A Conceptual Architecture of Distributed Web Services for Service Ecosystems

Chen Wu, Elizabeth Chang
Curtin University of Technology
School of Information Systems
Perth, Western Australia, 6845, Australia
E-mail: chen.wu@cbs.curtin.edu.au, change@cbs.curtin.edu.au

Abstract

The classical nature of centralized web services client/server architecture brings about many associated problems such as performance, bottlenecks, and scalability. Meanwhile some peer-to-peer research proposals have not respected the existing web services standards, thus leaving the compatibility and feasibility issue open. Hence existing web services architecture is not designed to accommodate large-scale, distributed, internet-wide applications. In this paper, we propose a conceptual architecture for distributed web services to enable web services-based software systems to "work well" in the heterogeneous and highly distributed service ecosystem environment.

1 INTRODUCTION

Current web services architecture is based on the classic broker architectural styles evolving from traditional distributed object technologies. The primary problem of such centralized broker architecture comes from the centralized indexing scheme provided by web services registry – UDDI. It does not scale well because the number and physical distribution of the UDDI clients can quickly overwhelm this centralized configuration and can lead to serious performance bottlenecks [3]. Moreover, the possible storage of vast numbers of advertisements on centralized registries hinders the timely updates, thus it is questionable whether centralized registries will scale up to the needs of web services [4]. Some related research attempted to introduce peer-to-peer network protocol into the existing web services architecture [5, 6, and 7]. However, such pure P2P architectural style has one critical problem: no existing registries (e.g. UDDI) are utilized. Hence the feasibility and compatibility of their research is questionable since they require the complete abolition of existing service discovery mechanisms, which are already well accepted as normative industry standards in web services practice. Moreover, it introduces more security and trust risk inherent in P2P computing paradigm.

Hence it is our belief that the existing web services architecture and related research falls short of support for internet-wide distributed applications in a native manner. In this paper, we propose a Distributed Web Services Architecture for Service Ecosystem (DWSASE) composed of selected architectural styles summarized in our previous work [11].

The remainder of this paper is organized as follows: Section 2 gives a brief overview of architectural styles in current web services architecture literature. In section 3, we introduce the architecture by elaborating its architectural topology, components, and connectors.. Section 4 elaborates the dynamic composition steps in super-broker. Section 6 concludes the paper.

2 Architectural Styles Summary

Since the proposed architecture aims at supporting the whole lifecycle of web services, Figure 1 depicts the corresponding web services lifecycle – service discovery, service invocation and service composition – that can be applied by different architectural styles in [11].

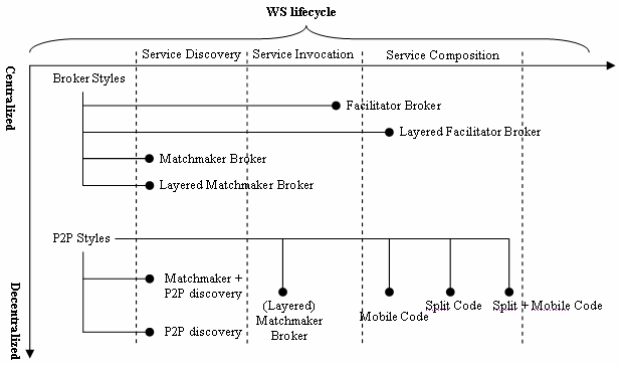

Figure 1. Summary of Architectural Styles

In Figure 1 the horizontal axis encompasses the WS-lifecycle and the vertical axis represents the degree of architectural decentralization. In [11] we argue that broker

and P2P are two extremes for the architectural design. They have their own benefits as well as drawbacks. Both are necessary to achieve the system objectives – the web services architectural properties identified in [11]. On one hand, it is imperative to have central broker facilitating service discovery, binding and interaction due to the heterogeneity, trust, and security reasons among highly autonomous web services from different organizations across the Internet; on the other hand, broker style raises scalability and performance issues since it originates from traditional distributed object systems where components are confined in a controlled domain with limited boundaries. Hence how to make trade-off during architecting the appropriate design based on the business context becomes a tough question associated with complex activity posed on web services architect. In the next section, we will describe how to leverage both styles" advantages while reducing their negative aspects and conflicts by proposing a hybrid architecture design.

