VEILLE TECHNOLOGIQUE

TAXINOMIE DES MIDDLEWARES

LIVRE BLANC

SOGETI HIGH TECH

BECK Valentin – MAZOYER David – NONGLATON Thomas – PATRICE Keenan – YUAN Ye

BU:

Sogeti High Tech - Montbonnot

Diffusion:

Organisme	Service	Nom

Résumé :

Visas :

Approuvé par	Date	Nom	Signature
(fonction)			
(fonction)			

Autorisé par	Date	Nom	Signature
(fonction)			
(fonction)			

Historiques des modifications :

Éd.	Auteur	Date	§	Nature de l'évolution
1.0		25/10/04		Création

SOGETI
High Tech

TABLE DES MATIERES

1. INTRODUCTION 11

1.1. BUT 11
1.2. DEMARCHE 11

2. LE CONTEXTE 12

2.1. MODELE OSI 12
2.2. DEFINITION D'UN PROTOCOLE 12
2.3. DEFINITION D'UN MIDDLEWARE 12
2.4. LES ACTEURS 13

3. LICENCES ET NORMES 14

3.1. LGPL 14
3.2. CPAL 14
3.3. GPL 14
3.4. IHM 14
3.5. HL7 14
3.6. DICOM 14

4. MIDDLEWARES PAR DOMAINES METIERS 15

4.1. DOMOTIQUE 15
4.2. TELECOMMUNICATION 15
4.3. SANTE 15
4.4. ENERGIE 15
4.5. AUTOMOBILE 16

5. LES PROTOCOLES 17

5.1. X10 17
5.2. KNX 18
5.3. UPNP 18
5.4. DPWS 19
5.5. LONWORKS - LONTALK 20
5.6. TCP (TRANSMISSION CONTROL PROTOCOL) 20
..... 21
5.7. ETHERNET 21
5.8. KERMIT 21
5.9. AMQP (ADVANCED MESSAGE QUEUE PROTOCOL) 22
5.10. STOMP 22
5.11. HTTP (HYPERTEXT TRANSFER PROTOCOL) 23

6. LES FICHES MIDDLEWARES 24

6.1. WOSH 24
6.1.1. Historique 24
6.1.2. Version actuelle 24
6.1.3. Date de sortie 24
6.1.4. Licence 24
6.1.5. Protocoles 25
6.1.6. Fonctionnalités 25
6.1.7. Domaine métier principal 25
6.1.8. Couche OSI 25
6.1.9. Architecture interne 25

6.1.10. Spécifications *25*
6.1.11. Site officiel *25*
6.2. ENTIMID 26
6.2.1. Historique *26*
6.2.2. Version actuelle *26*
6.2.3. Protocoles *26*
6.2.4. Fonctionnalités *26*
6.2.5. Domaine métier principal *26*
6.2.6. Couche OSI *26*
6.2.7. Architecture interne *27*
6.2.8. Implémentations *27*
6.2.9. Spécifications *28*
6.3. SM4ALL 29
6.3.1. Historique *29*
6.3.2. Version actuelle *29*
6.3.3. Date de sortie *29*
6.3.4. Licence *29*
6.3.5. Protocoles *29*
6.3.6. Fonctionnalités *29*
6.3.7. Domaines métiers principaux *29*
6.3.8. Couche OSI *30*
6.3.9. Architecture interne *30*
6.3.10. Implémentations *30*
6.3.11. Site officiel *30*
6.4. SIRILOG-RADIO 31
6.4.1. Historique *31*
6.4.2. Version actuelle *31*
6.4.3. Licence *31*
6.4.4. Protocoles *31*
6.4.5. Fonctionnalités *31*
6.4.6. Domaine métier principal *32*
6.4.7. Couche OSI *32*
6.4.8. Architecture interne *32*
6.4.9. Implémentations *32*
6.4.10. Site officiel *32*
6.5. CENTRICITY EE 33
6.5.1. Historique *33*
6.5.2. Version actuelle *33*
6.5.3. Licence *33*
6.5.4. Protocoles *33*
6.5.5. Fonctionnalités *33*
6.5.6. Domaine métier principal *33*
6.5.7. Couche OSI *33*
6.5.8. Architecture interne *34*
6.5.9. Spécifications *34*
6.5.10. Site officiel *35*
6.6. GAIA 36
6.6.1. Historique *36*
6.6.2. Version actuelle *36*
6.6.3. Protocoles *36*
6.6.4. Fonctionnalités *36*
6.6.5. Domaines métiers principaux *36*
6.6.6. Couche OSI *36*
6.6.7. Architecture interne *36*
6.6.8. Spécifications *38*

6.6.9. *Site officiel* 38
6.7. ExORB 39
6.7.1. *Protocoles* 39
6.7.2. *Fonctionnalités* 39
6.7.3. *Domaine métier principal* 39
6.7.4. *Couche OSI* 39
6.7.5. *Architecture interne* 39
6.7.6. *Spécifications* 40
6.8. WSAMI 41
6.8.1. *Historique* 41
6.8.2. *Date de sortie* 41
6.8.3. *Version actuelle* 41
6.8.4. *Protocoles* 41
6.8.5. *Fonctionnalités* 41
6.8.6. *Domaines métiers principaux* 41
6.8.7. *Couche OSI* 41
6.8.8. *Architecture interne* 42
6.8.9. *Spécifications* 42
6.8.10. *Site officiel* 43
6.9. CORTEX 44
6.9.1. *Historique* 44
6.9.2. *Date de sortie* 44
6.9.3. *Protocoles* 44
6.9.4. *Fonctionnalités* 44
6.9.5. *Domaines métiers principaux* 44
6.9.6. *Couche OSI* 44
6.9.7. *Architecture interne* 44
6.9.8. *Spécifications* 45
6.9.9. *Site officiel* 45
6.10. LIME 46
6.10.1. *Historique* 46
6.10.2. *Version actuelle* 46
6.10.3. *Date de sortie* 46
6.10.4. *Licence* 46
6.10.5. *Protocole* 46
6.10.6. *Fonctionnalités* 46
6.10.7. *Domaines métiers principaux* 46
6.10.8. *Couche OSI* 46
6.10.9. *Architecture interne* 47
6.10.10. *Implémentations* 47
6.10.11. *Spécifications* 47
6.10.12. *Site officiel* 47
6.11. REDS 48
6.11.1. *Historique* 48
6.11.2. *Version actuelle* 48
6.11.3. *Licence* 48
6.11.4. *Protocoles* 48
6.11.5. *Fonctionnalités* 48
6.11.6. *Domaines métiers principaux* 48
6.11.7. *Couche OSI* 48
6.11.8. *Architecture interne* 48
6.11.9. *Implémentation* 49
6.11.10. *Site officiel* 49
6.12. JacORB 50
6.12.1. *Historique* 50

6.12.2. *Version actuelle* 50
6.12.3. *Date de sortie* 50
6.12.4. *Licence* 50
6.12.5. *Protocoles* 50
6.12.6. *Fonctionnalités* 50
6.12.7. *Domaines métiers principaux* 50
6.12.8. *Couche OSI* 51
6.12.9. *Architecture interne* 51
6.12.10. *Spécifications* 51
6.12.11. *Site officiel* 51
6.13. TIBCO RENDEZVOUS 52
6.13.1. *Historique* 52
6.13.2. *Version actuelle* 52
6.13.3. *Date de sortie* 52
6.13.4. *Licence* 52
6.13.5. *Protocoles* 52
6.13.6. *Fonctionnalités* 52
6.13.7. *Domaines métiers principaux* 52
6.13.8. *Couche OSI* 52
6.13.9. *Architecture interne* 53
6.13.10. *Site officiel* 53
6.14. HORNETQ 54
6.14.1. *Historique* 54
6.14.2. *Version actuelle* 54
6.14.3. *Date de sortie* 54
6.14.4. *Licence* 54
6.14.5. *Protocoles* 54
6.14.6. *Fonctionnalités* 54
6.14.7. *Domaines métiers principaux* 54
6.14.8. *Couche OSI* 55
6.14.9. *Architecture interne* 55
6.14.10. *Site officiel* 55
6.15. OCEAN 56
6.15.1. *Historique* 56
6.15.2. *Version actuelle* 56
6.15.3. *Date de sortie* 56
6.15.4. *Licence* 56
6.15.5. *Fonctionnalités* 56
6.15.6. *Domaine métier principal* 56
6.15.7. *Couche OSI* 56
6.15.8. *Implémentations* 57
6.16. RRCAP 58
6.16.1. *Historique* 58
6.16.2. *Version actuelle* 58
6.16.3. *Date de sortie* 58
6.16.4. *Licence* 58
6.16.5. *Fonctionnalités* 58
6.16.6. *Domaine métier principal* 58
6.16.7. *Couche OSI* 58
6.16.8. *Implémentations* 58
6.17. ASPIRE 59
6.17.1. *Historique* 59
6.17.2. *Version actuelle* 59
6.17.3. *Date de sortie* 59
6.17.4. *Licence* 59

6.17.5. Protocole 59
6.17.6. Fonctionnalités 59
6.17.7. Domaine métier principal 60
6.17.8. Couche OSI 60
6.17.9. Architecture interne 60
6.17.10. Spécifications 61
6.17.11. Sites officiels 61
6.18. WISREED 62
6.18.1. Historique 62
6.18.2. Licence 62
6.18.3. Protocoles 62
6.18.4. Fonctionnalités 62
6.18.5. Domaine métier principal 63
6.18.6. Couche OSI 63
6.18.7. Architecture interne 63
6.18.8. Site officiel 64
6.19. OPENSOAP 65
6.19.1. Historique 65
6.19.2. Version actuelle 65
6.19.3. Date de sortie 65
6.19.4. Licence 65
6.19.5. Protocoles 65
6.19.6. Fonctionnalités 65
6.19.7. Domaine métier principal 65
6.19.8. Couche OSI 65
6.19.9. Architecture interne 66
6.19.10. Site officiel 66
6.20. APACHE ACTIVEMQ 67
6.20.1. Historique 67
6.20.2. Version actuelle 67
6.20.3. Date de sortie 67
6.20.4. Licence 67
6.20.5. Protocoles 67
6.20.6. Fonctionnalités 68
6.20.7. Couche OSI 68
6.20.8. Architecture interne 68
6.20.9. Spécifications 69
6.20.10. Site officiel/Forum 69
6.21. ORACLE STREAMS ADVANCED QUEUING 70
6.21.1. Historique 70
6.21.2. Version actuelle 70
6.21.3. Date de sortie 70
6.21.4. Licence 70
6.21.5. Protocoles 70
6.21.6. Fonctionnalités 70
6.21.7. Couche OSI 71
6.21.8. Architecture interne 71
6.21.9. Spécifications 72
6.21.10. Site officiel 72
6.22. STORMMQ 73
6.22.1. Historique 73
6.22.2. Version actuelle 73
6.22.3. Date de sortie 73
6.22.4. Licence 73
6.22.5. Protocole 73

6.22.6. Fonctionnalités 73
6.22.7. Domaines métiers principaux 74
6.22.8. Couche OSI 74
6.22.9. Architecture interne 74
6.22.10. Site officiel 74

7. CONCLUSION 75

LISTE DES FIGURES

Figure 1 : Trame X10 17
Figure 2 : Structure d'une trame IGMP 21
Figure 3 : Structure d'une trame Ethernet 21
Figure 4 : Liste des versions WOSH 24
Figure 5 : Architecture interne EnTiMid 27
Figure 6 : Implémentations EnTiMid 27
Figure 7 : Architecture interne Centricity EE 34
Figure 8 : Architecture interne GAIA 37
Figure 9 : Liste des MBBs de ExORB 40
Figure 10 : Architecture internet WSAMI 42
Figure 11 : Architecture interne REDS 49
Figure 12 : Architecture interne JacORB 51
Figure 13 : Architecture interne HornetQ 55
Figure 14 : Architecture interne ASPIRE 61
Figure 15 : Architecture prise en charge par WisReed 63
Figure 16 : Architecture interne OpenSOAP 66
Figure 17 : Architecture interne Apache ActiveMQ 68
Figure 18 : Architecture globale d'Oracle Streams AQ 71
Figure 19 : Architecture via http d'Oracle Streams AQ 71
Figure 20 : Propagation http d'Oracle Streams AQ 72
Figure 21 : Architecture AMQP 74

1. INTRODUCTION

1.1. BUT

Ce document constitue la réponse concernant la veille technologique effectuée sur les middlewares industriels. Nous avions pour but de réaliser une étude des différents middlewares et protocoles dans le monde de l'industrie. Cette étude concerne aussi bien les classiques tels que les MOM ou ceux dans des domaines spécifiques. Nous devons réaliser une étude comparative en termes de domaines métiers, types de protocoles et de parts de marché de ces différents protocoles et middlewares. Suite à cette première étape, nous devrons suivre l'évolution de ceux-ci pour étudier les différentes possibilités de mis en commun. Actuellement, de nombreux middlewares et protocoles différents sont utilisés en entreprise. L'incompatibilité de ceux-ci rend le travail de mis en place des applications difficiles. Il nous a été demandé de trouver des middlewares permettant de regrouper ceux-ci au sein de l'entreprise.

1.2. DEMARCHE

Pour la réalisation de cette étude, nous avons décidé de lister les protocoles implémentés dans les middlewares puis de lister les middlewares dans le monde de l'industrie (énergie, santé, domotique, téléphonie mobile et automobile). Ainsi, nous nous sommes réparti le travail de la manière suivante :

- ✓ Deux personnes sur la rédaction des fiches middlewares
- ✓ Trois personnes sur les recherches de middlewares : Travail en simultané dans une même salle ou communication par messagerie instantanée.

Dans un premier temps, nous avons rédigé l'historique avec les différentes dates importantes puis la licence sous laquelle il évolue. Ensuite, nous avons spécifié les données techniques et mis les différentes ressources d'informations.

2. LE CONTEXTE

2.1. MODELE OSI

C'est un modèle de communications entre ordinateurs proposé par l'ISO (International Organization for Standardization). Il décrit les fonctionnalités obligatoires à la communication et organisation de ces fonctions.

