République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Ibn Khaldoun -Tiaret-

Faculté des sciences et des sciences de l'ingénieur
Département d'Informatique

École Doctorale
Sciences et **T**echnologies de l'**I**nformation et de la **C**ommunication

Approche P2P pour une meilleure automatisation du processus de découverte distribuée des services web sémantiques

Mémoire

Pour l'obtention du diplôme de magistère
Option : Systèmes d'**I**nformation et de **C**onnaissances (SIC)

par

Hadj Madani MEGHAZI

Composition du jury

M. Amar BALLA	Maître de Conférences -A- à l'ESI -Alger-	Président
M. Djillali BAHLOUL	Maître de Conférences -A- à l'USTHB -Alger-	Examinateur
M. Samir KECHID	Maître de Conférences -A- à l'USTHB -Alger-	Examinateur
M. Youcef DAHMANI	Maître de Conférences -A- à Univ Ibn Khaldoun -Tiaret-	Examinateur
M. Khadhir BEKKI	Maître de Assistant -A- à Univ Ibn Khaldoun -Tiaret-	Examinateur
M. Youcef AKLOUF	Maître de Conférences -A- à l'USTHB -Alger-	Directeur de mémoire

ANNEE UNIVERSITAIRE 2009/2010

Merci mon Dieu de m'avoir donné
la force, la patience et la volonté d'arriver au terme de travail.

Je dédie ce modeste travail à mes parents…

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce Travail.

Je remercie Mr Youcef AKLOUF qui a accepté de diriger ce mémoire pour ses conseils et sa patience.

Je remercie aussi les membres du jury qui m'ont fait l'honneur de le constituer et surtout Mr BALLA qui a accepté le premier et du premier coup.

Du département Informatique de Tiaret, je remercie également Mr Maatoug et Mr Dahmani, sans oublier Mr Chadli pour leurs précieux conseils. Comme je remercie DJAMAL pour ses encouragements permanents.

Mes amitiés s'adressent à mes collègues de la promotion 2007-2008 de l'Ecole Doctorale, Hocine, Nacer, Kassa, Morad, Mohamed.KH, Adel, Ramzi et Djamel ; à El Hachemi, Rabeh et Mohamed.H de Bouraoui ; à Belkacem, Oussama, Lotfi, Noureddine et Hamouche du CDTA.

Résumé

Les services web sémantiques visent à automatiser l'utilisation des services web en ajoutant quelques annotations à ces services afin d'améliorer l'expression sémantique de leurs fonctionnalités. Avec l'évolution des services web au sein des organismes et de leurs utilisations à une grande envergure, la découverte et la localisation de tels services dans un contexte distribué est devenu un vrai défi. Un mécanisme de découverte des services web selon leurs fonctionnalités est alors nécessaire, puisque les fonctionnalités sont la plus importante chose que les partenaires recherchent. Dans ce mémoire, nous présentons un mécanisme de découverte automatique, scalable et distribuée pour l'environnement d'exécution des services web WSMX. Notre approche est basée sur les technologies P2P qui ont prouvé leur efficacité et leur robustesse en tant que systèmes distribués.

Mots clés: Services web sémantique, Découverte distribuée, WSMX, P2P, JXTA, JXTACast.

Abstract

Semantic web services aim to automate the use of web services by adding some annotations to these services in order to improve semantic expression of their functionalities. With the evolution of organization's web services and their widespread use, searching for them based on a distributed context has become a real challenge. A functionality-based web service discovery mechanism is then necessary since the functionalities are the most important thing partners look for. In this manuscript, we present an automatic, a scalable, and a distributed discovery mechanism for web service execution environments (WSMX). Our approach is based on P2P technologies that proved their efficiency and robustness as distributed systems.

Keywords: Semantic web services, Distributed discovery, WSMX, P2P, JXTA, JXTACast.

Sommaire

Liste des figures

Introduction générale

1. Contexte

Les services web sont devenus des composants technologiques importants dans le domaine d'intégration des applications, ce qui a rendu les organisations plus indépendantes et autonomes. Mais avec l'évolution du nombre des services disponibles sur le web et leurs utilisations, un nouveau problème a vu le jour, c'est le problème de la localisation et la découverte des services web, ce qui a devenu un domaine de recherche important.

Le processus de découverte classique est réalisé en utilisant des systèmes d'annuaires. Dans la majorité des cas ce sont des serveurs UDDI (*Universal Discovery, Discription and Integration*), où il est (le processus de découverte) basé essentiellement sur une recherche syntaxique des descriptions WSDL (*Web Service Discription Language*) des services web. Ces systèmes de découverte sont un petit peu limités dans la description des fonctionnalités des services web, comme ils sont réputés pour leur faible exactitude et leurs mauvaises performances, et même parfois pour leur faible disponibilité, ce qui est une conséquence directe de leur centralisation. En plus, après avoir effectué une recherche dans un UDDI, non pas tous les services énumérer dans le résultat de recherche sont accessible, puisqu'un serveur UDDI ne peut pas dépister le statut réel de chacun des services web (puisqu'il se peut que le fournisseur du service a arrêter de le fournir ou le fournisseur lui-même n'existe plus).

L'absence d'un mécanisme de découverte distribuée approprié constitue un problème réel qui peut limiter le grand potentiel des services web, qui requièrent une intervention humaine pour localiser les services désirés pour une requête donnée. Cette procédure limite considérablement la technologie des services web en termes d'efficacité et de scalabilité.

Les services web sémantique promettent de limiter l'intervention humaine tout au long de leur cycle de vie. Ceci en utilisant les standards du web sémantique qui rendent les services web auto-exploitable par les machines.

Une implémentation réussie du paradigme des services web sémantiques requiert un mécanisme de découverte automatique et scalable. En plus, la nature distribuée des services web sémantiques doit être prise en compte dans la conception d'un tel mécanisme.

Dans ce mémoire, nous allons proposer une approche pour remédier à certains aspects liés aux problèmes de coopération dans le processus de découverte distribuée des services web sémantiques dans un environnement d'exécution spécifique, qui se nomme WSMX (*Web Service Modeling eXecution environment*).

L'approche proposée fournie un mécanisme P2P de collaboration dans la découverte des descriptions qui sont enregistrées en utilisant la plateforme WSMX.

Couvrant tout le cycle de vie d'un service web, WSMX est notre banc d'essai.

2. Contribution

Plusieurs approches ont été proposées pour des mécanismes de découverte distribuée des services web sémantiques, mais dans toutes ces approches, pour coopérer, les nœuds participants (qui sont des fournisseurs des services, annuaires de ces services et même parfois des demandeurs de services), doivent tout d'abord localiser leurs semblables.

Pour ce faire, la quasi-totalité de ces approches utilisent des annuaires. Cette fois, ce ne sont pas des annuaires des services mais des annuaires des nœuds fournisseurs de services qui participent dans le processus de découverte.

Cependant, ces annuaires ont besoin de gestion (création, mises à jour et réplications) dans le cas des entrées ou des sorties des nœuds, comme ils ne reflètent pas le statut de ces nœuds en temps réel.

L'approche proposée permet de contourner l'utilisation d'un système d'annuaires pour les nœuds qui souhaite coopérer, en faisant recourt à une organisation en groupe et en procédant à une diffusion des demandes des services au sein de ce groupe. Cette approche offre une meilleure automatisation du processus de découverte distribuée des services et propose de l'alléger, comme elle offre une meilleure transparence concernant l'entrée et la sortie des nœuds.

3. Organisation du mémoire

Le présent rapport se subdivise en six chapitres :

***Le chapitre I* :** aborde les technologies du web sémantique et des services web avant de présenter le paradigme des services web sémantique, les langages des descriptions des services web sémantiques et quelques travaux dans le domaine.

Le chapitre II : donne une introduction des concepts de base de l'approche WSMO pour la description des services web sémantiques ainsi que sont langage de description WSML et la plateforme d'exécution WSMX.

Le chapitre III : donne un aperçu sur les systèmes P2P, les architectures existantes et les différente plateformes de développement d'applications P2P et plus précisément la plateforme JXTA.

Le chapitre IV : présente les approches qui existent pour une découverte distribuée des services web sémantiques et quelques travaux de recherche dans le domaine.

Le chapitre V : présente notre vision d'une approche de découverte distribuée des services web sémantiques issue des conclusions sur les avantages et les inconvénients des approches présentées dans le chapitre précédent.

Le chapitre VI : présente des détails techniques concernant l'implémentation de notre prototype « *WSMXDiscoCast »* basé sur l'approche proposée.

Chapitre I

Les Services Web sémantiques

1. Du web syntaxique au web sémantique

Le World Wide Web (WWW) a été développé en 1989 au laboratoire de physique européen (CERN[1]) à Genève, Suisse. C'était Tim Berners-Lee qui a développé le premier prototype du World Wide Web prévu pour servir de système d'information aux physiciens.

Le WWW original se constitué des documents (pages Web) et des liens entre ces documents. Les utilisateurs de navigateurs et de serveurs web se sont développés. Ils ont créé une infrastructure de partage d'informations attractive. Le Web est devenu bien plus intéressant au fur et à mesure que la quantité de l'information disponible augmentait. Dans cette infrastructure, une page Web peut être accédée par un URL (Uniform Resource Locator) via le protocole HTTP (HyperText Transfer Protocol), en utilisant un navigateur web (par exemple, Internet Explorer, Netscape, Mozilla, safari…etc.).

Actuellement, le World Wide Web se compose principalement de documents écrits en HTML (Hyper Text Markup Language), un langage qui est utile pour la présentation visuelle. La majeure partie d'information sur le Web est conçue seulement pour la consommation humaine. Un utilisateur humain peut lire des pages Web et les comprendre, mais leur signification n'est pas exprimée d'une manière à ce que des ordinateurs les interprètent.

L'information sur le Web peut être définie pour être utilisée par des ordinateurs non seulement pour l'affichage, mais également pour l'interopérabilité et l'intégration des applications et des systèmes. Une façon pour permettre l'échange et le traitement automatisé entre machines est de fournir les informations d'une manière à ce que les ordinateurs puissent les comprendre. Ceci est l'objectif du Web sémantique qui rend le traitement d'information sur le Web par des ordinateurs possible.

Tim Berners-Lee dans [1] affirme que : « le Web sémantique n'est pas un web à part mais juste une extension du web actuelle, dans lequel plus une information donnée a une signification bien définie, plus elle permet aux ordinateurs et aux gens de travailler en coopération ». Grau ajoute dans [2] que : « La prochaine génération du Web combinera les technologies existantes du Web avec des formalismes de représentation de connaissance ».

Le Web sémantique à été le résultat de plusieurs changements, en apportant des descriptions compréhensibles par machines aux données et aux documents qui existent déjà sur le Web. La figure 1.1 illustre les diverses technologies développées qui ont rendu possible le concept du Web sémantique.

[1] CERN : Désigne le Conseil Européen pour la Recherche Nucléaire.

	Statique	Dynamique	Syntaxe	Sémantique
codage	HTML	+RDBMS	+XML	+ RDF/OWL
Création	Manuel	Générer par des applications côté serveur	Générer par des applications basées sur des schémas	Générer par des applications basées sur des modèles
Utilisateurs	Humains	Humains	Humains et applications	Humains et applications
Paradigme	Recherche	Création/ consultation/ mise à jour	Intégration	Interopérabilité
applications	Navigateurs	Navigateurs	intégration de processus, EAI, BPMS, workflows	Agents intelligents, moteurs sémantiques

Fig. 1.1 L'évolution du Web [3].

Au début, on trouve le web statique qui est un ensemble de pages statiques liées ensemble, puis, passant par le web dynamique qui, au lieu d'un site web comportant une collection des pages manuellement construites avec HTML, définit des applications du côté serveur et des techniques d'accès aux bases de données employées pour dynamiser la création directe des pages web en réponse aux requêtes d'utilisateurs. Cette technique offre l'opportunité de livrer un contenu web adapté aux besoins des différents utilisateurs.

Néanmoins, les technologies disponibles pour créer dynamiquement des pages Web basées sur des informations des bases de données étaient insuffisantes pour les besoins des organisations recherchant des solutions d'intégration d'application. Extensible Markup Language (XML[2]) était l'une des solutions réussies développées pour produire l'intégration business-to-business (B2B). XML est devenu un moyen de transmission des données non structurées, semi-structurées, et même structurées entre les différents systèmes, augmentant ainsi l'intégration des applications et des entreprises.

Malheureusement, les solutions basées XML pour l'intégration des applications et des systèmes n'étaient pas suffisantes, puisque les données échangées ont manqué d'une description explicite de leurs signification. L'intégration des applications doit également inclure une intégration sémantique.

Vue l'étendu de l'intégration et de l'interopérabilité pour les processus d'intra et d'interentreprises, les chercheurs ont abordé ce problème et ont développé des normes sémantiques telles que RDF (Resource Description Framework) et OWL (Ontology Web Language). Les normes de RDF et de OWL permettent au web d'être une infrastructure globale pour partager les données et les documents, qui rendent la recherche et la réutilisation d'information fiable et plus facile, c'est l'idée du « web sémantique ».

[2] http://www.w3.org/TR/REC-xml/

Les recherches actuelles sur le Web Sémantique proposent de s'appuyer sur des techniques de représentation de connaissances (formalisme et raisonnement) pour munir l'information contenue dans les ressources web d'une sémantique.

Vers la fin des années 90 l'idée d'un web sémantique a amplifié l'intérêt pour le développement des structures connectives à base d'annotations, pour l'expression des liens qui existe entre les ressources, donnant naissance à ce que nous appelons maintenant des ***« ontologies »***.

2. Les ontologies

Le terme « ontologie » provient de la philosophie et a été adopté dans le domaine informatique avec une signification légèrement différente :

« *Une ontologie est une spécification explicite formelle d'une conceptualisation partagée* »

(Brost, 1997) [31].

« Une ontologie est une description formelle d'entités et leurs propriétés, relations, contraintes et comportement »

(Güringer et al, 1996)[32].

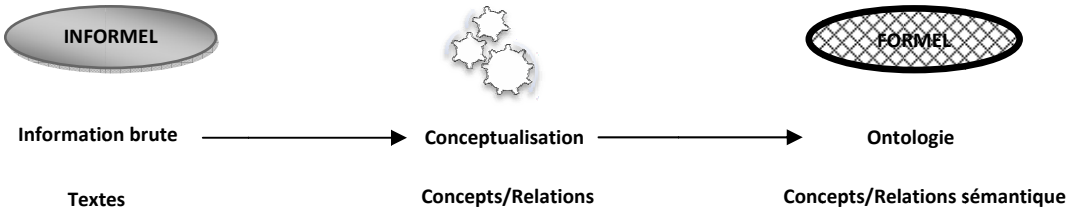


Fig. 1.2 L'ontologie comme une spécification formelle d'une conceptualisation.

Une ontologie définit les termes et les relations de base d'un vocabulaire dont une communauté est d'accord. Ce vocabulaire fournit des bases pour établir une connaissance de plus haut niveau pour spécifier la sémantique de la terminologie utilisée d'une façon bien définie et non ambiguë. Pour un domaine particulier, une ontologie représente un langage riche pour fournir des contraintes plus complexes sur les types des ressources et leurs propriétés.

Comparé à une taxonomie, les ontologies augmentent la sémantique des termes en fournissant des relations plus riches entre les termes d'un vocabulaire. Des ontologies peuvent être employées pour augmenter la communication entre hommes et ordinateurs.

Les trois utilisations principales des ontologies [4] sont :

- Assister la communication entre les utilisateurs humains.
- Réaliser l'interopérabilité et la communication entre les systèmes logiciels.
- Améliorer la conception et la qualité des systèmes logiciels.

Les Ontologies forment l'épine dorsale du web sémantique ; elles permettent aux machines la compréhension d'information à travers les liens entre les ressources

d'information et les termes dans les ontologies. En outre, les ontologies facilitent l'interopérabilité entre les ressources d'information à travers les liens au sein d'une même ontologie ou des liens entre des ontologies différentes.

Depuis que la recherche en matière d'ontologies a commencé dans le domaine informatique, les ontologies ont été considérées en tant que des moyens qui encouragent la réutilisation dans les technologies des systèmes à base de connaissances, et il s'est avéré que différents types d'ontologies montrent un potentiel différent pour la réutilisation [5].

Une catégorisation des ontologies peut être faite selon la façon utilisée pour leur conceptualisation. La figure 1.3 résume les types d'ontologies qui ont été publiés dans [6].

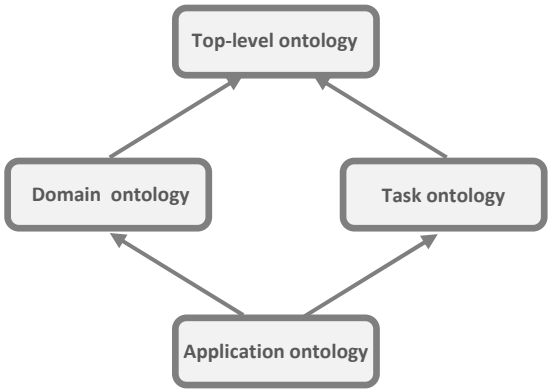


Fig. 1.3 Types d'ontologies.

- ***Top-level ontologies*** : ou bien les ontologies de haut niveau ou supérieures (Upper ontologies), essayent de décrire les notions très abstraites et générales qui peuvent être partagées dans beaucoup de domaines et d'applications.
- ***« Les ontologies de domaine » & « Les ontologies de tâches (générales)***» : Ces types d'ontologies capturent la connaissance dans un domaine spécifique, tel que la médecine, la géographie ...etc. ou la connaissance au sujet d'une tâche particulière, telle que le diagnostic ou la configuration.
- ***Application ontologies « d'application » :*** avec une portée restreinte, les ontologies d'applications fournissent le vocabulaire spécifique exigé pour décrire l'établissent d'une tâche particulière dans un contexte particulier d'application. Elles utilisent les ontologies de domaine et les ontologies de tâches, et décrivent, par exemple, le rôle joué par certaines entités de domaine dans une tâche spécifique.

En examinant la figure 1.3, on voit qu'elle représente une relation d'inclusion : les ontologies inférieures héritent et spécialisent des concepts et des relations, des ontologies supérieures. Les ontologies inférieurs sont plus spécifiques et ont ainsi un champ d'application plus étroit, tandis que les ontologies supérieures ont un potentiel plus large pour la réutilisation.

3. Les services web

Le web sémantique transforme le web actuel, le web syntaxique, en un web dont on se rend compte et on utilise la sémantique afin de surmonter le manque de compréhension et les limites dans l'intégration des structures de données et du vocabulaire.

Avec le web sémantique comme base fondamentale, il est possible que les utilisateurs humains trouvent des données pertinentes et peuvent interagir avec le web de manière sémantiquement définie et précise.

Mais en allant au delà des données et du traitement d'information, le web sémantique ne couvre pas l'introduction de l'aspect dynamique et distribué dans le web courant ; les services web est la solution qui promet de couvrir cet aspect dynamique et distribué en rendant l'infrastructure du web, un moyen pour les traitements distribués à une grande échelle.

Les services web sont basés sur le principe d'architecture orientée « service » SOA (Service-Oriented Architecture). SOA est une des dernières évolutions de l'informatique répartie, qui permet aux composants logiciels, y compris des fonctions d'applications, objets et processus de différents systèmes, d'être exposé comme services.

L'évolution a commencé quand les organisations ont voulu faire implémenter des solutions telles que le Business-to-Consumer et le Business-to-Business, ils ont réalisé que les technologies du web initiales associées au WWW ne sont pas suffisantes pour vendre des produits sur internet. En utilisant SSL, les organisations ont été en mesure d'implémenter des solutions pour avoir la confiance de leurs clients dans l'obtention des informations confidentielles (par exemple : le numéro d'une carte de crédit).

Afin de répondre aux espérances des clients et des partenaires, les organisations devraient lier leurs systèmes autonomes, hétérogènes et distribués, pour améliorer l'efficacité et la productivité, ce qui a mené au développement et au déploiement des solutions EAI (Entreprise Application Intégration). Les plateformes EAI ont été utilisées pour intégrer les systèmes incompatibles tels que : les ERP (Entreprise Ressource Planning), les CRM (Consumer Relationship management), les bases de données, les entrepôts de données et autres systèmes interne à l'organisation.

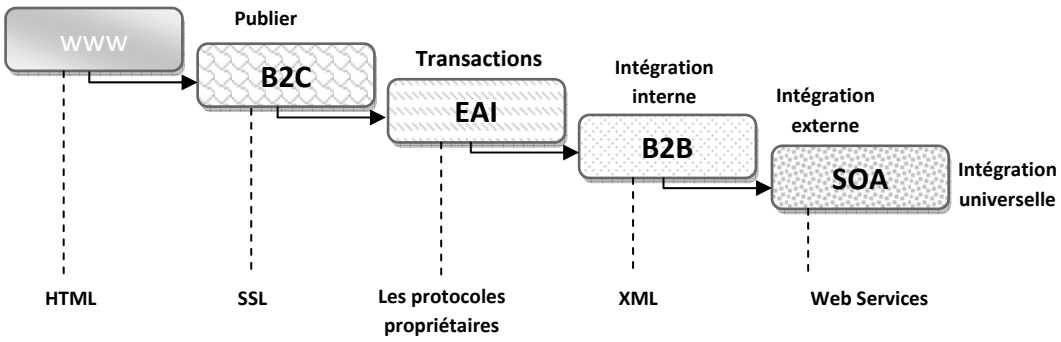


Fig. 1.5 Evolution de l'usage du web [3].

Cependant ces plateformes (EAI) utilisaient des protocoles propriétaires qui posent des problèmes quand il s'agissait de faire intégrer des systèmes internes à l'organisation avec d'autres qui ne le sont pas.

Les systèmes internes et externes devraient communiquer à travers des réseaux permettant ainsi aux entreprises d'achever leurs transactions. Pour atteindre un tel niveau d'intégration, les solutions Business-to-Business (B2B) se sont développées au tour de XML comme langage de représentation de données.

Après avoir vécu et expérimenté l'utilisation de XML, les organisations ont réalisé que leurs solutions architecturales montraient un couplage fort dans l'interaction des applications, ce qui limite la flexibilité et l'adaptation des systèmes. Le concept de l'architecture orientée services (SOA) a été présenté et a défini une méthode de concevoir, développer, déployer et gérer des portions logiques des programmes. Le but de SOA est de faiblir le couplage, structurer et standardiser les fonctionnalités d'entreprises dans les applications logiciels qui interagissent.

3.1. Qu'est-ce qu'un service web ?

Un service web est un service qui est disponible sur l'internet, utilise un système de communication XML standardisé et n'est pas spécifique à un système d'exploitation ou un langage de programmation [7].

Selon Gartner Research, « Les services web sont des composants logiciel faiblement couplées fournis au-dessus des technologies standardisées d'internet » [8].

Il y a différentes moyens pour effectuer une communication XML. Par exemple, on peut utiliser XML Remote Procedure Calls (XML-RPC) ou bien SOAP. Une autre alternative, est d'utiliser les méthodes GET/POST de HTTP pour faire circulé des documents XML.

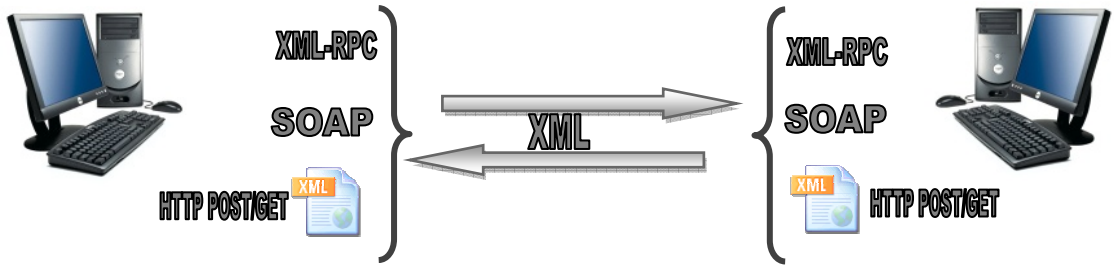


Fig. 1.6 Communication XML pour les services web

Une des caractéristiques des services web est qu'ils sont « ***auto descriptifs*** », ca veut dire, que si un nouveau service est publié, il doit également avoir à publier une interface publique du service. Au minimum, le service devrait inclure une documentation lisible afin que d'autres développeurs puissent plus facilement intégrer celui-là. S'il s'agit de créé un service SOAP, il est recommander d'inclure une interface publique écrite dans une

grammaire XML commune. Cette grammaire est utilisée pour identifier toutes les méthodes publiques, les arguments d'une méthode et les valeurs retournées.

Un service web complet est, donc, n'importe quel service qui :

- Est disponible à travers internet ou des réseaux privés (intranet).
- Utilise un système de communication XML standardisé.
- N'est pas lié à un système d'exploitation ou un langage de programmation.
- Est auto-descriptif via une grammaire XML commune.

3.2. Architecture des services web

Il y a deux manières de voir l'architecture d'un service web. La première est d'examiner les différents rôles de chaque acteur des services web ; la seconde est d'examiner la pile de protocole émergente des services web.

3.2.1. Le rôle d'acteurs des services web

Il y a trois rôles d'acteurs importants pour les services web :

Fournisseur de services : C'est le fournisseur du service web. Le fournisseur de service implémente le service et le rend disponible sur internet.

Demandeur de services : C'est n'importe quel consommateur du service web. Le demandeur utilise un service web existant en initiant une connexion réseau et en envoyant une requête XML.

Annuaire de services : C'est un annuaire logiquement centralisé des services. L'annuaire fournit un endroit central où les développeurs peuvent publier de nouveaux services ou trouver ceux qui existent. Il sert donc d'un point central aux compagnies et à leurs services.

