large message or variables (particularly large response messages), our main concern should be to provide a way to identify those types of service. While there is no standard way to know a priory which services will generate a large response message, we can use the return type as a telltale of those services. It is safe to assume that any efficiency gain will likely be much smaller when delegating the process if services are going to return an integer when compared with services that return an array of bytes. As such we suggest the usage of a simple rule: perform no process delegation if the next service return messages with simple types (numeric, boolean, strings) or complex types based on these types.

In line with previous work inner process delegation presents some difficulties when dealing with process monitoring. While keeping track of the progress of a business process in a centralised scenario is a simple task, doing so in a decentralised orchestration environment is not as trivial. This is a non-critical issue that only occurs for BPEL engines that support process delegation; nevertheless one should be aware of this limitation. Furthermore the delegation of branches that contain shared process variables can also become a source of problems.

V. Case Study

We present as an example the case of a digital newsstand website that allows registered users to view a range of newspapers as they are published. The website receives PDF files from the publishers which need to be converted into an image format (in the case JPEG) for display purposes and whose text must be extracted for search purposes. As part of the submission process there is the need to invoke several services (image conversion, resize and white space cropping, text extraction, OCR and storage). The sequence of services to be performed can be organised as a business process (a functional diagram of it can be found in Figure 3). The initial input of this process is a PDF file and an XML document with associated metadata. The process starts with two parallel branches. The first branch extracts text from the initial PDF, while the second branch converts the PDF into an array of PNG files and crops the white space around the generated PNG files. From this point on the process once again splits into two parallel branches, one that is responsible to convert the PNG files to TIFF format (the OCR service requires that the input files to be in TIFF format) with the resulting files being fed to an OCR service. Meanwhile the other branch converts the PNG files to the final JPEG format (with the appropriate screen resolution). The final activity consists on the use of a service that will store all non-intermediate files that were generated by the process.

In the worst case scenario each of the blocks in Figure 3 represents a service in a different peer. In a centralised orchestration this represents a significant amount of data that must be sent through the network. The total amount can be calculated by $T = 3S_{PDF|} + 5S_{PNG} + 2S_{TIF} + 2S_{OCR} +$

 ISBN: 978-1-61208-173-1

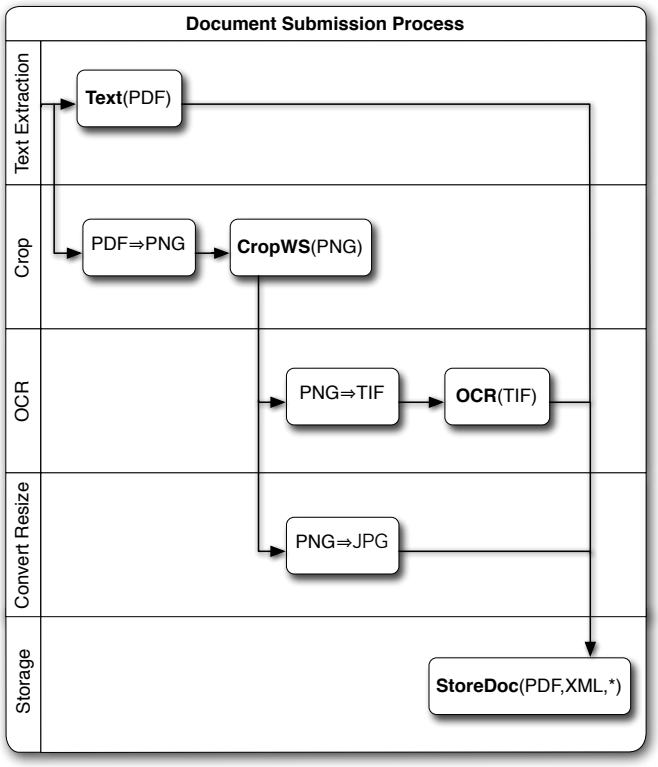

Figure 3: Functional diagram of the document submission process.

$2S_{TXT} + 2S_{JPG} + S_{XML} + S_{ID}$ where S_x represents the message size of the transmission of X. In a branched process such as this one it is possible to perform a simple optimisation by delegating an entire branch of activities. If one of the peers that provide an image conversion service also provides BPEL execution capabilities, we can reduce data that must circulate through the network by delegating all the activities in the "OCR" band of Figure 3. The call to the image conversion service would be a local one, thus avoiding sending the intermediary TIFF files that result from the image conversion back to the original caller trough the network. In this particular digital newsstand application the intermediary TIFF are about 3MB each, which for a 40-page newspaper would result on not having to send 120MB of data through the network if this optimisation is applied.