Il existe 7 couches de protocoles différentes :

- **Couche 7** : Application
- **Couche 6** : Présentation
- **Couche 5** : Session
- **Couche 4** : Transport
- **Couche 3** : Réseau
- **Couche 2** : Liaison
- **Couche 1** : Physique

2.2. DEFINITION D'UN PROTOCOLE

Le protocole est le langage utilisé pour la compréhension entre applications logicielles de même type dans un réseau. Les équipements et les applications doivent comprendre le même protocole pour pouvoir communiquer. Les protocoles nécessitent d'être sur la même couche OSI pour pouvoir fonctionner ensemble. Par exemple, le protocole TCP ou UDP fonctionnera sur la couche transport tandis que le protocole FTP ou SMTP fonctionnera sur la couche application. Pour ne citer qu'eux, sur la couche réseau IPv4 est utilisé, pour la couche présentation nous trouvons le protocole MIME ou ASCII, pour la couche session nous remarquons TLS, pour la couche liaison nous pouvons citer Ethernet et pour la couche physique le Bluetooth.

2.3. DEFINITION D'UN MIDDLEWARE

Un middleware est un logiciel qui permet d'échanger des informations entre plusieurs applications et qui se situe au-dessus de la couche transport du modèle OSI (couches 5, 6 et 7). Le réseau implique une utilisation d'une même technique pour échanger ces informations dans toutes les applications reliées grâce à des composants logiciels. Les middlewares garantissent la communication entre ces applications. Ils ne s'occupent pas des caractéristiques matérielles et logiciels. Ils sont utilisés pour l'échange de messages, l'appel de procédures à distance et la manipulation d'objet à distance. Ils sont utilisés pour relier les applications informatiques hétéroclites des systèmes d'information des industries ou institutions.

2.4. LES ACTEURS

Les différents acteurs des middlewares sont toutes les sociétés spécialisées dans les différents domaines telles que GE Healthcare pour la santé, ou différents centres de recherche pour la domotique afin de faciliter l'interopérabilité dans le domaine de la domotique. De plus, nous pouvons observer différents utilisateurs tels que les différents Centres Hospitalier Universitaire dans la santé ou les entreprises comme GDF/EDF dans l'énergie.

3. LICENCES ET NORMES

3.1. LGPL

Cette licence permet d'implémenter les fonctionnalités du protocole en s'affranchissant du caractère héréditaire de la licence GPL. Plus précisément, la licence LGPL n'a pas de copyleft. Elle permet le développement de logiciels propriétaires utilisant cette librairie libre. Cependant, une modification du code source de cette librairie implique une publication sous la licence LGPL.

3.2. CPAL

C'est une licence pour les logiciels libres approuvée en 2007. Toutes communications sur les modifications apportées peuvent être réutilisées. Cette licence n'est pas compatible avec la licence GPL et elle est approuvée pour le modèle OSI.

3.3. GPL

Proche de la licence LGPL, elle diffère du fait qu'une publication d'une modification peut être réutilisée. De plus, une modification du code source doit forcément être republiée sous cette licence.

3.4. IHM

Cette norme permet d'assurer une meilleure opérabilité entre les systèmes. Les développements sont aussi théoriquement facilités.

3.5. HL7

Cette norme définit la structure et le rôle des messages pour permettre une communication efficace des données liées au système de santé. C'est un langage qui fonctionne au niveau de la couche 7 du modèle OSI. (Version actuelle : 3)

3.6. DICOM

Cette norme s'applique sur la gestion des données issues de l'imagerie médicale. Ce standard définit un format de fichier, mais aussi un protocole de transmission des données (basé sur TCP/IP).

4. MIDDLEWARES PAR DOMAINES METIERS

Nous regroupons les différents middlewares étudiés par domaines métiers. Plusieurs middlewares peuvent être utilisés dans différents domaines. Les 5 principaux domaines que nous avons retenus sont la domotique, les télécommunications, la santé, l'énergie et l'automobile. Tous les détails des middlewares cités sont disponibles dans les fiches décrites au chapitre 8.

4.1. DOMOTIQUE

- WOSH
- EnTiMID
- SM4ALL
- GAIA
- WSAMI
- REDS

4.2. TELECOMMUNICATION

- ExORB
- CORTEX
- LIME
- REDS
- OCEAN
- RRCAP

4.3. SANTE

- Sirilog-Radio
- Centricity EE

4.4. ENERGIE

- WisReed
- STORMMQ

4.5. AUTOMOBILE

- GAIA
- SM4ALL
- STORMMQ

5. LES PROTOCOLES

5.1. X10

Crée en 1978 par Pico Electronics une petite entreprise écossaise qui réalise des travaux sur la domotique. Le protocole s'est très rapidement implanté aux États-Unis où il fut l'un des premiers standards pour la domotique.

Le protocole est un standard ouvert, ce qui a permis à de nombreux constructeurs de proposer des modules très variés, ce qui rend le protocole très populaire de nos jours aussi bien dans le monde professionnel que chez les particuliers.

Le X10 est un protocole dont le but est la communication entre émetteur et récepteur, par courant porteur haute fréquence (120 kHz). On définit un système de module permettant l'adressage de chaque périphérique. Ainsi chaque module est caractérisé par une lettre (de A à P) et un numéro (de 1 à 16).Les émetteurs envoient leurs ordres à tous les récepteurs, chaque ordre contient une partie adressage et une partie commande (entrecoupé par une partie de silence). La partie commande permet d'envoyer différents ordres tels que : éteindre ou allumer un périphérique, variation du courant, des informations de températures et d'autres informations recueillies à parti de capteurs d'informations … Le protocole permet aussi une communication via onde radio vers récepteur sur prise murale.

La construction de trames se fait de la manière suivante :

Adressage d'un module grâce à la lettre lui correspondant ainsi que son numéro, le message est répété deux fois pour éviter de le confondre avec une interférence.

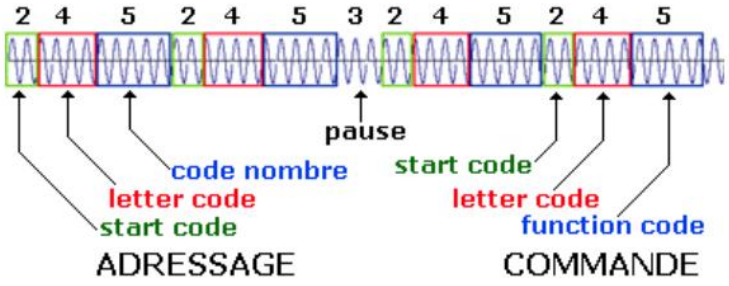

Figure 1 : Trame X10

La partie commande est composée d'un code lettre qui correspond à celui utilisé dans la partie adressage. La commande est caractérisée par un code fonction sur 4 bits (16 commandes possibles spécifiées dans le standard X10).La trame dure 47 périodes du réseau électrique (ce qui correspond à 940 ms soit près d'une seconde !)

Le protocole est principalement utilisé dans le domaine de la domotique. Elle est aussi bien utilisée pour le marché grand public que professionnelle, car le prix des composants utilisant le protocole est très compétitif et sa grande souplesse d'utilisation permet de réaliser des

réseaux complexes de façon aisée.

L'avantage du protocole X10 utilise le courant porteur pour transporter l'information, ce qui réduit le besoin en installation. Cependant l'utilisation de courant porteur peut poser problème vis-à-vis de la l'utilisation internationale, en effet pas tous les pays n'ont la même utilisation du courant électrique, de plus le courant peut être sujet à des perturbations à cause d'autres appareils. Le problème majeur de ce protocole et qu'il n'y a pas d'acquittement lorsqu'un appareil reçoit le message, ce qui ne permet pas d'avoir un contrôle d'erreur très précis (seulement utilisation de l'écho du message pour s'assurer qu'il a bien été envoyé). Le protocole semble vieillissant et de nombreuses avances sur l'utilisation des CPL (Courants porteurs en ligne) ont permis à d'autres protocoles d'offrir de meilleur débit d'utilisation, ainsi qu'une meilleure fiabilité.

5.2. KNX

KNX est un standard orienté pour le domaine de la domotique qui utilise les bus EIB/KNX qui reposent sur un standard uniforme européen (EN 50090). Le standard est ouvert et non propriétaire. Le système repose sur une architecture non centralisée, de ce fait il n'y a pas besoin de PC pour faire fonctionner une architecture KNX, cependant un logiciel existe pour configurer les équipements comme l'on veut, ce logiciel s'appelle ETS. L'ensemble des équipements qui communiquent via le bus EIB/KNX est appelé des participants.

L'avantage d'utiliser un bus et pas le courant porteur est de pouvoir réaliser un câblage sur mesure qui ne dérangera pas la communication entre les participants à cause des parasites sur le réseau électrique. En contrepartie cela oblige l'utilisateur d'investir dans une installation. Le bus étant non propriétaire, l'utilisateur a une grande liberté dans l'achat des équipements, ce qui permet d'avoir un réseau hétérogène sans se limiter aux produits d'un fabricant.

Il est utilisé par de grands noms de l'industrie telle que SIEMENS, SCHNEIDER, ABB.

5.3. UPnP

UPnP est un protocole réseau, il permet une simplification de la mise en réseau et la collaboration de périphériques et systèmes tels que les ordinateurs, les imprimantes, les routeurs, les points d'accès Wi-Fi mais aussi les périphériques portables. Le but est que lorsqu'un périphérique se connecte, sa présence est reconnue automatiquement sur le réseau et permet de proposer ou d'accéder à des services et fonctionnalités réseau.

Il est basé sur les protocoles UDP, TCP et HTTP. Il permet la communication entre deux machines dans le même réseau local.

Certains problèmes sont à noter :

- Pas de protocole d'authentification
- Utilise HTTP au-dessus d'UDP

5.4. DPWS

Initialement publié en mai 2004 par Microsoft, le protocole DPWS a été soumis à la standardisation par OASIS (pour Organization for the Advancement of Structured Information Standards) en 2008 qui est une organisation mondiale créée en 1993 et qui s'occupe de la standardisation des protocoles pour le e-commerce et les web services.

Le standard est libre et peut être téléchargé sur internet. Son utilisation est donc gratuite et libre d'accès.

Les objectifs de DPWS sont similaires à ceux d'UPnP mais DPWS est basé sur une utilisation Web Service.

DPWS reprend ces principes en se recentrant sur l'utilisation de Web Services. Devenu un standard (validé en 2008 par le consortium OASIS), DPWS est une spécification qui explique comment utiliser et assembler différents standards du Web Services tel que WS-Addressing (adressage de messages sur le réseau), WS-Discovery (découvrir des périphériques sur le réseau répondant à des services désirés), WS-Eventing(permet la souscription à un Web Service et permet ainsi l'envoie asynchrone de notifications), WS-Transfer (permet le transfert des ressources de Web Service) ou WS-MetadataExchange (permet d'obtenir les spécifications des formats des données à transmettre pour un Web Service) afin de créer des scénarios qui répondent aux besoins des utilisateurs de périphériques connectés. Afin de communiquer entre les différents périphériques, deux protocoles de communication sont utilisés, WSDL et SOAP.

On distingue trois entités: les équipements, les services et les points de contrôle. Des équipements peuvent héberger des services et utiliser des services, hébergés par d'autres équipements. Les points de contrôles permettent de découvrir les services disponibles sur le réseau, et de s'y abonner.

DPWS est utilisé dans le domaine de l'automatisation, comme l'atteste le projet de recherche Européen SOCRADES alliant des poids lourds du domaine tel que Schneider Electric et Siemens.

Grâce à son standard libre, on peut découvrir d'autres domaines d'applications : Le DPWS est une implémentation spécialement réalisée pour les périphériques embarqués (mobiles, tablettes, domotiques …) et pour les capteurs utilisant les réseaux sans fil.

De nombreux environnements de développements permettent d'utiliser DPWS, nous pouvons citer la célèbre API .NET de Windows qui permet de concevoir des applications utilisant DWPS, ou encore la WS4D API qui permet une utilisation de différents langages (C, C++,java) et même une utilisation via interface graphique.

L'avantage de l'utilisation de DPWS est de pouvoir utiliser de manière homogène différents équipements de type et de domaine différents (équipement de surveillance, téléphone portable, etc.). Un autre avantage est le fait que le fonctionnement du réseau ne dépend pas de la disponibilité d'un périphérique, un périphérique peut être mis en veille par un utilisateur ou alors déplacé hors du réseau ce qui est très utile pour les appareils transportables. Enfin, DPWS permet d'avoir une communication sécurisée (contrôle d'intégrité, contrôle d'authentification cf. chapitre 7.1 des spécifications du protocole).

5.5. LONWORKS - LONTALK

La technologie LonWorks a été créée par Echelon il y a une vingtaine d'années. Il s'agit d'une famille de produit dédié à la domotique. Cette technologie, tel qu'elle est utilisée actuellement, LonWorks 2.0, a été ratifiée comme un standard de la domotique en 2009.

Le protocole utilisé par celle-ci est LonTalk. Il s'agit d'un protocole propriétaire, qui au vu de la complexité de certains de ces concepts, nécessite l'utilisation de *Neuron*, une puce spécialement développée par Echelon. Après un accord basé sur des royalties, Motorola a obtenu une licence lui permettant d'utiliser cette puce.

Il est basé sur une architecture P2P. Il n'y a pas de maître qui détermine qui doit communiquer.
Chaque composant du système est libre d'émettre directement, sans forcément connaître la topologie du réseau.

Les réseaux LonWorks peuvent être implémentés selon différents types de support : par courant porteur, paires torsadées, radio, infrarouge, câble coaxial ou encore fibre optique.

Le choix du support se fait en fonction des besoins en débits du système.

Le protocole LonTalk est un standard utilisé dans de nombreux domaines tels que les réseaux de contrôle, la domotique, le transport ferroviaire, l'automatisation industrielle ou encore les éclairages publics. Il s'agit aussi d'un standard européen pour les stations-services.

Le principal avantage de cette technologie est la puce *Neuron* en elle-même. En effet, celle-ci permet, en plus d'offrir un moyen de communication, de contenir une couche application. Cela permet donc de limiter les coûts matériels.