Dans un scénario de fonctionnement normal, un fournisseur de services héberge un module implémentant un ou plusieurs services web accessible via le réseau. Il définit une description du service et la publie en la faisant enregistrée dans un annuaire des services. Le demandeur de service amorce une opération de recherche pour trouver la description du service. Afin d'invoquer le service et interagir avec l'implémentation du service web, le demandeur de service établit une connexion avec le fournisseur de service.

On peut distinguer trois opérations :

- La publication des descriptions des services (*publish*).
- La recherche et la découverte de la bonne description du service (*find*).
- L'association ou l'invocation des services basés sur la description (*bind*).

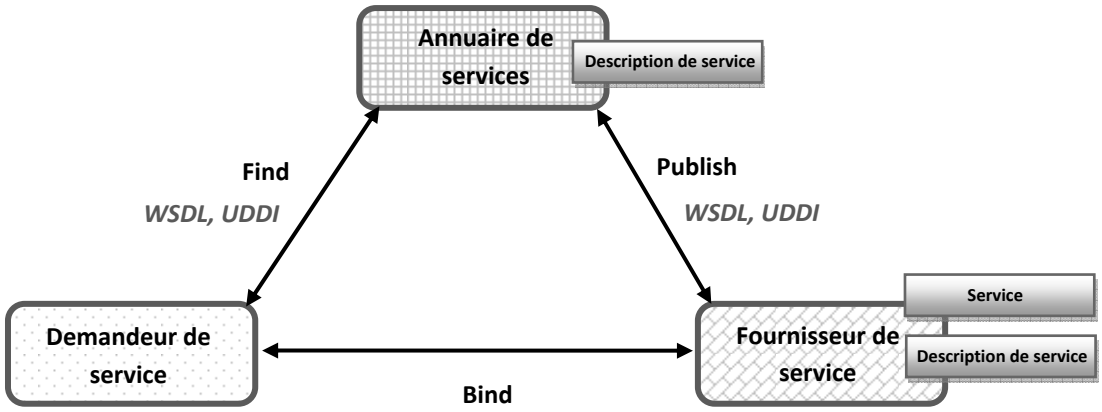


Fig. 1.7 Modèle d'interactions des services web

3.2.2. La pile des protocoles des services web

Une autre manière de voir l'architecture des services web est d'examiner leur pile de protocoles. La pile évolue toujours, mais a actuellement quatre couches principales.

Voilà une courte description de chaque couche :

- ***Transport***

Cette couche est responsable du transport des messages entres applications. Actuellement, cette couche inclut HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), et des nouveaux protocoles, comme Blocks Extensible Exchange Protocol (BEEP).

- ***Communication XML***

Cette couche est responsable de l'encodage des messages dans un format XML commun de telle sorte que les messages puissent être compris pour chacune des extrémités. Cette couche inclut XML-RPC et SOAP.

- ***Description du service***

Cette couche est responsable de la description de l'interface publique d'un service web donné. Actuellement, la description d'un service est manipulée par l'intermédiaire du langage WSDL (***W**eb **S**ervice **D**escription **L**anguage*).

- ***Découverte des services***

Cette couche est responsable de la centralisation des services dans un annuaire commun, et de fournir facilement des fonctionnalités de type Publish/Find (Publier/trouver). Actuellement, la découverte de services est manipulée par l'intermédiaire d'UDDI (***U**niversal **D**escription, **D**iscovery and **I**ntegration*).

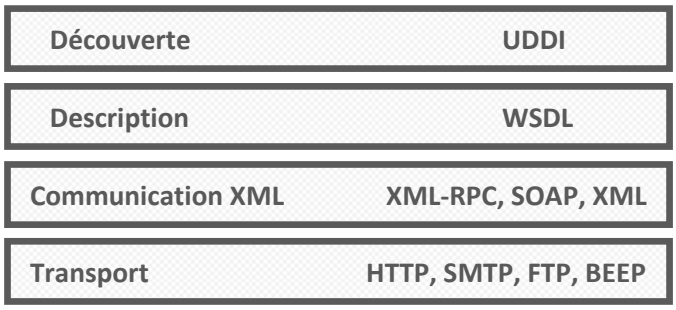


Fig. 1.8 La pile des protocoles pour un service web [7].

4. Les services web sémantiques

Le terme « services web sémantiques » est réservé pour l'automatisation des tâches d'utilisation des services, tels que : la découverte, la sélection, la composition et l'établissement des services appropriés. Les services web sémantiques se trouvent à la convergence de deux domaines de recherche importants concernant les technologies de l'internet : le web sémantique et les services web. Cette tâche est accomplie en rendant les services web auto-exploitables par machines. De la même façon que le web sémantique a promit de permettre aux machines de tirer parti du contenu statique du web en utilisant les annotations. L'idée d'appliquer des techniques semblables aux services web est très intéressante.

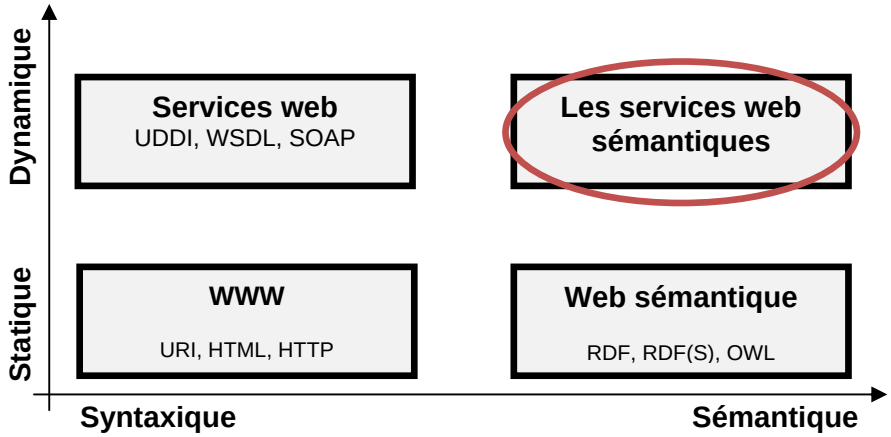


Fig. 1.9 L'évolution du web [10].

Les avantages potentiels des services web sémantiques ont mené à l'établissement d'un domaine de recherche important, dans le milieu industriel et académique. Plusieurs initiatives sont apparues pour faire ce qu'on appelle « l'annotation sémantique des services web », ce qui a produit une variété de descriptions des services web et leurs aspects relatifs, ce qui en retour a abouti à de divers genres de supports pour la découverte et la composition.

Le besoin d'automatisation du processus de conception, de mise en œuvre et de la découverte des services web est le même que celui du web sémantique, à savoir comment décrire formellement les connaissances de manière à les rendre exploitables par des

machines. En conséquence, les technologies et les outils développés dans le contexte du web sémantique peuvent compléter la technologie des services web.

Le concept fondamental du web sémantique et des SWS[3] (Services Web Sémantique) est l'ontologie, qui produit « une signification bien définie » des informations contenu dans le web. Une ontologie est un ensemble de termes de la connaissance, qui inclut un vocabulaire, des liens sémantiques et quelques règles simples pour un domaine donné. L'avantage principal des ontologies est que plusieurs communautés sur internet ont maintenant des définitions partagées sur un bon nombre de concepts clés. Par exemple, la notion d'ontologie peut jouer un rôle important pour permettre d'expliciter la sémantique des services facilitant ainsi les communications hommes-machines, d'une part, et d'autre part les communications machines-machines.

L'objectif des services web sémantique est de créer un web sémantique des services web dont les propriétés, les capacités, les interfaces et les effets sont décrits de manière non ambiguë et exploitable par des machines et ce en utilisant les couches techniques sans pour autant en être conceptuellement dépendants [9]. La sémantique exprimé permet d'automatisé les fonctionnalités suivantes :

- Processus de description et de publication des services.
- Découverte des services.
- Sélection des services.
- Composition des services.
- Fourniture et administration des services (Négociation des contrats etc.).

Donc les services web ont besoin des langages et d'APIs pour ajouter cette dimension sémantique qui va leurs permettre d'avoir les fonctionnalités suivantes :

- ***Une découverte automatique des services web***

  La découverte automatique d'un service web implique la localisation automatique des services web qui produisent une certaine fonctionnalité et qui répondent aux contraintes demandées. Par exemple, un utilisateur qui veut trouver un service qui vent des billets d'avion entre deux villes données et accepte une carte de crédit. Actuellement, cette tâche doit être fait par un humain qui peut utiliser un moteur de recherche pour trouver le service approprié, lire une page web et exécuter le service manuellement, pour déterminer si les contraintes sont satisfaites. Avec des langages appropriés[4], les informations nécessaires pour la découverte des services web doivent être spécifiées sous forme de balises sémantiques (par exemple une utilisation d'ontologie sous forme « OWL » ou « DAML ») qui peut être interprétées par des machines au niveau du site qui héberge le service web et l'annuaire de services ou au niveau d'un moteur de recherche avancé à base d'ontologies. Ou bien, un service peut pro-activement

[3] **SWS :** Semantic web services.

[4] Ce type de langage correspond à DAML-S et OWL-S qui vont être détaillés plus loin dans ce chapitre.

s'annoncer auprès d'un annuaire, pour que les demandeurs peuvent le trouvé en adressant leurs requêtes à l'annuaire.

Ces langages doivent produire des annonces déclaratives des propriétés et des fonctionnalités du service qui peuvent être utilisées pour la découverte automatique du service web.

- ***L'invocation automatique des services web***

L'invocation automatique d'un service web implique l'exécution d'un service web identifié par programme informatique ou un agent. Par exemple l'utilisateur peut demander d'acheter un billet d'avion d'un site donné et d'un vol donné. Actuellement, l'utilisateur doit consulter le site web qui offre le service, remplir un formulaire et cliquer sur un bouton pour exécuter le service. Une alternative est que l'utilisateur envoi directement une requête HTTP au service avec les paramètres appropriés en HTML. Dans les deux cas, un humain dans la boucle est nécessaire. L'exécution d'un service web peut être considérée comme une suite d'appels de fonctions. En utilisant des langages qui fournissent des APIs déclaratives interprétable par machines, on peut exécuter ces appels de fonctions. Un agent logiciel doit être capable d'interpréter les balises du langage utilisé pour qu'il soit capable de comprendre les entrées nécessaires pour l'invocation du service, quelles sont les informations qui vont être retournées et comment le service peut être exécuté automatiquement. Ainsi le langage utilisé doit fournir des APIs déclaratives pour les services web qui vont être nécessaire pour l'automatisation d'exécution des services web.

- ***Une composition et une interopérabilité automatique des services web***

Cette tâche implique une sélection, une composition et une interopérabilité automatique des services web pour l'exécution de certaines tâches, étant donnée une description de haut niveau d'un objectif. Par exemple, un utilisateur peut vouloir avoir prendre tous les arrangements de voyages pour un voyage à une conférence. Actuellement, l'utilisateur doit sélectionner les services web, spécifier la composition manuellement et de faire en sorte que chaque logiciel requis pour l'interopération est créé. Donc il faut des langages qui vont coder les informations nécessaires pour la sélection et la composition des services web sur leurs sites. Un logiciel peu être développé pour manipuler ces représentations avec des spécifications des objectifs de la tâche. Le langage utilisé doit fournir des spécifications déclaratives des pré-requis et des conséquences de l'utilisation individuelle d'un service qui sont nécessaires pour l'automatisation de la composition des services et leurs interopérabilités.

- ***Une surveillance d'exécution automatique des services web***

  Les services individuels et même la composition des services exigera souvent du temps pour s'exécuter complètement. Un utilisateur peu se demander pendant cette période sur l'état de sa requête, où les plans ont pu avoir changé, ce qui requière une altération dans les actions prises de l'agent logiciel. Par exemple, un utilisateur veut s'assurer qu'une réservation d'une chambre d'hôtel est déjà faite. Ça serait préférable d'avoir un moyen de découvrir l'état de la requête, et si des problèmes sont apparus dans le processus. Ainsi un langage choisi doit fournir des descriptions pour l'exécution des services. Ce point à été le sujet de plusieurs travaux de recherches, mais aucune approche n'a pu produire un langage qui fait une description satisfaisante de l'exécution des services web [3].

4.1. L'architecture des services web sémantiques

Le groupe architecture du W3C travail activement à l'élaboration d'une architecture étendue standard. Cette architecture est basée sur celle présenté dans la figure (Fig. 1.8) des services web, la pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment.

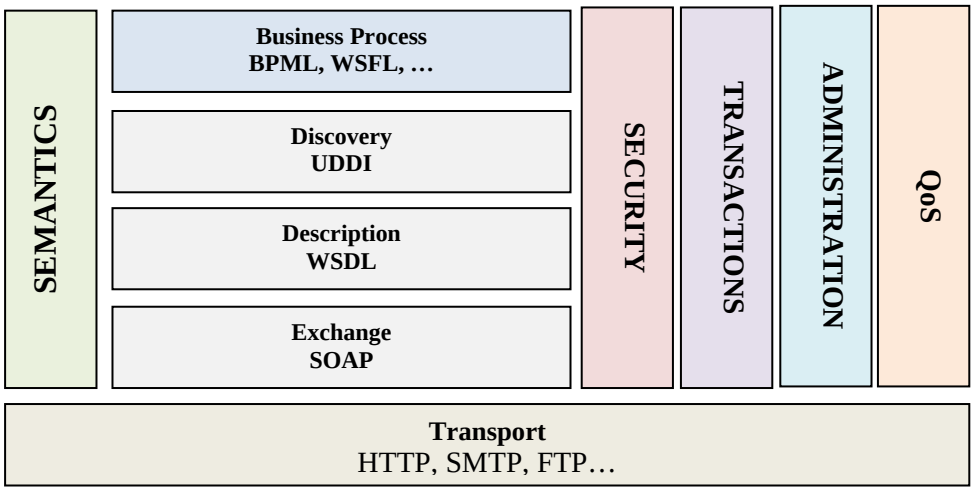


Fig. 1.10 Pile des services web sémantiques [9].

Ces couches s'appuient sur les standards émergeants SOAP, WSDL et UDDI. Une couche de « business process » est ajoutée, permettant de rendre les processus métiers accessibles à l'intérieur d'une entreprise et au-delà même de ces frontières. Les couches transversales : sécurité, transactions, administration et QoS, permettent de compléter la couche « *business process* ».

La couche « *SEMANTICS* » vient s'intégrer à chacune des quatre couches supérieures et permettant ainsi d'automatiser ces processus.

4.2. Descriptions sémantiques des services web

Actuellement plusieurs descriptions sémantique des services web existent, cependant cette section va donner une brève présentation des descriptions les plus importante dont on site : DAML-S, OWL-S et WSDL-S. Par contre, tout un chapitre est dédié à l'approche WSMO où elle sera détaillée.

4.2.1. DAML-S

DAML-S est un langage de description des services basé sur XML utilisant le modèle des logiques de descriptions (et plus précisément DAML+OIL). Son intérêt est qu'il est un langage de haut niveau pour la description et l'invocation des services web dans lequel la sémantique est incluse (contrairement à UDDI).

DAML-S est composé de trois parties principales : Le ***« Service profile »*** qui permet la description, la promotion et la découverte des services, en décrivant non seulement les services fournis, mais également des pré-conditions à la fourniture de ce service. Le ***« Service model »*** qui présente le fonctionnement du service en décrivant dans le détail et d'une manière relativement abstraite les opérations à effectuer pour y accéder. Certains éléments du *Service Model* peuvent être utilisés à la manière du *Service Profile* afin de fournir des informations supplémentaires à un utilisateur dans le cas où les opérations à effectuer seraient également un critère de choix. C'est le *Service Model* qui va permettre une composition des services si nécessaire. Il permet également d'effectuer un contrôle du déroulement du service. Le ***« Service Grounding »*** va présenter clairement et dans le détail la manière d'accéder à un service. C'est dans cette partie que les protocoles et les formats des messages échangés, sont spécifiés.

4.2.2. OWL-S

OWL-S est une ontologie OWL des services, pour décrire les différents aspects des services web. L'initiative d'OWL-S est issue de l'ontologie DAML pour les services (DAML-S) dont sa première version en mai 2001, c'était le premier effort progressif vers l'annotation sémantique des services web. OWL-S dans sa version 1.1 a été soumis au W3C en novembre 2004 par Nokia, l'Université de Maryland, le National Institute of Standards and Technology (NIST), Network Inference, SRI International, France Telecom, Stanford University, Toshiba, et l'Université de Southampton.

En passant de DAML+OIL à OWL, OWL-S a essayé d'adopté des recommandations existantes du web sémantique les couplant ainsi au monde des services web, en liant les descriptions d'OWL-S avec celles existantes de WSDL.

L'ontologie OWL-S comprend des concepts de haut niveau, elle définie le concept de haut niveau « Service » et trois sous ontologies OWL-S qui sont : « *Service Profile* », « *Service Model* » et « *Service Grounding* », qui sont illustrés dans la figure 1.11.

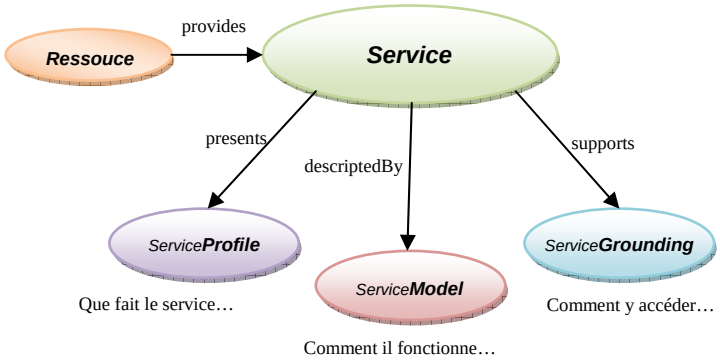


Fig. 1.11 Le model conceptuel d'OWL-S [10].

4.2.3. WSDL-S

En utilisant XML-Schema et ayant le soutient des industriels, WSDL-S constitue une approche incrémentale pour l'ajout d'annotations sémantiques aux documents WSDL. Dans WSDL-S les utilisateurs peuvent ajouter des annotations sémantiques aux documents WSDL en utilisant l'extension définie dans la spécification WSDL. Les annotations sémantiques peuvent être des références à des concepts définis dans des ontologies externes.

WSDL-S en tant que tel ne prescrit aucun langage particulier pour les ontologies et il n'est pas défini pour être lié aux langages des représentations sémantiques. Les utilisateurs peuvent utiliser OWL, WSMO (voir le deuxième chapitre) ou n'importe quel langage de modélisation. WSDL-S est issu du projet METEOR-S[5] de l'université de Georgia qui a été significativement modifié par IBM et l'équipe METEOR-S. la figure 1.12 illustre le principe de WSDL-S et met le point sur comment le model de domaine est maintenu externe au modèle des services web et comment les associations entre les concepts de WSDL et leurs annotations sémantiques correspondantes sont maintenues en utilisant des références (URI).

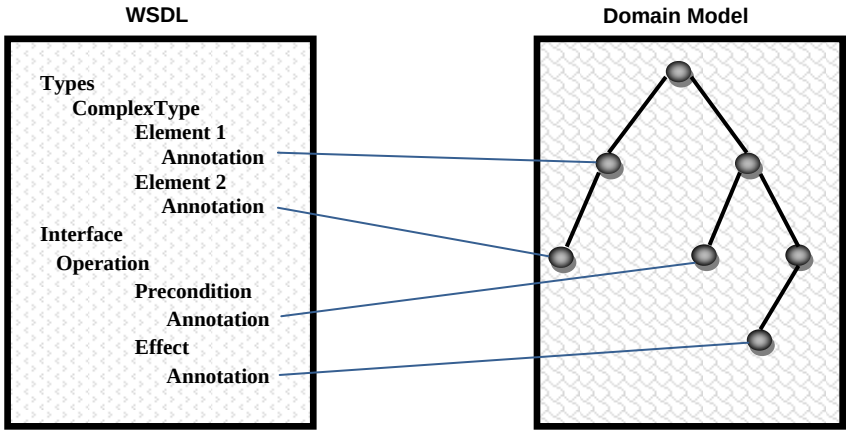


Fig. 1.12 L'externalisation de la représentation et de l'association de la sémantique aux éléments de WSDL[3].

[5] **METEOR-S :** http://lsdis.cs.uga.edu/projects/meteor-s/.

5. Synthèse

Les services web deviennent des composants technologiques importants dans le domaine d'intégration d'applications. Aujourd'hui, les services web sémantiques constituent une voie nouvelle permettant de mieux exploiter les services en automatisant, autant que possible, les différentes tâches liées à un service.

Les services web sémantiques se retrouvent à l'intersection de deux domaines de recherche actifs qui sont le web sémantique et les services web. Le web sémantique ayant pour objectif d'enrichir le web en lui ajoutant une couche de métadonnées compréhensibles aux machines pour permettre aux programmes informatique de faire un traitement automatique de l'information. Et les services web qui sont des composants logiciels faiblement couplés fournis au dessus des technologies standardisées, offrant ainsi un vrai support pour l'intégration des applications et des services.

Entre dynamique et sémantique, les services web sémantiques est l'évidente solution qui promet l'automatisation des tâches de : description, découverte, sélection, composition et fourniture des services afin de créer ce qu'on appelle un web sémantique des services web.

De multiples travaux de recherche ont été menés pour la description des SWS, cependant l'approche WSMO (qui est l'objet du prochain chapitre), selon [11], est favorite. Ce qui est justifiable puisque aucune des autres approches n'a attaqué, de manière unifiée, tous les aspects d'une plateforme pour la définition et l'exécution des SWS.

Le chapitre suivant a pour objectif de détailler l'approche « WSMO » pour une description sémantique des SWS, ces fondements théoriques, son langage de description « WSML » et la plateforme d'exécution des services web sémantiques « WSMX ».

Chapitre II

L' Approche WSMO

1. Introduction

Le WSMO[6] (Web Service Modeling Ontology) est un projet de l'union européenne qui constitue un cadre compréhensible pour SESA (Semantically Enabled Service-Oriented Architectures) et défini un model conceptuel avec un langage de spécification, comme il fourni une implémentation avec plusieurs outils. Le langage de spécification WSML[7] fourni un ensemble de raisonneurs, et couvre tous les types de langages qui sont considérés comme appropriés pour le web sémantique. L'implémentation WSMX[8] fournie un environnement de développement et d'exécution pour SESA à base de WSMO.

Le concept SESA a pour objectif d'offrir un support pour tout le cycle de vie et de production dans les systèmes à architecture orientée service (SOA). C'est pourquoi, les approches des SWS se sont étendues avec des descriptions sémantiques : les buts comme des descriptions des objectifs de l'utilisateur (demandeur des services), et les médiateurs pour la prise en charge des problèmes d'hétérogénéités [12].

L'approche WSMO définie les ontologies, les services web, les buts et les médiateurs comme ses éléments de haut niveau avec un model conceptuel qui prend en charge ces derniers. Ce model conceptuel a pour but la structuration des annotations sémantique des services.

2. Concepts de base de WSMO

Afin d'automatiser totalement ou partiellement le processus d'intégration de nombreux services dans une architecture orientée service, il est nécessaire de décrire sémantiquement tous les aspects liés aux services qui sont disponibles via une interface d'un service web.

L'approche WSMO fournit un modèle conceptuel pour la création de telles descriptions des services. WSMO se base principalement sur WSMF (Web Service Modeling Framework) [13], qui fournit une spécification ontologique des éléments de base des services web sémantique en étendant et en raffinant ces derniers.

Les principes de base de l'approche WSMO sont :

- ***Services vs services web:*** Les services sont vus comme des instances concrètes des services web où toutes les entrées sont spécifiées. Ils sont les entités réelles prévues pour l'invocation. Un service web, en revanche, est une entité abstraite, une classe d'un ensemble des services concrets qui sont en mesure de fournir des fonctionnalités à l'utilisateur. WSMO est conçu comme un moyen de décrire les services web et non pas de les remplacer.

[6] www.wsmo.org
[7] www.wsmo.org/wsml/
[8] www.wsmx.org

- ***Description vs mise en œuvre:*** WSMO fait une nette distinction entre la description des éléments des services web sémantique et les technologies qui permettent leur exécution. WSMO vise à fournir un modèle de description ontologique des éléments des SWS et d'être conforme avec leurs technologies d'exécution.
- ***Basée sur des ontologies :*** Les ontologies sont à la base même du méta model de WSMO, permettant un traitement d'information avancé et une résolution des problèmes d'hétérogénéité des ressources. Les ontologies sont le model de données de WSMO, ainsi que, toutes les descriptions des ressources et des données échangées sont ontologiquement décrites.
- ***Une séparation ontologique des rôles :*** Le contexte dans lequel les demandeurs et les fournisseurs d'une fonctionnalité d'un service donné se présentent peut être très différent les uns des autres. C'est pour cela que les auteurs de l'approche WSMO ont vu qu'une séparation entre les exigences qu'un demandeur donné et la fonctionnalité que les fournisseurs des services sont disposés à offrir est nécessaire.
- ***découplage Strict:*** Le web est un espace ouvert et distribué, où les ressources sont indépendamment développées. WSMO soutient ce fait et précise que les ressources devraient être strictement découplées. Cela signifie que des descriptions sémantiques sont développés de manière isolée les uns des autres sans tenir compte de leurs utilisations possibles ou leurs interactions avec d'autres ressources.
- ***La médiation:*** Les ressources sont strictement découplées les uns des autres, il doit exister un mécanisme pour résoudre les problèmes d'hétérogénéité entre les ressources. La médiation joue ce rôle et permet de résoudre les problèmes de correspondances entre les ressources qui peuvent se produire au niveau des données, des protocoles, ou des processus. La médiation est un concept de base du métamodèle WSMO et un élément de haut niveau du métamodèle WSMO.
- ***Le web Compilance :*** WSMO hérite du concept de *Universal Resource Identifier* (URI) à partir du web, en tant que mécanisme pour l'identification unique des ressources. En fournissant ce principe au sein de WSMO, signifie que les langages qui formalisent WSMO devrait utiliser des URI pour l'identification des ressources afin d'être conforme aux technologies du web (pour être se qu'on appel « Web-Compilant »).
- ***Les sémantiques d'exécution :*** Des implémentations de référence de WSMO, tels que « *Web Service eXecution Environment* » (WSMX) fournissent un mécanisme permettant de vérifier la spécification WSMO d'une manière formelle.