3 DWSASE Architectural Design

To describe our architectural design, we will follow the architectural approach proposed in [2], where architecture style is defined by component, connector, data, and configuration. One can regards configurations as a set of constraints on how architectural elements (connectors, components, and data) can be combined [8]. In particular, such constraints on elements interactions can be expressed in the form of topology [1].

3.1 Contexts and Overall Topology

Here we introduce the contexts within which our architectural components are constructed. Such context can be seen as a consistent service ecosystem, which in turn can be further divided into three levels explained as follows:

Global-Space – Global space refers to area that does not belong to any existing domains. It reflects the initial and existing web services architecture style – centralized broker, where service consumers and providers centre on a single global registry Global-space is a rather course-grained habitat for web services.

Domain – A virtual community where related web services come together in an attempt to provide quality and added-value services to their customers. Examples of domains are: PC hardware industry in Western Australia, Logistics industry in New Southern Wales. Domain is a habitat where services are willing to actively cooperate with each other rather than passively coordinated by third party middleware or enforced by other protocols.

Dynamic-Alliance – A close bond established in an ad-hoc manner among trusted web services within the same domain driven by certain business factors. Services within alliance communicate with each other in a peer-to-peer manner.

Composing these contexts together, the architectural topology of DWSASE is depicted in Figure 2.

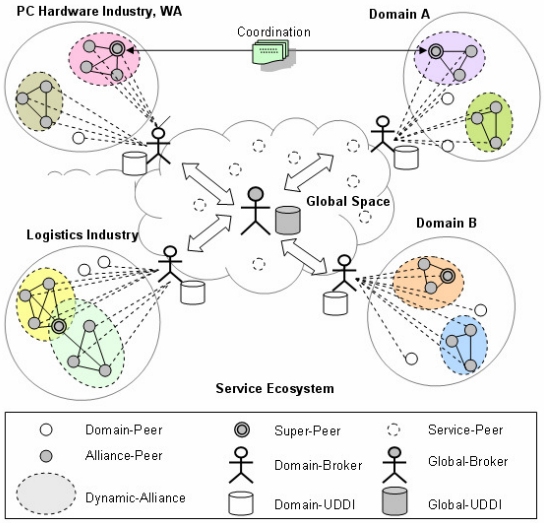

Figure 2. Architectural Topology View

3.2 Architectural Components

The core components for this architecture are: Service-Peer, Domain-Peer, Alliance-Peer, Super-Peer, Domain-Broker, Domain UDDI, Global-Broker, and Global-UDDI.

Service-Peer – an ordinary service provider or/and service consumer scattered in the global-space. They are able to find each other through the global UDDI, and bind, invoke each other through the fundamental functionalities summarized as follows:

- Basic UDDI client to interact (i.e. inquiry, publication, subscription) with UDDI server.
- The mechanism to wrap the native local resource using web services.
- The runtime of web services to be invoked by other services.
- Message handling for both inbound and outbound message (SOAP in particular) processing.

Domain-Peer – A kind of service-peer located within a domain where fine-grained service environment are fostered. Peers within the same domain are ready to collaborate with selected partner peers to provide quality and added-value services in response to the dynamic requirements from customers based on their own capabilities and willingness. In order to become a Domain-Peer, a qualified service-peer has to be able to use the "join" speech act defined in certain domain broker protocol

to explicitly notify the domain broker about its joining action, which, if succeeds, will activate a series of events. For instance, the domain-peer needs to register (i.e. use the UDDI Publication API set) the detailed service meta-data with the Domain-UDDI maintained by the Domain-Broker. The subsequent events are discussed in the following sub-section Domain-Broker component. Domain-Peer wraps the local resources into services and facilitates the service invocation with other web services.

Technically, migrating service-peer to domain-peer necessitates the following additional functionalities:

- Supporting the Domain-Broker protocol command.
- Providing agile service invocation mechanism so that service consumer can bind and interact with selected web service provider across the organization boundary.