5.6. TCP (TRANSMISSION CONTROL PROTOCOL)

Premier réseau à transfert de fichier développé en 1973 par la DARPA (Defense Advanced Research Projects Agency), le protocole TCP a été adopté par Arpanet (Advanced Research Projects Agency Network) en 1976 avant d'être déployé dans le reste du monde.

Ce protocole TCP est un protocole de transport fiable, en mode connecté, documenté dans la RFC 793. Dans le modèle TCP/IP, TCP est situé au niveau de la couche de transport (Couche 4 sur le modèle OSI). Les applications transmettent des flux de données sur une connexion réseau, et TCP découpe le flux d'octets en segments, dont la taille dépend de la MTU du réseau sous-jacent (couche liaison de données).

Une session TCP fonctionne en trois phases :

- l'établissement de la connexion
- les transferts de données
- la fin de la connexion

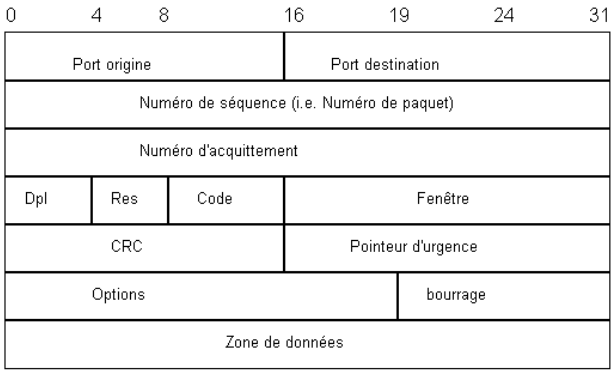

Figure 2 : Structure d'une trame IGMP

5.7. ETHERNET

Le premier réseau utilisant Ethernet a été réalisé par Robert Metcalfe et son assistant David Boggs dans les années 70. Le protocole Ethernet est un protocole de réseau local à commutation de paquets. Il est devenu une norme internationale ISO/IEC 8802-3 et a été standardisé sous le nom IEEE 802.3.

Son fonctionnement est basé sur un principe de dialogue sans connexion et donc sans fiabilité. Les trames sont envoyées par la carte réseau sans aucune procédure de type « handshake » avec l'adaptateur destinataire. Ce service de couche 2 du modèle OSI est similaire au service en mode datagramme de couche 3 assuré par IP et au service sans connexion de couche 4 d'UDP.

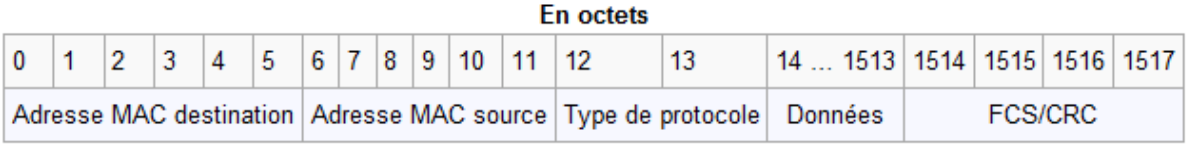

Figure 3 : Structure d'une trame Ethernet

Le protocole Ethernet est notamment utilisé aujourd'hui dans le cadre de liaison entre les ordinateurs et les routeurs internet via l'utilisation de câble RJ45 ou dans la téléphonie mobile avec l'installation de ce type de technologie sur les nouvelles baies 3G. La démocratisation de ce protocole a pu se produire grâce à de nombreux avantages tels qu'une rapidité d'échange (100 Mbits/s couramment utilisé), une connexion stable et permanente et la sécurisation d'un échange par voie filaire. Ce dernier point reste cependant un défaut de cette technologie, car qui dit réseau filaire dit encombrement de câbles et mobilité réduite.

5.8. KERMIT

Développé en 1981 au sein de l'université de Columbia, le protocole Kermit permet le transfert de fichier en ajoutant à ce dernier une couche de contrôle d'erreur. Ce protocole qui n'est désormais plus supporté fut beaucoup utilisé au début des années 1980, car ce dernier permet aux fichiers d'être échangés sur de nombreux types de machines tels que les systèmes à microprocesseur, les mainframes aux OS propriétaires...

Kermit permet l'envoi de plusieurs fichiers par lot, avec leurs attributs. La force de ce protocole réside donc dans sa disponibilité sur un grand nombre de types de machine, cependant cet avantage constitue également sa plus grande faiblesse. En effet, cette compatibilité ampute de beaucoup ses performances ce qui fait que l'utilisation de ce protocole est extrêmement couteuse en temps machine.

5.9. AMQP (ADVANCED MESSAGE QUEUE PROTOCOL)

Développé entre 2004 et 2006 par la banque JPMorgan Chase et iMatix Corporation, il avait pour but de concurrencer le monopole établi par IBM et permettre à tous les éditeurs de l'implémenter dans leurs solutions d'intégration.

AMQP est sous licence GPL (General Public License). Par définition, c'est un protocole gratuit et adaptable aux différentes solutions souhaitées.

Les Middlewares Orientés Message ne se soucient pas du message échangé comme expliqué précédemment. Cependant, le protocole AMQP est un protocole « wire-level ». Il décrit le format de données envoyées par l'application A sur le réseau sous forme de flux d'octets. Ainsi, toutes applications pouvant créer et interpréter ce format de données peut interagir avec d'autres applications quel que soit le langage utilisé. Une application JAVA pourra communiquer avec une application .NET sans aucun problème. Le protocole est utilisé entre un client et un serveur. Ce lien est appelé « broker ». L'application doit se connecter via ce lien pour rentrer sur le réseau AMQP. Une fois la connexion établie, l'application ouvrira une session et pourra utiliser le protocole pour envoyer ses messages aux différentes applications. Ce protocole s'appuie sur le protocole réseau TCP/IP.

Le cahier des charges définit, pour ce protocole, un système de files d'attente des messages envoyés. Ce n'est pas défini dans les MOM mais AMQP permet de créer des files avec des conditions particulières pour que le message intègre celles-ci. Chaque queue est identifiée par un nom et une adresse. Selon les conditions créées, le protocole enregistrera le message dans la file correspondante. Par exemple, dans le monde industriel, si les messages envoyés sont la production journalière de pièces, la quantité va varie. Les conditions de stockage sur deux files peuvent être les productions inférieures et supérieures à un certain nombre. Les messages seront lisibles que par l'application concernée.

L'avantage de ce protocole est sa souplesse. Il est utilisable dans n'importe quel environnement. De plus, il gère la communication avec simplicité grâce à ses différents systèmes d'échanges applicables. Cependant, le monopole d'IBM ne permet pas ou peu à AMQP de se développer dans ce secteur. Le paiement d'une licence assure au client un suivi. C'est un gage de tranquillité tandis qu'un protocole gratuit demande un déploiement et une gestion quotidienne.

5.10. STOMP

STOMP est un protocole utilisé pour les MOM. C'est un protocole basé sur TCP/IP et sous licence GPL. La communication entre l'application et le serveur se fait au moyen de texte au travers d'une frame spéciale constituée d'une série de lignes de commandes. Ce protocole est similaire au protocole OpenWire. Pour exemple, nous pouvons citer quelques lignes de commandes servant à envoyer ou recevoir des messages telles que :

```
SEND

destination:/queue/a

hello queue a

^@
```

```
RECEIPT

receipt-id:message-12345

^@
```

5.11.HTTP (HYPERTEXT TRANSFER PROTOCOL)

Inventé par Tim Berners-Lee depuis 1996, ce protocole permet la communication entre un client et un serveur et a été développé pour le World Wide Web. C'est un protocole de la couche 7 application du modèle OSI et utilise TCP comme couche de transport. Les principaux clients de ce protocole sont les navigateurs internet qui permettent à un utilisateur d'accéder à un serveur contenant des données.

Ce protocole est le seul utilisé sur le marché dans le monde de l'internet. Cependant, il a de grosses failles de sécurité concernant les échanges de données par l'utilisateur vers le serveur. Il a été mis en place un autre protocole HTTPS fonctionnant de la même manière mais cryptant les informations échangées.

6. LES FICHES MIDDLEWARES

6.1. WOSH

6.1.1. Historique

Depuis 2009, plusieurs versions ont été implémentées. Dans un premier temps, « byron » et « faust » ont fait leur apparition avec l'implémentation du Data Mining. Ensuite, « icarus » a permis d'améliorer la stabilité et la performance grâce à des connexions. Pour obtenir une standardisation du protocole, la version « icarus » utilise des appels de la librairie STL. « Qed » implémente différents plugins tandis que « wolf » améliore la sécurité et les permissions, et implémente les persistances ainsi que les bases de données. La dernière version, « phoenix », offre une stabilité dans le Shell WOSH.

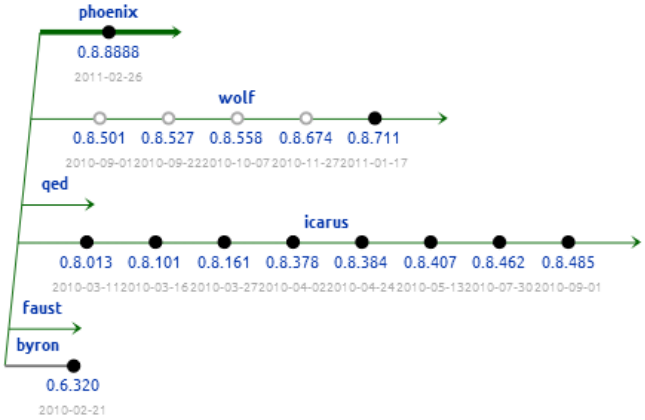

Figure 4 : Liste des versions WOSH

6.1.2. Version actuelle

WOSH-phoenix version 0.8.888 sortie le 26/02/2011

6.1.3. Date de sortie

2009

6.1.4. Licence

Common Public Attribution License 1.0 (CPAL)

6.1.5. Protocoles

- X10
- UDP/TCP
- RNDIS

De plus, il utilise différents langages tels qu'AINSI, C++ et QT4.

6.1.6. Fonctionnalités

WOSH est une multi plateforme de framework pour la domotique.

6.1.7. Domaine métier principal

Domotique

6.1.8. Couche OSI

Couche 7 - Applicative

6.1.9. Architecture interne

- Utilisation en tant que logiciel utilisateur, permet la manipulation directe des différents appareils connectés avec une interface graphique ou par ligne de commande.
- Utilisation à distance grâce à la messagerie instantanée (utilisation de libgloox pour l'interaction avec un serveur XMPP), par SMS ou avec un appareil mobile utilisant une connexion internet.
- Multiplateforme (Windows 98+/XP/Vista/7, POSIX Linux, MAC OS, Windows CE/Mobile System)

6.1.10. Spécifications

http://wosh.sourceforge.net/docs/

6.1.11. Site officiel

http://wosh.sourceforge.net/

6.2. ENTIMID

6.2.1. Historique

Ce middleware est en cours de recherche et permet de répondre à différentes exigences :

- L'interconnexion des réseaux
- L'ouverture à des protocoles de haut niveau
- Favoriser les accès nomades
- Des solutions dynamiques de déploiement
- La sûreté et sécurité

Ce logiciel se veut comme une plateforme permettant de faire communiquer les différents middlewares dans la domotique grâce à des passerelles liées aux différents protocoles. Il permettra une standardisation des middlewares utilisés.

6.2.2. Version actuelle

Projet de recherche

6.2.3. Protocoles

- KNX
- X10
- DPWS
- UPnP
- SOAP
- LonWorks
- Protocoles à venir (X2D, IOBL et d'autres en cours de spécification)

6.2.4. Fonctionnalités

Il permet l'interopérabilité des différents logiciels utilisés dans la domotique grâce à une large gamme de protocoles pris en compte.

6.2.5. Domaine métier principal

Domotique

6.2.6. Couche OSI

Couche 7 - Applicative

6.2.7. Architecture interne

- Architecture en couche sous forme de module permettant de prendre en compte de nouveaux protocoles
- Communication par messages décentralisés ce qui permet de limiter les perturbations entre les différents appareils et de prendre en compte l'asynchronisme de la communication entre les appareils.
- Interopérabilité entre protocoles ou entre intergiciels grâce à une couche de communication utilisant les protocoles SOAP, DPWS ou UPnP.
- Couche d'abstraction permettant un encodage et décodage propre à chaque protocole.

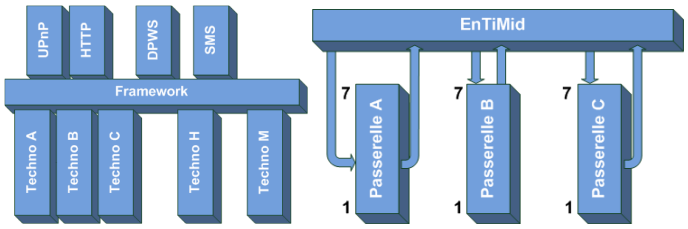

Figure 5 : Architecture interne EnTiMid

6.2.8. Implémentations

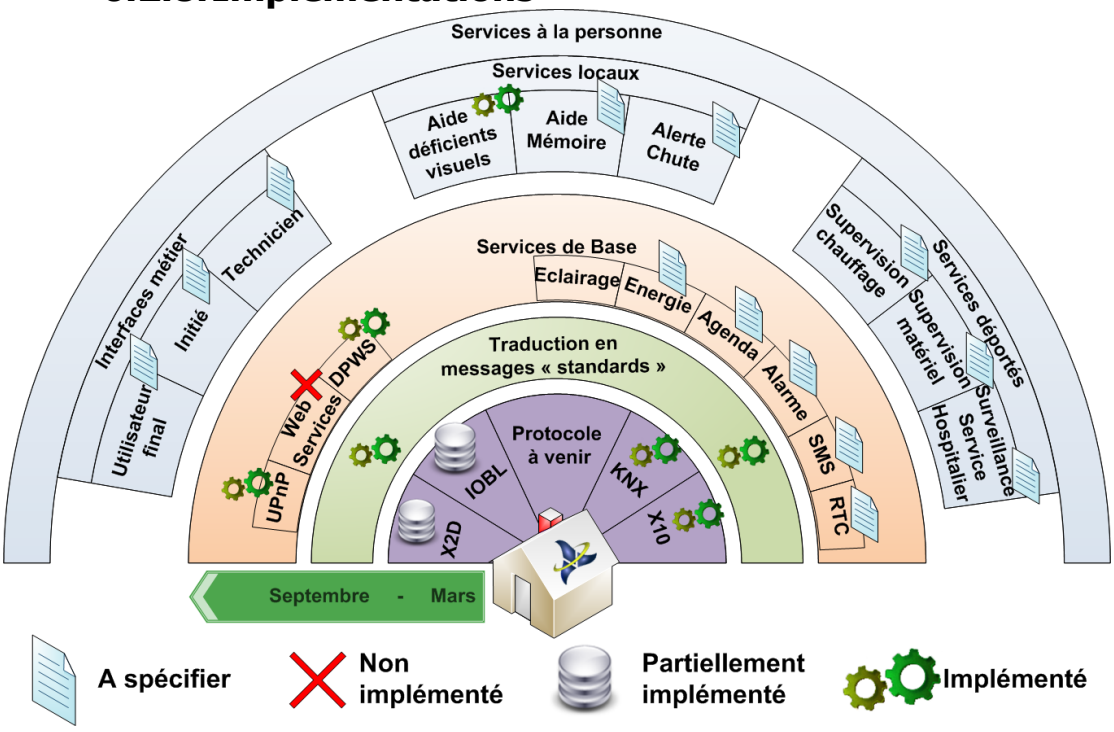

Figure 6 : Implémentations EnTiMid

Il implémente plusieurs protocoles et leur traduction en messages standards. Il manque à implémenter beaucoup de services de base et de services à la personne.