A base de ces principes de conception du métamodèle WSMO, les auteurs de l'approche définissent quatre éléments de haut niveau, à savoir, les ontologies, les services web, les buts, et les médiateurs. Pour bien décrire les services web sémantique selon cette approche, une compréhension de chacun de ces quatre éléments est nécessaire. La figure 2.1 montre ces quatre éléments.

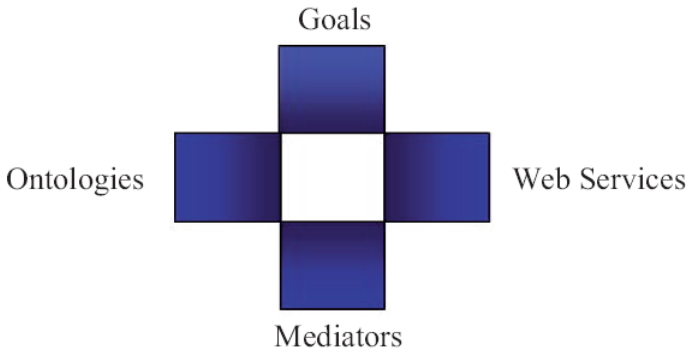


Fig. 2.1 Les éléments de haut niveau de WSMO

2.1. Ontologies WSMO

Les ontologies fournissent la terminologie à employer pour les autres éléments de WSMO. Elles fournissent aussi le premier et le plus important moyen pour réaliser l'interopérabilité entres les buts (Goals) et les services web aussi bien qu'entres les différents services web. En utilisant des technologies standardisées, différent éléments peuvent être liées directement ou indirectement en utilisant des alignements prédéfinis.

La définition d'une ontologie a la forme suivante :

```
Class ontology
    hasNonFunctionalProperty type nonFunctionalProperty
    importsOntology type ontology
    usesMediator type ooMediator
    hasConcept type concept
    hasRelation type relation
    hasFunction type function
    hasInstance type instance
    hasRelationInstance type relationInstance
    hasAxiom type axiom
```

- ***Les propriétés non-fonctionnelles :*** Il est possible d'ajouter des propriétés non fonctionnelles à tous les éléments de WSMO. Ces propriétés non fonctionnelles sont principalement utilisées pour décrire des aspects non fonctionnels, tels que le nom du créateur, la date de création, des descriptions en langage naturel, etc.
- ***Les ontologies importées :*** WSMO est conçu pour réutiliser des ontologies existantes afin d'en créer des autres. Ceci est utile sachant que le processus de création d'une ontologie pour décrire un domaine donné peut être complexe et coûteux. Tous les éléments de haut niveau de WSMO utilisent l'expression

« *importsOntology* » pour importer les ontologies qui contiennent les concepts qui sont jugé nécessaires pour concevoir une description.

- ***Les médiateurs pour les ontologies importées :*** En important une ontologie on n'est pas sûr qu'elle soit directement utilisée, puisque des problèmes de correspondance entre des expressions utilisées dans cette ontologie avec d'autres sur une autre ontologie peuvent surgir. Dans ce cas un médiateur est requis, ce dernier a pour rôle d'aligner, migrer ou transformer l'ontologie importée pour résoudre les problèmes d'hétérogénéité qui peuvent apparaitre. Comme l'expression « *importsOntology* », l'expression « *usesMediators* » peut être utilisée sur tous les éléments de haut niveau de WSMO.
- ***Les concepts :*** comme des éléments de base du domaine du problème.
- ***Les relations :*** comme un model d'interdépendance entre les concepts.
- ***Les fonctions :*** pour permettre la transformation des données.
- ***Les instances :*** doivent être spécifiées (peuvent pointer sur des objets externes).
- ***Les axiomes :*** sont des expressions logiques.

2.2. Services web WSMO

Un service web WSMO est une description formelle de la fonctionnalité d'un service web, en termes de ce qui est capable de faire « *Capability* », et en termes d'interfaces (les méthodes utilisées pour interagir). Comme tout élément de WSMO, un service peut importer des ontologies dans sa description. Cette description peut utiliser deux types de médiateurs : les « *ooMediator* » et les « *wwMediator* ».

La définition d'un service web a la forme suivante :

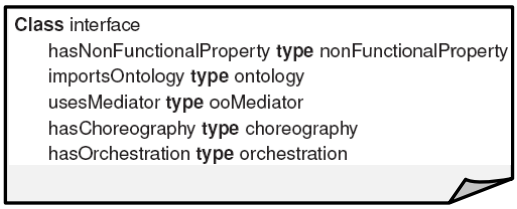
```
Class service
    hasNonFunctionalProperty type nonFunctionalProperty
    importsOntology type ontology
    usesMediator type {ooMediator, wwMediator}
    hasCapability type capability  multiplicity = single-valued
    hasInterface type interface
```

- ***Capabilities:*** Décrivent la fonctionnalité fournie d'un service. Ecrite avec un ensemble d'axiomes qui décrive l'état de l'environnement d'exécution avant et après que l'exécution du service est faite, en utilisant les expressions « *hasPrecondition* », « *hasPostcondition* », « *hasAssumption* » et « *hasEffect* ».

```
Class capability
    hasNonFunctionalProperty type nonFunctionalProperty
    importsOntology type ontology
    usesMediator type ooMediator
    hasPrecondition type axiom
    hasAssumption type axiom
    hasPostcondition type axiom
    hasEffect type axiom
```

- ***Les interfaces:*** Spécifient comment le service se comporte pour achever ses fonctionnalités. Une interface d'un service se compose d'une *Chorégraphie* qui décrit l'interface pour l'interaction Client-Service requise pour la consommation d'un service, et une *Orchestration* qui décrit comment les fonctionnalités d'un service web sont achevées par agrégation avec d'autres services.

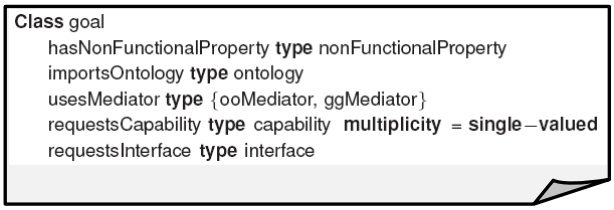
Une interface est définie comme suit :



2.3. Butes WSMO

Représentent les désirs de l'utilisateur et fournissent les moyens qui caractérisent ces demandes en terme de besoins fonctionnels et non-fonctionnels. Un but WSMO peut être vu comme une description des services qui peuvent satisfaire les désires d'un demandeur de service. Un but peut importer des ontologies, et peut utiliser deux types de médiateurs : les « *ooMediator* » (entre les ontologies) et les « *ggMediator* » (entre les buts).

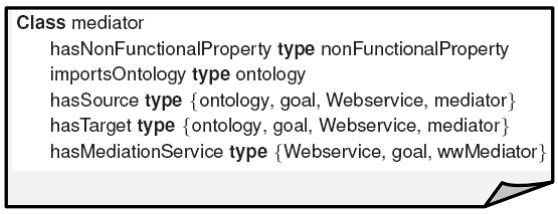
La définition d'un but a la forme suivante :



2.4. Médiateurs WSMO

Visent à traiter automatiquement des problèmes d'interopérabilité entre les différents éléments de WSMO et fournissent des éléments procédurales supplémentaires pour spécifier d'autres *mappings* non capturés par l'utilisation des ontologies. Un médiateur WSMO établit des liens faiblement couplés entre les éléments de WSMO et fournit des moyens de médiation pour la résolution des incohérences qui peuvent surgir lors de la connexion des différents éléments définis par WSMO. Plus précisément, WSMO définit quatre types de médiateurs pour connecter les éléments WSMO : ***les médiateurs OO*** connectent et font la médiation entre les ontologies hétérogènes, ***les médiateurs GG*** connectent les buts, ***les médiateurs WG*** lient les services web aux buts, et ***les médiateurs WW*** connectent les services web en interaction résolvant ainsi leurs incohérences.

La définition d'une classe de médiateurs a la forme suivante :



Pour résumer les médiateurs de type :

- **ooMediator :** résolvent les conflits entre les ontologies importées.
- **wwMediator :** résolvent les incompatibilités d'interaction avec d'autres services web.
- **ggMediator :** permettent la réutilisation et le raffinement d'un but existant.
- **wgMediator :** permettent de lier les services aux buts.

3. Web Service Modeling Language (WSML)

WSML (Web Service Modeling Language) est un langage formel pour WSMO qui sert à décrire les ontologies et les services web sémantique de façon à prendre en charge tous leurs aspects descriptifs. Il comprend une variété de formalismes afin d'étudier leur pertinence dans la description des services web sémantique.

Il existe cinq variantes du langage WSML qui sont le « ***WSML-Core*** », « ***WSML-DL*** », « ***WSML Flight*** », « ***WSML-Rule*** » et le « ***WSML Full*** ».

Ces variantes diffèrent de leur expressivité logique et des paradigmes des langages utilisés, elles permettent aux utilisateurs de faire un compromis entre l'expressivité d'une variante et la complexité du raisonnement. Ce qui mène à dire que certaines variantes sont plus adaptées à certains types de raisonnement et de représentation.

La réutilisation des paradigmes des langages connus pour leurs descriptions logiques dans une seule plateforme syntaxique et être un langage (à part), basé sur WSMO, pour le model conceptuel des ontologies, des services web, des buts et des médiateurs ; ne sont pas les seules caractéristiques qui ont fait de WSML un langage remarquable. WSML est aussi connu par sa syntaxe qui est compréhensible par les simples utilisateurs, par l'utilisation des formalismes connus (Les variantes de WSML permettent la réutilisation des outils déjà développés pour ces formalismes) et par un ensemble de dispositifs qui permettent son intégration avec les technologies du web. Toutes ces caractéristiques et plus ont fait de WSML un langage unique en le comparant avec les autres propositions pour le web sémantique et les SWS [14].

4. Web Service Modeling eXecution environment (WSMX)

Il existe deux implémentations d'environnements d'exécution principaux dans la communauté WSMO, le WSMX (***W**eb **S**ervice **M**odeling e**X**ecution environment*) et IRS-III[9] (***I**nternet **R**easoning **S**ervice*). Tous les deux sont considérés comme plateformes complètes servant à intégrer les tâches impliqués dans l'utilisation des services web sémantique de la spécification des buts jusqu'à leur exécution. Dans notre cas, nous allons voir seulement la plateforme WSMX qui est notre banc d'essai.

WSMX est une initiative menée par l'institue de recherche DERI[10] et les projets européens (DIP[11], SEKT[12] et Knowledge Web[13]) [15]. WSMX est une implémentation de référence de WSMO et un environnement d'exécution [16] dans lequel les descriptions sémantique des buts des utilisateurs et celles des services web des fournisseurs peuvent être utilisés pour découvrir, composer, faire la *médiation*, *sélectionner* et *invoquer* les services web qui correspondent aux besoins de l'utilisateur. Une caractéristique de WSMX est qu'il ne définie pas les services en terme de leurs détails de conception ou d'implémentation, mais seulement en terme des fonctionnalités attendues et leur comportent externe visible.

4.1. Architecture de WSMX

La plateforme WSMX est conçue selon les standards du SOA, et a une architecture basée sur des composants, où chaque composant fait une part d'une fonctionnalité. Dans cette section une brève description des composants (figure 2.2) de WSMX et leur rôle est donnée.

- ***Data & communication protocols Adapters:*** comme leur nom indique, ce sont des adaptateurs pour les applications qui veulent communiquer avec WSMX. Ce type de composants est considéré par les auteurs de WSMX comme externe, mais quand les interfaces des applications n'intègrent pas des API WSMO, ces adaptateurs sont nécessaires pour se connecter à l'interface du serveur WSMX. Ils traduisent n'importe quel format de message au format WSML.
- ***WSMT (Web Service Modeling Toolkit):*** est un environnement de développement qui regroupe un bon nombre d'outils qui facilitent la création des descriptions des ontologies, des services web, des buts et des médiateurs, graphiquement ou en utilisant des éditeurs WSML. Comme ils permettent de se connecter au serveur WSMX et faire des inférences à base des descriptions créées.
- ***Communication Manager :*** a deux rôles principaux, le premier est de fournir aux adaptateurs une interface qui leur permet de recevoir et d'envoyer des messages WSML. Le deuxième est de traduire les messages dans n'importe quel autre format requis (pour une éventuelle invocation par des entités externes et n'utilisant pas le format WSML, c'est l'opération du « *Lowering* »). En invoquant

[9] IRS-III : http://technologies.kmi.open.ac.uk/irs/

[10] Digital Enterprise Research Institute (DERI) : http://www.deri.org/

[11] DIP: http://dip.semanticweb.org/

[12] SEKT : http://sekt.semanticweb.org/

[13] Knowledge Web : http://knowledgeweb.semanticweb.org/

le « *Communication Manager* », il est responsable de la traduction d'un message reçu d'une entité externe (qui utilise n'importe quelle représentation de données) en WSML. Cette traduction est souvent appelée « *Lifting* ».

- ***WSMX Manager :*** joue le rôle de coordinateur dans l'architecture, il gère le traitement de toutes les notifications en les passants aux composants appropriés.

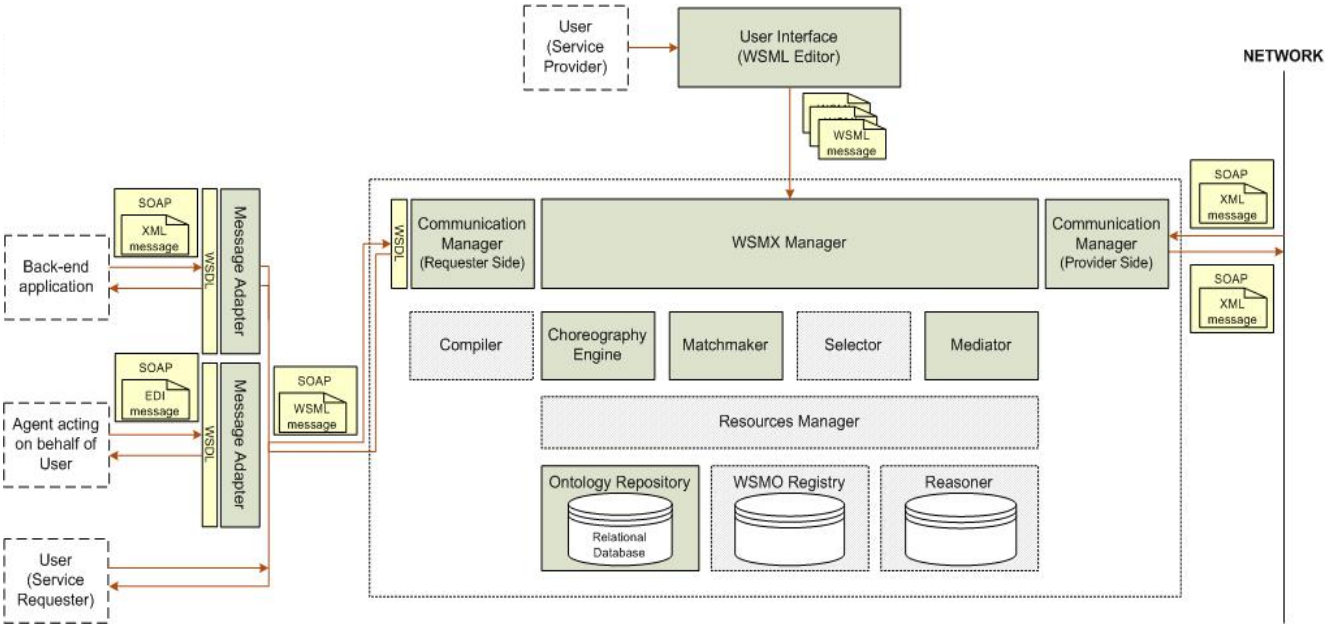


Fig. 2.2 Architecture de WSMX [www.wsmo.org]

- ***Compiler :*** Considéré comme le composant essentiel, il est utilisé dans toutes les opérations. Comme il sert à valider les documents WSML.
- ***Matchmaker :*** Offre un ensemble des services web en faisant correspondre les « *Capapilities* » déjà enregistrées avec les buts de l'utilisateur. Il accepte comme entrées un ensemble de services web enregistrés et le but du demandeur. Il fournie en sortie un ensemble de services web qui correspondent au but décrit.
- ***Mediator :*** Offre une médiation pour les données communiquées. Il permet, si nécessaire, de trouver un médiateur approprié pour une requête donnée. Fait la correspondance entre une ou plusieurs ontologies sources vers une ou plusieurs ontologies de destinations, qui sont utilisés soit dans les opérations de découverte-sélection ou bien dans l'opération d'exécution des services web. Ce composant est basé essentiellement sur les « *ooMediator* » de la spécification WSMO.
- ***Choreography Engine :*** Définie les modèles des communications des services web, en faisant la médiation entre les modèles des communications du demandeur et ceux du fournisseur des services. Pour ce faire, il utilise des médiateurs pour compenser le manque (ou la non-correspondance) dans les modèles de communication. Les règles de chorégraphie conçues, sont créées et enregistrer au même temps de leur conception.

4.2. Scénarios d'enregistrement, de sélection et d'invocation des SWS dans WSMX

Dans sa manipulation des services web, WSMX distingue entre deux phases principales qui sont : l'enregistrement du service web et son exécution.

Le processus d'enregistrement est fait par un éditeur WSMO et le compilateur (le composant).

Cette phase s'exécute en quatre étapes, illustrées dans la figure 2.3 :

1. Les descriptions WSML du service web sont définies dans un éditeur.
2. Le « *Communication Manager* » prend en charge les invocations entre l'éditeur et le compilateur.
3. Le « *Compiler* » vérifie la syntaxe du document WSML.
4. Le document WSML est ajouté au dépôt de services.

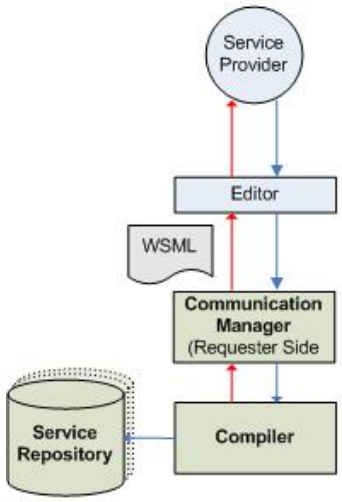


Fig. 2.3 processus d'enregistrement des services web dans WSMX [17]

Le processus d'exécution a deux phases : la découverte et l'invocation des services web. La première phase à pour objectif d'identifier tous les services web qui correspondent au but défini, tandis que la deuxième fait l'invocation d'un service sélectionné.

La phase de découverte, illustrée dans la figure 2.4, se fait aussi en quatre étapes :

1. L' « *Adapter* » fait convertir la requête en un but écris en WSML.
2. Le « *Communication Manager* » reçoit l'invocation.
3. Le « *Compiler* » vérifie la syntaxe du document WSML.

4. Le « *Matchmaker* » sélectionne les services qui correspondent au but décrit par le document WSML vérifié et utilise des médiateurs si nécessaire.

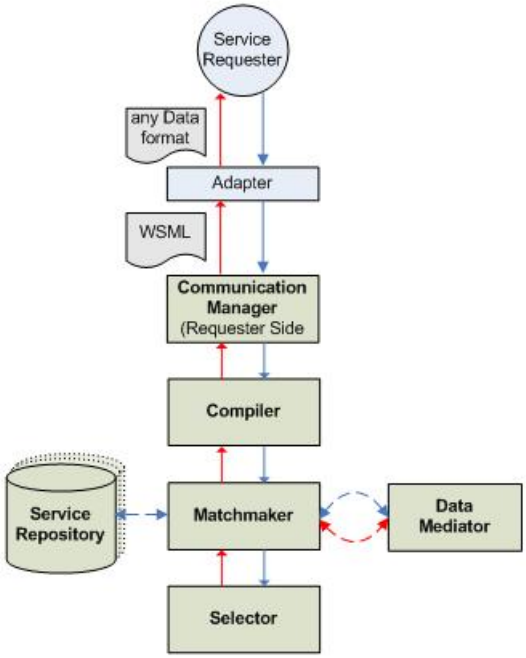


Fig. 2.4 processus de découverte des services web dans WSMX [17]

La phase d'invocation des services web, illustrée dans la figure 2.5, se fait en six étapes :

1. Le demandeur spécifie le service web et fournie une instance d'ontologie (non nécessairement en WSML).
2. L' « *Adapter* » s'assure que la requête est au format WSML.
3. Le « *Communication Manager* » du côté du demandeur envoi une description du service web et l'instance d'ontologie au « *choreography Engine* » et au « *Compiler* ».
4. Le « *choreography Engine* » fait la médiation de communication entre les deux « *Communication Manager* » (demandeur & fournisseur), tandis que le « *Compiler* » vérifie que les documents sont syntaxiquement valides.
5. Le « *Communication Manager* » du fournisseur convertit les données requises en XML, si nécessaire, en utilisant le « *Data Mediator* ».
6. Le fournisseur de service exécute ce dernier et renvoi des données, si nécessaire, soit d'une manière synchrone ou asynchrone.

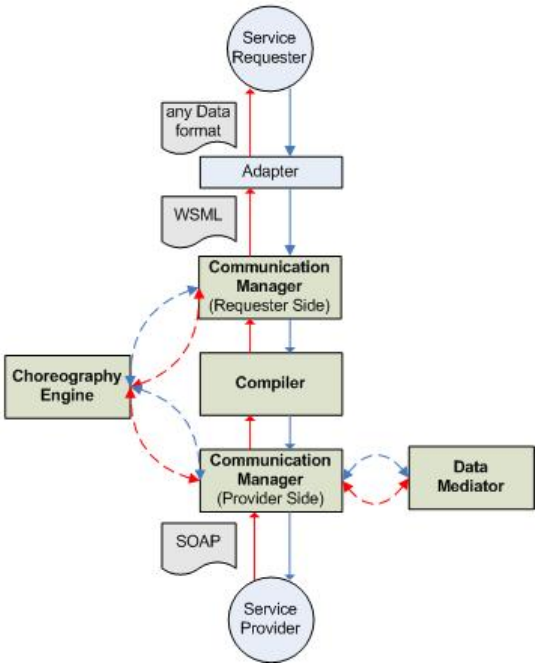


Fig. 2.5 processus d'invocation des services web dans WSMX [17]

5. Synthèse

Dans ce chapitre nous avons donné une introduction des concepts de base de l'approche WSMO, des spécifications du langage WSML et un aperçu de l'environnement d'exécution WSMX.

WSMO a pour but d'automatiser les processus d'enregistrement, de découverte et d'exécution des services web sémantique. Cependant WSMX comme son implémentation de référence, a pour objectif de permettre à n'importe quels systèmes d'information d'interagir avec d'autres tout en préservant la sémantique de leurs messages, leurs traitements et leurs protocoles. Ce qui permet de découvrir et d'exécuter des SWS en toute transparence vis-à-vis des spécifications utilisées dans leurs descriptions. Ce qui justifie notre choix, dans les chapitres suivants, d'utilisé des nœuds WSMX comme représentant des systèmes (nœuds) WSMO et non-WSMO dans leurs interactions.

Le chapitre suivant a pour objectif de donner une présentation des systèmes et des plateforme P2P pour mieux illustrer d'une part quelques approches existantes de découverte des SWS du chapitre d'après (chapitre IV), et d'autre part l'architecture de notre solution proposée.

Chapitre III

Aperçu sur les systèmes P2P

1. Présentation

Les systèmes pair-à-pair[14] constituent une alternative à l'approche client/serveur. L'idée de base est que les applications distribuées ne sont plus conçues en séparant les clients, consommateurs de services, et les serveurs, fournisseurs de services. Par contre, un système pair à pair est composé de pairs égaux en fonctionnalités, qui décident séparément de se joindre au système pour proposer un ou plusieurs services. Donc un système pair à pair est un réseau dynamique de pairs interconnectés, consommateurs et fournisseurs de services.

Contrairement aux idées reçues, le concept même du « pair à pair » est loin d'être récent. Bien au contraire, ce concept est à l'origine même d'internet [18]. Le protocole de communication d'internet a été conçu dès le début comme un système symétrique (pair à pair). Ainsi dès 1969, année de naissance d'internet, par le biais du réseau précurseur ARPANET, projet militaire créé par l'ARPA[15] ; Internet fonctionnait déjà de manière autonome. Les premières applications et protocoles d'internet, par exemple FTP pour l'échange des fichiers et Telnet pour utiliser une machine à distance, étaient déjà des applications pair à pair dans le sens où chaque pair pouvait être à la fois client et serveur. Le modèle P2P connait un très grand succès depuis la fin des années 90, époque à laquelle « *Napster* » une application P2P de partage des fichiers, créée par « *Shawn Fanning* », permet à des millions d'usagers connectés à Internet de télécharger et partager librement des fichiers multimédia.