Domain-Broker – As mentioned earlier, Domain-Broker manages Domain-Peers as well as provides some crucial add-on services to Domain-Peers inside the domain. It handles the joining and leave request from Domain-Peers, generates matching tables for each Domain-Peer, and maintains the transaction history data for Domain-Peers as well. Formative domain protocol is employed inside each domain to allow Service-Peer (service provider and/or consumer) 'join' and 'leave' a particular domain for some reason. Once a Service-Peer **SPnew** turns into a new Domain-Peer **DPnew**, it is firstly granted privilege to register (i.e. apply the UDDI Publication API set) its detailed service metadata with the Domain-UDDI. If the metadata is found entirely new to this domain, the Domain-Broker creates new QoS entry for **DPnew** in the QoS database maintained by the DSS module. Otherwise, related QoS and trustworthiness value can be obtained directly from existing records stored in the QoS database for future service selection processing.

Suppose set $DP = \{dp_1, \mathrm{K}\ , dp_n\}$ where DP represents all the Domain-Peers in the current domain. The Domain-Broker then propagates DPnew's service metadata (mainly high-level data such as name, interface, classification, etc.) to a set of Domain-Peers $IDP \subset DP$, where

$$IDP = \left\{ dp_i \mid dp_i \in DP \,\&\, dp_i \, is\, a\, potential\ \ consumer\, of\ DP_{new} \right\}$$

The Domain-Broker needs to calculate the potential consumer list for such propagation by comparing DPnew's service metadata with IDP's Service Request Subscriptions (SRS) which include QoS requirement as well as functional requirements. Each $idp_i \in IDP$ is able to check the detail service metadata by querying Domain-UDDI and DSS module before DPnew can be appended to its local matching table. For instance, it may have specific QoS requirement at different time slot which is not publicly stated in their SRS. Meanwhile, the Domain-Broker also generates the matching table for **DPnew** itself. This matching table stores a set of Domain-Peers $PDP \subset DP$ where

$$PDP = \{dp_i \mid dp_i \in DP \,\&\, dp_i \, is\, a\, potential\ \ provider\, of\ DP_{new}\ \}$$

The potential provider is meant each $pdp_i \in PDP$'s published service matches with DPnew's intent of consumption and potential requirements – both Context and QoS – embedded in its SRS. DPnew can furthermore choose the GUI-based matching list so that the human (i.e. the decision maker) can mediate the service selection process and update the matching table interactively. As a result, each one of the involved $idp_i \in IDP$ as well as DPnew obtains an updated local matching table which is afterwards used for service interaction as a peer-to-peer routing mechanism. On leaving the domain, the DPnew notifies the Domain-Broker, who will then take the following steps 1) retrieve transaction history data (e.g. QoS report) from the involved $idp_i \in IDP$ and report them to DSS module for further review; 2) propagate the leaving message to all involved $idp_i \in IDP$ and $pdp_i \in PDP$, which will in turn perform certain routine operations accordingly (e.g. removing the entry from the matching table); 3) unregistered the DPnew from the Domain-UDDI.

Figure 3 indicates the module design of Domain-Broker, which consists of three core components: the subscription queue, the matching engine, and the UDDI client. The domain interface corresponds to the Domain-Broker protocol, while the service interface corresponds to the Global-Broker protocol.

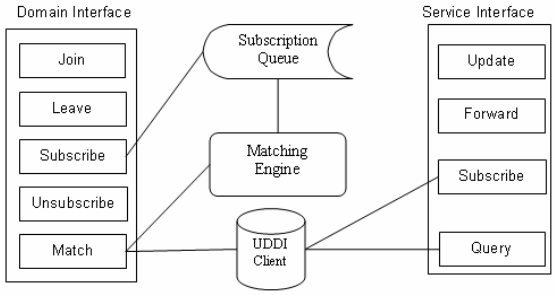

Figure 3. Domain-Broker Architecture

Alliance-Peer – A kind of Domain-Peer dwelling within a certain Dynamic-Alliance. When a concrete requirement from a customer is presented to a specific domain-peer who is unable to suffice such requirement alone, this domain-peer (normally referred to as "Super-Peer" in the next sub-section) attempts to initiate a dynamic-alliance by selecting several other peers based-on certain criteria (e.g. relevance to the current requirement, QoS, current loading, etc) from its local matching table. If, somehow, this domain-peer is unable to select any other