6.2.9. Spécifications

http://www.irisa.fr/triskell/publis/2009/Nain09a.pdf

http://videos.rennes.inria.fr/fete-sciences-2009/index.html

6.3. SM4ALL

6.3.1. Historique

Ce middleware est en cours de recherche et vise à constituer des passerelles entre les différents systèmes embarqués que ce soit dans la domotique ou l'automobile. C'est un projet européen regroupant différents groupes de travailleurs de différentes universités ou entreprises telles que Vienna University of Technology ou Telefonica.

6.3.2. Version actuelle

Version 8.1b le 22/09/2011

6.3.3. Date de sortie

Le 17/10/2009

6.3.4. Licence

GPL version 3

6.3.5. Protocoles

- X10 (PLC)
- OSGI, UPnP
- Ethernet
- Wifi
- Zigbee (bluetooth)
- SOAP

6.3.6. Fonctionnalités

Il propose une interopérabilité des différents systèmes embarqués utilisés dans l'automobile ou la domotique.

6.3.7. Domaines métiers principaux

Automobile et domotique

6.3.8. Couche OSI

Couche 7 - Applicative

6.3.9. Architecture interne

- Architecture en couche d'abstraction, permettant à l'utilisateur final d'utiliser les appareils utilisant des protocoles différents de façon uniforme via le protocole SOAP.
- Architecture centralisée
- Middleware orienté service, utilisation de UPnP (avec le framework OSGI)
- Basé sur l'architecture REST

6.3.10. Implémentations

Il permet l'interactivité des évènements entre producteurs et consommateurs. De plus, il offre l'opportunité de simuler l'environnement de la maison pour savoir les évènements qu'il s'y produit. Par exemple, dès qu'une lumière est allumée, le logiciel nous avertit en inscrivant dans l'interface qu'une lumière s'est allumée.

6.3.11. Site officiel

http://www.sm4all-project.eu/

6.4. SIRILOG-RADIO

6.4.1. Historique

Anciennement propriétaire d'InfoConcept, ce middleware revient à Medasys lorsque le groupe rachète l'entreprise. Il fait partie de la famille des RIS (Radiology Information System).

6.4.2. Version actuelle

Version 3.4.x en 2005

6.4.3. Licence

Propriété du groupe Medasys.

6.4.4. Protocoles

- KERMIT pour la norme H.PR.I.M. 2.
- TCP/IP pour la norme H.PR.I.M. 3.
- DICOM s'appuyant sur TCP/IP.

Il répond aux normes IHM (Integrating the Healthcare Enterprise), HL7 (Health Level 7), DICOM (Digital Imaging and COmmuncations in Medecine) et H.PR.I.M. 2 (Harmonie et PRomotion de l'Informatique Médicale).

6.4.5. Fonctionnalités

Ce middleware est utilisé dans l'imagerie médicale et permet de :

- Gérer des salles et des équipements
- Prendre des rendez-vous
- Réaliser l'examen
- Interpréter les résultats
- Mettre en place les comptes rendus
- Faire des cotations
- Établir des statistiques
- Réaliser de la recherche médicale

De plus, il fonctionne avec une base de données Oracle permettant d'enregistrer les différentes informations.

6.4.6. Domaine métier principal

Santé

6.4.7. Couche OSI

Couche 7 - Applicative

6.4.8. Architecture interne

Il est doté d'une architecture technique ouverte et interopérable puisqu'il utilise les différentes normes établies dans le domaine de la santé. Il alimente le circuit d'information médicale de l'établissement suivant les protocoles normalisés et sécurisés (TCP, DICOM).

Il satisfait les échanges avec les systèmes médico-administratifs de l'hôpital. Il fédère la diffusion des comptes-rendus et images de tous les services. De plus, il dialogue par l'émission et la réception de messages en respectant les normes.

6.4.9. Implémentations

Il est implémenté dans la plupart des CHU de France. En mars 2012, il a été déployé au CHRU de Montpellier.

6.4.10. Site officiel

http://www.medasys.com/company/index.html

6.5. CENTRICITY EE

6.5.1. Historique

Développé par GE Healthcare, il fait partie de la famille des PACS (Picture Archiving and Communication System).

6.5.2. Version actuelle

Version 3.1.x.

6.5.3. Licence

Propriété du groupe GE Healthcare

6.5.4. Protocoles

DICOM s'appuyant sur TCP/IP.

6.5.5. Fonctionnalités

Ce middleware est aussi utilisé dans l'imagerie médicale et permet de :

- Acquérir des images issues des modalités
- Visualiser et interpréter les résultats
- Stocker et archiver
- Diffuser vers les services cliniques

6.5.6. Domaine métier principal

Santé

6.5.7. Couche OSI

Couche 7 – Applicative

6.5.8. Architecture interne

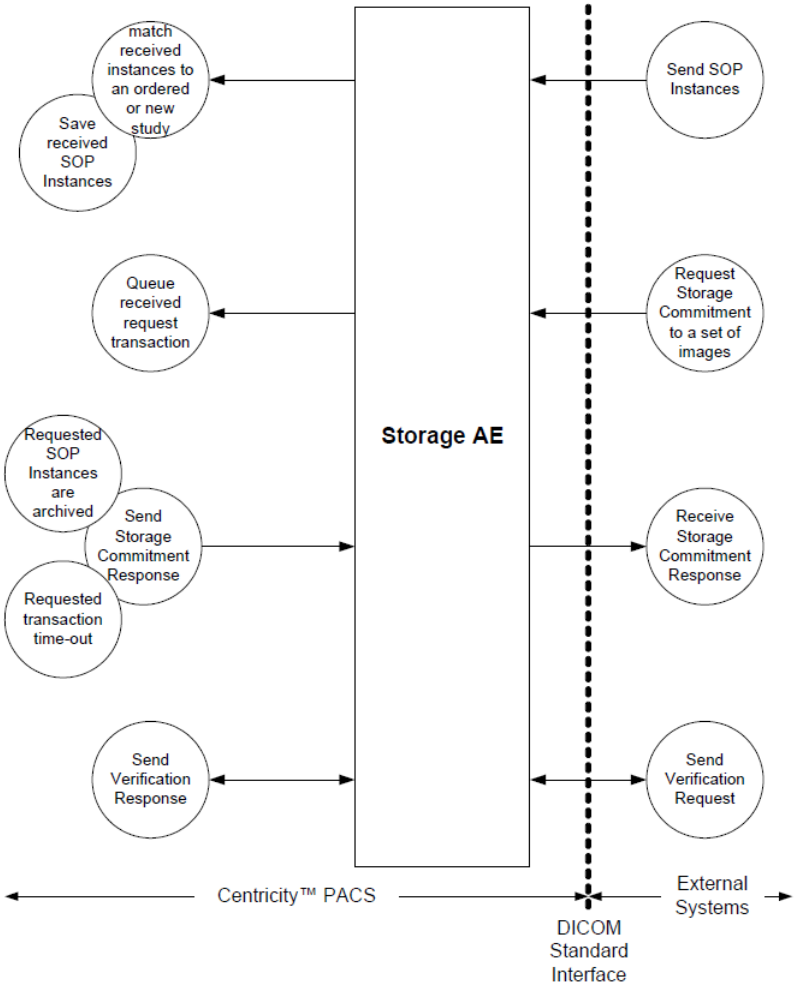

Figure 7 : Architecture interne Centricity EE

Pour stocker et archiver les différentes données, il supporte les fonctions suivantes :

- Réception d'instances SOP d'un objet AE.
- Il relit chaque instance reçue des études commandées ou planifiées de la base de données en faisant correspondre les données du patient.
- Si aucun lien n'est trouvé, il crée une nouvelle étude en utilisant directement les informations du patient.

6.5.9. Spécifications

http://www.gehealthcare.com/eufr/interoperability/dicom/docs/2017753_209r09.pdf

6.5.10.Site officiel

http://www.gehealthcare.com/

6.6. GAIA

6.6.1. Historique

Ce middleware est un projet de recherche réalisé par une équipe provenant de l'université de l'Illinois. Il a pour but de créer une plateforme permettant d'interagir et d'acquérir des informations sur différents lieux tels que les bureaux, les maisons ou bien les voitures.

6.6.2. Version actuelle

Projet de recherche

6.6.3. Protocoles

- X10
- Bluetooth
- Wifi
- Ethernet

Ils permettent l'utilisation de divers appareils tels que : Badge radio fréquence, des équipements utilisant l'infrarouge, scanneur d'iris. Un des composants de GAIA (Code Updater) permet d'intégrer de nouveaux middlewares.

6.6.4. Fonctionnalités

Il permet aux utilisateurs de configurer le comportement de leur habitat. De plus, les nouveaux matériels sont reconnus dynamiquement et s'intègrent simplement avec l'environnement existant. Ce projet est une interface orientée utilisateur permettant de transformer des espaces publics ou privés peuplés de ressources informatiques en réseau compatible.

6.6.5. Domaines métiers principaux

Domotique et automobile

6.6.6. Couche OSI

Couche 7 – Applicative

6.6.7. Architecture interne

GAIA utilise une architecture qui permet d'utiliser un réseau domotique comme un système d'exploitation qui gère l'ensemble des appareils (supporte les signaux / évènements,

système de fichier, sécurité, processus…). Son architecture se découpe ainsi:

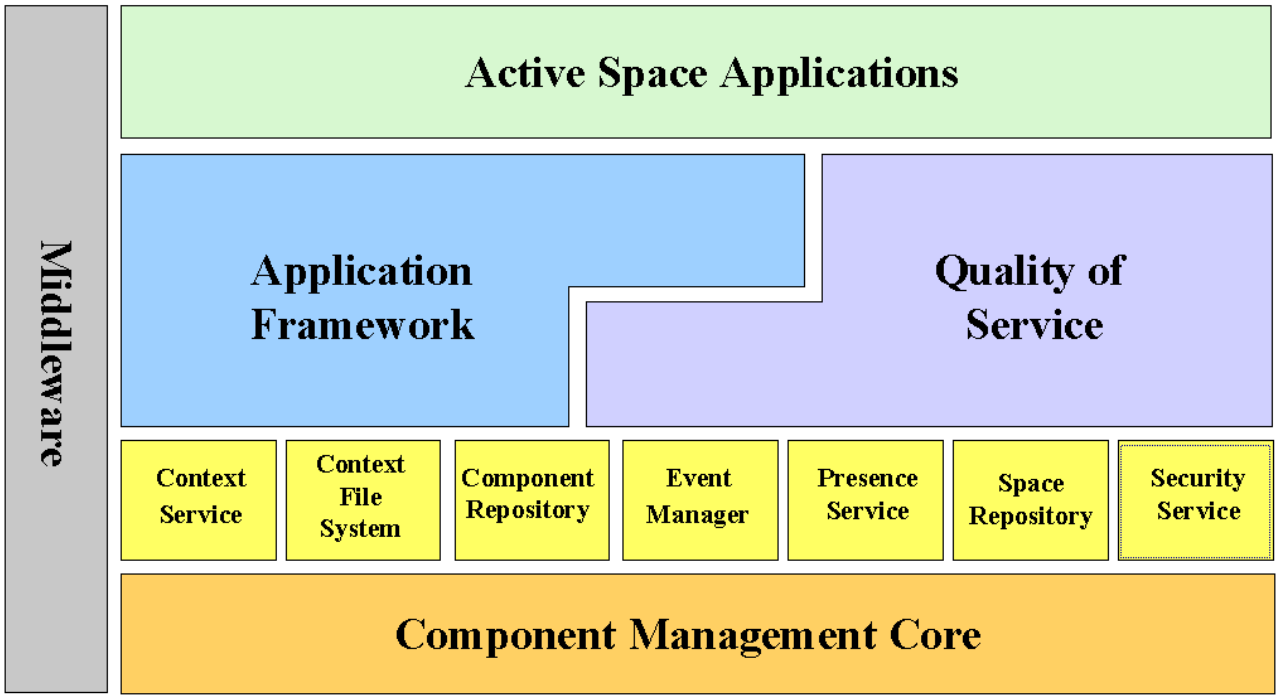

Figure 8 : Architecture interne GAIA

Component Management Core: Permets de manipuler l'ensemble des composants de GAIA, c'est-à-dire ajouter/modifier ou supprimer des objets. GAIA utilise trois éléments:

- GAIA node, un composant dans le réseau qui peut utiliser les fonctionnalités d'un GAIA component, il regroupe en général plusieurs GAIA component dans un GAIA container.
- GAIA component, l'unité logicielle la plus petite, il permet les appels à distance.
- GAIA component container, groupe de GAIA component proposant un ensemble de fonctionnalités pour un composant du réseau.