Le succès des systèmes P2P est justifiable grâce à des points qui sont considérés comme des inconvénients des systèmes Client/Serveur et peut se résumer en :

- Le nombre de clients, le téléchargement et la demande en bande passante grandissante peuvent amener les serveurs à ne plus servir d'avantage de clients.
- Une caractéristique de cette architecture (Client/Serveur), est qu'elle nécessite très peu de puissance de calcul et de ressources du côté client, ce qui a été justifié à l'ère où cette architecture a vu le jour, alors qu'à notre ère, la plupart des machines clientes sont devenus très puissantes.
- Le client dans une telle architecture agit de manière passive : il demande des services auprès des serveurs mais il est incapable d'offrir des services aux autres.

Contrairement à l'architecture client/serveur, un réseau P2P ne dépend pas d'un serveur central pour accéder à un service quelconque. Le réseau P2P évite l'organisation centralisée de l'architecture client/serveur, en adoptant une organisation horizontale, fortement connectée et permettant aux machines à connexion intermittente de se retrouver sur le réseau en agissant tantôt en client et tantôt en serveur. L'avantage principal des

[14] Plusieurs appellations peuvent être utilisée pour identifier le concept du peet -to-peer parmi celles-ci : P2P, Pair à Pair Poste à Poste ou bien Servent.

[15] **ARPA:** Advenced Research Projects Agency dependant du DoD, Departement of Defense USA.

réseaux P2P est qu'ils distribuent les responsabilités de fournir des services sur l'ensemble des pairs du réseau.

En plus, ces réseaux exploitent mieux la bande passante en utilisant différents canaux de communications. Le problème des goulots d'étranglement au niveau des serveurs fortement sollicités ne se pose pas dans les réseaux P2P, car plusieurs pairs peuvent fournir des services et router d'autres pairs aux fournisseurs de services similaires afin de réduire la congestion du réseau.

2. Définition

Plusieurs définitions ont étés attribués au concept P2P dont voici une des plus significatives et complètes :

« Le P2P est une classe de systèmes ***auto-organisants*** qui permettent de réaliser une fonction globale[16] de manière décentralisée en tirant parti du partage des ressources et de la puissance de calcul disponibles sur un ***grand nombre d'extrémités d'Internet*** » [19].

Les termes « *auto-organisants, grand nombre, extrémités de l'internet* » expriment clairement les trois principaux enjeux du paradigme P2P qui sont (respectivement) : l'auto-organisation, scalabilité et volatilité des pairs.

- **L'auto-organisation :** Qui se traduit par le fait que le réseau n'est sous l'emprise d'aucune administration ou contrôle centralisé. Les pairs communiquent et s'organisent entre eux de manière autonome et coopérative. Vu l'étendue des réseaux et la diversité des plates-formes (système d'exploitation et équipements réseaux) une gestion du réseau est nécessaire à cause de :
    - l'hétérogénéité des plates-formes ;
    - l'accès aux ressources à distance (contourner les Firewalls et les NAT[17]).
- **La scalabilité :** Qui veut dire qu'un véritable réseau P2P devrait être capable de garder les mêmes performances en terme de localisation de données, de routage de messages et d'insertion de nouveaux pairs quelque soit le nombre de pairs composant le réseau P2P.
- **La volatilité des pairs :** Qui est synonyme d'un problème de disponibilité des ressources. En effet, le fait que les pairs deviennent détenteurs des ressources, il faudrait s'assurer de la disponibilité de ces ressources en absence des pairs qui les détiennent (chose qui est prise en charge par le serveur dans les réseaux client/serveur) ce qui constitue l'un des défis majeurs des développeurs des applications P2P (surtout pour le calcul distribué), dont il faut prendre en compte:
    - Les déconnexions violentes des pairs;
    - La découverte automatique de ressources nouvellement disponibles sur le réseau.

[16] Offrir à une communauté un service de manière décentralisée

[17] **NAT :** Network Address Translation

3. Architectures des systèmes P2P

Il existe trois grandes familles d'architecture des systèmes P2P, qui sont : l'architecture *centralisée*, l'architecture *décentralisée* et l'architecture *hybride*.

3.1. Architecture centralisées

Ce type d'architecture repose sur un serveur central sur lequel se connectent les utilisateurs et qui permet la gestion des partages, la recherche et l'insertion d'informations. Chaque utilisateur recherche un service désiré sur le serveur. Ce dernier lui indique alors la liste des pairs sur lesquels il est susceptible de le trouver. L'utilisateur se connecte alors directement à l'un de ces pairs pour l'exécution du service. Ce qui est très similaire au processus de découverte-sélection et d'invocation des services web en utilisant un simple UDDI.

Cette façon de procéder est très fragile puisque le serveur central est indispensable au réseau. Ainsi, si le serveur est débranché, à la suite d'une panne (par exemple) c'est tout le réseau qui s'effondre. Le réseau Napster est un très bon exemple de cette architecture.

Les avantages de cette architecture sont :

- l'efficacité des recherches par une indexation centralisée;
- la facilité de la mise en œuvre;
- la mise à jour en temps réel de l'index central.

En contre partie :

- le réseau est complètement dépendant du serveur central qui est une source de vulnérabilité;
- et l'anonymat n'est pas garanti ;

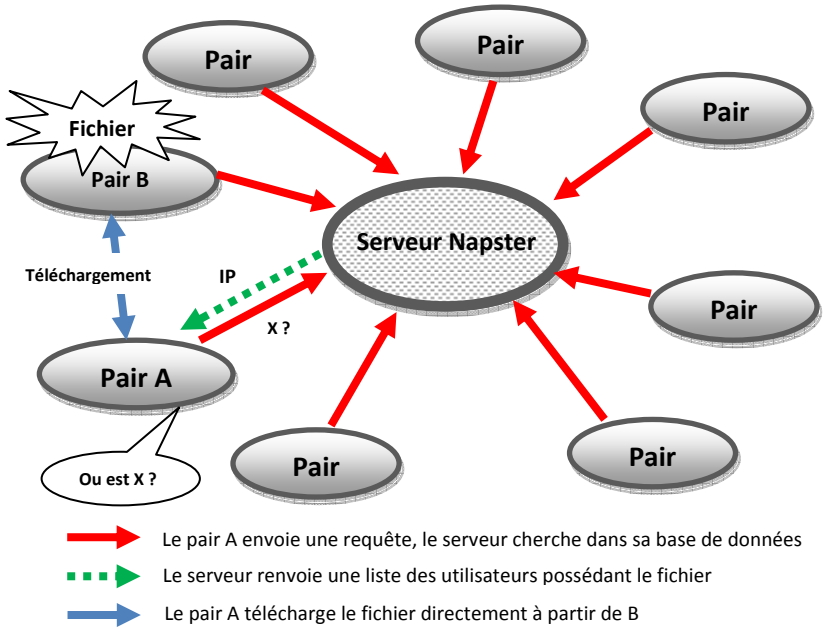


Fig. 3.1 .Architecture P2P centralisée.

3.2. Architectures décentralisées

C'est l'architecture P2P pure où il n'y a pas de serveur central et tous les pairs ont un rôle équivalent. Les membres du réseau se découvrent dynamiquement en relayant les requêtes de proche en proche (Figure 3.2). Un pair transmet une requête à ses voisins qui font de même et ainsi de suite (notion d'inondation) jusqu'à arriver à destination. Pour limiter la génération d'un nombre exponentiel des messages par ce procédé, on limite la propagation des messages jusqu'à un horizon défini par un paramètre de durée de vie de la requête, dit : « TTL[18]» qui est décrémenté sur chaque pair. Une fois un pair possédant la ressource est trouvée, une connexion directe s'établit entre les pairs impliqués.

Ce modèle a pour avantages la non vulnérabilité des pairs, du moment qu'il est un système pair à pair pur, ainsi que l'anonymat des utilisateurs. En contrepartie, il génère beaucoup de trafic. Cependant, des protocoles optimisés ont pu être mis en place, basés sur les tables de hachage distribuées permettant de réaliser des recherches en un nombre de messages croissant de façon logarithmique en fonction du nombre des pairs du réseau comme CAN, Chord, Freenet, Tapestry et Pastry [18],[20]. Les réseaux Gnutella sont un très bon exemple de cette architecture.

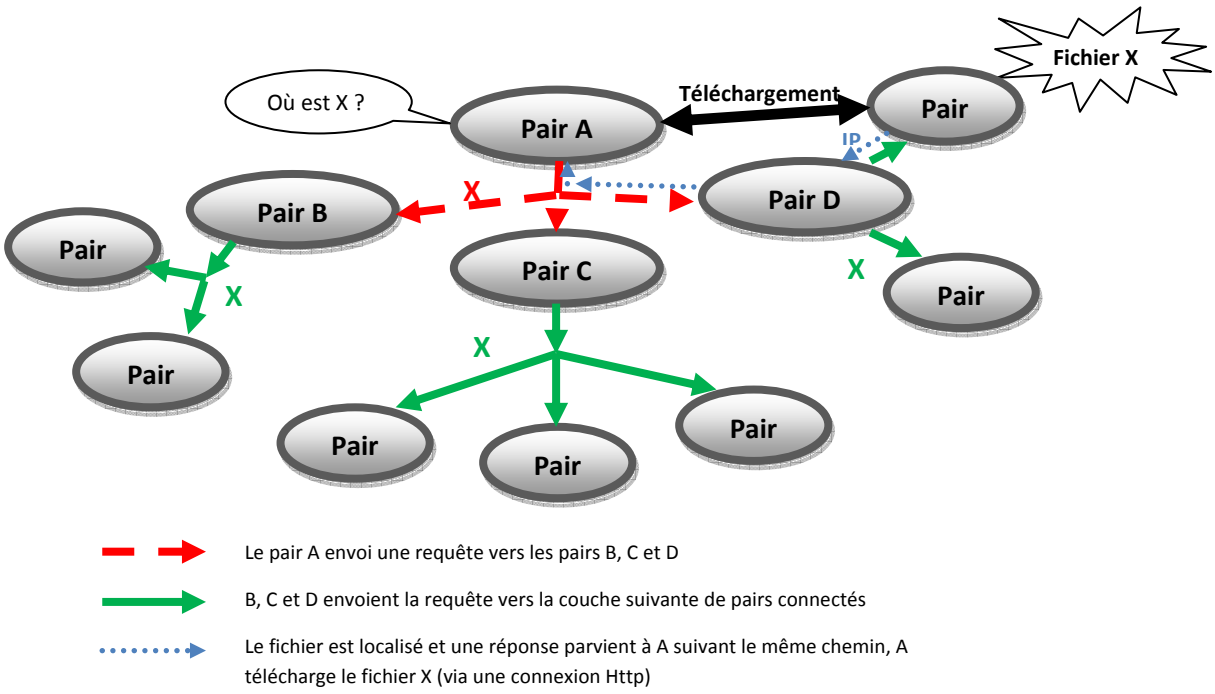


Fig. 3.2. Architecture P2P décentralisée (réseau GNUtella).

3.3. Architectures hybrides

Une solution hybride consiste à utiliser des Superpeers (super pairs), éléments du réseau choisis en fonction de leur puissance de calcul et de leur bande passante. Ces derniers gèrent un groupe de postes, jouent le rôle d'intermédiaire dans la transmission des requêtes et réalisent des fonctions utiles au réseau, comme l'indexation des informations. Lorsqu'un poste se connecte, il est affecté à l'un de ces groupes.

[18] **TTL** : Time To Live

Les serveurs sont reliés entre eux suivant un mode décentralisé. Chaque serveur référence le contenu de son groupe, si un serveur ne possède pas le fichier (la ressource) demandé par un des pairs de son groupe, il transmet à ce dernier l'adresse d'un autre super pair où sera renouvelée la requête. Cette solution, rend le réseau un peu moins robuste que dans un réseau de type Gnutella. Cependant, si un super-pair tombe en pane, son groupe de pairs sera déconnecté au reste du réseau et non plus le réseau tout entier comme le cas d'un réseau Napster. Cette architecture est illustrée dans la figure 3.3.

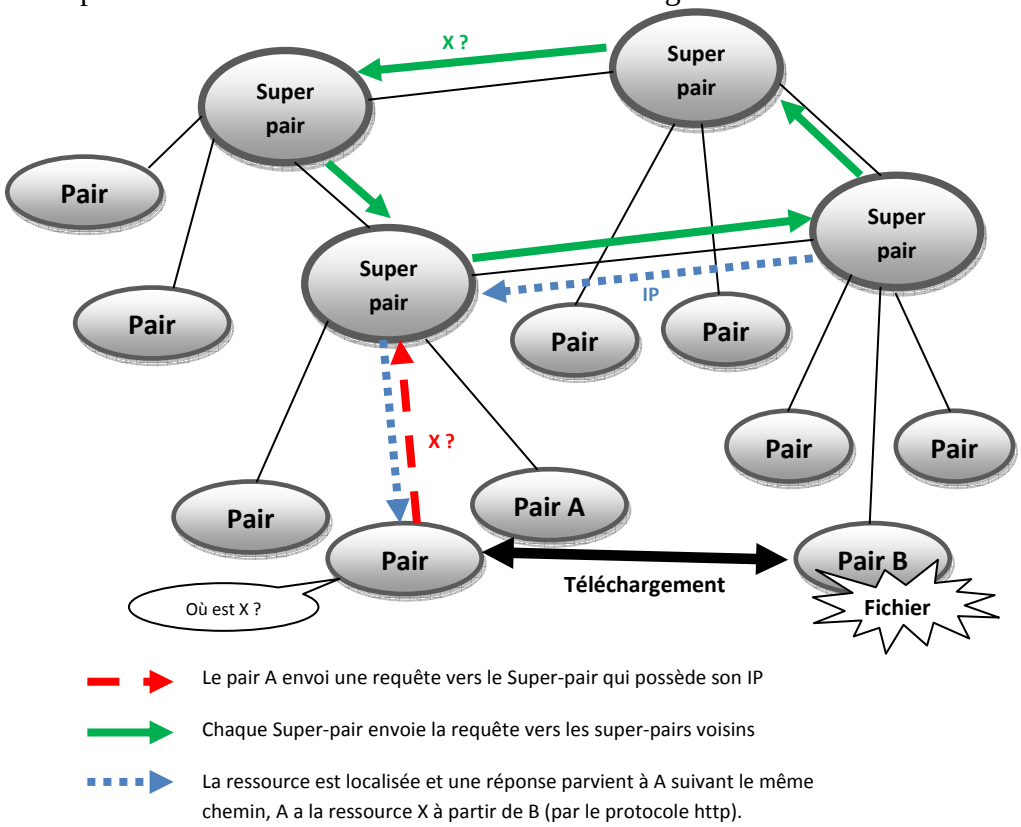


Fig. 3.3. Architecture P2P Hybride.

4. Tables de hachage distribuées

La première problématique posée pour les systèmes P2P est celle de la découverte et de l'accès aux ressources. Cette problématique est un résultat évident vu les caractéristiques de décentralisation et de connectivité Ad-Hoc des pairs.

Pour répondre à cette problématique, des solutions diverses dites basées DHT[19] ont été proposées par plusieurs acteurs et laboratoires de recherche qui travaillent dans le domaine. Ces solutions portent essentiellement sur l'utilisation des DHT pour localiser les ressources dans un réseau P2P ainsi que le routage des requêtes et le téléchargement des fichiers avec un moindre coût (en terme de gourmandise en bande passante et de temps de téléchargement) et plus d'efficacité (en terme de scalabilité et de dynamicité), car c'est la localisation des ressources et le routages des différents messages entre pairs qui constituent le premier souci des développeurs d'applications P2P.

[19] **DHT :** Distributed Hash Tables (Tables de hachage distribuées).

Les DHT offrent des méthodes proches de celles associées aux structures de données de type table de hachage. On utilise donc le plus souvent ces systèmes pour publier des objets et les récupérer grâce à une clé d'accès arbitraire qui identifie l'objet de manière unique. On utilise alors deux primitives de base qui sont l'insertion d'un objet identifié par une clé et la récupération d'un objet via sa clé identifiante. On peut également avoir une méthode de suppression d'un objet grâce à sa clé identifiante.

Les divers systèmes de DHT existants sont ceux motionnés dans la figure 3.4 différent principalement par la nature et la taille de l'espace de nommage utilisé ainsi que par la métrique de distance utilisée lors du routage des messages.

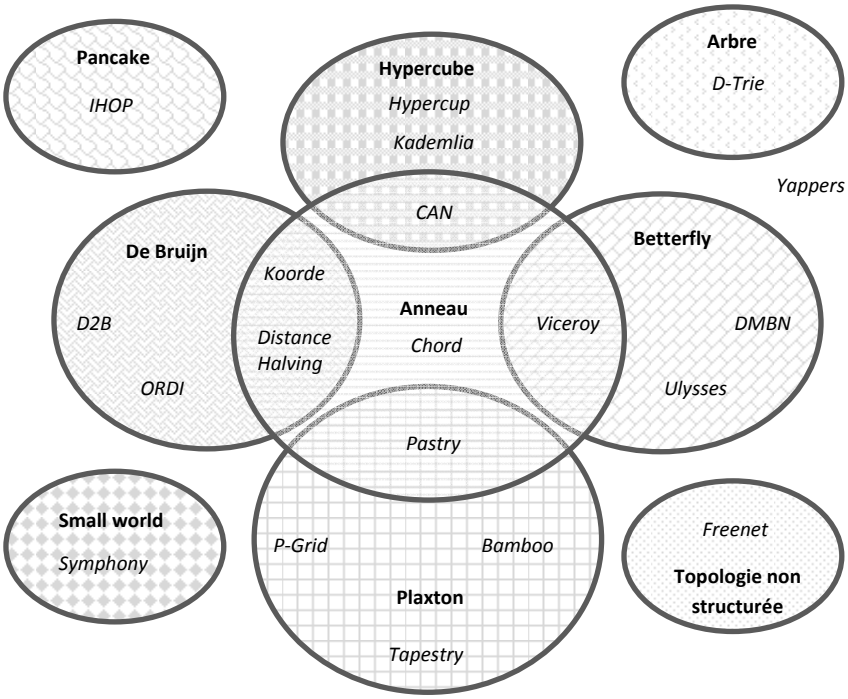


Fig. 3.4 .Taxonomie topologique des différentes DHTs. Les familles sont inscrites en caractères gras et les propositions en italique [20].

Les systèmes basés sur les DHT ont prouvé leur efficacité en terme de scalabilité, de routage des messages, de non gourmandise en bande passante et de dynamique dans la gestion du réseau (insertion et suppression des pairs) [21]. Malgré leurs bonnes intentions, les applications bâties à base de ces solutions pèchent par leur hétérogénéité et leur non interopérabilité. Pour cela, d'autres solutions ont vue le jour, ce sont les plates-formes de développement d'application P2P.

5. Plate-formes de développenment d'applications P2P

La plupart des logiciels P2P existants ont été développés d'une manière spécifique sans se référencer à des standards propres au P2P. Des initiatives ont été entreprises pour offrir un tel standard. Elles proposent un environnement permettant de développer des applications P2P en offrant les fonctionnalités de base telles que la gestion des pairs, des groupes de pairs, l'indexation des ressources, la découverte des ressources, la communication entre pairs et autres aspects comme la sécurité.

Ainsi, *Sun* propose sa plateforme P2P *JXTA*[20] basée sur les technologies Java et XML, *Microsoft* de sa part, intègre dans sa plate-forme *.Net* des outils pour développer des applications P2P, tels que les *services web*, les *services processes*[21], *Windows Forms*[22] et les *Web Forms*[23]. Une autre importante plateforme est celle de *Jini*[24] qui délivre un ensemble de mécanismes (un ensemble d'APIs sous forme d'interfaces Java) qui permettent aux pairs de s'interconnecter et de fournir des services pour les membres du réseau.

- **JXTA vs Microsoft.Net :** Le développement des applications P2P avec l'architecture .NET a comme principal désavantage par rapport à JXTA de ne pas être créé spécifiquement pour ce genre d'applications et demande de la part des programmeurs une plus grande charge de travail. En effet, l'utilisateur doit, par exemple, utiliser les services web pour construire une application P2P. JXTA propose par contre une architecture étendue et complète spécifique aux systèmes P2P. De plus, cette architecture est en pleine expansion et intègre de nombreux concepts propres aux systèmes P2P.

- **JXTA vs Jini [22] :** Même si les deux plateformes partagent les mêmes objectifs de permettre une organisation et une découverte dynamique des ressources d'un réseau en respectant l'*ubiquité* et l'interopérabilité entre les dispositifs membres du réseau, ils diffèrent par le fait que *Jini* est intimement lié au langage de programmation Java. En effet, contrairement à JXTA qui défini un ensemble de protocole pour publier et rechercher des ressources dans le réseau, dont l'implémentation peut être faite avec n'importe quelle langage de programmation, *Jini* défini un ensemble d'API sous forme d'interfaces Java qui implémentent les concepts utilisés pour manipuler les services.
  Autre inconvénient de *Jini* par rapport à JXTA est l'utilisation du modèle de données de Java, donc pour faire passer ces données dans le réseau, *Jini* est obligé d'encapsuler ces données dans des objets Java. Par contre, JXTA s'appuie sur les documents XML pour représenter les données, et dans ce cas un simple parsing[25] du document est suffisant.

6. La plate-forme JXTA

JXTA est une spécification en XML d'un ensemble de protocoles P2P génériques qui permettent à n'importe quel périphérique connecté au réseau (téléphones portables, PDA, PCs et serveurs) de communiquer et de collaborer en tant que pairs. Une des caractéristiques de ces protocoles est qu'ils sont indépendants des langages de programmation.

[20] JXTA pour JuXTApose -- https://jxta.dev.java.net.
[21] Permettent de gérer la découverte des pairs lorsque le protocole HTTP n'est pas utilisé.
[22] Permettent d'écrire des interfaces utilisateurs graphiques permettant à un utilisateur de s'authentifier, d'effectuer des recherches et de partager du contenu.
[23] Permettent de retourner facilement le contenu des pairs sous forme HTML.
[24] http://www.jini.org
[25] Le **parsing** est le traitement syntaxique d'un document.

JXTA est une plateforme de développement d'applications P2P tout à fait originale par la façon dont elle perçoit le problème ainsi que par les mécanismes de mise en œuvre d'un réseau P2P qu'elle offre en ce situant à un niveau d'abstraction assez élevé.

Au moyen de la plate-forme JXTA, *Sun* offre aux développeurs et aux utilisateurs un ensemble de services de haut niveau permettant la création et l'utilisation d'applications P2P. Ces services et ces applications ont l'avantage d'être interopérables et multi-plateformes. JXTA est un ensemble des protocoles qui peuvent être implémentés dans tout langage de programmation. Pour l'instant, *Sun* fournit une implémentation de JXTA en Java, en C, et en C#, mais d'autres implémentations existent en Perl, en Python et en Smalltalk.

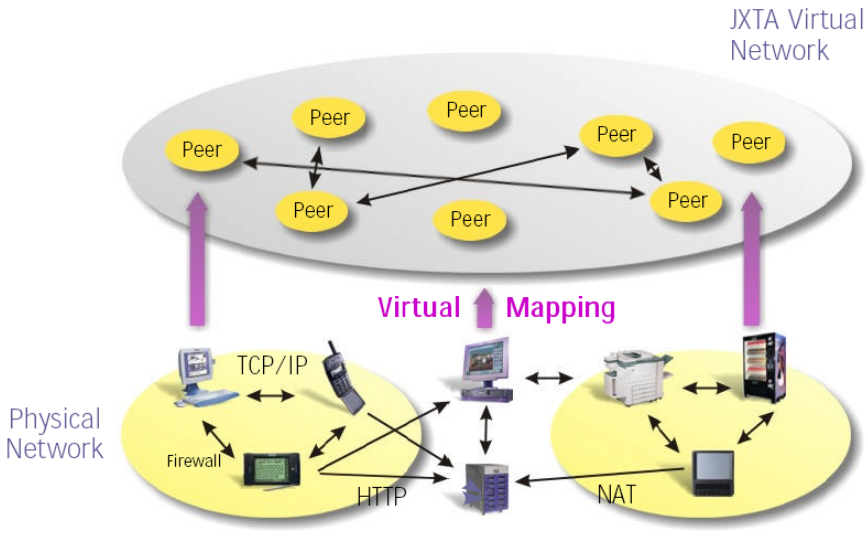


Fig. 3.5 .La vision générique (Overlay) de la plate-forme JXTA [23].

6.1. Objectifs de la plate-forme JXTA

La technologie JXTA cherche à surmonter les points faibles des différents systèmes P2P existants. Elle est désignée pour répondre aux objectifs suivants:

- **La factorisation :** C'est la mise en commun des fonctionnalités identiques par découpage en couches modifiables indépendantes les unes des autres. En effet, les systèmes pair-à-pair utilisent souvent des protocoles différents mais qui ont les mêmes fonctionnalités, comme la découverte de ressources, la communication entre pairs et la gestion des groupes. Ainsi, pour une amélioration de ces fonctionnalités, celles-ci doivent être découpées en couches indépendantes les unes des autres.

- **Interopérabilité :** Les différents systèmes élaborés sont pour l'instant incompatibles entre eux. Or, il serait souhaitable de pouvoir bénéficier des ponts d'échanges entre ces systèmes. Le développement de services spécialisés basés sur un même environnement générique est nécessaire afin de rendre ces services interopérables.

- **Indépendance vis-à-vis de la plate-forme d'accueil :** L'objectif d'un système pair à-pair à grande échelle est d'être déployé sur un grand nombre de pairs. Il faut donc disposer d'un environnement générique et indépendant de la plateforme d'accueil. Une spécification précise et claire de l'ensemble des protocoles et de leurs formats à respecter lors de l'implémentation de l'environnement, est nécessaire. Ainsi, le choix du langage ne doit pas influencer l'interopérabilité des différentes implémentations existantes.