Domain-Peers from its local matching table it will generate a new service request sent to the Domain-Broker in the form of asynchronous subscription. The Domain-Broker will in turn treat such subscription as if from a newly joined domain-peer (i.e. this domain-peer), thus repeating the process stated in last sub-section. Once such subscription is matched with some publication, this domain-peer will again attempt to initiate a dynamic alliance. Otherwise, Global-Broker will be resorted to forward such subscription to other relevant Domain-Brokers. Once confirmed message from each one of those Domain-Peers is received, the ad-hoc alliance is thus formed without replying on any support or permit from the Domain-Broker. Here we assume that all Domain-Peers are capable of commanding the same alliance peer-to-peer protocol which dictates the basic actions (e.g. "join", "leave") necessary to create, maintain, and remove such dynamic-alliance. Once the alliance is formed, alliance-peers will be able to work autonomously by exchanging messages with each other to fulfill the end user requirements introduced by the alliance-initiator. Meanwhile, it is the alliance-initiator's responsibility to discompose current dynamic-alliance if necessary.

During the process of message exchanges, each alliance-peer will make evaluation for its alliance-peer partner by storing at local site some value against some metrics such as trustworthiness or QoS value. Some alliance-peers might find that trustworthiness or QoS value are out of their expectations or exceed the threshold for particular peer partners. In such circumstances, they have the channel to file to Domain-Broker. If two alliance-peers have difficult in interacting, they may resort to super-peer to mediate such heterogeneity inherent in autonomy nature of web services.

It is worth noting that not every domain-peer will become an alliance-peer since dynamic-alliance relies on the external concrete requirement, whose versatility makes it difficult to find appropriate service provider with desirable capabilities each time. On the other hand, Domain-Broker might trigger the formation of some relatively stable and useful "static-alliance" by imitating some external requirements in the form of service request subscription. To enable a domain-peer to become an alliance-peer, additional functionalities are necessary:

- Supporting certain alliance peer-to-peer protocol.
- Supporting certain the run-time execution of business protocol such as BPEL4WS
- Providing service process execution run-time environment so that process instance or specification can be interpreted and executed in the local site.

Super-Peer – A kind of Alliance-Peer that initiates the formation of a particular Dynamic-Alliance by sending invitation messages to selected partner peers in a manner consistent with alliance peer-to-peer protocol. Super-Peer to the dynamic-alliance is as Domain-Broker to the domain; its major responsibility includes maintaining the ad-hoc alliance – to rearrange the alliance according to end customer requirements. Moreover, super-peer has the capability to negotiate and interact with super-peers on behalf of other dynamic-alliance from alternative domains, thus coordinating two (or more) alliances across domains. Alliance-Peers with adequate computation capacity will be nominated and hence supplied with super-peer capability from Domain-Broker upon their requests upon applying to become the super-peer. However, it is not mandatory for each Dynamic-Alliance to have at least one super-peer to survive the changes from requirements without recreating the alliance repetitively. While that how to nominate and enable a super-peer is a topic beyond the scope of this paper, numerous algorithms and software implementation methods can be leveraged (such as Mobile Agent) during our further prototyping phase of the research. There exist some domain built-in super-peers nominated and enabled by the Domain-Broker to organize the domain under the guidance of the Domain-Broker at the initial phase of the domain formation. In order to become a super-peer, the following extra functionalities have to be added:

- Supporting the basic mediation engine for heterogeneity Supporting the design-time composition of certain service business protocol such as BPEL4WS
- Supporting the process specification split and delivery at run-time
- Supporting the service coordination protocol such as WS-Coordination

Domain-UDDI – A UDDI registry maintained by the Domain-Broker and used by all the Domain-Peers, who have corresponding access to maintain their own service meta-data and retrieve the detailed via Publication API set, low-level service information using the UDDI Inquiry API set (e.g. get_bindingDetail(), get_serviceDetail(), etc.). While domain-uddi is the index for all the domain services, it behaves more like a passive service "hashtable" used for reference rather than an active matchmaker used to match the service providers and consumers. In a word, in this architectural design, service discovery and selection is covered by Domain-Broker in stead of domain-uddi.