In an optimal scenario where all peers have BPEL execution capabilities it would be possible to apply the partition algorithms that were previously mentioned in Section II. It should be noted that some delegation could prove to be counter-productive. If there were services just before the storage service, dedicated to provide unique identifiers, produce checksums or calculate hashes based on the metadata of the new document, delegating the orchestration of one of those services and the storage to those providers would actually increase the network usage since instead of invoking the first service, receive its results and send everything to the storage service, both the initial PDF and final JPEG images would have to be sent first to the new service provider and only after to the storage service. In this scenario instead of transferring $T_{final} = 2S_{XML} + 2S_{ID} + S_{PDF} + S_{JPG}$ we would be transferring $T_{final} = 2S_{XML} + S_{ID} + 2S_{PDF} + 2S_{JPG}$. If we apply the criteria that was proposed earlier, since the id/checksum service return type will probably be of a simple type (that we can safely predict to be small when compared with byte arrays that hold the original PDF or JPEG files) no delegation would occur, thus avoiding generating extra network traffic. Other optimisations could be considered, such as trying to merge activities in peers that provide multiple consecutive services. This optimisation could greatly reduce network traffic but it would be difficult to analyse its beneficial impact if factors such as throughput were also to be considered. In this particular example it could also be possible to further explore the P2P network by using it as a storage medium, which would allow the storage service to be executed by any available peer.

VI. Conclusion and Future Work

In this work we have discussed and implemented strategies to better integrate web services, SOA, and business process execution in peer-to-peer environments. Our proposed strategy allows the deployment of replicated web services in multiple peers without requiring any major change to the services themselves or to the clients. It accomplishes this with the use of a small proxy that allows access to services hosted on the P2P network to clients that are not aware of the presence of the P2P network providing the following benefits:

- Does not require modifications to existing services or clients.
- Does not mandate a specific technology for the development or deployment of new services.
- Provides a way to tap into otherwise wasted computational resources.
- Transparently manages access to replicated web services.

While it can be argued that the adopted strategy adds an additional step that has the potential of slowing the access to a given service in low load scenarios, it also has the potential to shield clients in high load scenarios, provided that multiple peers provide an entry proxy to the P2P network and that popular services are properly replicated. Business process execution engines can use the replicated services transparently, providing an added layer of reliability to business process execution, yet a P2P-aware business process engine can obtain the following additional benefits:

- Delegate parts of a business process to other engines.
- Directly select service providers to distribute workload.
- Reduce the amount of data that must be sent through the network.

Of the previously stated benefits, process delegation and reduced network data transfers depend directly on the existence of additional business process execution engines in the P2P network. Since we are able to discover them in runtime we can alleviate the "BPEL in every node" assumption present

 ISBN: 978-1-61208-173-1

in previous works [15][16] and avoid the need to perform the partition process before actually deploying the business process [17][18] thus ensuring that there is an "always working" solution for the execution of the business process, even when no other business process execution engines are available in the P2P network.

A point to be improved in the service discovery mechanism is that the current approach is still based on the traditional WSDL, which only provides a syntactic description of the service. While this description provides enough information to discover and execute a service based only on an incoming request, the use of a semantic description would enable more refined queries. One of the goals of a semantically improved discovery mechanism would be to further improve service execution resilience by allowing the exchange of a missing or faulty services with semantically equivalent ones (or composition of multiple services if applicable) in an automated way (a similar approach can be seen in PANIC [20]). This requires knowledge about what the service does that cannot be obtained from the current WSDL description, but might be available with the introduction of WSDL-S or OWL-S. The additional knowledge gathered about each service could also be used to improve peer selection process, and open the way to more efficient process delegation strategies. Two peers can be providing the exact same set of services, yet due to differences in hardware the performance obtained from each one can be very different, making one of those peer a less desirable choice to perform some classes of services. As an example a storage service would benefit from being executed on a peer with more available storage space while a video conversion service would benefit from being executed in a peer with dedicated encoding hardware. By taking into account the requirements of the service when selecting the service-providing peer it is possible to promote an even more rational use of available hardware resources. It should be noted that having advanced peer/service selection algorithms is a an important step that to achieve further performance gains, and is an important research topic [21][22][23].

As was said before, the proposed P2P service discovery mechanism assumes that a web service is going to be described by a WSDL. While this assumption holds true for SOAP based web services, it collapses when dealing with REST services. This has a double impact since it prevents the use of REST services in BPEL processes and prevents proper integration of REST services with our P2P network. Taking into account the work described in [24] where REST services have been described in WSDL and in [25], where REST services were composed into BPEL processes (with the use of extensions) we believe that it will be possible to support REST services in parallel with SOAP based web services using our proposed P2P architecture with only minor modifications.

ACKNOWLEDGEMENT

This work was funded in part by the Portuguese Foundation for Science and Technology grant SFRH/BD/62554/2009.

REFERENCES

[1] I. J. Taylor and A. Harrison, *From P2P to Web Services and Grids. Peers in a Client/Server World*. Springer, 2005.

[2] E. A. Marks and M. Bell, *Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, June 2006.

[3] M. Bichler and K.-J. Lin, "Service-oriented computing," *Computer*, vol. 39, pp. 99–101, 2006. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MC.2006.102

[4] O. Zimmermann, V. Doubrovski, J. Grundler, and K. Hogg, "Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned," in *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, ser. OOPSLA '05, 2005, pp. 301–312. [Online]. Available: http://doi.acm.org/10.1145/1094855.1094965

[5] JXTA Community Board, "Jxta homepage," 2010. [Online]. Available: http://jxta.kenai.com/

[6] ——, *JXTA v2.0 Protocol Specification*, 2007. [Online]. Available: http://jxta.kenai.com/Specifications/JXTAProtocols2_0.pdf

[7] M. Amoretti, "Enabling peer-to-peer web service architectures with jxta.soap," in *IADIS e-Society 2008*, 2008.