- **Ubiquité :** Le dernier objectif est l'ubiquité. Il matérialise le désir que la plateforme JXTA soit implantable sur n'importe quel périphérique doté d'un cœur digital « *digital heartbeat* ». Cette notion est très générale puisqu'elle comprend les PDA, les GSM, les routeurs réseaux, les ordinateurs de bureau, les serveurs de données (ou d'applications).

6.2. Architecture de la plate-forme JXTA

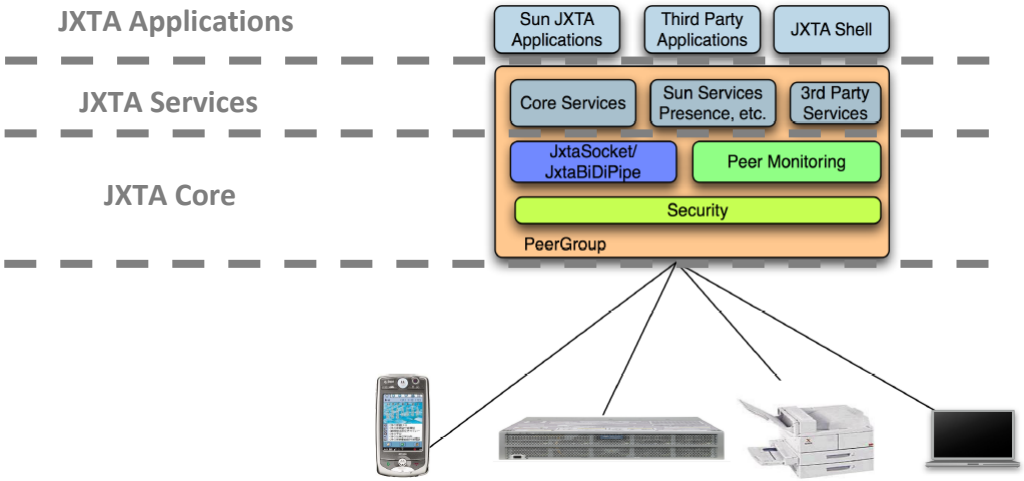


Figure 3.6. Architecture logique du framework JXTA [24].

- **La couche noyau (Core)** Le noyau de JXTA encapsule les primitifs minimaux et essentiels qui sont communs à la gestion d'un réseau P2P. Il offre aux développeurs un ensemble de mécanismes de base pour les applications P2P permettant la découverte, le transport des messages (y compris de traverser les Firewalls et les NATs), la création des pairs, des groupes des pairs, et les primitifs de sécurité telles que l'anonymat et le cryptage.

- **La couche Service** Permet d'étendre les mécanismes de base offerts par la couche noyau et ainsi faciliter le développement d'applications de plus haut niveau. Les principaux services proposés dans cette couche sont des mécanismes de recherche et d'indexation (***JXTA Search***), de partage de ressources (***JXTA cms***), de stockage (***jxtaspaces***), de traduction de protocoles (***JXTAxmlrpc***,***jxta-rmi***,***jxtasoap***,…),etc.

- **La couche Applications** Au sein de cette couche, les applications seront développées sur la base des services disponibles dans la couche services. Exemples de ces applications : la messagerie instantanée P2P, partage de document et de ressource, la gestion et la livraison de contenu multimédia, les systèmes d'email P2P, et beaucoup d'autres.

La frontière entre les services et les applications n'est pas rigide. Une application d'un pair donné peut être vue comme service par un autre pair. Le système entier est conçu pour être modulaire, permettant aux développeurs de sélectionner et choisir une collection des services et d'applications qui convient le plus à leurs besoins.

6.3. Composants d'un réseau JXTA

Un réseau JXTA consiste en un ensemble de pairs (***peers***). Ces pairs peuvent s'auto-organiser dans des groupes des pairs (***peer groups***) fournissant un ensemble des services.

Conséquence de la nature de l'architecture adoptée par JXTA dans l'organisation des pairs et le flux des données qui est l'architecture *hybride*, il existe deux types de pairs JXTA, les simples pairs et les super pairs (***Super peers***). Les super pairs jouent, en plus de leur rôle de simples pairs, le rôle d'annuaires[26] pour les pairs qui leurs sont directement connectés, comme ils gèrent périodiquement la liste des autres super pairs.

Les pairs JXTA annoncent leurs services sous forme de documents XML appelés annonces (***advertisements***). Les annonces permettent aux autres pairs dans le réseau de savoir comment se connecter et interagir avec les pairs qui offrent ces services.

Les pairs JXTA utilisent les tubes (***Pipes***[27]) pour envoyer des messages vers les autres pairs. Les pipes constituent un mécanisme de transfert asynchrone et unidirectionnel utilisé pour assurer la communication entre les pairs. Le message est un simple document XML qui constitue l'unité de base d'échanges entre pairs. Les tubes (***pipes***) sont reliés à des extrémités spécifiques (***Endpoints***).

Quatre aspects essentiels distinguent JXTA des autres modèles des réseaux distribués :

- Utilisation des documents XML pour décrire les ressources disponibles dans le réseau;
- Abstraction des tubes par rapport aux pairs, et des pairs par rapport aux extrémités, sans dépendre d'un moyen de nommage centralisé tel que le DNS ;
- Utilisation d'un plan d'adressage uniforme et plat (non hiérarchique) qui repose sur les Peer IDs ;
- Une infrastructure décentralisée de recherche basée sur la table de hachage distribuée (***SRDI***[28]) pour l'indexation des ressources.

[26] Ils sont des annuaires des pairs et des ressources détenues par ces pairs.

[27] A la base de la simple définition des pipes JXTA, d'autres types de pipe ont vu le jour tels que : Unicast pipe, Propagate pipe, SecureUnicast pipe, Bidirectionnal pipe…

[28] **SRDI: S**hared **R**essource **D**istributed **I**ndex (la DHT de JXTA); www.di.unipi.it/~ricci/jxta-dht.pdf

7. Synthèse

Le P2P est un modèle distribué dont les entités de base sont des pairs qui jouent à la fois le rôle des clients et des serveurs. Les systèmes qui utilisent ce modèle peuvent être classés en trois catégories : centralisée, pure et hybride.

Les nœuds dans un tel modèle se caractérisent par leur décentralisation et leur connectivité Ad-Hoc, leur utilisation d'overlay et l'anonymat. Les deux premières caractéristiques du modèle P2P pose un problème majeur est celui de « la localisation des ressources ». Pour répondre à cette problématique plusieurs solutions à la base des DHT ont été proposées, ces solutions ont prouvé leur efficacité en terme de scalabilité, de routage des messages, de non gourmandise en bande passante et de dynamique dans la gestion du réseau (insertion et suppression des pairs).

Malgré leurs bonnes intentions, les applications bâties à base de ces solutions pèchent par leur hétérogénéité et leur non interopérabilité. Pour cela, d'autres solutions ont vue le jour, ce sont les plateformes de développement d'applications P2P. Plusieurs plateformes de développement d'applications P2P ont été proposées dont les plus importantes sont : Microsoft .Net, JINI et JXTA. Et nous avons vu que la plateforme JXTA de *Sun* Microsystems, par rapport à JINI et .Net, a du potentiel et de divers avantages à offrir, tel que : l'échange des messages XML, l'indépendance vis-à-vis de la plateforme d'accueil et l'ubiquité.

Le projet JXTA a été conçu en respectant la norme SOA (Service-Oriented Architecture) et comprend plusieurs services de haut niveau qui facilite la création et l'utilisation des applications P2P. En plus, JXTA intègre un nombre de protocoles générique qui permet à n'importe quel dispositif connecté au réseau de communiquer et de collaborer en tant que pair.

Dans la solution que nous avons donné pour la collaboration dans le processus découverte distribuée des SWS, nous avons choisi la plateforme JXTA pour faire gérer les communications P2P entre les différents nœuds participant à ce processus, mais notre choix d'une approche P2P et d'utikiser JXTA comme plateforme de communication, n'a pas été fait juste à base des avantages cités dans ce chapitre, néanmoins ce choix se base aussi sur les travaux qui sont présentés dans le chapitre suivant.

Chapitre IV

Approches existantes pour la découverte distribuée des SWS

1. Introduction

Comme mentionné dans la section 4 du premier chapitre, les services web sémantiques ont pour objectif d'automatiser les processus de description, de publication, de découverte, de sélection, de composition et d'invocation (fourniture) des services web, soit tout le cycle de vie d'un service web.

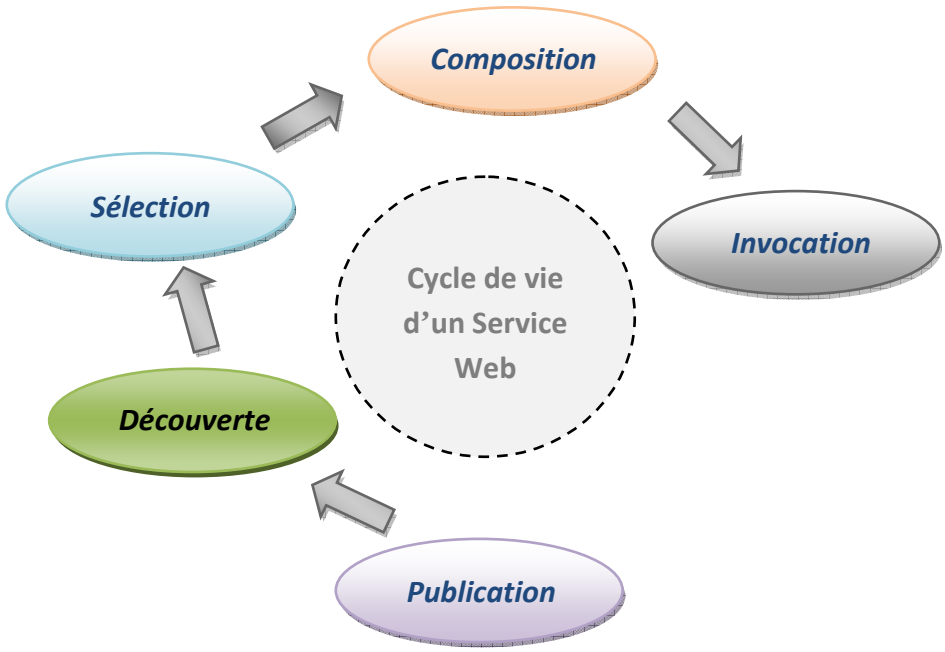


Fig. 4.1 Cycle de vie d'un service web.

De ces cinq étapes du cycle de vie d'un service web, la découverte est l'étape la plus importante. Au moins elle attire la majeure attention des communautés de recherche et des industriels. Ceci est justifiable en considérant les deux points suivant :

- On ne peut pas utiliser un service que si on est informé de son existence ou qu'on l'a déjà découvert.
- La différence entre les fonctionnalités qu'un service fourni et les fonctionnalités désirées par un demandeur de service, dépend énormément de la localisation des meilleurs services disponibles.

Plusieurs propositions pour des mécanismes de découverte ont été faites. Mais une solution efficace doit prendre en compte deux critères important, qui sont : la ***complétude*** (*Completeness*) et la ***justesse*** (*Correctness*), dans un model issu d'une analyse profonde de la majorité des problèmes de découverte des services [25].

La complétude dans de telles solutions implique que toutes les entités importantes doivent être découvertes. Tandis que la justesse implique que seulement les entités importantes sont découvertes.

Un processus classique de découverte est fait en utilisant des annuaires des services web, tels que ebXML[29] et UDDI qui sont un peut limités dans la description des services web et ce qu'ils fournissent, en plus de leur centralisation qui est vue comme inconvénient dans certaines circonstances.

De ce fait, l'absence d'un mécanisme de découverte distribué est un réel problème qui peut limiter le grand potentiel de cette technologie, qui requiert l'intervention humaine pour localiser les services désirés pour une requête donnée. Cependant, cette intervention limite considérablement ces technologies en termes de scalabilité et d'efficacité. Les services web sémantiques est la promesse qui veut limiter l'intervention humaine en utilisant les standards du web sémantique.

Une bonne implémentation du paradigme des services web sémantiques nécessite un mécanisme de découverte automatique et scalable. Sans oublier la nature distribuée des services web sémantique qui doit être prise en considération dans la conception de tels mécanismes [25].

Comme la plateforme WSMX couvre tout le cycle de vie d'un service web et interagit avec d'autres systèmes[30] en utilisant les « *WSMX Adapters* ». Pour ce qui suit, nous allons présenter les différentes approches de découverte distribuée des services web sémantiques en se focalisant sur des nœuds implémentant la plateforme WSMX. Ceci a pour but de simplifier la complexité du domaine du problème et d'éliminer toute source d'ambigüité.

2. Découverte distribuée dans WSMX (approches possibles)

Une des plus importantes étapes dans la réalisation d'un but (goal) d'utilisateur en utilisant WSMX est la découverte. Dans son processus de découverte WSMX fait la distinction entre la notion de *service* et celle de *service web*. Un service est vu comme une instance concrète d'un service web qui a toutes les entrées bien spécifiées. Tandis qu'un *service web* est considéré comme une entité abstraite (une classe des services concrets).

Le processus de découverte peut être défini par ces trois suivantes étapes : la *découverte des buts (**Goal discovery**),* la *découverte des services web (**web service discovery**) et la découverte des services (**service discovery**).*

Dans la première étape, des buts prédéfinis et abstraits vont être sélectionnés. Puisque WSMX défini des modèles (*templates*) des buts pour l'ensemble des services web qu'il fourni et mesure le degré de similarité entre le but de l'utilisateur et ces buts prédéfinis, ensuite selon ce degré de similarité un ensemble de buts est sélectionné.

La deuxième étape, *découverte des services web*, donne un ensemble des descriptions abstraites des services web qui correspondent aux buts déjà sélectionnés.

La dernière étape, *découverte des services*, permet de trouver tous les services réels, services concrets qui leurs descriptions abstraites sont découvertes dans l'étape précédente.

[29] **ebXML:** (Electronic Business using eXtensible Markup Language). http://www.ebxml.org

[30] Des systèmes utilisant autres specifications pour la description des SWS tels que : OWL-S, WSDL-S…

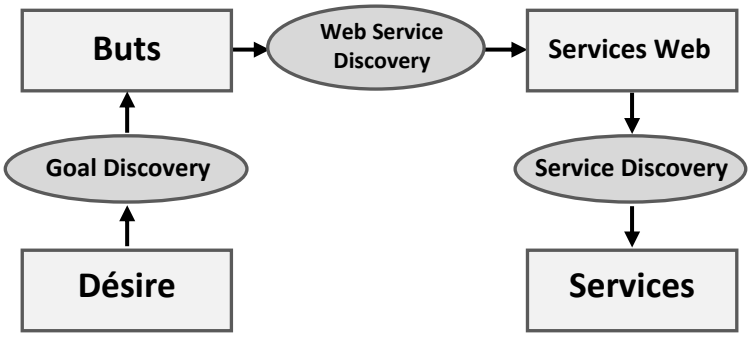


Fig. 4.2 Les majeures étapes dans le processus de découverte d'un service web [10].

Cependant le processus de découverte décrit ci-dessus est fait localement. Mais quand les services enregistrés localement vont ne pas satisfaire les désirs de l'utilisateur, le module « *WSMX Discovery Engine* » va collaborer en transmettant les requêtes à d'autres systèmes dans le réseau.

La découverte distribuée peut se faire en utilisant n'importe quelle des trois approches suivantes [25] : l'approche centralisée qui utilise des solutions similaires à ceux des UDDI, l'approche P2P et l'utilisation du paradigme du « *Triple Space* ».

2.1. Approche centralisée pour une découverte distribuée dans WSMX

Connue comme la plus simple à mettre en œuvre, cette approche dédie tout un serveur pour jouer le rôle d'un annuaire central pour les plateformes WSMX existantes.

Dans cette approche, chaque plateforme WSMX doit être enregistrée au niveau d'un annuaire central. Cet annuaire permet de faire enregistrer les plateformes WSMX et non pas les services web, ces derniers sont enregistrés au niveau des plateformes WSMX[31].

Ainsi, un demandeur de service (*Service Requester*) doit, tout d'abord, adresser une requête à un annuaire, central et connu au préalable, des *nœuds WSMX*[32] pour avoir la liste des nœuds WSMX existant dans le réseau. L'annuaire central répond à cette requête et fourni cette liste au demandeur de service.

Ayant la liste des nœuds WSMX, le demandeur de service parcourt la liste obtenue dans l'étape précédente et adresse sa requête (la vraie requête concernant les services web) aux nœuds figurant dans cette liste un par un. A chaque fois qu'un nœud WSMX reçoit une requête, la plateforme WSMX implémentée dans ce nœud traite cette requête localement et renvoie le résultat (même si ce résultat est négatif) au demandeur de service.

Ce processus s'arrête quand le demandeur du service reçoit un résultat positif ou que tout les nœuds WSMX dans la liste sont consultés. La figure 4.3 illustre ce scénario de découverte.

[31] Couvrant tout le cycle de vie d'un service web, Les plateformes WSMX peuvent jouer le rôle de fournisseur de service, d'annuaire de ses services et demandeur de services aux près d'autres nœuds WSMX.
[32] Un nœud WSMX est un nœud qui implémente une plateforme WSMX.

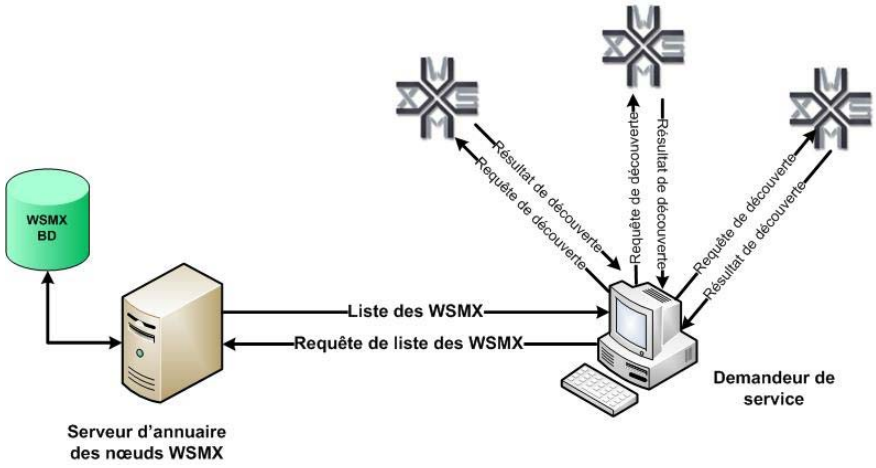


Fig. 4.3 Scénario d'annuaire centralisé pour la découverte distribuée dans WSMX.

Cette approche a l'inconvénient d'être non scalable et extrêmement inefficace, puisqu'elle surcharge le demandeur de service à faire de multiple requêtes à différents plateforme WSMX. Sans oublier que si le serveur d'annuaire tombe en panne c'est tout le réseau qui s'effondre.

2.2. Approches P2P pour une découverte distribuée dans WSMX

La caractéristique la plus importante est que dans ce type d'approches il n'y a pas d'annuaire centralisé dédié. Tous les pairs sont égaux et coopèrent avec les autres pour répondre aux requêtes d'utilisateurs. Ce qui permet de contourner le problème du seul point d'échec de tout le système.

Dans les approches de ce type, on peut imaginer des solutions basées sur des réseaux P2P des plateformes WSMX dont les pairs font usage des protocoles P2P pour se trouver les un les autres. Plusieurs alternatives peuvent être considérées en se basant sur la manière dont les pairs coopèrent avec leurs semblables. Les auteurs de [25] mentionnent deux alternatives, présentées ci-dessous, une approche P2P pure et une approche P2P hybride. Les deux approches incorpore localement sur les pairs un (des) annuaire(s) des nœuds WSMX.

Autre caractéristique est qu'un demandeur de service, dans de telles approches, doit connaitre au moins un pair WSMX pour lequel la requête va être adressée.

2.2.1. Approche P2P pure

Dans cette approche tous les pairs ont des rôles égaux, un pair est serveur de ces propres services et client chez les autres pairs.

Puisqu'il n'y a pas d'annuaire central pour les autres pairs dans le réseau (qui sont des nœuds WSMX dans notre cas), pour coopérer un pair doit transmettre sa requête à un des pairs qu'il connait. Si le but défini dans la requête n'est pas satisfait, donc la requête est transmise à un autre pair connu.

La requête sera transmise de proche en proche, ce qui peut entrainer l'inondation du réseau avec des messages véhiculant cette requête. Ceci est perçu comme inconvénient majeur pour ce type d'approche. Pour optimiser cette approche, certains critères peuvent être définis pour sélectionner le pair pour lequel la requête va être transmise.

Un autre inconvénient est que les pairs dans cette approche ont toujours besoin de gérer les annuaires locaux des pairs WSMX qu'ils connaissent.

Dans le cas où un pair a une réponse positive, un message est retourné au client. Le service peut alors être exécuté sur la plateforme WSMX du pair qui le détient et le résultat est transmis au demandeur de service. La figure 4.4 illustre le fonctionnement de cette approche.

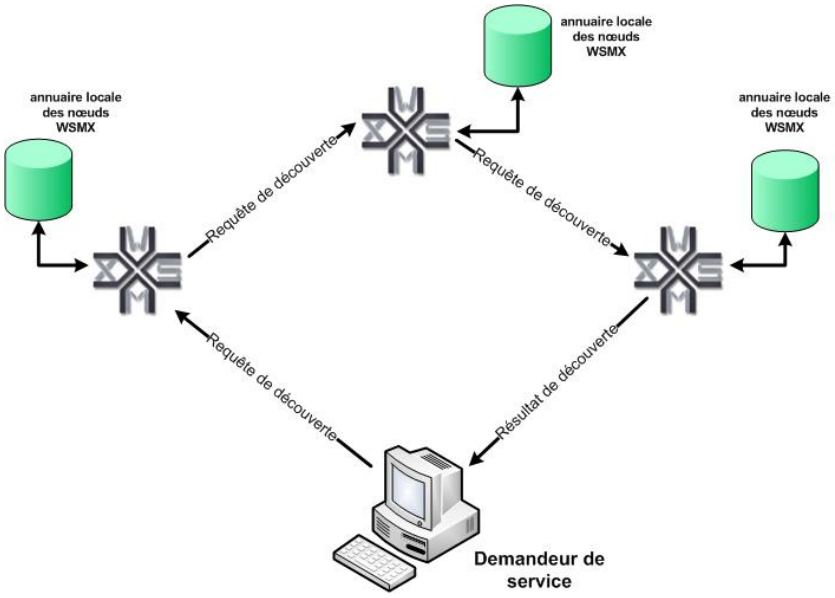


Fig. 4.4 Une approche P2P pure pour la découverte distribuée dans WSMX.

2.2.2. Approche P2P Hybride

Cette approche est une combinaison des deux approches centralisée et pure. On peut distinguer deux types de pairs dans cette approche. Les nœuds fils (*Child Nodes*) et les super nœuds (*super Nodes*).

Un nœud fils est connecté et communique avec un seul super nœud. D'autre part, les super nœuds, en plus de leur rôle qui est le même avec celui des nœuds fils, agissent comme des annuaires de leurs semblables.

Les super nœuds gèrent deux listes : une ayant des informations sur les nœuds fils qui leurs sont connectés, et l'autre enregistre des informations sur les super nœuds voisins. La figure 4.5 montre l'architecture de cette approche.

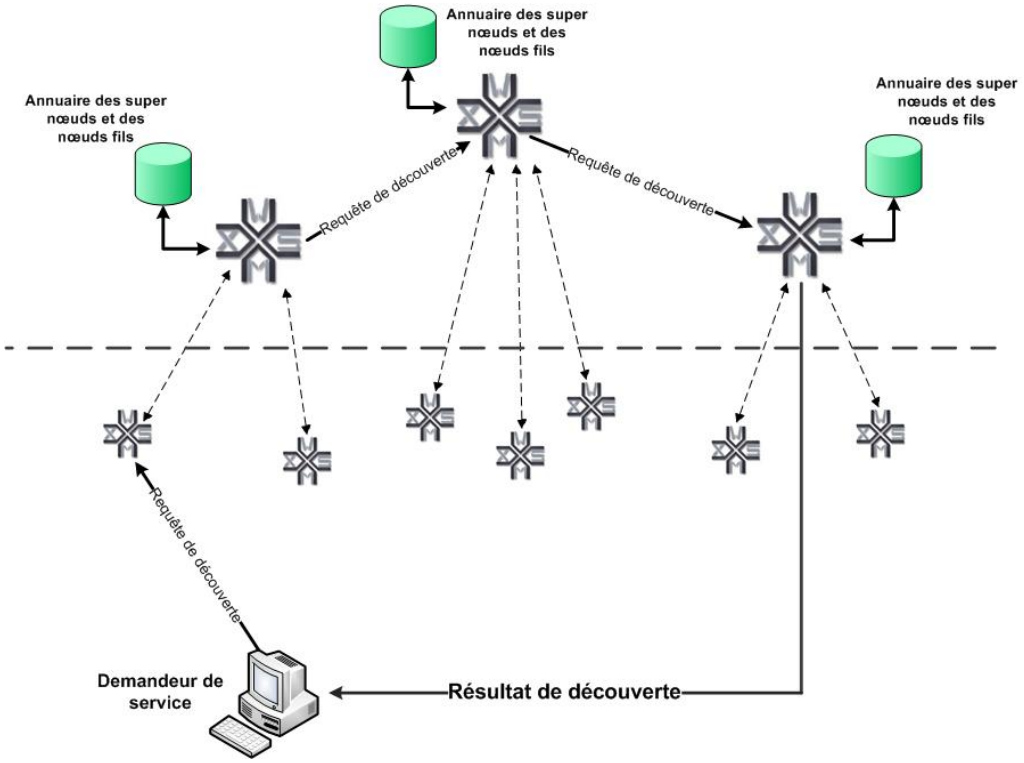


Fig. 4.5 Une approche P2P hybride pour la découverte distribuée dans WSMX.

