Role of Triple Space Computing in Semantic Web Services*

Brahmananda Sapkota, Edward Kilgarriff, Christoph Bussler

DERI, National University of Ireland, Galway, Ireland
<Brahmananda.Sapkota, Edward.Kilgarriff, Chris.Bussler>@deri.org

Abstract. This paper presents Triple space computing and described the role it can play in bringing Semantic Web Services to the next level of maturity. In particular the shortcomings of current Semantic Web services are identified and the role of Triple Space computing can play in resolving these shortcomings is described.

1 Introduction

In the recent years the Internet has become a popular business hub. Lately, efforts have been put to bring machine level communication through the use of Semantic Web technology [1]. Initiatives such as [2], [3], [4] are aiming at enabling seamless integration of business processes through the combination of Semantic Web and Web services thus supporting automation of service discovery, mediation and invocation [2]. The synchronous communication technology used in Semantic Web services (SWS) requires both parties to be 'live' at the time of their interaction reducing scalability [5]. In a distributed scenario, such communication can be costly because of network congestion, low processing power, third party invocation, etc. Timing in communication is also important as most business transactions are time bounded in the business world. The time difference problems can be resolved through a priori agreement but in automated business process reaching such an agreement beforehand is unpractical.

In the past, tuple-based computing was largely investigated for introducing communication asynchrony between processes in parallel computing. Linda [6], TSpace [7] approaches were intended to facilitate communication between applications that require distribution of data. Messaging techniques like publish-subscribe, emails, etc., were introduced to support communication asynchrony [8]. These techniques, however, do not provide semantic support for communication. In this paper, we identify some of the roles of a new communication approach called Triple Space Computing [9] in the context of Semantic Web Services focusing on the communication aspects. Other aspects of Semantic Web Services such as mediation, discovery, etc are out of the scope of this paper.

* This material is based upon works supported by the EU funding under the projects DIP (FP6 - 507483) and by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131. This paper reflects the author's views and the Community is not liable for any use that may be made of the information contained therein.

This paper is structured as follows. In Section 2, similar approaches are discussed. Section 3 describes SWS, discusses its missing features and other requirements. The Triple Space Computing (TSC) architecture is presented in Section 4. Section 5 discusses the role of TSC in SWS paradigm. The paper is concluded in Section 6 presenting intended future works.

2 Similar Works

The shared space communication paradigm had already taken the attention of both industry and academia of which some relevant ones are described below.

Message Queuing technology is designed to support communication asynchrony in distributed environment [10]. In this scheme, sending application puts (enqueue) messages on to the queue where as the receiving application extracts (dequeue) them from that queue [11]. The flexibility and scalability of this technique is limited by queue size and their location [12].

The **Publish-Subscribe** interaction mechanism is aimed at supporting communication asynchrony through publication of an event and a subscription to that event [8]. The events registered by the publishers are propagated asynchronously to the subscribers of that event. Though, it has been largely recognized as a promising communication infrastructure in distributed environment [13], the importance of semantics in distributed communication is largely ignored.

The **Tuple-Based Communication** concept introduced for Linda by Gelernter [6] allow inter processes communication by writing and reading information (i.e. tuple) to and from a shared space called Tuple Space. The tuple space is persistent and the communication is blocking and transient. Thus the reading process waits until the matching tuples are available [14]. The tuple matching is content based, nesting is not supported by the used data model, and physical representation of tuple space is difficult. Tuples having the same number and order of fields but with different semantics can not be matched [15] thus they do not scale [16].

Semantic Tuple Spaces is introduced in sTuple [16]. It envisions to cross-fertilize the Semantic Web and Tuple Space technologies. The tuples in sTuple are similar to that of JavaSpace[17] where one of the fields must contain a data item semantically defined using DAML+OIL [18]. The Semantic Tuple Spaces enables semantic matching on top of object based polymorphic matching. The underlying data model of sTuple consists of both Semantic and and non-Semantic technologies thus inherits all the problems of tuple-based communication.

3 Semantic Web Services

Semantic Web services aim to realize seamless integration of applications, semantically enhanced information processing and distributed computing on the Web through the combination of Semantic Web and Web services. Ontologies are vital for semantic interoperability and advanced information processing, Web services

enable computation over the Web. Web service technology is built around Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). SOAP facilitates message exchanging between Web services [19]. WSDL [20] is used for describing access interfaces of a Web service. UDDI registries [21] are used to advertise service descriptions. These technologies work at a syntactic level thus do not support dynamic inspection and integration of Web services. In this context, Semantic Web services (SWS) exploits Ontologies to automate Web service discovery, composition and invocation, thus enabling seamless interoperation between them with minimum human intervention.