Event manager : Cet élément permet de répondre aux évènements de façon asynchrone. Tous les GAIA components l'utilisent pour connaître les changements sur le réseau (ajout ou suppression de composant sur le réseau).

Context Service : Permets de concevoir un contexte et de le déployer facilement sur le réseau. Un contexte est un moyen d'interaction entre les différents éléments selon des conditions prédéfinies.

Component Repository : Les composants logiciels (GAIA component) sont stockés dans une table appelée Component Repository. Il permet de stocker les informations de plus bas niveau essentielles à l'exécution de ces composantes et donc du GAIA node associé (type de plateforme matériel, système d'exploitation requis, etc.)

Space Repository : Base de données centralisée, contient toutes les informations à propos des appareils et les services proposés par le réseau. Chaque mise à jour des informations

fait appel à l'Event manager. Chaque composant est décrit sous format XML. Cette base de données permet aux entités du réseau de demander des informations à propos d'autres composants.

6.6.8. Spécifications

http://gaia.cs.uiuc.edu/papers/GaiaSubmitted3.pdf

6.6.9. Site officiel

http://dl.acm.org/citation.cfm?id=643558

6.7. ExORB

6.7.1. Protocoles

- IIOP
- XML-RPC

6.7.2. Fonctionnalités

C'est un middleware orienté service. Il repose sur le principe des RPC, c'est-à-dire des procédures à distances. Il permet de mettre

6.7.3. Domaine métier principal

Téléphonie mobile

6.7.4. Couche OSI

Couche 7 – Applicative

6.7.5. Architecture interne

Sa structure repose sur des composants entièrement modifiables en temps réels. Il se compose de trois éléments :

- Les **micro-building block** (MBB) sont les procédures utilisées par le middleware. Des paramètres d'entrées sont demandés et la procédure retourne des paramètres de sortie. Il stocke aussi des informations sur les états, liées à l'exécution du MBB, la somme des états des MBBs permet d'obtenir l'état d'ExORB.
- Les **Actions** sont les éléments spécifiant la façon dont les MBB doivent être enchainés. Ceci permet de définir la logique du système.
- Les **domaines** regroupent les MBB et les Actions dans une structure de stockage. Ainsi il est possible de manipuler une collection de MBB facilement.

Par défaut, ExORB est composé de 28 MBBs découpés en 11 domaines (cf. figure 2) qui permettent l'utilisation des protocoles IIOP et XML-RPC, mais il est possible d'ajouter de nouveau MBB, Actions et domaines pour d'autres protocoles.

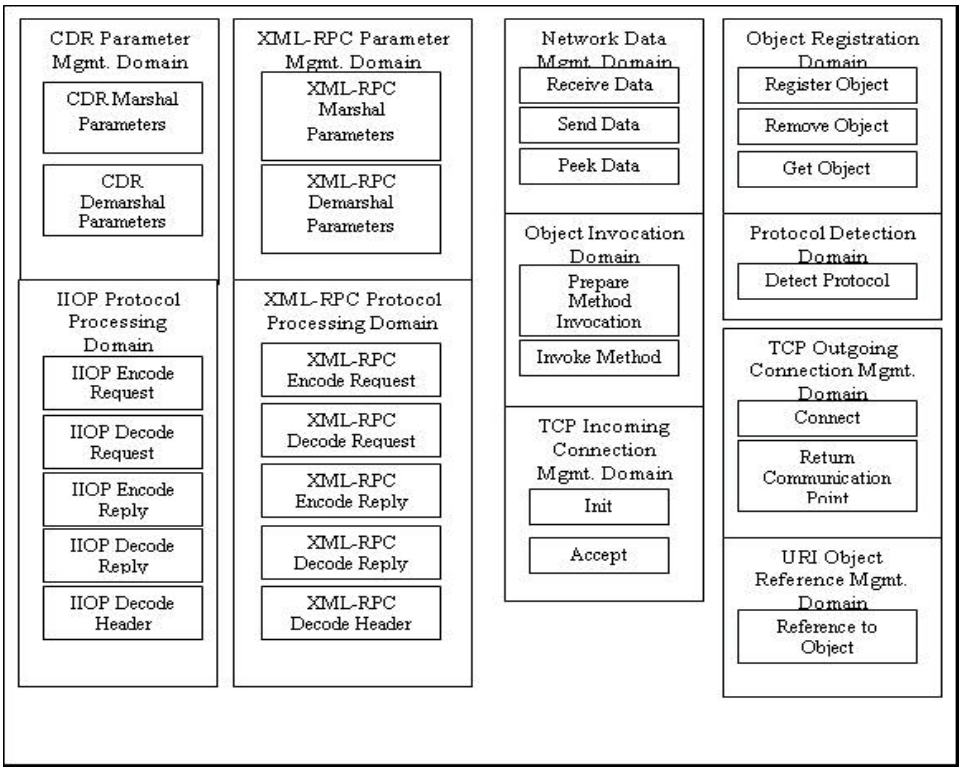

Figure 9 : Liste des MBBs de ExORB

De plus, il se base sur l'architecture CORBA d'où son nom.

6.7.6. Spécifications

http://www.minema.di.fc.ul.pt/summerschool/Blair_ReflectionChapter.pdf

6.8. WSAMI

6.8.1. Historique

Ce middleware fait partie d'un projet de recherche le IST Ozone Project. Ce projet de 32mois a débuté en octobre 2001 et s'est terminé en 2004. Il a été sponsorisé par la commission européenne. Ce nom signifie Web Services for AMbient Intelligence.

6.8.2. Date de sortie

Eté 2004

6.8.3. Version actuelle

Licence LGPL

6.8.4. Protocoles

- WSDL
- SOAP
- UDDI

6.8.5. Fonctionnalités

Il est utilisé afin de gérer un réseau d'appareil pour la domotique.

Le protocole WSDL est l'interface permettant d'utiliser les différents web services

Le protocole SOAP permet d'avoir un mécanisme d'échange d'informations

Le protocole UDDI permet d'enregistrer de nouveaux web services et de localiser les web services existants.

6.8.6. Domaines métiers principaux

Intelligence ambiante, domotique.

6.8.7. Couche OSI

Couche 7 – Applicative.

6.8.8. Architecture interne

C'est un middleware basé sur les architectures web services WSA.

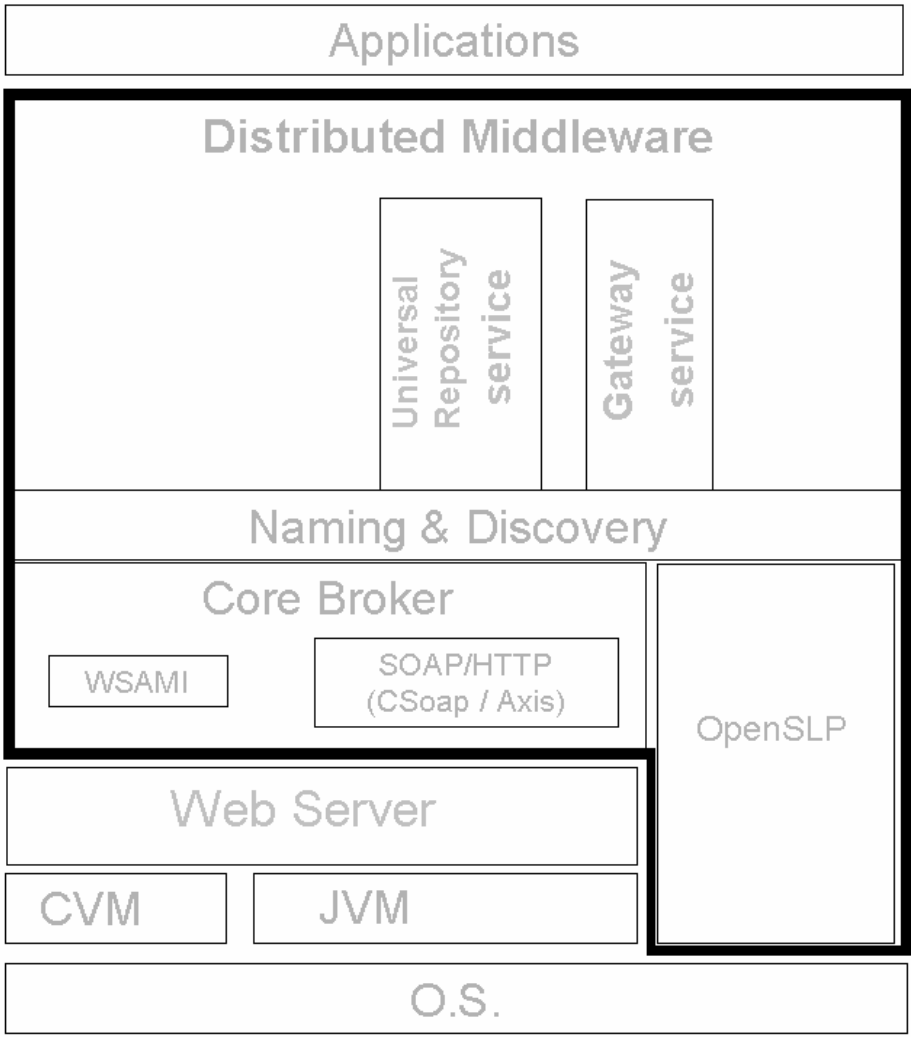

Figure 10 : Architecture internet WSAMI

6.8.9. Spécifications

https://www.rocq.inria.fr/arles/documents/software/wsami/WSAMIUserGuide.pdf

https://www.rocq.inria.fr/arles/documents/software/wsami/WSAMIArch.pdf

https://www.rocq.inria.fr/arles/documents/software/wsami/WSAMIDevel.pdf

6.8.10.Site officiel

https://www.rocq.inria.fr/arles/index.php/software/64-wsami-a-middleware-infrastructure-for-ambient-intelligence-based-on-web-services.html

6.9. CORTEX

6.9.1. Historique

Ce projet de recherche initié par l'IST. Développé en 2001, il a pour but de combler le fossé entre les exigences du système et les lacunes des différents middlewares et architectures.

6.9.2. Date de sortie

2001

6.9.3. Protocoles

WLAN 802.11b (ad hoc)

6.9.4. Fonctionnalités

Basé sur la théorie du modèle de programmation d'objet sensitive, ce middleware a pour objectif de faire communiquer de façon intelligente un réseau d'appareil mobile. Dans cette approche il y a trois éléments :

- Les capteurs produisent des évènements logiciels en réponse à un stimulus matériel
- Les actuateurs, ce sont des acteurs qui consomment des événements logiciels et réagissent en essayant de changer l'état du monde réel grâce aux périphériques physiques
- Les objets sensibles sont définis comme des entités qui peuvent à la fois consommer et produire des événements logiciels, il se trouve entre le capteur et l'actuateur.

Les objets sensibles peuvent communiquer entre eux, mais aussi avec les capteurs et actuateurs via un paradigme de communication basé sur les événements.

6.9.5. Domaines métiers principaux

MANET, réseau mobile

6.9.6. Couche OSI

Couche 7 – Applicative

6.9.7. Architecture interne

Modèle de communication : Cortex est basé sur une communication asynchrone non

centralisée, il utilise le pattern Publish-subscribe.

Protocole de routage performant : Le routage des mobiles sur un réseau Ad-hoc doit prendre en compte le fait du changement fréquent et rapide de la topologie du réseau. Cortex utilise un algorithme de routage probabiliste qui permet de s'adapter rapidement à de nouvelles situations.

Moteur d'inférence prise de conscience du contexte : Il est important de pouvoir dynamiquement analyser l'environnement physique et s'adapter selon les besoins. Pour cela, Cortex utilise un moteur d'inférence écrit en CLIPS.

6.9.8. Spécifications

http://cortex.di.fc.ul.pt/Deliverables/WP5-D14.pdf

6.9.9. Site officiel

http://cortex.di.fc.ul.pt/rel_research.htm

6.10.LIME

6.10.1.Historique

Ce middleware est le résultat de recherches effectuées par des personnes de l'université de Washington.

6.10.2.Version actuelle

La dernière version est la 1.1

6.10.3.Date de sortie

2006

6.10.4.Licence

LGPL

6.10.5.Protocole

LINDA

6.10.6.Fonctionnalités

C'est un middleware écrit en java. Il a pour but de simplifier la programmation d'échange d'informations sur un réseau ad-hoc dont l'un des grands problèmes est le changement constant des ressources (périphérique en mouvement, données en actions, etc.).

6.10.7.Domaines métiers principaux

MANET, réseau mobile

6.10.8.Couche OSI

Couche 7 – Applicative

6.10.9.Architecture interne

LIME utilise le principe de LINDA qui est un modèle pour le partage mémoire. Chaque donnée est représentée sous forme de données élémentaires associées à un tuple qui n'est qu'un tableau contenant un ensemble de données typées (ex : < « foo », 15 , 24.3> est un tuple). Ces tuples sont partagés en mémoire pour tous les processus qui peuvent lire, écrire et modifier les tuples. Une méthode non déterministe permet de lire un tuple ayant un format spécifique, par exemple read(« foo » , float ? , float ?) permet de lire le tuple contenant « foo » et ayant deux nombres entiers.

LIME reprends ce principe sur un réseau mobile ad-hoc, en faisant certaines modifications. Tout d'abord, il n'y a pas de place unique où peut être stockée la table de tuple. Il faut donc stocker chaque table de tuple sur le mobile ainsi que des règles de partage des tuples. La méthode de lecture non déterministe permet ainsi de spécifier dans le tuple à lire, un identifiant qui correspond au possesseur de la machine dans le réseau. Un autre ajout a été les Reactions qui permettent de spécifier des actions à réaliser lorsqu'un nouveau tuple est détecté sur les réseaux, ce qui est intéressant dans un contexte de réseau en changement constant.

6.10.10.Implémentations

LIME : Implémentation pour mobile

TINYLIME : Implémentation pour capteurs sans fil

TEENLIME : Gestion d'application sur un réseau d'appareils embarqués

6.10.11.Spécifications

http://lime.sourceforge.net/Lime/index.html

6.10.12.Site officiel

http://lime.sourceforge.net/

6.11.REDS

6.11.1.Historique

C'est un projet de recherche établi par deux professeurs de polytechnique Milan. Plusieurs versions ont été sorties : la 1.0 et 1.1.