Un nœud fil qui vient de recevoir une requête va l'exécuter localement, s'il n'y a pas de résultats positifs, il va transmettre la requête à son super nœud. Si le super nœud n'a pas de résultats positifs, donc il va consulter ces nœuds fils pour un éventuel service qui correspond aux descriptions du but. Si tous les nœuds fils échouent, la requête sera transmise au prochain super nœud où ce processus sera répété jusqu'à ce que tous les nœuds seront consultés ou un résultat positif sera donné. Il se peut qu'aucun résultat positive ne sera trouvé, dans se cas le résultat qui sera envoyé par le dernier pair consulté sera négatif.

2.3. Approche basée sur le paradigme du « Triple Space »

Le « *Triple Space Computing* » hérite le model de communication basé sur la publication du paradigme « *Space-based Computing*[33] » et l'étend sémantiquement [26]. Le « *Triple Space Computing* » permet aux applications de communiquer en faisant les deux opérations de lecture et d'écriture des triplets RDF dans l'espace partagé.

L'approche de découverte distribuée du *Triple Space Computing* dans WSMX est illustrée dans la figure 4.6.

[33] Le Space Computing a ces racines dans le domaine du traitement parallèle et plus précisément dans le modèle de mémoire proposé pour le langage ***Linda*** développé par D. Gelernter au milieu des années 80 à l'université de Yale.

Dans ce modèle, une application du demandeur de services écrit la requête dans le *Space*. Les nœuds WSMX peuvent lire la requête du *Space* et commencent l'exécution du processus de découverte localement. Si le processus de découverte est fructueux chaque nœud WSMX écrit (répond) dans le *Space*. Dans le cas où le processus de découverte échoue (réponse négative, nulle), les nœuds WSMX n'écrivent rien dans le *Space*.

Dans cette approche les nœuds WSMX lisent et évaluent la requête localement d'une manière indépendante. Cette approche présente plusieurs avantages comme l'autonomie en termes de temps, d'emplacement, de référence et de schéma de données [26].

Cependant, les auteurs de [26] affirment que le « *Triple Space Computing* » c'est une approche qui peut donner quelques choses de plus pour les services web, mais dans aucun cas peut remplacer les approches existantes (basées sur l'échanges directes des messages).

L'inconvénient de cette approche est que les nœuds WSMX doivent connaitre (localiser) le *Space* qui est en cours d'utilisation.

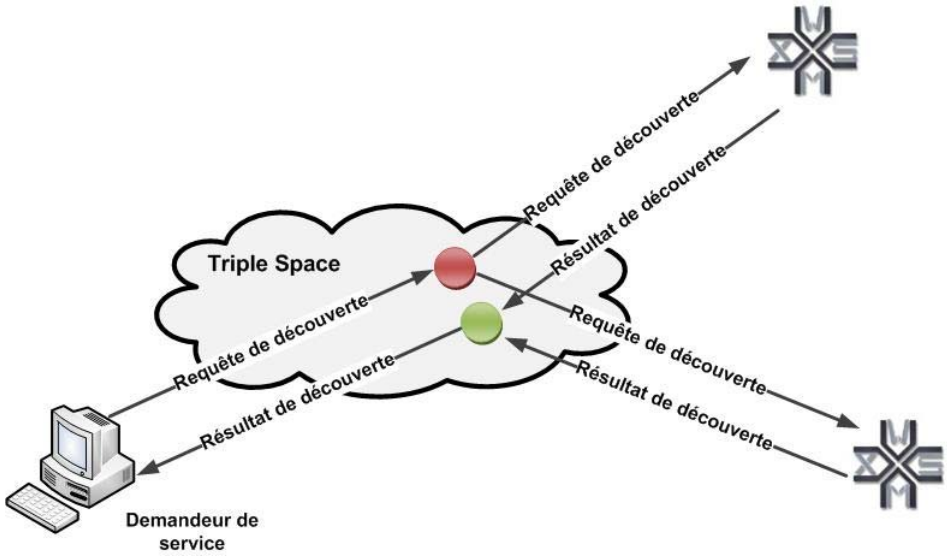


Fig. 4.6 Une approche basée sur le Triple Space pour la découverte distribuée dans WSMX.

3. Quelques travaux connexes

Vu son importance dans la réalisation de la vision des SWS, plusieurs propositions ont vu le jour pour des éventuelles améliorations du processus de découverte des SWS. Cette section a pour objectif de donner un aperçu sur quelques importantes propositions dans ce contexte.

3.1. **METEOR-S WSDI**[34]

Les auteurs de [27] proposent une infrastructure P2P pour les processus de publication et de découverte sémantique des services web. Ils fournissent une organisation pour les annuaires qui se base sur les ontologies au dessus d'une infrastructure P2P.

Cette approche a voulu surmonter le problème du seul point d'échec et le problème de congestion au niveau d'annuaires (UDDI) des services web, qui nécessitaient une centralisation du stockage des descriptions des services web. Mais vu l'évolution du nombre des services web, le nombre des annuaires et la demande sur ces services, la décentralisation est une évidence. Cependant cette décentralisation nécessite une réplication de toutes les descriptions des services web sur l'ensemble des annuaires, ce qui est vu comme fastidieux en termes d'espace de stockage global et du trafic généré.

L'implémentation du MWSDI (METEOR-S Web Service Discovery Infrastructure) permet aux annuaires de s'enregistrer dans un réseau P2P. Les annuaires sont classés dans des catégories d'annuaires, cette classification est faite selon une ontologie de domaine. Ainsi, les descriptions des services web seront, selon la vision des auteurs de cette approche, eux aussi classifiés d'une manière sémantique.

MWSDI fournit quatre types de pairs dans son réseau P2P :

- **Les pairs opérateurs :** actent comme des opérateurs des annuaires des services (permettent d'accéder aux annuaires), comme ils assurent le maintient de la fourniture de l'ontologie des annuaires.
- **Le pair Gateway (GWP) :** actent comme un point d'entrer pour les annuaires pour joindre le MWSDI. Il est responsable de la mise à jour de l'ontologie des annuaires quand un nouvel annuaire joint le réseau. Il est aussi responsable de la propagation de la mise à jour de l'ontologie vers tous les autres pairs. Cependant ce pair n'est pas associé à aucun annuaire.
- **Les pairs auxiliaires :** agissent comme des fournisseurs de l'ontologie des annuaires.
- **Les pairs clients :** sont des membres passagers du réseau P2P et qui sont instanciés seulement pour utiliser le MWSDI.

La figure 4.7 illustre l'implémentation du MWSDI et les différents acteurs qui figurent.

[34] METEOR-S Web Service Discovery Infrastructure

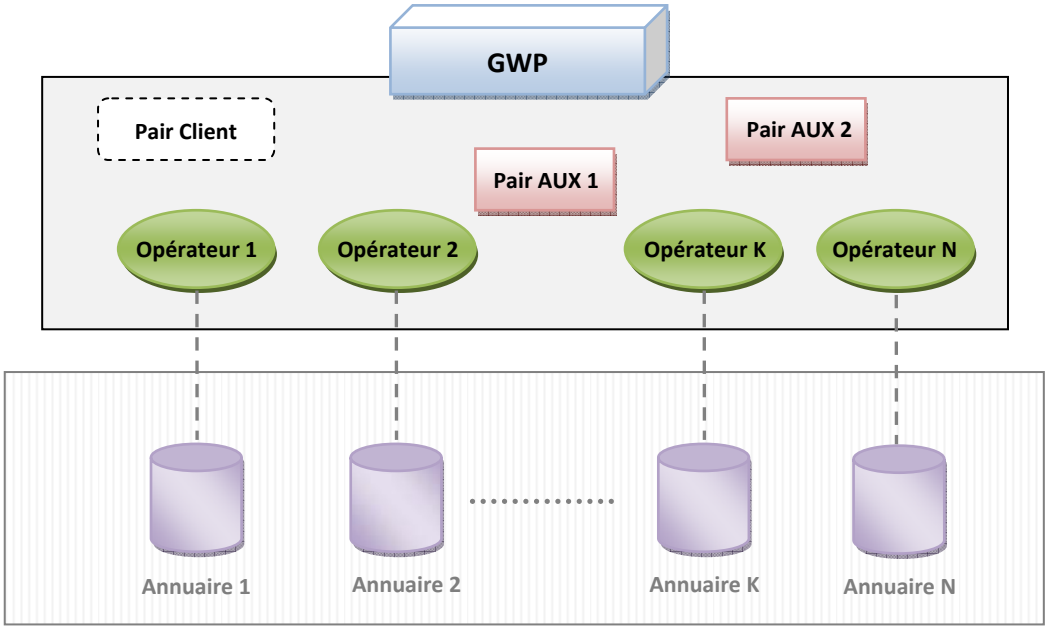


Fig. 4.7 Composants du MWSDI [27].

Cependant, le réseau P2P des annuaires dans METEOR-S souffre du problème du seul point d'échec étant donné qu'il y a un seul point d'entrée prévu (le pair Gateway) pour accéder au réseau P2P des annuaires. En plus la manière dont les services web sont classés et répartis sur l'ensemble des annuaires, semble limiter le potentiel d'une éventuelle découverte sémantique, cela est dû à l'utilisation d'une ontologie pour classer tous les services web. Cette ontologie dont on doit assurer la disponibilité, la maintenance et la mise à jour à chaque entrée ou sortie d'un annuaire du réseau P2P.

3.2. **Découverte basé sur DAML-S** [28]

Les auteurs de [28] ont aussi proposé une approche P2P pour la découverte des services web. Dans cette approche on utilise *Gnutella* comme un protocole P2P et DAML-S comme un langage de description des services.

Gnutella est un réseau P2P pure utilisé principalement pour le partage des fichiers et qui n'utilise aucun annuaire central. Tandis que DAML-S (voir *Chap 1, section 4.2.1*) est un langage de description des services web qui essai de faire un rapprochement entre les infrastructures émergeantes qui se basent sur WSDL, UDDI, BPEL4WS[35] et le web sémantique, en proposant une ontologie de haut niveau pour une éventuelle intégration de la sémantique dans les descriptions des services web.

[35] http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

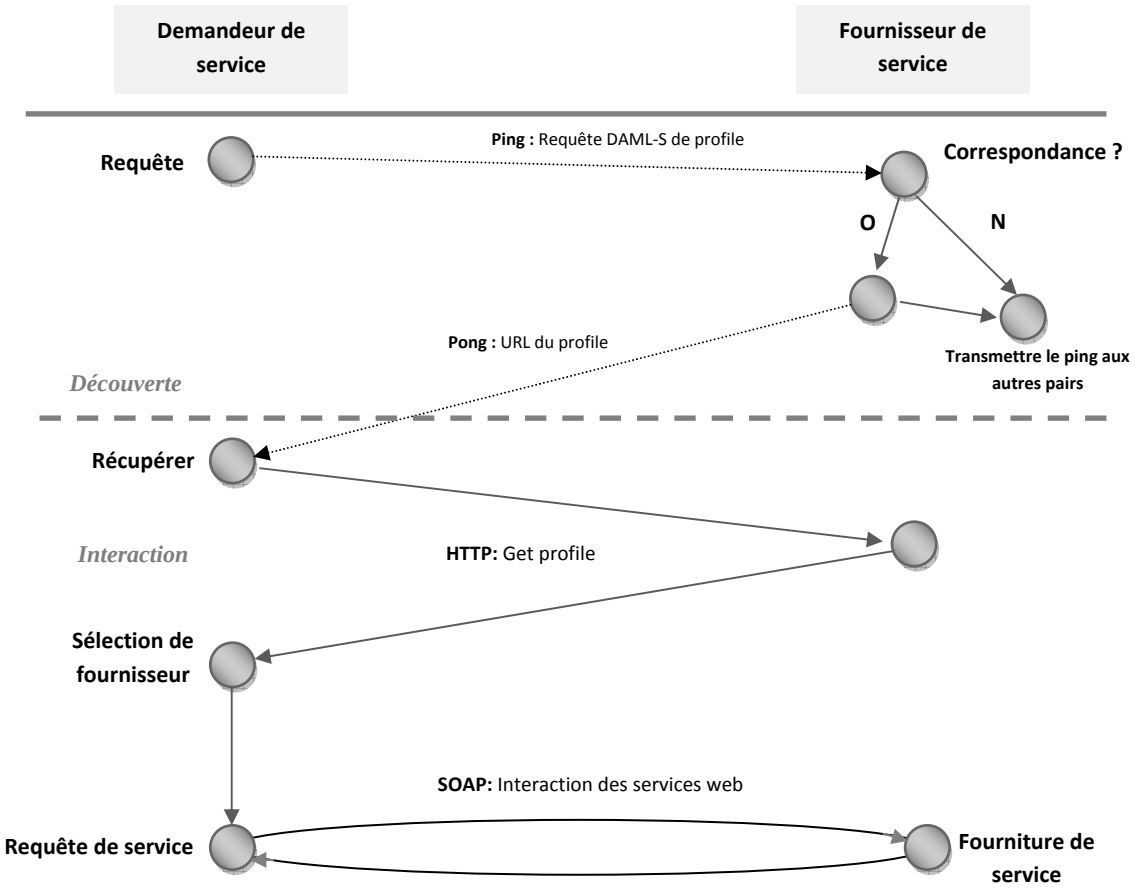


Fig. 4.8 Protocol de découverte et d'interaction des services web [28].

Le travail présenté a pour objectif de montrer comment DAML-S peut faire une recherche dans un réseau P2P basée sur les capacités des services web.

Le processus de découverte, dans cette approche, consiste à envoyer un message de type « *Ping* » contenant la requête, décrite en DAML-S, sur le profile de service. Ce message se fait transmettre d'un pair proche à un autre, et ceci dans les deux cas où le pair destinataire a un résultat positif ou négatif jusqu'à ce que tous les pairs seront consultés.

Dans le cas d'un résultat positif, le pair en cause va répondre avec un message « *Pong* » contenant l'URL du profile. La figure 4.8 illustre les interactions entre les pairs dans cette approche.

Mais cette approche présente certains inconvénients liés aux technologies utilisés. Cette approche hérite les limitations de DAML-S comme model de description, et du protocole *Gnutella* qui est connu pour sa génération d'un trafic énorme.

3.3. **Triple Space Kernel de WSMX**

Les auteurs de WSMX ont proposé une solution basée sur le paradigme du *Triple Space*. Les fonctionnalités de base du *Triple Space* sont réalisé par un module qui se nome « *Triple Space Kernel* ».

Le TSK[36] est développé en utilisant JXTA comme plateforme de communication P2P, comme il utilise un espace d'adressage plat pour identifier tous les intervenants dans les processus de découverte et d'interaction.

Il existe trois types de participants dans cette approche, les serveurs TSK, les clients lourds et les clients légers. Les serveurs TSK et les clients lourds opèrent dans le même espace d'adressage, tandis que les clients légers ont besoin des *proxies* pour accéder aux TSK. Comme variante, les clients légers peuvent accéder aux TSK en utilisant le protocole HTTP, ceci en communiquant avec la plateforme WSMX (avec le TSK intégré) comme un service web. Ces clients peuvent être des simples clients demandeurs de services ou des systèmes utilisant autres infrastructure que WSMX.

Le TSK fournis une interface avec une multitude d'opérations spécifiques pour la création et le paramétrage des nouveaux *Spaces*, la manipulation des *Spaces* déjà existants et la gestion des transactions.

La vision d'une collaboration, utilisant le TSC, des auteurs de WSMX est illustrée dans la figure 4.9.

Cette vision ne néglige pas la possibilité d'une éventuelle collaboration en utilisant un système de communication basé sur le simple échange des messages (SOAP/HTTP), vu leur efficacité et leur simplicité.

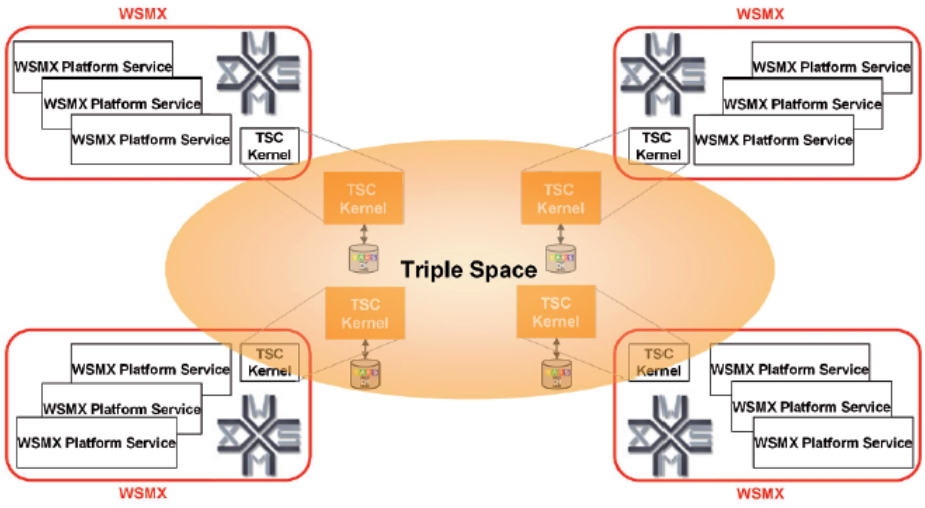


Fig. 4.9 Communication Inter-WSMX pour WSMX utilisant le TSC[37] [26].

[36] *Triple Space Kernel*, http://tsc.sti2.at/.
[37] **TSC :** Triple Space Computing

Malgré qu'aucune application ne peut rapidement parcourir tous le web sémantique pour déterminer s'il existe un triplet pertinent. Comme il n'existe aucune application qui publie un triplet et attend éternellement pour qu'une application le récupère. L'approche du TSC pèche par sa lenteur, puisqu'un demandeur de service doit publier sa requête dans le *space* et attend que les autres participants la consultent, puis traitent la requête localement et répondre dans le *space*.

4. Synthèse

Dans ce chapitre nous avons présenté les différentes approches qui existent pour faire une découverte distribuée des services web sémantiques.

Nous avons vu qu'il existe trois grandes classes d'approches : l'approche centralisée, l'approche P2P et l'approche basée sur le paradigme du Triple *Space Computing*.

L'approche centralisée est la plus simple à réaliser, mais cette approche a l'inconvénient de surcharger le demandeur de service de découvrir tous d'abord les fournisseurs de services existants et de consulter chacun d'eux. Ainsi que la centralisation de l'annuaire des fournisseurs des services est vu comme un maillon faible dans cette approche.

Les approches P2P existantes ont surmonté quelques inconvénients des approches centralisées. Mais ces approches héritent les inconvénients des protocoles utilisés pour les communications entre les pairs et ceux des langages de description utilisés pour les descriptions sémantiques des SWS. Certaines ont même ignoré la nature distribuée des services web sémantiques, en jugeant suffisant de centraliser les descriptions dans des clusters d'annuaires et de répartir les descriptions des services selon une ontologie sur ces annuaires (METEOR-S WSDI).

Concernant la dernière classe d'approches, celle du paradigme du *TSC*, cette approche combine les mécanismes du traitement parallèle issus du domaine du *Space Computing* et les *triplets* de *RDF* du web sémantique pour concevoir un système de collaboration entre intervenants qui partagent un *Space* faisant le rôle d'un tableau dont tous le monde écrivent leurs requêtes et récupèrent les réponses. L'inconvénient de cette approche est la lenteur de son processus de découverte proposé. Mais ça n'empêche pas qu'elle est meilleure des autres approches proposées, puisqu'elle n'utilise pas d'annuaires (centrales ou distribués) pour la localisation des autres nœuds. Donc il n'y a pas un besoin de faire des opérations supplémentaires telles que la mise à jour et la réplication des informations liées aux annuaires.

Une meilleure approche pour une découverte distribuée des services web doit prendre en considération la nature distribuée des services web sémantique ainsi que leur promesse d'une meilleure automatisation de tous les processus liés à leur cycle de vie.

Dans le chapitre suivant notre approche de découverte distribuée des services web sémantiques sera présentée et notre choix de l'architecture adopté et des plateformes utilisées sera expliqué.

Chapitre V

Approche P2P pour une meilleure automatisation du processus de découverte distribuée des SWS pour WSMX

1. Introduction

Après avoir vu les différentes approches proposées, leurs avantages et leurs inconvénients, dans notre tentative de proposition d'une approche pour une découverte distribuée des services web sémantiques, nous avons essayé de fixer quelques objectifs pour notre solution et de prendre en considération les inconvénients des travaux déjà existants.

Nous avons pensé à une solution qui respecte la nature distribuée des services web sémantiques et qui ne fait pas une séparation entre les services et leurs descriptions. En plus, cette solution ne doit pas dépendre d'un système d'annuaires des fournisseurs de services, puisque ces systèmes ont besoin de gestion, (configurations, mises à jour, réplications, …) ce qui créée des tâches supplémentaires à prendre en charge. Cette solution doit aussi contribuée à la vision d'une meilleure automatisation des tâches liées au cycle de vie des SWS (dans notre cas c'est le processus de découverte qui nous intéresse).

2. Approche proposée pour une découverte distribuée des SWS pour WSMX

Des approches décrites précédemment pour le mécanisme de découverte distribuée dans WSMX, nous avons choisi une approche P2P pour notre solution de découverte distribuée. Ceci est parce que :

- Les approches P2P sont des solutions scalable comparées aux approches centralisées.
- Pour plus de deux décennies, les systèmes P2P ont prouvé leur efficacité comme des systèmes distribués pour l'échange d'information.

Pour remédier au problème de non utilisation des annuaires, la première solution qui vient à l'esprit c'est que pour collaborer les nœuds implémentant WSMX n'ont qu'à faire une diffusion de leurs requêtes dans un réseau P2P.

Pour chaque requête de service faire diffuser un message dans un réseau P2P qui ne contient pas seulement les pairs WSMX (qui sont dispersés sur la toile d'internet). Une telle solution sera fastidieuse !

Pour limiter le flot des messages généré par la diffusion, nous avons pensé à une organisation logique des pairs WSMX (qui jouent le rôle des fournisseurs des services, d'annuaires pour leurs services et même parfois des demandeurs des services). L'image la plus proche pour une telle solution est une organisation en groupe des pairs WSMX (voir la figure 5.1). Sans s'oublier qu'il faut associer au sein cette organisation un mécanisme de diffusion sélective des messages de requêtes.

Donc pour faire parvenir les requêtes des services aux pairs WSMX sans avoir recours à un système d'annuaires il faut :

1. Créer un groupe de pairs WSMX.
2. Tous les pairs WSMX doivent figurer dans ce groupe.
3. Avoir un mécanisme de diffusion des messages au sein de ce groupe.

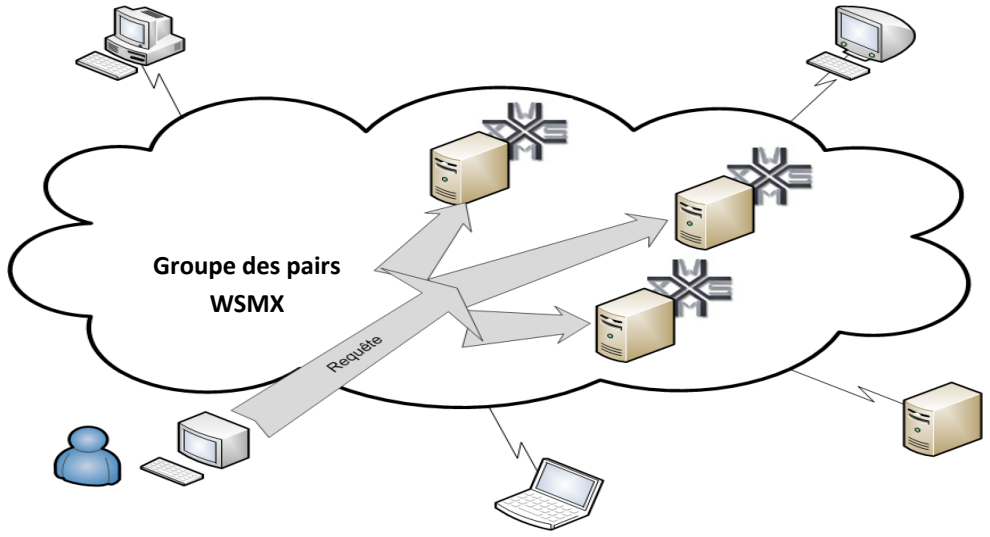


Fig. 5.1 Une organisation en groupe des pairs WSMX.

Notre approche de découverte distribuée pour WSMX, est conçue au dessus d'une infrastructure P2P hybride[38] en utilisant les spécifications de la plateforme JXTA. La raison pour laquelle nous avons choisi la plateforme JXTA est que JXTA fournie une infrastructure qui permet de définir et d'implémenter rapidement des services P2P personnalisés. JXTA fournie aussi un support des mécanismes P2P de base tels que : la gestion des pairs, des groupes de pairs et des communications P2P, tout en laissant l'utilisateur se focaliser seulement sur ce qui concerne l'implémentation de son service.

Malgré que nous avons utilisé une architecture hybride, tous les nœuds WSMX ont les mêmes rôles (comme il n'y a pas d'annuaires des nœuds WSMX). La seule différence entre les pairs est celle induite par la nature des pairs JXTA, ce qui ne fait aucune différence dans notre cas.

Nous avons maintenant des nœuds WSMX éparpillés sur un réseau P2P JXTA. Le problème maintenant est de faire parvenir une requête d'un demandeur de service à ces nœuds WSMX sachant que nous n'allons pas utiliser des annuaires pour localiser ces nœuds.