The use of SOAP as communication protocol is one of the problems of (Semantic) Web services. Supporting asynchronous communication, it requires both the service owner and the invoker be available at the same time. The current (Semantic) Web services are moving apart from the Web paradigm that allow independent information publishing and retrieval [9]. In addition, information can not be reused and read multiple times.

In addition to the existing ones, the basic requirements of the (Semantic) Web service technologies are: ***Persistent Publication*** - the reuse of information is possible only if the information is published persistently. ***Communication Asynchrony***- the information should be made accessible asynchronously in order to allow SWS to work in Web scale. ***Reuse and Provenance*** - in order to avoid conflict of communication, logging is needed. For example, in case of Web service discovery and composition, if similar request were processed before the previous result can be reused.

4 Triple Space Communication

The fundamental concept of TSC (Triple Space Computing) is to provide a semantic communication infrastructure such that machine-to-machine interaction can be achieved at Web scale. It was first envisioned in [9] and ideally, it extends traditional tuple-based computing in the direction of RDF. It aims at providing a persistent shared semantic space called triple space where application write and read information to asynchronously communicate with each other. Providing communication asynchrony, TSC provides autonomy to applications in terms of *time*, *reference*, *location* [22], [9].

The visible advantage of using TSC in Web service paradigm are threefold. Firstly, the Web service paradigm is brought onto the Web paradigm, i.e., the persistent publication of information. Secondly, the communication is asynchronous [9]. Thirdly, TSC provides middleware support hiding from internal application complexities.

4.1 TSC Architecture and Interaction

Building on proven technologies and their combination, TSC has (Figure 1) a simple but powerful architecture.

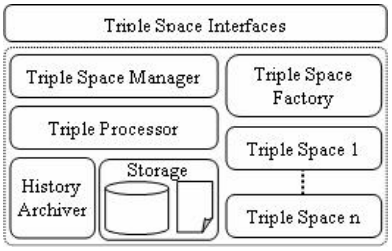

Fig. 1. Triple Space Computing Architecture

It consists of loosely coupled components and is based on SOA design principles. It uses RDF as an underlying data model. The architecture presented here is the extension of a minimal triple space architecture defined in [22].

Triple Space Interfaces are defined to enable applications to interact with TSC. It allows applications to *create* and *destroy* triple spaces, to *write* to and *read* from a triple space, to retrieve *history* of communication. The read, write, and history operations are relative to the triple space, i.e., these operations can be performed on a triple space that already exists. Applications can invoke these operations through HTTP. The semantics of these operations are defined as follows:

create (TSID) **returns** SUCCESS | TSID_ALREADY_EXISTS
destroy (TSID) **returns** SUCCESS | TSID_DOES_NOT_EXIST
TSID.write (Triple*) **returns** SUCCESS | TSID_DOES_NOT_EXIST
TSID.read (Triples) **returns** Triple∗ | TSID_DOES_NOT_EXIST
TSID.history (Triples) **returns** Triple∗ | TSID_DOES_NOT_EXIST

Where TSID and Triple∗ represent the triple space ID and zero or more RDF Triples respectively. The triple space ID is unique with respect to the URL [23] where TSC infrastructure is hosted. The parameters to the write and history operations are the queries expressed in RDF triples. The parameter to read operation is the Triple(s) to be stored in the triple space identified by TSID. The **Triple Space Manager** receives a request from the applications and informs one of the necessary components to carry out their operation. If the operation invoked is *create* then the **Triple Space Factory** is informed to create a new triple space. Triple Space Factory creates a new triple space having id TSID, if it is not already created. The **Triple Processor** component implements the functionality of the *write*, *read* and the *history* operations. The storage space can be anything from a simple file system to an advanced RDF stores but YARS (Yet Another RDF Store [24]) is used in the current implementation. The **History Archiver** component records every flow of triples to and from triple space.

The information in the Web-like open environment should be kept securely. At present only primitive security functionalities such as access policy and information provenance have been implemented. We acknowledge the need of more secure strategy and is left for future work.