Global-Broker – A broker managing the whole broker community consisting of various domain brokers. Global-Broker will not involve detailed issues of service-peers or Domain-Peers, though some of them may publish their service information into the Global-UDDI (explained in the next sub-section). Global-broker only concerns problems issued from Domain-Broker, such as the formation of a specific domain, the nomination of the Domain-Broker, and the propagation of the service request subscription raised from one Domain-Broker towards other relevant Domain-Brokers. However, the routing of service request from one domain to a matched service provider from another domain is the work belongs to Domain-Broker and super-peer rather than the global-broker.

Global-UDDI – A UDDI registry that only stores the indices for two types of components – Domain-Brokers and ordinary service-peers. While Domain-Brokers' index is maintained by Global-Broker, no other components maintain the index of those ordinary service-peers located in the global-space.

3.3 Architectural Connectors

The major architectural connectors that mediate communication among components presented above are: web services communication protocol, broker protocol, alliance peer-to-peer protocol and super-peer protocol.

Web Service Communication Protocol – the foundation communication mechanism for any components in this architectural design. Other connectors are based on WS-Protocol. In particular, it is built on the standard protocols such as HTTP 1.1, SOAP 1.2, WSDL 1.1 and UDDI V3 Programmers API.

Broker protocol – There are basically two types of broker protocol: Domain-Broker protocol and global-broker protocol. Domain-Broker protocol stipulates the following (but not limited to) atomic speech acts: join, departure, subscribe, unsubscribe, and match. Similarly, global-broker protocol includes performatives such as apply, create, destroy and subscribe. Some related work will explain these protocols at length during the later phase of our research.

Alliance P2P Protocol – A P2P protocol mainly used to form and maintain the dynamic-alliance. Relevant actions includes: invite, join, leave, rearrange, etc.

Super-Peer Protocol – It is employed when two (or more) dynamic alliances from different domains are working together. The basic semantics of Super-Peer protocol will be based on the WS-Coordination, a meta-specification that will govern specification – in this case, Super-Peer Protocol – that implements concrete forms of coordination.

WS Business Protocol – A meta-protocol for domain business protocol. Example of WS Business Protocol is the BPEL4WS.

Domain Protocol – A domain specific application-level protocol, describing the detailed and well-accepted conventions regarding the process in a specific industry. One of the examples is the RosettaNet PIPs supply-chain standards in the semi-conductor industry. Domain Business Protocol is mainly employed by connected alliance-peers.

4 Dynamic-Alliance Composition

In this section, we present the services composition process facilitated by the super-peer in the dynamic-alliance. We introduce the alliance composition process illustrated by Figure 4. Upon receiving the user requirement profile, the domain-peer becomes the super-peer for the potential dynamic-alliance. It starts the alliance composition processes including the following five major steps:

Step 1. Function Graph Generation. The domain-peer first evaluates the functional requirements based on the user requested function described by the user requirements profile. From these requirements, the Domain-Broker will generate the function graph, an abstract level view of the composition process specification. Currently, extensive research has been conducted in automatic process generation. In this paper, we assume that the process is generated using certain mechanism leveraged from these previous work [9, 10]. Future work will cover the detailed automatic process composition algorithm.

Step 2. Composition Agent Generation. Taking the function graph generated in step 1 as input, the super-peer is able to further generate the composition agent A, which contains the process instance, current status information, and some data structure encompassing the non-functional user requirements. Once the composition agent is generated, it migrates from one function to another along the function graph.

Step 3. Function Service Selection. The composition agent starts to seek the desired service providers from current Alliance-Peer's local matching table against the functionality specified by the function graph. Note that here the searching space has already been limited to service providers SPR = $\{SP_{r1}, SP_{r2} \ldots SP_{ri}. \ldots SP_{rn}\}$ that match current peer's request subscription R. This reduces the service selection time complexity and hence improves the composition performance. Here the matching table works as if a routing table of a peer in a normal peer-to-peer overlay. For each SPri., a matching algorithm is used to check if it is functionally matched with the current function node F on the function graph. Such algorithm can be leveraged from the one used by matching the service provider's publication and service consumer's request subscription mentioned in Section 3. This step will end up with reducing the SPR to a smaller set of functional-qualified service provider SPf = $\{SP_{f1}, SP_{f2} \ldots SP_{fi}. \ldots SP_{fn}\}$.