6.11.2.Version actuelle

La version actuelle stable est la 1.3. Cependant, il existe une version expérimentale la 1.5.

6.11.3. Licence

LGPL

6.11.4.Protocoles

Il se base sur des protocoles réseau qui ne sont pas spécifiés.

6.11.5.Fonctionnalités

C'est un framework permettant de construire des applications basées sur le principe d'émetteur et récepteur pour des réseaux importants et dynamiques.

6.11.6.Domaines métiers principaux

Grâce à sa fonctionnalité qui lui permet de prendre en compte la reconfiguration du réseau, le middleware peut être utilisé dans de nombreux domaines, que ce soit en domotique ou en réseau mobile ad-hoc.

6.11.7.Couche OSI

Couche 7 – Applicative

6.11.8.Architecture interne

Middleware écrit en Java. Il utilise le pattern Publish-Subscribe.

Architecture modulaire: L'architecture permet à l'utilisateur de l'adapter facilement à ses besoins, en définissant le format pour les messages ou les mécanismes internes tels que le routage ou le suivi des messages.

Prise en compte de la reconfiguration topologique du réseau : Il est le premier Middleware à mettre en place un système qui lui permet de prendre en considération le changement de topologie du réseau pour mieux répondre aux besoins et rééquilibrer la charge du trafic ou le nombre trop important de demandes de connexions.

Support natif de l'envoi de réponses aux messages : Chaque message marquer comme « Repliable » stocke le chemin parcouru pour arriver à sa destination, pour être utilisé par la réponse. Comme le procédé est supporté nativement, il permet d'avoir une trace sur le nombre de messages réponses qui sont attendus et vérifier que tous ont été reçus.

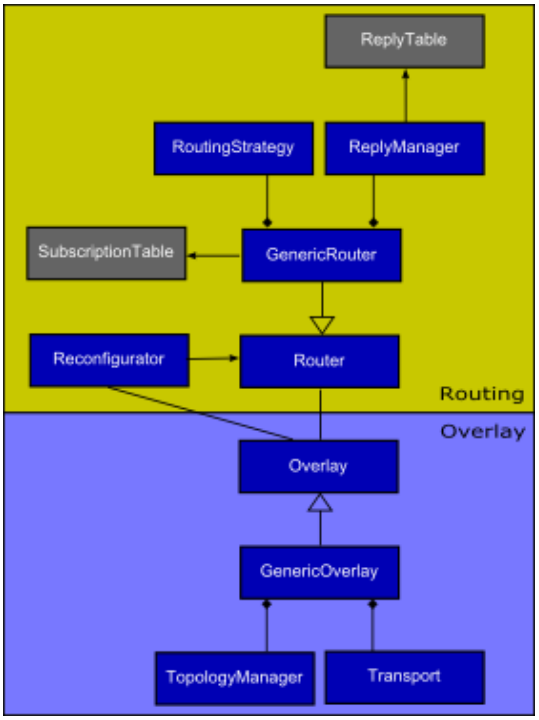

Figure 11 : Architecture interne REDS

6.11.9.Implémentation

Ce middleware a été utilisé pour implémenter un autre middleware permettant le management et la précision des machines dans l'agriculture.

Source : http://users-cs.au.dk/tpj/publications/reds.pdf

6.11.10.Site officiel

http://zeus.ws.dei.polimi.it/reds/

6.12.JACORB

6.12.1.Historique

Ce middleware date de 1995 et a été créé par deux étudiants à l'université de Berlin. Au départ, ce projet était un amusement et par curiosité. Depuis, il a évolué et est utilisé dans de nombreux domaines.

6.12.2.Version actuelle

JacORB 3.0 Beta 2 en mars 2012. La version dernière version stable est la 2.3.1 sortie en mai 2009.

6.12.3.Date de sortie

Ce middleware est sorti en 1995.

6.12.4.Licence

LGPL

6.12.5.Protocoles

- TCP/IP
- IIOP

ETF (Extensible Transport Framework for OMG) permet le support de plusieurs autres protocoles.

6.12.6.Fonctionnalités

JacOrb est un Orb, c'est-à-dire un ensemble de fonctions qui permet à des objets d'envoyer et de recevoir des requêtes et des réponses. Les objets invoqués sont le plus souvent des services. Ces envois et réceptions se font de manière transparente et portable.

6.12.7.Domaines métiers principaux

Il est utilisé dans plusieurs applications demandant d'excellentes performances et une parfaite flexibilité.

Il est par exemple utilisé dans des domaines comme :

- l'aéronautique
- la défense des télécommunications
- la santé

6.12.8.Couche OSI

Couche 7 – Applicative.

6.12.9.Architecture interne

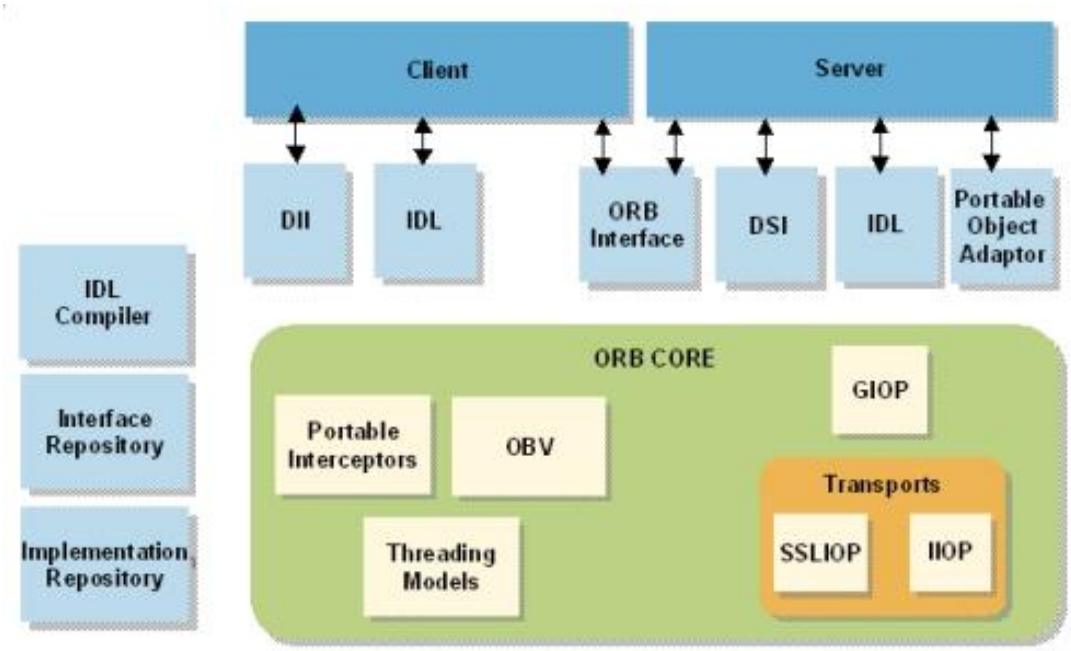

Figure 12 : Architecture interne JacORB

6.12.10.Spécifications

http://www.jacorb.org/documentation.html

6.12.11.Site officiel

http://www.jacorb.org/

6.13.TIBCO RENDEZVOUS

6.13.1.Historique

L'entreprise Tibco Software a été créée en 1997. Elle édite des solutions d'intégration et de gestion des processus métiers.

6.13.2.Version actuelle

La version actuelle est la 8.3.2

6.13.3.Date de sortie

2003

6.13.4.Licence

Propriété de la société TIBCO Software

6.13.5.Protocoles

- UDP
- TRDP(TIB Reliable Data Protocol)
- PGM(Pragmatic General Multicasting)
- AMQP(Advanced Message Queuing Protocol)

6.13.6.Fonctionnalités

Il transporte des données entre les processus ou les threads du programme. Les messages contiennent des champs de données qui se décrivent eux-mêmes. Les programmes peuvent manipuler les champs, envoyer et recevoir des messages.

6.13.7.Domaines métiers principaux

Le middleware TIBCO Rendezvous est utilisé par plus de 2000 entreprises dans tous types de domaines.

6.13.8.Couche OSI

Couche 7 – Applicative

6.13.9.Architecture interne

- Un démon (RVD) par site
- Protocole fiable de diffusion sur UDP
- Chaque démon filtre les messages qu'il doit retransmettre à ses clients
- WAN: routeurs "intelligents" de messages

6.13.10.Site officiel

http://www.tibco.com/products/soa/messaging/rendezvous/default.jsp

6.14.HORNETQ

6.14.1.Historique

HornetQ est un projet open source de messagerie asynchrone lancé par JBoss le 24 août 2009.

6.14.2.Version actuelle

La version actuelle est la 2.2.14 le 20 avril 2012

6.14.3.Date de sortie

Le 31 octobre 2009

6.14.4.Licence

Apache 2.0

6.14.5.Protocoles

- STOMP
- TCP

6.14.6.Fonctionnalités

- C'est un Middleware Orienté Message (envoi de message asynchrone).
- Il peut être utilisé sur différents systèmes d'exploitation (utilisation du Java Runtime Environment)
- Il simplifie l'utilisation de JMS (pour l'envoi de message asynchrone), avec des environnements complexes.
- Des applications abonnées consomment des messages stockés dans deux types de file de messages : Queue (un message n'est reçu que par un seul abonné) et Topic(les messages sont envoyés à tous les abonnés)
- Il peut être utilisé à partir d'un serveur d'application JEE (peut être intégré à une application stand-alone via netty)

6.14.7.Domaines métiers principaux

Ce n'est pas un middleware purement industriel. Cependant, il peut être utilisé par d'autres

middlewares afin d'en créer des spécifiques à un domaine métier.

6.14.8.Couche OSI

Couche 7 – Applicative

6.14.9.Architecture interne

- Un serveur HornetQ a son propre journal à haute performance persistante. Il l'utilise pour le message et la persistance.
- Les clients HornetQ peuvent potentiellement, à partir de machines physiques, interagir avec le serveur HornetQ.

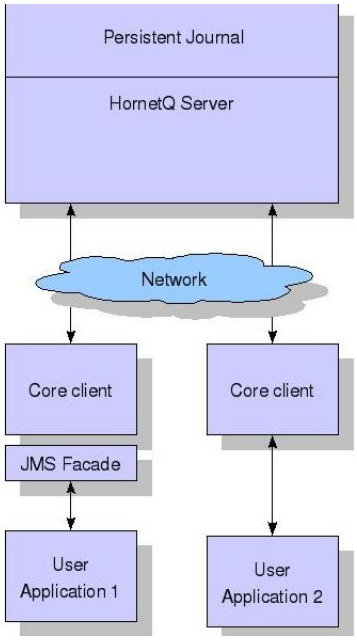

Figure 13 : Architecture interne HornetQ

6.14.10.Site officiel

http://www.jboss.org/hornetq

6.15.OCEAN

6.15.1.Historique

OCEAN (Optimization & Configuration of the Extended Access Network) est un middleware propre à l'entreprise de télécommunication SFR. Cette application a été réalisée par les équipes franco-espagnoles de la société SOPRA GROUP Division Telecom. La branche française s'est occupée de la conception du middleware alors que les équipes espagnoles se sont chargées de développer le middleware.

Aujourd'hui, la maintenance et les mises à jour sont assurées par la société WIPRO Limited.

6.15.2.Version actuelle

La version actuelle est la 5.3.2.

6.15.3.Date de sortie

Le développement a débuté en 2002 et il est officiellement sorti en 2005.

6.15.4.Licence

La société SFR est l'entière dépositaire du middleware.

6.15.5.Fonctionnalités

OCEAN est le référentiel radio de la société SFR. À l'intérieur de cette application est recensée la totalité des sites mobiles de la société de télécommunication française. On retrouve des informations diverses tels que les coordonnées géographiques du site, le type et nombre d'antennes présents sur site, etc. Cette gigantesque base de données est donc quotidiennement mise à jour en fonction des différentes opérations réalisées sur les sites par les partenaires de SFR.

6.15.6.Domaine métier principal

Télécommunication

6.15.7.Couche OSI

Couche 7 – Applicative

6.15.8.Implémentations

Portail internet

6.16.RRCAP

6.16.1.Historique

RRCAP est un middleware propre à l'entreprise de télécommunication SFR. Cette application a été réalisée par les équipes franco-espagnoles de la société SOPRA GROUP DIvision Telecom qui se sont occupées d'OCEAN. Les deux applications ont été développées en parallèle et communiquent entre elles.

Aujourd'hui la maintenance et mise à jour du site est assurée par la société WIPRO Limited.

6.16.2.Version actuelle

La version actuelle du middleware est la version 4.1.

6.16.3.Date de sortie

Le développement a débuté en 2002 et il est officiellement sorti en 2005.

6.16.4.Licence

La société SFR est l'entière dépositaire du middleware.

6.16.5.Fonctionnalités

RRCAP est utilisé dans le cadre de la modélisation et le rattachement virtuels des sites et antennes SFR aux différents BSC de l'opérateur. Ce lien permet de visualiser les ponts de connexion qu'il existe entre les différents sites et l'émetteur principal de la zone géographique.

6.16.6.Domaine métier principal

Télécommunication

6.16.7.Couche OSI

Couche 7 – Applicative

6.16.8.Implémentations

Portail internet.

6.17.ASPIRE

6.17.1.Historique

Le projet Aspire est un projet collaboratif de l'UE né en janvier 2008. Cette équipe était composée d'universitaires français, portugais, anglais, danois, belge et suisse. Cette association multiethnique a passé 36 mois pour développer le middleware ASPIRE.

Le projet ASPIRE va changer le paradigme actuel de déploiement de radios identification, grâce à l'introduction et le renforcement de l'orientation d'identification radio vers le libre de middleware. Dans ce paradigme, la solution place le middleware aux mains des utilisateurs finaux (en particulier les PME). En conséquence, le middleware peut s'intégrer à faible coût, grâce aux infrastructures IT et de réseautage de l'entreprise. Pour soutenir ce paradigme, ASPIRE développe et fournit une plate-forme middleware légère, libre, programmable, conviviale, conforme aux normes, évolutive, intégrée et intelligente qui facilite un faible coût de développement et le déploiement de solutions innovantes d'identification radio entièrement automatique.