5 Role of TSC in SWS

TSC brings machine-to-machine (Semantic) Web service communication to the Web scale. TSC has many roles to play in (Semantic) Web service technologies, the most notable being:

a **Communication Asynchrony** - Current (Semantic) Web service technologies lack scalability because the communication is synchronous. TSC provides communication asynchrony thus allowing SWS applications to communicate without knowing each other explicitly.

b **Persistent Publication** - Current SWS technologies are difficult to manage and do not scale as it lacks persistent publication. Thus making service discovery a tedious, tiresome and time consuming task. Triple space provides a persistent storage where Web service descriptions, Ontologies and access interfaces can be published persistently. The published information can be uniquely identified by their URIs. In addition, the published information can be read similar to that in the Web. Thus significantly simplifying communication set up between applications supporting seamless integration and computations on a Web scale.

c **Scalability** - Enabling asynchronous communication and using semantically enriched resources much of the process can be automated achieving scalability significantly.

d **Complexity Management** - The communication complexity is managed efficiently in TSC through persistent publication and synchronous communication. If the success message is not returned, the writing or reading application will know that the request need to executed again. Thus, information will never be lost in case of system failures.

e **Communication Archiving** - TSC allows the storing of a history of all flow of messages and requests to and from the triple space. This enables monitoring of communicating applications helping in reuse of already available Web services.

f **Provenance** - TSC provides a mechanism to ensure that the information arrived from the said source and time helping in reducing denial of service or non-repudiation.

g **Middleware Support** - Due to the fact of the above mentioned support, TSC also plays a role of middleware in SWS.

6 Conclusions and Future Work

The traditional Web services are synchronous and destroying. The information exchanged between applications can not be reused and read multiple times. TSC provides a semantic communication infrastructure where information can be published persistently. All communication are asynchronous and information exchanged are described using RDF Triples. Each triples are identified through URIs. In this paper, we presented what role TSC can play in Semantic Web services. By providing persistent shared space, TSC brings Semantic Web services

to Web scale. It plays the role of middleware that hides internal complexity from applications allowing them to communicate without changing their internal storage structure. Though it provides support for resolving communication disputes much more security needs have to be supported. Advanced support for security will be incorporated in future architecture.

References

1. Berners-Lee, T., et. al: The Semantic Web. Scientific American (2001)
2. Roman, D., Lausen, H., U.Keller, eds.: Web Service Modelling Ontoloty (WSMO). WSMO Deliverable, version 1.2 (2005)
3. Martin, D., et. al: Bringing Semantics to Web Services: The OWL-S Approach. In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (2004)
4. Patil, A., Oundhakar, S., Verma, K.: METEOR-S: Web Service Annotation Framework. In Proceedings of World Wide Web Conference (2004)
5. Engleberg, I., Wynn, D.: Working in groups: Communication Principles and Strategies. Houghton Mifflin (2003)
6. Gelernter, D.: Mirror Worlds. Oxford University Press (1991)
7. TSpace, http://www.icc3.com/ec/tspace/.
8. Eugster, P., et. al: The Many Faces of Publish/Subscribe. ACM Computing Surveys (2003)
9. Fensel, D.: Triple Space Computing. Technical Report, Digital Enterprise Research Institute (DERI) (2004)
10. Blakeley, B., et. al: Messaging and Queuing using MQI. McGraw-Hill (1995)
11. Gawlik, D.: Message Queuing for Business Integration. eAI Journal (2002)
12. Leymann, F.: A Practioners Approach to Database Federation. In Proceedings of 4th Workshop on Federated Databases (1999)
13. Huang, Y., Garcia-Molina, H.: Publish/Subscribe in a Mobile Environment. In Proceedings of MobiDE (2001)
14. Matsouka, S.: Using Tuple Space Communication in Distributed Object-Oriented Languages. In Proceedings of OOPSLA '88 (1988)
15. Johanson, B., Fox, A.: Extending Tuplespaces for Coordination in Interactive Workspaces. Journal of Systems and Software **69** (2004) 243–266
16. Khushraj, D., et. al: sTuple: Semantic Tuple Spaces. In Proceeding of MobiQuitous '04 (2004)
17. JavaSpace, http://java.sun.com/products/jini/specs.
18. DAML+OIL, http://www.daml.org/2001/03/daml+oil-index.html.
19. Gudgin, M., et. al, eds.: SOAP Version 1.2. W3C Recommendation (2003)
20. Chinnici, R., et. al, eds.: WSDL Working Draft. W3C (2004)
21. Clement, L., et. al, eds.: UDDI Spec. Technical Committee Draft. OASIS (2004)
22. Bussler, C.: A Minimal Triple Space Computing Architecture. In Proceedings of WIW 2005 (2005)
23. URL, http://www.cse.ohio-state.edu/cs/Services/rfc/rfc-text/rfc1738.txt.
24. Yet Aother RDF Store, http://sw.deri.org/2004/06/yars/yars.html.