[8] A. S. Foundation, "Web services - axis," 2005 Published. [Online]. Available: http://axis.apache.org/axis/

[9] M. P. Papazoglou, B. J. Krämer, and J. Yang, "Leveraging web-services and peer-to-peer networks," in *Proceedings of the 15th international conference on Advanced information systems engineering*, ser. CAiSE'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 485–501.

[10] Z. Zhengdong, H. Yahong, L. Ronggui, W. Weiguo, and L. Zengzhi, "A p2p-based semantic web services composition architecture," in *IEEE International Conference on E-Business Engineering*, oct. 2009, pp. 403–408. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICEBE.2009.63

[11] S. Dustdar and L. Juszczyk, "Dynamic replication and synchronization of web services for high availability in mobile ad-hoc networks," *Service Oriented Computing and Applications*, vol. 1, no. 1, pp. 19–33, 2007. [Online]. Available: http://dx.doi.org/10.1007/s11761-007-0006-z

[12] L. Juszczyk, J. Lazowski, and S. Dustdar, "Web service discovery, replication, and synchronization in ad-hoc networks," in *Proceedings of the First International Conference on Availability, Reliability and Security*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 847–854. [Online]. Available: http://dx.doi.org/10.1109/ARES.2006.143

[13] S. Dustdar and M. Treiber, "Integration of transient web services into a virtual peer to peer web service registry," *Distributed and Parallel Databases*, vol. 20, pp. 91–115, September 2006. [Online]. Available: http://dx.doi.org/10.1007/s10619-006-9447-1

[14] J. G. R. Jr., G. T. do Carmo, M. T. Valente, and N. C. Mendonça, "Smart proxies for accessing replicated web services," *IEEE Distributed Systems Online*, vol. 8, no. 12, 2007.

[15] M. G. Nanda, S. Chandra, and V. Sarkar, "Decentralizing execution of composite web services," *SIGPLAN Not.*, vol. 39, pp. 170–187, October 2004. [Online]. Available: http://doi.acm.org/10.1145/1035292.1028991

[16] F. Montagut and R. Molva, "Enabling pervasive execution of workflows," in *Collaborative Computing: Networking, Applications and Worksharing, 2005 International Conference on*, 2005, p. 10 pp. [Online]. Available: http://dx.doi.org/10.1109/COLCOM.2005.1651227

[17] R. Khalaf, O. Kopp, and F. Leymann, "Maintaining data dependencies across bpel process fragments," *International Journal of Cooperative Information Systems.*, vol. 17, no. 3, pp. 259–282, 2008. [Online]. Available: http://dx.doi.org/10.1142/S0218843008001828

[18] R. Khalaf, "Supporting business process fragmentation while maintaining operational semantics: a bpel perspective," Ph.D. dissertation, University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, 2008. [Online]. Available: http://elib.uni-stuttgart.de/opus/volltexte/2008/3514/

[19] M. Pereira, M. Fernandes, J. A. Martins, and J. S. Pinto, "Service oriented p2p networks for digital libraries, based on jxta." in *ICSOFT 2009 - Proceedings of the 4th International Conference on Software and Data Technologies*, B. Shishkov, J. Cordeiro, and A. Ranchordas, Eds. INSTICC Press, 2009, pp. 141–146.

[20] J. Hunter and S. Choudhury, "Panic: an integrated approach to the preservation of composite digital objects using semantic web services,"

 ISBN: 978-1-61208-173-1

International Journal on Digital Libraries, vol. 6, no. 2, pp. 174–183, 2006. [Online]. Available: http://dx.doi.org/10.1007/s00799-005-0134-z

[21] F. Xhafa, L. Barolli, T. Daradoumis, R. Fernández, and S. Caballé, "Jxta-overlay: An interface for efficient peer selection in p2p jxta-based systems," *Computer Standards & Interfaces*, vol. 31, no. 5, pp. 886–893, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYV-4S2TRXK-1/2/ac83a5f48d7beeac9f84cb21e6182d83

[22] N. C. Mendonça and J. A. F. Silva, "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*. New York, NY, USA: ACM, 2005, pp. 1704–1708. [Online]. Available: http://doi.acm.org/10.1145/1066677.1067062

[23] S. Dykes, K. Robbins, and C. Jeffery, "An empirical evaluation of client-side server selection algorithms," in *Proceedings of INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, March 2000, pp. 1361–1370.

[24] L. Mandel, "Describe rest web services with wsdl 2.0: A how-to guide," IBM, 2008. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-restwsdl/

[25] C. Pautasso, "Bpel for rest," *Business Process Management*, vol. 5240, pp. 278–293, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85758-7_21

 ISBN: 978-1-61208-173-1