6.17.2.Version actuelle

La version est la 0.5.1.

6.17.3.Date de sortie

Il est sorti en 2010.

6.17.4.Licence

Licence LGPL V2.1.

6.17.5.Protocole

RFID

6.17.6.Fonctionnalités

Le projet Aspire vise à développer et à promouvoir un middleware open source intégrable avec plusieurs outils afin de faciliter le développement, le déploiement et la gestion des applications basées sur l'identification de fréquence radio et de capteurs d'applications. Il met en œuvre plusieurs spécifications de consortiums tels qu'EPC Global, NFC Forum, JCP et OSGi Alliance.

Aspire fournit également un ensemble d'outils permettant aux consultants RFID de déployer des solutions sans avoir besoin de programmation fastidieuse. Aspire permet la spécification

des processus RFID. En conséquence, les outils génèrent tous les artefacts nécessaires pour déployer ces solutions.

6.17.7.Domaine métier principal

Il est utilisé dans le cadre de la radio-identification.

6.17.8.Couche OSI

Couche 7 – Applicative

6.17.9.Architecture interne

L'architecture spécifie l'existence d'un IDE permettant la gestion visuelle de tous les fichiers de configuration et des métadonnées qui sont nécessaires pour l'exploitation d'une solution RFID. Il s'agit notamment de:

- L'éditeur de lecteur physique de configuration pour configurer les lecteurs physiques et leurs paramètres opérationnels et les environnements.
- L'éditeur de configuration du lecteur logique qui sera en charge de la définition de lecteurs logiques (LLRP, RP, HAL et lecteurs Simulator).
- L'éditeur de lecture des spécifications qui permettront l'édition, ainsi que la gestion du serveur F & C Filtrage Spécifications.
- Le F & C des commandes d'exécution qui aura comme objectif de fournir un client de contrôle des commandes sur un lecteur ou d'un composant qui implémente la spécification ALE.
- Le connecteur qui sera capable d'interagir avec l'application pour révéler toutes ses fonctionnalités et configurations. Il devrait permettre la configuration des connecteurs pour les différents systèmes et bases de données.
- L'éditeur de Master Data qui permettra aux utilisateurs et / ou aux consultants de modifier les données d'entreprise, y compris les informations sur l'emplacement de l'entreprise, ses lieux d'affaires, ainsi que ses processus.
- Le 'Workflow Management' éditeur qui peut offrir, aux utilisateurs professionnels et les consultants RFID, une interface utilisateur graphique pour la manipulation des processus d'affaires complexes. L'outil de gestion de flux de travail tirera parti de la fonctionnalité de ces outils afin de soutenir les déploiements RFID conformément aux processus d'affaires particuliers.

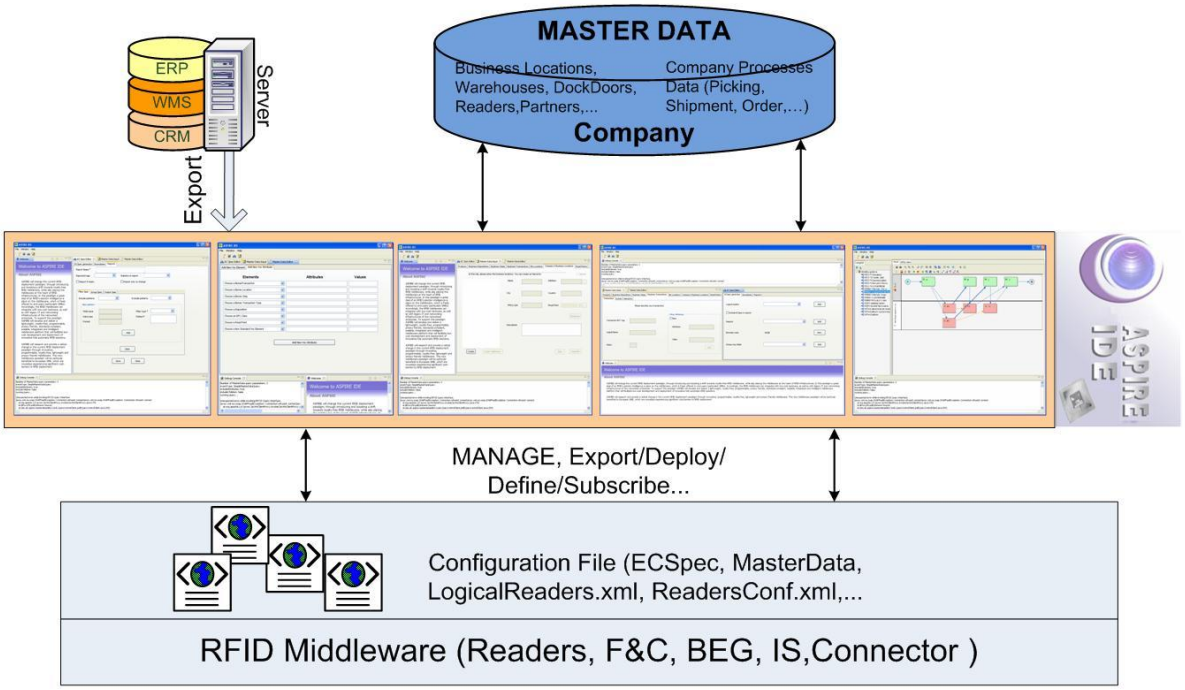

Figure 14 : Architecture interne ASPIRE

6.17.10.Spécifications

http://wiki.aspire.ow2.org/xwiki/bin/view/Main/Documentation

6.17.11.Sites officiels

- Forge: http://forge.ow2.org/projects/aspire
- ASPIRE Project: http://www.fp7-aspire.eu
- LinkedIn group: http://www.linkedin.com/groups?gid=1179007
- The AspireRFID Moodle: http://moodle.fp7-aspire.eu

6.18. WISREED

6.18.1.Historique

Créé par l'entreprise FUJITSU WisReed est un logiciel commercial permettant la communication avec des réseaux de capteur intelligent.

6.18.2.Licence

Propriété du groupe FUJISTU

6.18.3.Protocoles

Le protocole Zigbee avait été pensé pour le modèle Ad-hoc mais ne correspondait pas aux besoins de l'application. Fujitsu a créé un protocole maison pour y répondre.

Cependant, WisReed permet l'interconnexion avec différents réseaux autre que le protocole créé pour WisReed, comme ZigBee, Bluetooth ou TCP/IP grâce à sa partie middleware.

6.18.4.Fonctionnalités

WisReed peut être découpé en deux parties, une partie protocole gérant un réseau de communication entre différents appareils en mode de communication Ad-hoc et une partie middleware permettant l'échange d'informations entre capteurs et services de collecte d'informations.

Le WisReed Protocole pour la communication en Ad-hoc permet de répondre aux besoins liés à un réseau de capteur intelligent, c'est-à-dire :

- Construction d'un réseau à grande échelle, les capteurs pouvant se trouver très éloignés et être nombreux.
- Un réseau ayant une très grande fiabilité, les données traitées ayant une très grande importance.
- Rendre les opérations d'ajout et de suppression d'appareil sur le réseau moins pénible, en effet le réseau doit pouvoir s'adapter à n'importe quel type d'infrastructure.

La partie Middleware est composée de trois fonctions :

- Un protocole d'abstraction : Permets d'utiliser différents protocoles tels que Zigbee et TCP de façon uniforme.
- L'abstraction du réseau : Un numéro d'identifiant pour accéder à un appareil est défini par WisReed afin d'avoir une abstraction des différents réseaux interconnectés.
- Une API pour le middleware.

Des protocoles peuvent être ajoutés pour être pris en charge par WisReed grâce à un format de plug-in spécifique. La correspondance entre le réseau utilisé et le protocole est maintenue dans une table de conversion.

6.18.5.Domaine métier principal

Le domaine principal est le domaine de l'énergie, des capteurs intelligents, des systèmes de compteur AMI système (compteur « intelligent ») ou encore la gestion de la consommation électrique des Datacenters.

6.18.6.Couche OSI

Couche 7 – Applicative

6.18.7.Architecture interne

Pour le routage des informations, les protocoles Ad-hoc existants utilisent une méthode qui consiste à envoyer une demande de chemin sur l'ensemble du réseau avant d'envoyer le message à sa destination finale. Ceci n'est pas recommandé sur un très large réseau. De plus le nombre de réponses pour le chemin à parcourir peut-être très important dû au fait que le réseau n'est pas stable (ajout, suppression d'appareils). C'est pour cela que WisReed utilise une autre méthode qui consiste à parcourir le réseau par un parcours en profondeur, chaque terminal envoie un message « Hello » qui permet de savoir qui sont ses voisins, il constitue ainsi une table de routage. Lorsqu'un appareil veut envoyer un message, il va l'envoyer à son premier voisin de sa table si le message n'est pas à destination d'un de ses voisins. Lorsqu'un appareil reçoit un message, soit ce message est pour lui et il le garde soit il n'est pas pour lui et alors il va l'envoyer à son premier voisin. Si le message remonte jusqu'à lui, il pourra essayer un autre voisin.

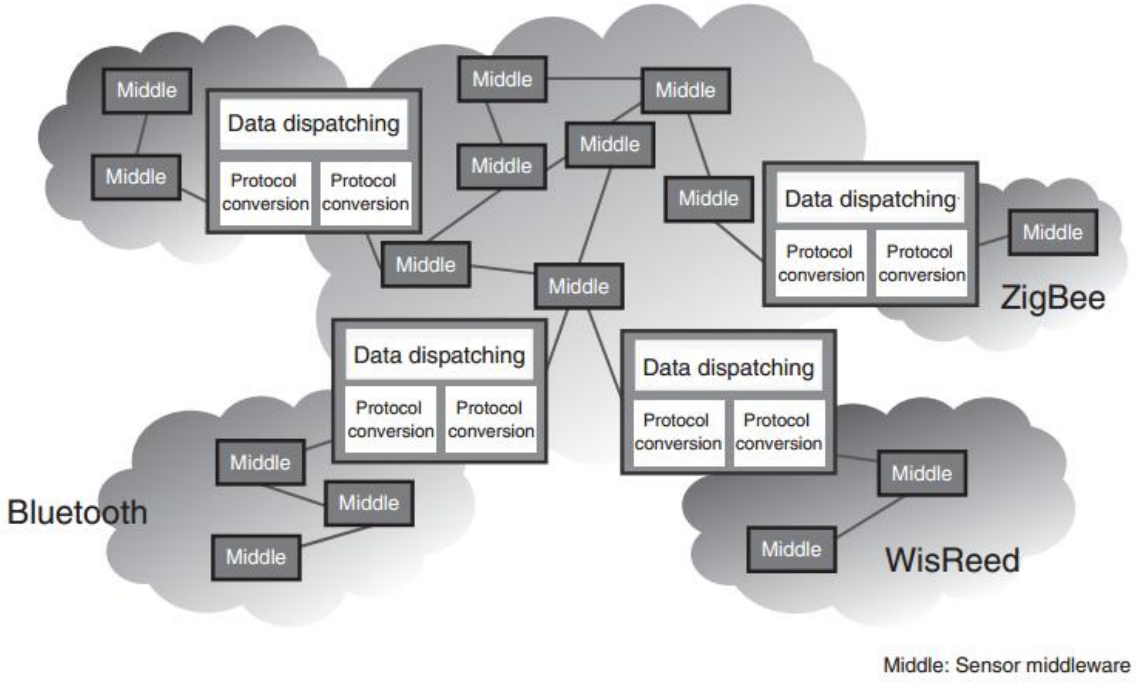

Figure 15 : Architecture prise en charge par WisReed

Sur l'image ci-dessus, on voit ce que pourrait être un réseau utilisé par WisReed. On a 3 types de réseau : un réseau ad-hoc WisReed, Zigbee et Bluetooth. Chaque réseau, en interne, utilise leur propre protocole et/ou middleware. Lorsque les données doivent être envoyées d'un réseau à un autre, l'application WisReed permet de traduire le message du protocole interne au protocole adjacent correspondant au réseau (par exemple on peut passer du Bluetooth à TCP/IP et ensuite passer de TCP/IP à Zigbee. WisReed s'occupera de transmettre correctement l'ensemble des informations nécessaires aux différents réseaux). La partie réception qui contiendra à son tour une partie applicative WisReed traduira dans le protocole réseau auquel il est attaché.

Le middleware permet ainsi de récupérer diverses informations à partir de différents capteurs (AMI, Gestion du niveau de l'eau, gestion du fonctionnement de l'air conditionné d'un immeuble, etc.) qui pourront être tous regroupés dans un DataCenter et permettre ainsi leur observation.

Le middleware possède des fonctionnalités pour gérer les capteurs tels que :

- L'ajout d'un capteur sur le réseau simplement en le branchant.
- Le filtre et la transmission uniquement des données des capteurs que l'on veut (utilisation de règle, par exemple transmettre l'information d'un capteur de température s'il dépasse un seuil)
- Le formatage des données récupérées par les capteurs avant de les envoyer (calculer des moyennes ou d'autres informations statistiques)
- L'utilisation des capteurs virtuels. Un capteur virtuel est une combinaison des capteurs de différents types ou se trouvant dans différents lieux qui sont reconnus comme étant une seule et unique entité. La combinaison permet d'obtenir un indice qui sera en fonction de différentes règles et des données récupérées par les capteurs associés.
- La gestion des erreurs du réseau. Le logiciel permet de prévenir directement l'opérateur dans le cas où un capteur ne fonctionne plus. Il permet aussi de détecter la saturation de la bande passante ou différents problèmes liés à la qualité du réseau.

6.18.8.Site officiel

http://www.fujitsu.com/global/services/solutions/sensor-network/about/

6.19.OPENSOAP

6.19.1.Historique

Le projet Open Soap a commencé dans le but de développer un middleware open source basé sur le standard SOAP.

6.19.2.Version actuelle

OpenSoap V 2.0.1.

6.19.3.Date de sortie

Le 2 avril 2002.