Des outils JXTA qui sont à notre disposition, nous avons pensé à rassembler ces nœuds WSMX dans un groupe logique (JXTA PeerGroup). Pour collaborer, ces nœuds font diffuser leurs requêtes dans ce groupe.

En diffusant une requête dans un groupe JXTA avec tous les nœuds WSMX dedans, un pair (nœud) WSMX peut ne pas connaitre ses semblables pour coopérer à la résolution d'une requête donnée, cependant, il doit connaitre leur groupe d'appartenance.

[38] En utilisant un bon mécanisme de localisation des ressources basé sur les DHT, l'architecture P2P hybride permet une meilleure exploitation des ressources du réseau même si on diffuse des messages.

La plateforme JXTA n'offre aucun mécanisme qui permet de diffuser un message dans un groupe de pairs. Cependant elle offre des pipes de diffusion (*propagation pipes*[39]) qui permettent de faire la diffusion en faisant parvenir des copies d'un message donné aux pairs implémentant des « *InputPipes* » avec l'annonce (*Advertisement*) du « *propagation pipe* » [29].

Une solution se présente ici, est de créer des Input et des Output Pipes pour tous les pairs d'un groupe avec des annonces de pipes contenant un identificateur de pipe (*Pipe ID*) connu à l'avance (et qui est bien sûr unique). Ainsi on peut communiquer directement sur ces pipes dans un groupe sans avoir à les découvrir.

Développé dans la communauté de JXTA, nous avons ajusté le ***JXTACast***[40] pour acheminer les requêtes aux pairs WSMX. Le *JXTACast* permet de diffuser des messages et des fichiers, en utilisant des pipes de diffusion JXTA, dans un groupe de pairs donné (avec le même principe cité ci-dessus).

Maintenant que nous avons le moyen de faire parvenir le message de requête d'un demandeur de service aux pairs WSMX, ces dernier n'ont qu'à traiter localement cette requête et en utilisant les informations contenues dans le message de requête faire retourner les résultats au demandeur de service.

Ainsi, on peut distinguer trois grandes phases dans notre approche de découverte distribuée des SWS pour WSMX :

1. Une fois bien formulée, la requête d'un service sera transmise aux pairs WSMX en utilisant JxtaCast.

2. Les pairs WSMX traitent localement la requête reçue et préparent les résultats.

3. Les résultats seront encapsulés dans des messages et seront retournés au pair demandeur du service, en créant un pipe direct, avec les informations contenues dans le message de requête (cette fois sans passer par *JxtaCast*).

3. JXTACast

JxtaCast est un protocole de messagerie conçu pour faire la diffusion des messages et des fichiers à tous les membres d'un *PeerGroup* (Groupe de pairs). Les fichiers sont découpés en bloques, encapsuler chaque bloc dans un message et les messages sont en suite transmis en utilisant un *Propagation Pipe*.

[39] C'est le même principe d'un « Walkie-talkie», faire une transmission sur une fréquence radio fixée à l'avance. Donc tous les récepteurs qui sont réglés sur cette fréquence recevront la transmission.

[40] https://jxtacast.dev.java.net/

JxtaCast défini cinq types de messages lui permettant de s'assurer de la bonne transmission d'un fichier donné :

- un message de type ***FILE*** qui contient des informations sur le fichier en cours de transmission (un bloc du fichier par exemple),
- un message de type ***FILE_ACK*** pour acquitter positivement un message de type « *FILE* »,
- un message ***FILE_REQ*** pour demander un bloc spécifique,
- un message ***FILE_REQ_RESP*** pour répondre à une demande de bloc d'un fichier,
- un message de type ***CHAT*** pour envoyer un message texte.

Chacun de ces messages contient plusieurs champs. Le premier champ permet d'identifier le type du message tandis que le reste des champs sont spécifiques à la fonctionnalité du message.

La plateforme JXTA avec sa classe « *net.jxta.endpoint.Message* » permet de créer des messages personnalisés. Une fois créés, ces messages peuvent être transmis en utilisant *JxtaCast*. Ce qui rend *JxtaCast* utilisable pour encapsuler et faire transmettre une requête d'un demandeur de service aux pairs WSMX qui sont tous déjà réunis dans un groupe de pairs.

En intégrant *JxtaCast* dans notre approche pour la conception d'un mécanisme de découverte distribuée pour WSMX, une pile des technologies utilisées comptera quatre couches :

- Une ***couche transport*** qui regroupe les protocoles de base pour le bon acheminement des messages.

- La ***couche JXTA*** qui regroupe tous les mécanismes de base pour les communications P2P (définition des pairs, des groupes, les pipes …). JXTACast fait aussi parti de cette couche et qui interviens seulement dans les processus d'envoi et de réception des requêtes.

- La ***couche WSMX*** qui s'en charge de la gestion des SWS (tout leur cycle de vie) selon la vision WSMO et du traitement des requêtes.

- La ***couche Application***, représente toutes les applications qui intègrent WSMX.

Une couche *sécurité* peut être définie en intégrant des mécanismes de sécurité aux trois couches supérieures.

La figure 5.2 illustre cette pile et les couches qu'elle intègre.

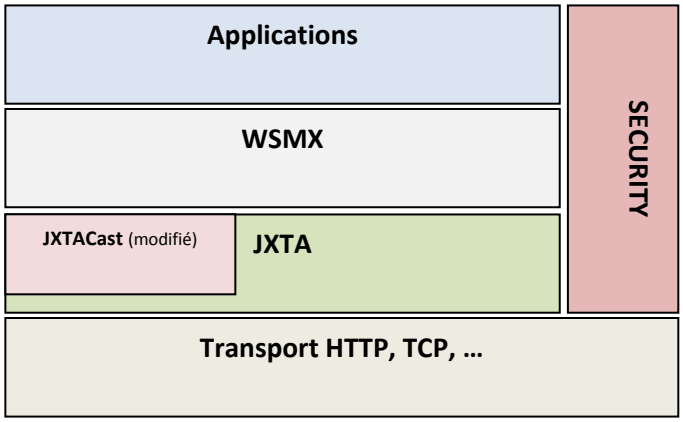


Fig. 5.2 Pile des technologies proposée pour une solution de découverte distribuée des SWS pour WSMX.

4. Scénario d'exécution de la solution proposée pour une découverte distribuée des SWS pour WSMX

Pour une meilleure compréhension de la solution proposée, nous allons considérer la séquence des évènements pour un simple scénario de découverte distribuée (voir la figure 5.3) :

Etape 1 : Un demandeur de services formule sa requête dans le format WSML (*Goal*) et l'encapsule dans un message (*message de requête de services*).

Etape 2 : Le demandeur de services qui est enregistré avec JXTACast et qui fait parti du groupe des pairs WSMX, envoi avant tout la requête en utilisant le mécanisme de diffusion de JXTACast.

Etape 3 : La requête est transmise vers tous les pairs WSMX.

Etape 4 : JXTACast qui est implémenté sur chacun des pairs WSMX reçoit la requête et la transmet à la plateforme WSMX implémentée sur chacun des pairs en utilisant l'interface *WSMXEntryPoints*[41]. Chaque plateforme WSMX sur chacun des pairs qui ont reçu la requête traitent la requête localement.

Etape 5 : le résultat de découverte sur chaque pair est encapsulé dans un message. L'entête de ce message est rempli en utilisant les informations extraites de l'entête du message de la requête. Le message de réponse est renvoyé au pair demandeur du service en utilisant un pipe bidirectionnel JXTA.

[41] La plateforme WSMX donne la possibilité de l'utiliser comme un serveur à part et d'interagir avec elle comme un simple service web dont elle fournie un fichier WSDL (***WSMXEntryPoints***) qui donne toutes les informations nécessaires pour accéder à cette plateforme et les méthodes utilisées possibles.

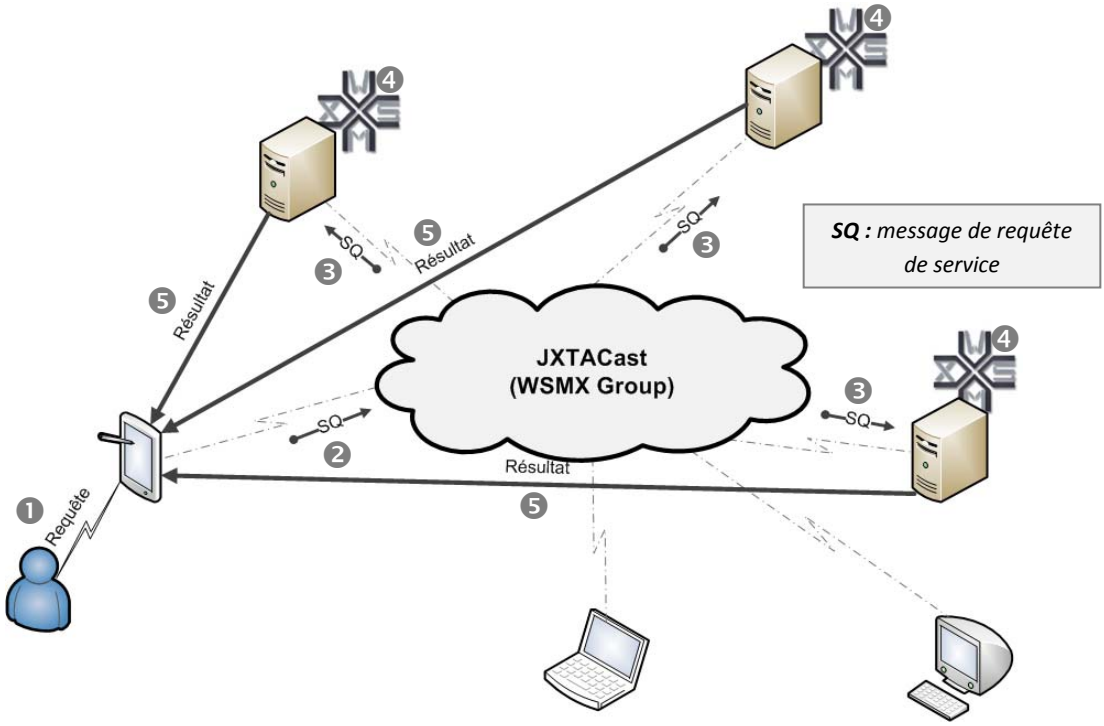


Fig. 5.3 Scénario d'exécution pour l'approche proposée pour une découverte distribuée pour WSMX.

5. Apports de l' approche proposée pour une découverte distribuée des SWS

Notre travail surmonte plusieurs des inconvénients des approches précédentes en utilisant :

- **WSMO**[42] : qui offre un model complet et précis pour la description des services web et des requêtes des utilisateurs.
- **JXTA** : qui offre un réseau P2P générique qui fait abstraction des particularités physique du réseau réel (notion d'*Overlay*). Comme elle offre des mécanismes de base pour la définition et l'identification des pairs, des groupes de pairs et des pipes des communications.

Dans la solution proposée, nous avons utilisé *JXTACast*, un protocole de messagerie JXTA, qui utilise le mécanisme du *PeerGroup* (groupe de pair) et les *Propagation Pipes* (pipes de diffusion) de JXTA pour diffuser les requêtes dans le groupe des pairs WSMX.

[42] WSMX permet de réaliser ceci tout en préservant la nature distribuée des SWS.

L'avantage de cette solution est qu'elle offre une meilleure automatisation et une plus grande transparence concernant l'entrée et la sortie des pairs WSMX. Donc pour être un des pairs WSMX opérationnels, un nouveau pair WSMX doit joindre le groupe des pairs WSMX, comme il doit implémenter notre version modifiée de JXTACast.

Ce qui rend possible la transmission d'une requête donnée dans le groupe des pairs WSMX sans connaitre aucun de ses membres et sans avoir besoin d'autres configurations supplémentaires sur les autres pairs WSMX au cas d'un changement dans le groupe des pairs WSMX (dans le cas où un pair WSMX quitte le groupe ou un autre qui joint ce dernier).

Le mécanisme de diffusion des requêtes dans un groupe de pairs WSMX, permet de ne pas avoir besoin d'annuaires pour localiser les pairs WSMX. Ces annuaires qui posaient des problèmes de gestion (créations, mises à jour et réplications) et de configuration, dans certaines approches, ce qui allège considérablement le processus de découverte et facilite l'implémentation de tels systèmes.

On peut dire que l'organisation en groupes permet de mieux structurer les pairs WSMX en communautés, ce qui permet une meilleure exploitation des ressources du réseau. Ce qui a permit malgré que la diffusion concerne seulement le message de la requête, comme elle concerne uniquement les pairs implémentant WSMX (faisant parti du groupe) et non pas tous les pairs qui figurent dans le réseau.

On peut aussi imaginer un ensemble des partenaires qui forment des communautés se faisant confiance des services développés au sein de ces communautés et qui s'organisent en groupes pour collaborer. La solution proposée permet de contribuer à cette vision et facilite sa réalisation.

6. Synthèse

Dans ce chapitre, nous avons tout d'abord exprimé notre vision d'une approche (de collaboration) pour le processus de découverte distribuée des SWS qui est issue des avantages et des inconvénients des approches décrites dans le chapitre précédent.

Ensuite, nous avons présenté notre approche P2P de découverte distribuée des services web sémantique pour WSMX, dans laquelle nous avons proposé de réunir les pairs WSMX dans un groupe de pairs. Pour coopérer, ces pairs peuvent diffuser leurs requêtes dans ce groupe. Chaque pair WSMX traite les requêtes reçues localement et renvoi la réponse aux pairs qui sont à l'origine de ces requêtes.

Vers la fin du chapitre nous avons soulevé quelques points positifs de l'approche proposée qui se résument en une:

- Minimisation des configurations requises lors des entrées et des sorties des pairs WSMX.
- Meilleure automatisation et une plus grande transparence lors de l'envoi des requêtes grâce à une organisation en groupes.
- Elimination des annuaires utilisés pour la localisation des pairs WSMX et des tâches liées à la gestion de ces annuaires.

Dans le chapitre suivant nous allons donner plus de détailles techniques concernant l'implémentation de notre solution.

Chapitre VI

Implémentation & mise en œuvre

1. Introduction

L'objectif essentiel de ce travail de réalisation est la validation d'un prototype de l'approche proposée pour une meilleure automatisation du processus de découverte distribuée des services web sémantiques pour la plateforme WSMX que nous avons nommé « ***WSMXDiscoCast*** ».

Pour implémenter ce prototype, nous avons essentiellement fait appel à deux plateformes : la plateforme JXTA pour implémenter une infrastructure de communication P2P et la plateforme WSMX pour la définition et la description des services web sémantique (selon l'approche WSMO).

WSMXDiscoCast a été développé dans un environnement Windows XP, en utilisant Java comme langage de programmation (JDK 1.6) avec l'aide de l'environnement de développement *Eclipse*[43], la version 2.4.1 de la plateforme JXTA, la version 0.5 de la plateforme WSMX et la version 2.0 de JxtaCast. Pour la bonne exécution de la plateforme WSMX une installation d'*Apache*[44] et d'*Axis*[45] est nécessaire.

La figure 6.1 illustre comment les interactions entre les deux plateformes WSMX et JXTA sont définies dans l'approche proposée de découverte *WSMXDiscoCast* sur chaque pair WSMX.

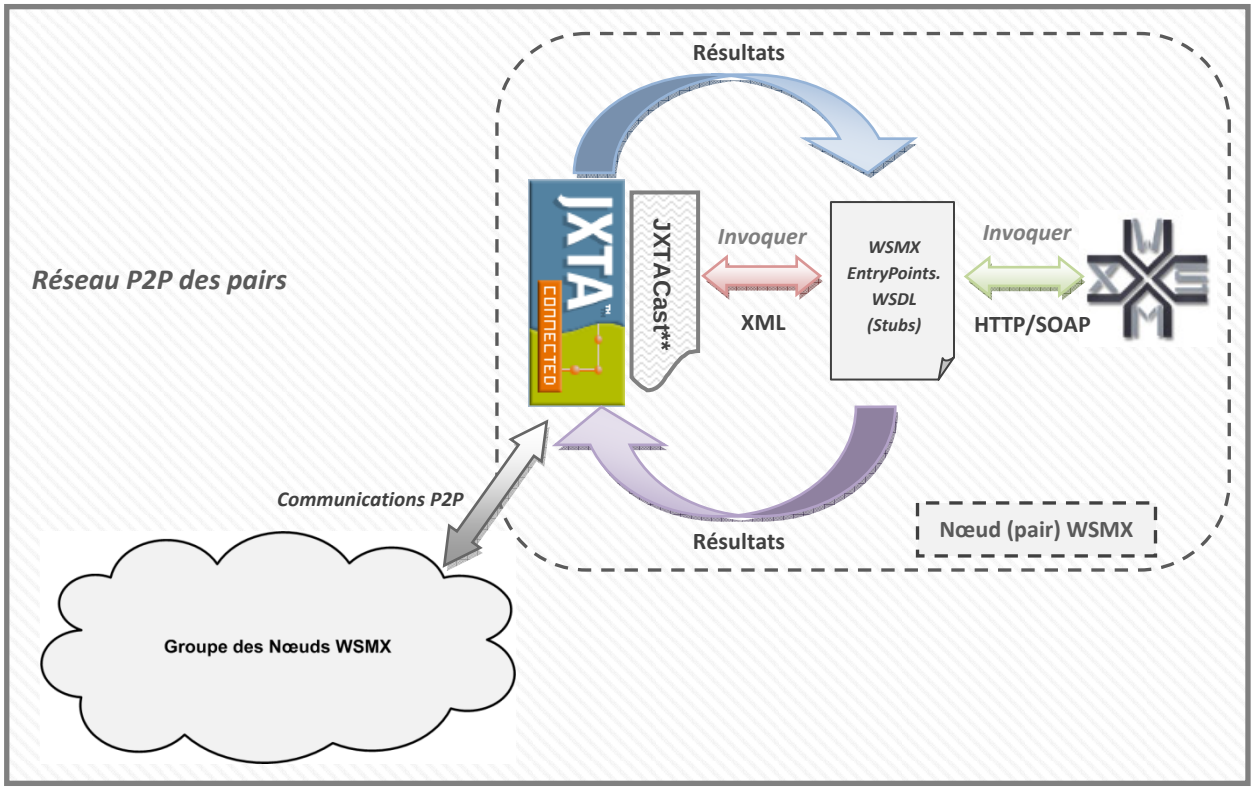


Fig. 6.1 Architecture utilisée pour implémenter WSMXDiscoCast.

[43] http://www.eclipse.org
[44] http://www.apache.org
[45] ws.apache.org/axis/

En examinant la figure 6.1 on peut distinguer quatre importantes briques que nous avons utilisées pour implémenter *WSMXDiscoCast* :

- La plateforme WSMX.
- Le fichier de description WSDL du service *WSMXEntryPoints*.
- La plateforme JXTA.
- JxtaCast.

Dans ce qui suit nous allons présenter des détailles techniques sur ces briques du prototype de l'approche proposée et les interfaces graphiques utilisées pour illustrer son exécution.

2. WSMXEntryPoints

La plateforme WSMX n'offre pas d'API pour invoquer ses modules à partir d'un code, mais elle offre la possibilité de l'utiliser en tant qu'un serveur à part (*Stand-alone server*) en utilisant ses *WSMXEntryPoints*.

Comme son nom l'indique, *WSMXEntryPoints* est une classe Java de WSMX qui offre des points d'entrées (interfaces) pour accéder aux fonctionnalités importantes de la plateforme WSMX.

Cependant pour une éventuelle utilisation sans avoir besoin de Java pour recompiler toute la plateforme, les développeurs de WSMX ont pensé à offrir aux autres développeurs qui souhaitent utiliser cette plateforme la possibilité d'interagir avec un serveur WSMX comme un simple service web.

Le fichier de description WSDL de ce service web est disponible depuis l'URL « ***http://Nom_de_la_machine:8050/axis/services/WSMXEntryPoints?wsdl*** », avec une instance en exécution de la plateforme WSMX sur la machine. Ce fichier permet de lister les méthodes publiques disponibles ainsi qu'une description de leurs paramètres, comme il fourni des détails techniques sur la manière d'accéder au service.

Voici ci-dessous une concise description de quelques importantes méthodes du service *WSMXEntryPoints* :

- **DiscoverWebServices**(String ***wsmlGoal***) : Prend en paramètre un but (désirs du demandeur de services) écrit en WSML et qui retourne un ensemble de services web correspondants.

- **AchieveGoal**(*String* ***wsmlGoal***) : permet de faire les trois processus de découverte, de sélection et d'exécution des services web qui correspondent au but donné en paramètre.

- **InvokeWebService**(*String* ***webService***, *String* ***wsmlGoal***) : permet d'invoquer le service web qui figure dans le premier paramètre. Le deuxième paramètre est un but d'utilisateur avec des instances pour l'exécution du service.
- **Store** (String ***wsmlEntity***) : permet d'enregistrer localement n'importe quelle entité définie en WSML.

Dans la version actuelle de WSMX le *WSMXEntryPoints* compte en toutes 45 méthodes. Les 41 méthodes qui ne sont pas décrites ici permettent essentiellement de donner des informations complémentaires sur les différentes entités enregistrées (services web, ontologies, buts et médiateurs).

Pour utiliser le service *WSMXEntryPoints* depuis notre code Java nous avons utilisé un programme qui vient avec *Axis* qui se nome « *WSDL2Java »*. *WSDL2Java* permet de générer à partir d'un fichier WSDL d'un service des classes Java qu'on peut les utiliser pour accéder directement aux méthodes de ce service.

3. JxtaCast**

Comme précédemment évoqué, nous avons modifié *JxtaCast* pour l'adapter au prototype de l'approche proposée pour une découverte distribuée des SWS pour WSMX.

JxtaCast (notre version modifiée de JxtaCast) est différente de la version original de *JxtaCast* en deux points :

3.1. Types des messages *JxtaCast***

Dans sa version originale, JxtaCast offre la possibilité de transmettre deux types de messages : les messages de type « *FILE »* pour transférer des fichiers et les messages de type « *CHAT »* pour transmettre des messages textes.

Grace à la classe « *net.jxta.endpoint.Message* » de la plateforme JXTA nous avons pu créer des messages personnalisés pour faire encapsuler les requêtes des demandeurs des services. Des informations supplémentaires peuvent être ajoutées telles que : l'identificateur et le nom du pair demandeur du service, le type du message, le type de la requête, champs pour les paramètres de la requête, date d'envoi du message... etc.

La figure 6.2 permet de donner une courte description des champs d'un message de requête dans *JxtaCast***.

Champs	Descriptions
MessageType	Permet de reconnaitre le type du message (***WSMXDiscoCast***)
SenderID	Identificateur unique (ID JXTA) du pair demandeur du service
SenderName	Nom du pair demandeur de service.
mType	Permet d'identifier la méthode à invoquer dans WSMXEntryPoints
Param1	Premier paramètre de la méthode à invoquer
Param2	Deuxième paramètre de la méthode à invoquer
Date	Date et heure d'envoi du message
Caption	Message texte

Fig. 6.2 Champs d'un message WSMXDiscoCast dans JxtaCast**.

3.2. **Invocation de WSMX**

Nous avons aussi ajusté les écouteurs d'évènements de *JxtaCast* dans *JxtaCast*** afin qu'ils puissent invoquer la plateforme WSMX à la réception d'un message de type « *WSMXDiscoCast* », ceci en déchiffrant le message reçu et en invoquant ensuite la méthode appropriée du service *WSMXEntrypoints* local.

A la réception d'un message, JxtaCast** utilise les lignes suivantes pour lire les messages de requêtes des services :

```
String msgType = getMsgString(msg, MESSAGETYPE);
String sender   = getMsgString(msg, SENDERNAME);
String caption  = getMsgString(msg, CAPTION);
String param1  = getMsgString(msg, "param1");
String param2  = getMsgString(msg, "param2");
String sType  = getMsgString(msg, "mType");
```

En suite, en instanciant un objet « *service* » de la classe « *WSMXEntryPointsPortType* » générer par le *WSDL2Java* on peut accéder à toutes les méthodes du service *WSMXEntryPoints* de la plateforme WSMX sur le pair. La méthode à utiliser dépend de la valeur du champ « *mType* ».

```
public WSMXEntryPointsPortType service;
service = new WSMXEntryPointsLocator().getWSMXEntryPointsSOAP12port_http();
```

Par exemple si on veut faire une découverte des services qui correspondent à un but donné, et que le champ « *mType* » a la valeur « 00 », ce qui correspond à la méthode « *achieveGoal* », une invocation de la méthode sera comme suit :

```
service.achieveGoal(param);
```

4. Transfert des résultats

Une fois la découverte est faite localement sur les nœuds WSMX qui ont reçu le message de la requête, les résultats sont maintenant prêts pour être transmis au demandeur de services.

Comme *JxtaCast* ne supporte pas les communications en unicast, nous avons eu besoin de concevoir un mécanisme de communication spécifique d'un pair à un pair pour faire véhiculer les messages des résultats.

Les pipes bidirectionnels (*Bidirectionnel Pipes*) sont un des moyens possibles dans ce cas pour une communication en unicast entre deux pairs. Cependant, la spécification de la plateforme JXTA n'offre pas une gestion de tels moyens de communication. Un développeur doit créer et gérer un pipe bidirectionnel à partir de deux « *Unicast pipes* » et faire paramétrer les écouteurs sur chacune des deux extrémités.