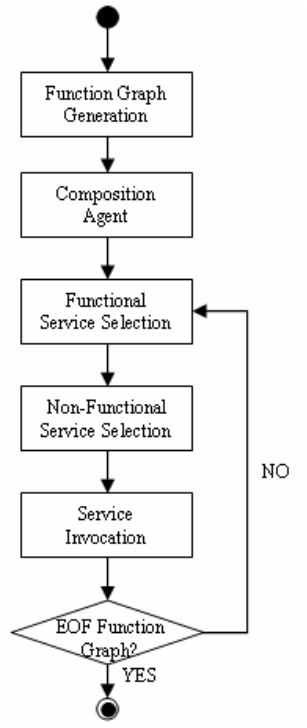

Figure 4. Alliance Composition Steps

Step 4. Non-Function Service Selection. In this step the composition agent A checks and compares each SPi in the set SPf and finally choose one SPf as SPresult that best fulfills the non-functional requirements (such as QoS, trustworthiness, etc) contained in the agent and specified by the end user. Relevant existing research on non-functional service selection can be leveraged to implement the algorithm in this step. This domain-peer might resort to Domain-Broker to retrieve those non-function data relating to current service provider candidate yet to be decided.

Step 5. Service Invocation. The composition agent A executes the process by migrating to the alliance-peer hosting the SPresult. Then the service provided by the SPresult will be invoked locally. If in some occasion the parallel execution of the process is desired, the process clones and each clone migrates to a dedicated alliance-peer. After all parallel service invocations have succeeded, the next function in the function graph, is to be executed. Again step 2 and step 3 will iteratively perform until all the functions finished.

5 CONCLUSION

In this paper, we propose an improved Internet-based distributed web services architecture (DWSASE) composed of a set of well-identified architectural styles. Detailed architectural components and connectors and motivations are described. The major contribution of this paper is that it provides a novel distributed web services architecture in the distributed Internet computing environment. The new architecture encompasses combinations of architectural styles, which can be used as guidance for designing and implementing the general web services application software systems. Currently, the implementation work is underway.

6 REFERENCES

[1] Perry, D. E. and Wolf, A. L., "Foundations for the Study of Software Architecture", ACM SIGSOFT Software Engineering Notes, 17(4), Oct. 1992, pp. 40–52.

[2] Fielding, R.T., "Architectural Styles and the Design of Network-based Software Architectures", PhD Dissertations, University of California, Irvine CA, USA

[3] Papazoglou, M. P., Kr¨amer, B. J., and Yang, J., "Leveraging Web-Services and Peer-to-Peer Networks", Springer-Verlag Berlin Heidelberg 2003.

[4] Paolucci, M., Soudry J., Srinivasan, N., and Semantic Sycara, K., "A Broker for OWL-S Web Services", Proceedings of First Internatioinal Web Services Symposium, 2004

[5] Emekci, F., Sahin, O., Agrawal, D., and Abbadi, A. 2004, "A Peer-to-Peer Framework for Web Service Discovery with Ranking", Proceedings of the IEEE ICWS'04, 2004

[6] Schmidt, C. and Parashar, M., "A Peer-to-Peer Approach to Web Service Discovery", World Wide Web, 7(2): 211-229, 2004

[7] Banaei-Kashani, F., Chen, C-C., Shahabi, C., 2004, "WSPDS: Web Services Peer-to-peer Discovery Service", Proceedings of International Symposium on Web Services and Applications (ISWS'04), USA, pp 733-743, June 2004

[8] Shaw, M., Garlan, D. Software Architecture – Perspectives on an Emerging Discipline, Prentice-Hall, Inc. 1996

[9] Yang, J., Papazoglou, M.P., "Service Component for Managing Service Composition Life-Cycle", Information Systems 29 97-125, Elsevier, 2004

[10] Liang, Q., Chakarapani, L.N., Su, S.Y.W., Chikkamagalur, R.N., Lam, H., "A Semi-Automatic Approach to Composite Web Services Discovery, Description and Invocation", International Journal of Web Services Research Vol.1, No. 4, 2004

[11] Wu, C. and Chang, E., "Comparison of web service architectures based on architecture quality properties", Proceedings of 3rd IEEE International Conference on Industrial Informatics, August 10-12, Perth, Australia.