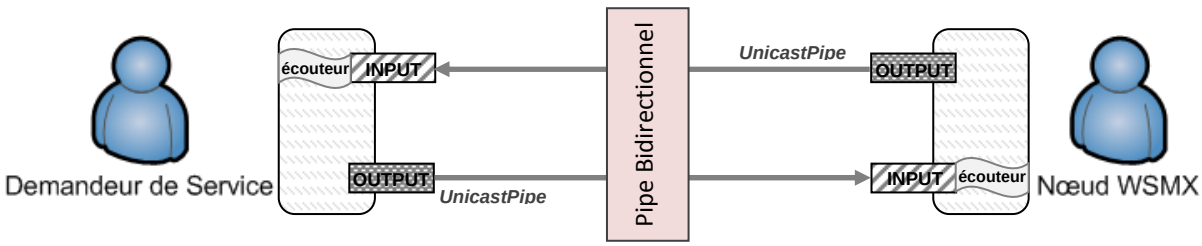


Fig. 6.3 Assemblage d'un pipe bidirectionnel JXTA.

Malgré si on a les deux identificateurs des deux pairs et les écouteurs paramétrés pour le format[46] des messages à recevoir, avant d'établir le pipe bidirectionnel, il faut que les deux pairs définissent une annonce des pipes Unicast utilisés dans le pipe bidirectionnel (une seule annonce suffit pour les deux pipes unicast). C'est comme se mettre d'accord sur une fréquence radio pour émettre et recevoir les communications.

Le fichier de l'annonce du pipe a la forme suivante (voir la figure 6.4):

[46] Ce qui est nécessaire pour déchiffrer le message reçu et distinguer ses différents champs.

```
<!DOCTYPE jxta:PipeAdvertisement>

<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
    <Id>
        urn:jxta:uuid-59616261646162614E50472050325033894 4BCED387C4A2BBD8E9415B78C484104
    </Id>
    <Type>
        JxtaUnicast
    </Type>
    <Name>
        ServerPipe WSMXDiscoCast
    </Name>
</jxta:PipeAdvertisement>
```

Fig. 6.4 Annonce d'un pipe Unidirectionnel JXTA.

Maintenant qu'on a un résultat fourni par la plateforme WSMX et un pipe Bidirectionnel établi entre les deux extrémités, il ne reste qu'à encapsuler le résultat dans un message en utilisant toujours la classe « *net.jxta.endpoint.Message* » et initié l'envoi du message.

La figure 6.5 permet de donner une courte description des champs d'un message qui encapsule les résultats d'une requête de découverte.

Champs	Descriptions
MessageType	Permet de reconnaitre le type du message (***WSMXDiscoCast_Response***)
SenderID	Identificateur unique (ID JXTA) du pair WSMX
SenderName	Nom du pair WSMX
SType	Permet d'identifier la méthode invoquée dans le WSMXEntryPoints
Maches	Nombre des réponses positifs
RESPONSE	Les réponses sur la requête de découverte
Date	Date et heure d'envoi du message
RequesterName	Nom du demandeur du service
Caption	Message Texte

*Fig. 6.5 Champs d'un message WSMXDiscoCast dans JxtaCast**.*

Pour envoyer un message en utilisant ce pipe, il suffit de mettre le message sur le « *Output Endpoint* » du pipe et faire exécuter les lignes suivantes :

```
while (!BiDiPipe.sendMessage(message)){

 try {
            wait(1000);

      } catch (InterruptedException e) {

            e.printStackTrace();
                                    }
```

A la réception du message de réponse, l'écouteur implémenté et paramétré sur l'extrémité « *Input Endpoint* » du pair demandeur de service va désencaspsuler de la même façon décrite dans la section 3.2 et récupérer la réponse qui est la liste des services qui correspondent au but du demandeur de services.

5. Préparation d'un nœud WSMX

Pour qu'un nouveau nœud WSMX coopère dans le processus de découverte en utilisant l'approche proposée, seule sa configuration est requise.

Après avoir installé la plateforme JXTA (configurer le pair et le connecter au réseau) et implémenté notre prototype, il reste à ajuster les paramètres spécifiques du service « *WSMXEntryPoints* » de la plateforme WSMX de ce nouveau pair et à faire joindre ce dernier au groupe des nœuds WSMX.

Pour faire intégrer les paramètres du service « *WSMXEntryPoints* » (pour communiquer avec la plateforme WSMX), il suffit de récupérer le fichier de description de l'URL « ***http://Nom_de_la_machine:8050/axis/services/WSMXEntryPoints?wsdl*** » et de l'enregistrer dans répertoire du prototype. Une fois en possession du fichier de description, une exécution du programme (*Axis*) *WSDL2Java* générera un package « *com.wsmx.client* ». Ce package contient des classes Java utilisées par le JxtaCast** pour accéder à la plateforme WSMX du pair.

Cependant la manière la plus simple pour faire joindre le pair au groupe d'appartenance des nœuds WSMX est de spécifier le nom du groupe dans les lignes suivantes :

```
WSMXGroupID = createPeerGroupID("jxta:uuid-De l'annonce du
groupe des Noeuds WSMX");

WSMXGroup = createPeerGroup(WSMXGroupID, "WSMXPeerGroup",
"Experimentation Group");

joinGroup(WSMXGroup);
```

6. Interfaces graphiques

Le prototype que nous avons développé fourni deux interfaces graphiques pour mieux illustrer l'exécution du processus de découverte distribuée des services web sémantiques avec WSMX.

6.1. JxWSMX Requester

Le GUI *JxWSMX Requester* est décrit ci-dessous dans la figure 6.6 :

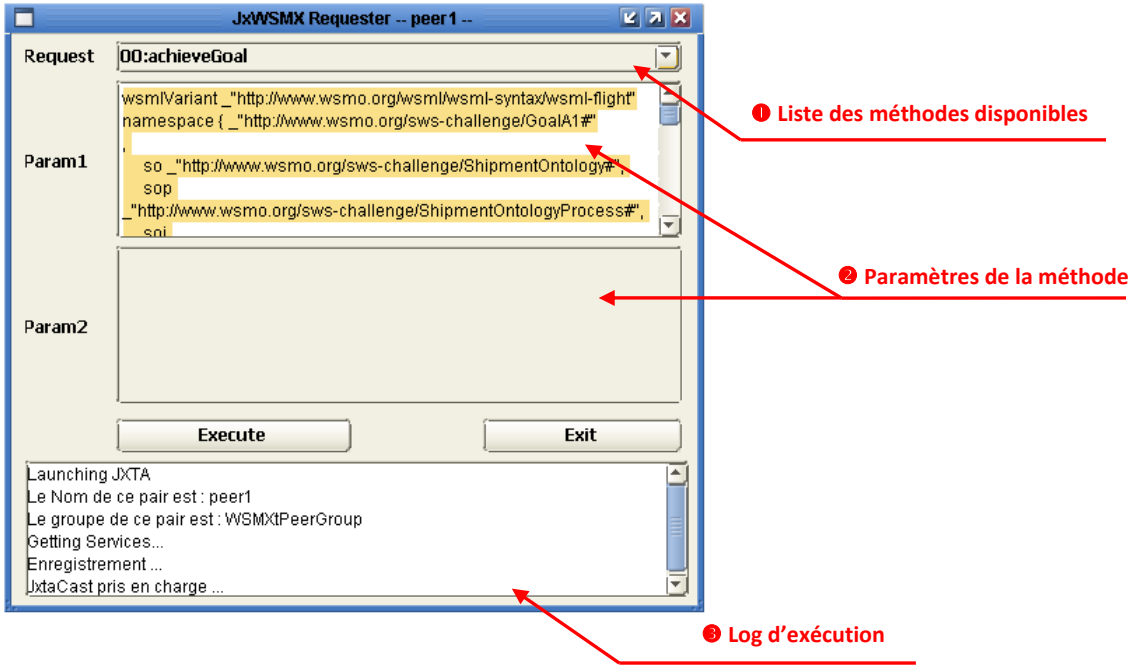


Fig. 6.6 JxWSMX Requester: interface graphique du demandeur des services

Cette interface permet tout d'abord de lancer la plateforme JXTA et le pair demandeur de service. Et puisqu'un pair demandeur de services doit soit :

1. connaitre un pair WSMX pour lequel la requête sera transmise,
2. être membre du groupe des nœuds WSMX et implémente JxtaCast**.

En optant pour le deuxième point, cette interface permet (en background) de faire joindre le pair dans le groupe d'appartenance des nœuds WSMX et de lancer le JxtaCast**.

JxWSMX Requester fournie une liste des méthodes possibles pour invoquer les plateformes WSMX distantes ❶ (Voir la figure 6.6). Deux zones de textes pour introduire les paramètres des méthodes ❷ (Voir la figure 6.6) avec un bouton « *Execute* » pour valider l'invocation de la méthode sélectionnée. Une zone de notification ❸ « log d'exécution », pour donner une idée sur le déroulement de l'exécution.

6.2. JxWSMX Response Monitor

Une fois que le JxtaCast** sur le pair demandeur de services reçoit une réponse, ce dernier créé une interface « *JxWSMX Response Monitor* ». Cette interface a pour seul but d'afficher les résultats et des informations supplémentaires contenues dans le message de réponse tels que le nom du nœud qui a répondu, le nombre des résultats positifs et le type du message pour lequel cette réponse est envoyée.

Cette interface est créée pour illustrer l'opération de réception des messages de réponse, mais ceci n'est pas nécessaire dans un vrai scénario de découverte. Dans ce cas une structure de données est créée pour stocker les messages de réponses reçus.

Le GUI *JxWSMX Requester* est présenté ci-dessous dans la figure 6.7 :

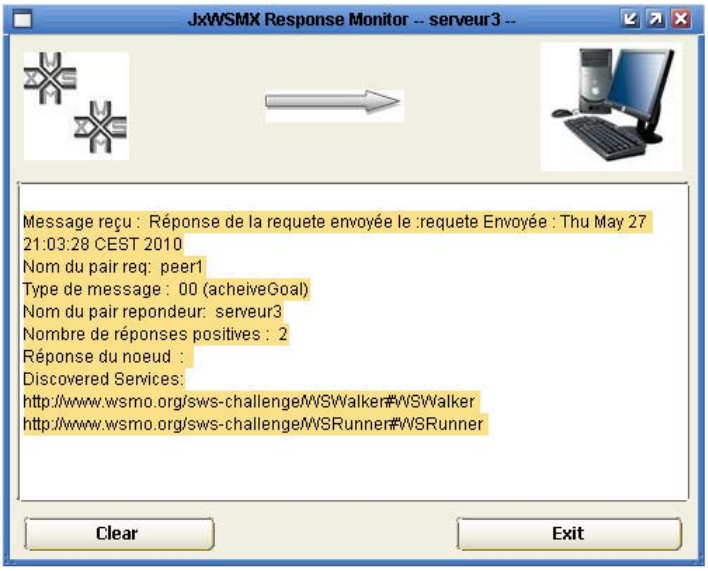


Fig. 6.7 JxWSMX Response Monitor

7. Synthèse

Dans ce chapitre nous avons présenté l'implémentation et le déploiement un prototype pour l'approche proposée pour une meilleure automatisation du processus de découverte distribuée des services web sémantique pour la plateforme WSMX.

Nous avons tout d'abord donné une description sur l'architecture adoptée pour l'implémentation du prototype. Ensuite, nous avons dévoilé quelques détails techniques concernant *JxtaCast*** qui se résument aux changements faits sur la forme des messages de *JxtaCast* et l'appareillage réalisé sur ce dernier afin qu'il puisse communiquer avec la plateforme WSMX en utilisant le service *WSMXEntrypoints*.

Après cela, nous avons montré comment nous avons manipulé les pipes bidirectionnels pour acheminer les messages des résultats vers le demandeur de services. Comme nous avons indiqué les configurations requises pour qu'un nouveau nœud WSMX puisse coopérer dans le processus de découverte distribuée en utilisant notre prototype. Vers la fin du chapitre, quelques prises d'écran ont été effectuées pour mieux illustrer les différentes fonctionnalités de notre prototype.

En examinant de près la manière dont nous avons implémenté notre prototype (*WSMXDiscoCast*), un pair WSMX, en disposition de ce prototype, n'a qu'à fournir (uniquement) un package correspondant à son service *WSMXEntrypoints* pour qu'il coopère dans le processus de découverte distribuée des SWS selon l'approche proposée.

Conclusion et perspectives

1. Conclusion

Dans ce mémoire nous avons présenté une approche P2P pour une meilleure automatisation du processus de découverte distribuée des services web sémantiques pour la plateforme WSMX. Nous avons justifié notre choix de la plateforme WSMX comme notre banc d'essai en trois points. (1) WSMX est une implémentation de référence de l'approche WSMO qui offre un model complet et précis pour la description des services et des requêtes des utilisateurs. (2) WSMX couvre tout le cycle de vie d'un service, sert d'un annuaire pour ses propres services, et coopère avec ces semblables comme des systèmes distribués. (3) la plateforme WSMX offre la possibilité d'interagir avec d'autres systèmes qui ne sont pas des nœuds WSMX grâce à son module « *WSMX Adapters* », ce qui nous permet de simuler une coopération des systèmes non-WSMX en utilisant des nœuds WSMX.

Nous avons aussi justifié notre choix d'une approche P2P par le fait que les systèmes P2P ont prouvé leur efficacité comme des systèmes distribués pour l'échange d'informations et comme des solutions scalable, pour plus de deux décennies.

L'étude des différentes solutions existantes pour une découverte distribuée des services web sémantiques, nous a permis de proposer une approche que dans sa conception, nous avons essayé de prendre en compte les avantages et les limites des approches déjà proposées.

L'approche proposée est donc basée sur une infrastructure P2P hybride (semi-décentralisée) en utilisant les spécifications de la plateforme JXTA. Comme nous avons besoin qu'une requête soit diffusée dans un groupe de pairs avec tous les nœuds WSMX dedans, nous avons fait ajuster JxtaCast, un protocole d'échange des messages développé dans la communauté JXTA, pour acheminer les messages des requêtes vers les nœuds WSMX. Ces derniers peuvent répondre en établissant un pipe de communication directe avec le demandeur de services.

Nous avons pu soulever quelques points positifs de l'approche proposée :

- La non utilisation des annuaires pour la localisation des nœuds WSMX, ce qui allège considérablement le processus de découverte distribuée.
- Un demandeur de service n'a pas à connaitre un ou plusieurs nœuds WSMX pour adresser sa requête, mais il lui suffit seulement de connaitre leur groupe d'appartenance, ce qui offre un meilleur anonymat dans le processus de découverte.

- L’organisation en groupe permet une meilleure structuration des nœuds WSMX, comme elle offre une meilleure transparence lors de l’entrée et la sortie des nœuds WSMX.

- Minimiser les configurations qui se limitent seulement (dans notre cas) à la fourniture des informations nécessaires pour l’accès à la plateforme WSMX sur le nouveau nœud qui arrive.

2. Perspectives

Malgré que notre prototype donne des résultats positifs, nous voyons plus juste de faire une évaluation appropriée de la solution proposée dans un réseau de nœuds WSMX plus grand. Une telle évaluation permet de faire des testes mesurant les performances des nœuds WSMX avant et après l’utilisation de notre prototype, comme elle permet de donner des statistiques concernant le temps de réponse et la bande passante utilisée.

Nous envisageons aussi de faire intégrer des mécanismes de sécurité dans le prototype développé, en faisant appel aux mécanismes de sécurité de JXTA : Groupes sécurisés (*secure Groupes*) et pipes sécurisés (*secure pipes*) [30], pour sécuriser toutes les communications.

L’organisation en groupes peut être aussi exploitée pour créer des communautés de partenaires qui se font confiance et souhaitent coopérer entre partenaires.

Pour terminer, une éventuelle extension de l’approche proposée dans ce mémoire avec l’usage des folksonomies [33],[34] donne la possibilité de mieux structurer les nœuds WSMX en groupes, ce qui permet de mieux cibler les requêtes tout en réduisant l’espace de leur diffusion (*pour quels groupes (communautés) de pairs WSMX doit-on s’adresser pour répondre le mieux à une requête donnée ?*).

Une étude plus poussée dans cette direction va combiner des techniques multidisciplinaire de domaines tel que : les réseaux sociaux, la recherche d’information et le traitement du langage naturelle, afin d’élaborer un mécanisme permettant une création dynamique des groupes de pairs WSMX en utilisant une sorte d’intelligence collective émergeante des interactions sociales entre les fournisseurs et les demandeurs de services [35].

Enfin, une comparaison des performances de cette extension avec celle de notre prototype et ceux des propositions existantes peut contribuer à une étude intéressante.

Bibliographie

[1]: *Berners-Lee, T., Hendler, J., & Lassila, O.”* The Semantic Web”. *Scientific American, pp.34-43. (2001,May). disponible depuis: http://www.ryerson.ca/~dgrimsha/courses/cps720_02/resources/. [dernier accès : 31/05/2010].*

[2]: *Grau, B.C.*, “A possible simplification of the Semantic Web architecture”, *International World Wide Web Conference, Proceedings of the 13th international conference on World Wide Web, New York 2004. pp.704-713. ISBN: 1-58113-844-X*

[3]: *Jorge Cardoso,* “Semantic Web Services: Theory, Tools, and Applications”. *University of Madeira, Portugal, Information Science reference, Hershey - New York, 2007.pp 1-20, 134-152, 191-210, 240-312; ISBN 978-1-59904-047-9.*

[4]: *Mike Uschold, Robert Jasper*, “A Framework for Understanding and Classifying Ontology Applications”. *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-SolvingMethods (KRR5) Stockholm, Sweden, August 2, 1999.*

[5]: *Rudi Studer, Stephan Grimm, Andreas Abeck* , “SemanticWeb Services Concepts, Technologies, and Applications”. *Verlag Berlin Heidelberg 2007, ISBN-978-3-540-70893-3.*

[6]: *N. Guarino.* “Semantic Matching : Formal Ontological Distinctions for Information Organization,Extraction, and Integration”. *In M.T. Pazienza, editor, Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, number 1299 in LNCS, pages 139–170. Springer-Verlag, 1997.*

[7]: *Ethan Cerami,* “Web Services Essentials, Distributed Applications with XML-RPC, SOAP, UDDI & WSDL”. *O'Reilly First Edition February 2002, ISBN: 0-596-00224-6.*

[8]: *Ramesh Nagappan, Robert Skoczylas, Rima Patel Sriganesh,* “Developing Java™ Web Services Architecting and Developing Secure Web Services Using Java”, *Wiley Publishing Inc 2003, ISBN 0-471-23640-3*

[9]: *Patrick Kellert et Farouk Toumani*, “Les Web services sémantiques”. *Revue hors série 2004.disponible depuis: http://www.revue-i3.org/hors_serie/annee2004/revue_i3_hs2004_01_07.pdf. [dernier accès : 31/05/2010].*

[10]: *Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, John Domingue,* “Enabling SemanticWeb Services : TheWeb Service Modeling Ontology”. *Springer- Verlag Berlin Heidelberg 2007, ISBN-13 978-3-540-34519-0*

[11] : *Dumitru Roman1, Jos de Bruijn, Adrian Mocan, Holger Lausen, John Domingue, Christoph Bussler, and Dieter Fensel,* "WWW: WSMO, WSML, and WSMX in a Nutshell", *ASWC 2006, LNCS 4185, pp. 516–522, Springer-Verlag Berlin Heidelberg 2006.*

[12]: *Michael Stollberg, Omair Shafiq, Darko Anicic,* "TUTORIAL Semantically Enabled Service-Oriented Architectures (SESA)", *Seoul, South Korea, September 2007. disponible depuis: http://www.wsmo.org/TR/d17/sesa-tutorial/SESA-tutorial-printouts-update.pdf. [dernier accès : 11/02/2010].*

[13]: *Dieter Fensel, Mick Kerrigan, Michal Zaremba.* "Imlementing semantic web services: the SESA framework". *Springer-Verlag Berlin Heidelberg (2008), ISBN: 978-3-540-77019-0*

[14]: *Jos de Bruijn, Dieter Fensel, Mick KerriganUwe Keller, Holger Lausen, James Scicluna.* "Modeling Semantic Web Services The Web Service Modeling Language". *Springer-Verlag Berlin Heidelberg(2008) , e-ISBN: 978-3-540-68172-4*

[15]: *Zaremba, M., Haller, A., Zaremba, M., Moran, M.,* "WSMX - Infrastructure for execution of semantic web services". *In the Proceedings of the 2nd International Conference on Web Services (ICWS'04), Springer (2004).*

[16]: *D. Roman, H. Lausen, and U. Keller,* "Web Service Modeling Ontology Standard: WSMO", *Working Draft v02, 2004. disponible depuis: http://www.wsmo.org/2004/d2/v02/20040306/. [dernier accès : 05/02/2010].*

[17]: *Robert Zaremba,* "The Dynamic Discovery of Web Services Using WSMX". *disponible depuis: www.cs.pitt.edu/~chang/231/seminars/S07roz.ppt. [dernier accès : 05/02/2010].*

[18]: *Olivier Soyez,* « Stockage dans les systems pair à pair », *dans sa thèse de doctorat soutenue à l'université de Picardie Jules verne d'Amiens, novembre 2005. Disponible depuis : http://oliviersoyez.free.fr. [dernier accès : 31/05/2010].*

[19] : *Bruno Richard,* « Les technologies pair à pair », Hewlett packard (Laboratoire Grenoble), 2000. *Disponible depuis :* http://rangiroa.essi.fr/cours/systeme2/03-pair-a-pair.ppt. *[dernier accès : 05/02/2010].*

[20]: *Guillaume Doyen,* « Supervision des réseaux et services pair à pair », *dans sa thèse de doctorat soutenue à l'université de Henri Poincaré- Nancy 1 , pp. 29-46, 12 decembre 2005.*

[21] : *M. F. Kaashoek and D. R. Karger Koorde,* "A simple degree-optimal distributed hash table". *In F. Kaashoek and I. Stoica, editors, Proceedings of the 2nd International Workshop on Peer-to-Peer Systems - IPTPS'03, number 2735 in LNCS, pp. 98–107. Springer-Verlag, 2003.*

[22]: *Kurt Vanmechelen,Jan Broeckhove, Gunther Stuer,T om Dhaene.* "Jini and JXTA Based Lightweight Grids", (*Chapitre 13 du livre* "Engineering The Grid: Status and Perspective", *pp 4-11 by American Scientific Publishers,2006,ISBN: 1-58883-038-1. Disponible depuis : www.comp.ua.ac.be/publications/files/Jini_JXTA_Chapter13.pdf [dernier accès : 31/05/2010]*.

[23]: *Juan Carlos Soto,* "Project JXTA: Project JXTA: An Open P2P Applications Platform An Open P2P Applications Platform Introduction and Update". *Disponible depuis : http://www.cs.rpi.edu/academics/courses/fall02/netprog/notes/jxta/jxta.pdf. [dernier accès : 31/05/2010]*.

[24]: JXTA Java™Standard Edition v2.5: Programmers GuideSeptember 10th, 2007. *Disponible depuis : https://jxta-guide.dev.java.net/source/browse/*checkout*/jxta-guide/trunk/src/guide_v2.5/JXSE_ProgGuide_v2.5.pdf [dernier accès : 03/05/2010]*.

[25] : *Ioan Toma, Brahmananda Sapkota, James Scicluna, Juan Miguel Gomez, Dumitru Roman, and Dieter Fensel*, "A P2P Discovery mechanism for Web Service Execution Environment", *Digital Enterprise Research Institute (DERI), Galway, Ireland and Innsbruck, Austria 2004.*

[26]: *Dieter Fensel, Mick Kerrigan, Michal Zaremba* "Imlementing semantic web services: the SESA framework", *pp236. Springer-Verlag Berlin Heidelberg (2008) , ISBN: 978-3-540-77019-0.*.

[27]: *K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar and J. Miller,* "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services", *Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers. (pre-publication version)*

[28]: *Massimo Paolucci, Katia P. Sycara, Takuya Nishimura, and Naveen Srinivasan.* "Using DAML-S for P2P Discovery". *In ICWS(2003). pp 203–207.*

[29]: *Brendon J. Wilson,* "JXTA : p2p jxta reference implementation", (*2002) by New Riders Publishing. ISBN 0-73571-234-4*

[30]: *Joseph D. Gradecki,* "Mastring JXTA: Building Java Peer-to-Peer Applications". *pp15-38. Wiley Publishing(2003)., Inc., ISBN: 0-471-25084-8.*

[31]: *Borst, Willem.,* "Construction of Engineering Ontologies for Knowledge Sharing and Reuse"*: Ph.D. Dissertation, University of Twente.(1997), ISSN: 1381-3617 (CTIT Ph. D-series No. 97-14).*

[32]: *Mike USCHOLD et Michae GRÜNINGER.,* "Ontologies: principles, methods and applications, (1996). *Knowledge Engineering Review, 11(2):93-155.*

[33]: *Evelyne BROUDOUX, "* Folksonomies et indexation. collaborative. Rôle des réseaux sociaux dans la fabrique de l'information", *2006, Disponible depuis : http://www.docforum.tm.fr/documents/23&24nov06SavResPar06InterBroudouxE.pdf [dernier accès : 31/05/2010].*

[34]: *Olivier ERTZSCHEID, "* Indexation sociale et folksonomies : le monde comme catalogue", *mai 2008, Disponible depuis : http://www.slideshare.net/olivier/oe-abes-mai2008 [dernier accès : 31/05/2010].*

[35]: *Pantazoglou, M. Tsalgatidou, A. ,"* A P2P Platform for Socially Intelligent Web Service Publication and Discovery", *The Third International Multi-Conference on Computing in the Global Information Technology (ICCGI '08), 2008, IEEE. pp 271-276. ISBN: 978-0-7695-3275-2. Disponible depuis : http://cgi.di.uoa.gr/~michaelp/papers/iccgi08.pdf [dernier accès : 31/05/2010].*