6.19.4.Licence

6.19.5.Protocoles

- HTTP
- SMTP

6.19.6.Fonctionnalités

SOAP est une abréviation pour Simple Object Access Protocol. C'est une spécification proposée par IBM au W3C qui aurait pour but d'interconnecter tous les web services distribués.

6.19.7.Domaine métier principal

Ce middleware est de type généraliste et peut être implémenté dans n'importe quel domaine métier.

6.19.8.Couche OSI

Couche 7 – Applicative

6.19.9.Architecture interne

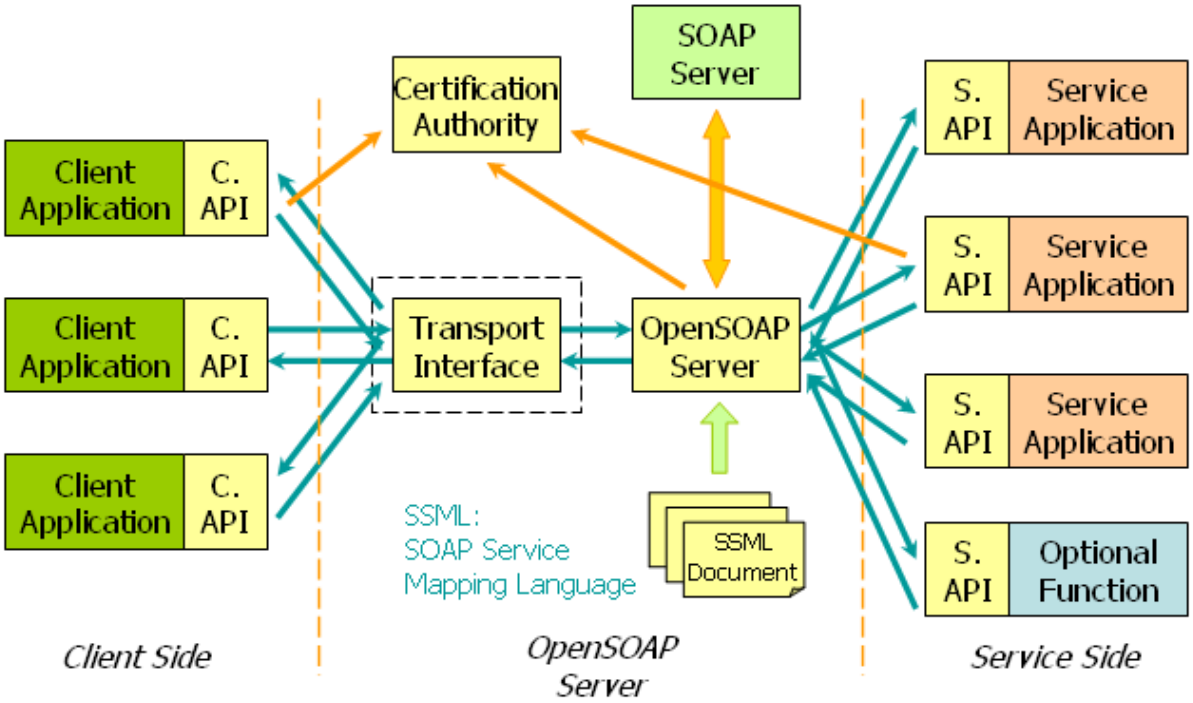

Figure 16 : Architecture interne OpenSOAP

6.19.10.Site officiel

http://www.opensoap.jp/doc/index.html.en

6.20. APACHE ACTIVEMQ

6.20.1. Historique

Il s'agit d'un middleware qui implémente le Java Message Service 1.1. Il fait partie des MOM Open Source les plus utilisés. Sa force repose sur la grande variété de langages et de protocoles supportés.

6.20.2. Version actuelle

La version actuelle est la 5.6.0, sortie le 7 mai 2012.

6.20.3. Date de sortie

La version 1.1 release date de 2000 et la version actuellement utilisée date quant à elle de 2004.

6.20.4. Licence

Apache License 2.0.

6.20.5. Protocoles

- OpenWire
- REST
- STOMP
- WS Notification
- XMPP
- AMQP
- VM
- TCP
- NIO
- SSL
- NIO SSL
- UDP
- Multicast
- HTTP/HTTPS
- WebScokets
-

6.20.6.Fonctionnalités

- Support complet pour les EIP à la fois dans le client JMS et le Message Broker.
- Prise en charge de nombreuses fonctionnalités avancées telles que le regroupement de messages, les destinations virtuelles, les jokers et les destinations composites.
- Compatibilité JMS 1.1 et J2EE 1.4 avec prise en charge transitoire, persistante, transactionnel et XA messagerie.
- Prise en charge de Spring de telle sorte qu'ActiveMQ peut être facilement intégré dans des applications de Spring et configuré en utilisant le mécanisme de configuration XML.
- Compatible avec les serveurs J2EE populaires tels que Geronimo, JBoss 4, GlassFish et WebLogic.
- Conception orientée clustering de haute performance basée sur la communication client-serveur.

6.20.7.Couche OSI

Couche 7 – Applicative

6.20.8.Architecture interne

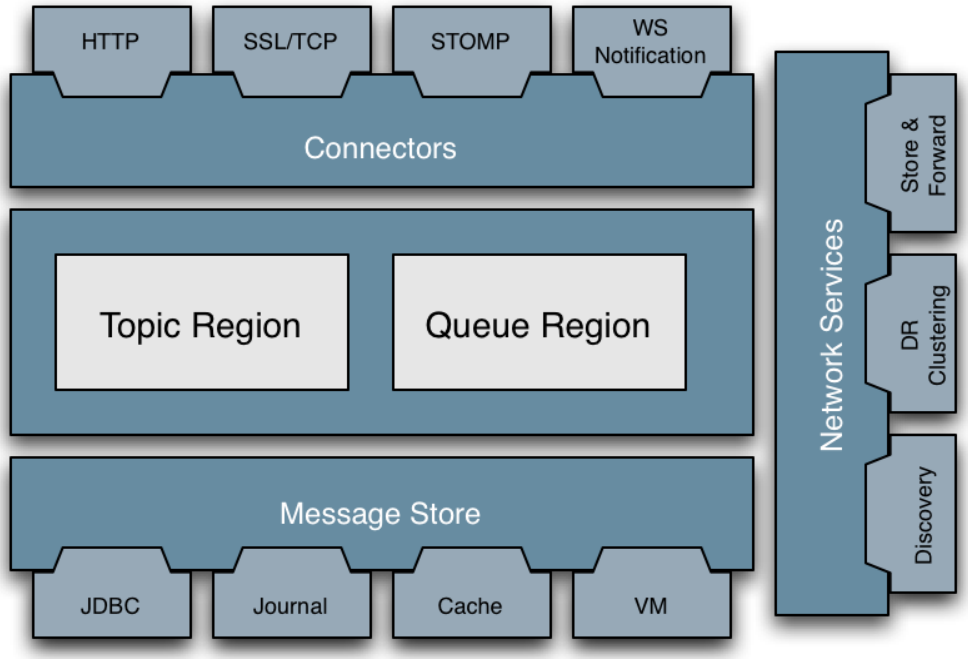

Figure 17 : Architecture interne Apache ActiveMQ

6.20.9.Spécifications

http://activemq.apache.org/getting-started.html

6.20.10.Site officiel/Forum

http://activemq.apache.org/

http://activemq.apache.org/discussion-forums.html

6.21. ORACLE STREAMS ADVANCED QUEUING

6.21.1. Historique

Oracle Streams AQ est inclus de base avec le système de gestion de base de données depuis la sortie de la version 8, en juin 1997. Il était connu sous le nom Oracle Advanced Queuing avant son intégration dans Oracle Streams depuis la version 10.1.

6.21.2. Version actuelle

La version actuelle est la 11g release 2.

6.21.3. Date de sortie

1997

6.21.4. Licence

Propriété de la société ORACLE.

6.21.5. Protocoles

- HTTP
- HTTPS
- SMTP
- SOAP

6.21.6. Fonctionnalités

Oracle Streams AQ fournit une base de données intégrant la fonctionnalité d'envoi et réception de messages. Ce dernier est construit à partir d'Oracle Streams et tire parti des fonctionnalités de base de données Oracle. Ainsi, les messages peuvent être stockés, diffusés et transmis via Oracle Net Services et HTTP (S) entre les files d'attente sur des ordinateurs et bases de données différents.

Parce qu'Oracle Streams AQ est construit à partir de tableaux de base de données, tous les avantages opérationnels de disponibilité, d'évolutivité et de fiabilité propre à Oracle BDD sont également applicables aux données des files d'attente. De plus, les caractéristiques de base des BDD standards telles que la récupération, le redémarrage et la sécurité sont prises en charge par Oracle Streams AQ. Il est également possible d'utiliser l'outil de développement de BDD et de gestions tel qu'Oracle Enterprise Manager afin de surveiller les files d'attente de messagerie. Enfin, il est également possible d'importer ou exporter les files d'attente de messagerie.

Les messages peuvent être interrogés à l'aide du standard SQL. Cela signifie qu'il est possible d'utiliser le langage SQL pour accéder aux propriétés du compte de messagerie et des messages, à l'historique de messagerie, etc. Il est également possible d'auditer les files d'attente et d'utiliser les propriétés de la technologie SQL, tel que l'index, pour optimiser l'accès aux messages.

6.21.7.Couche OSI

Couche 7 – Applicative

6.21.8.Architecture interne

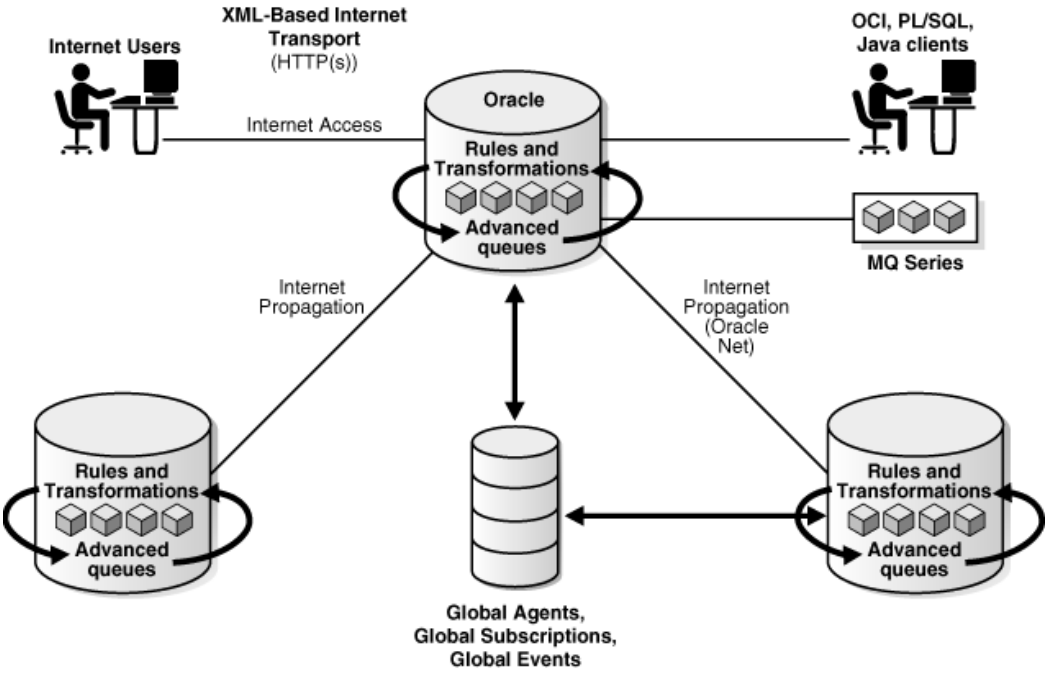

Figure 18 : Architecture globale d'Oracle Streams AQ

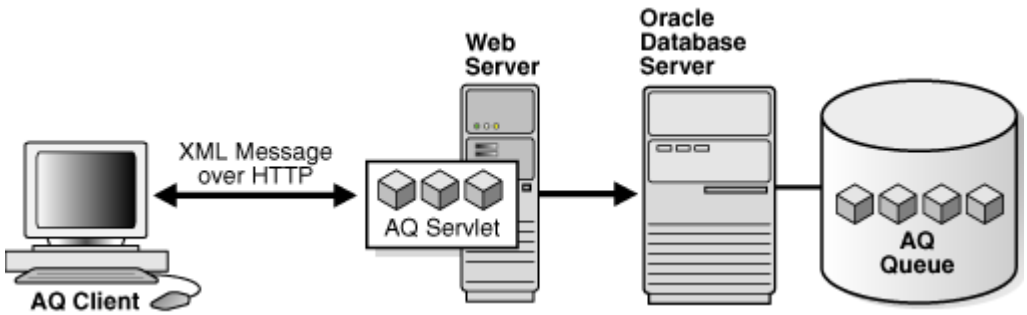

Figure 19 : Architecture via http d'Oracle Streams AQ

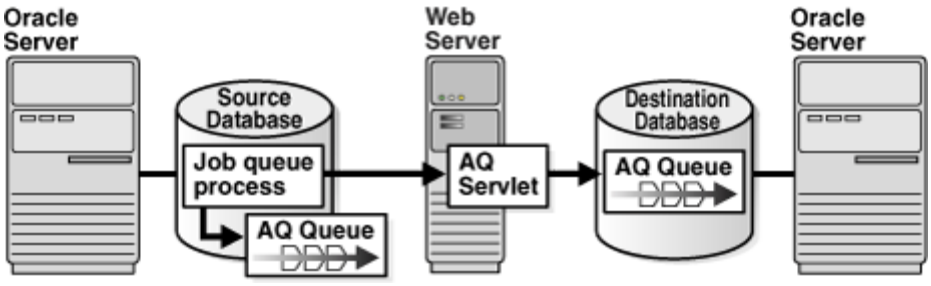

Figure 20 : Propagation http d'Oracle Streams AQ

6.21.9.Spécifications

Version 9.2 : http://docs.oracle.com/cd/B10500_01/appdev.920/a96587/qintro.htm

Version 10.2 : http://docs.oracle.com/cd/B19306_01/server.102/b14257/aq_intro.htm

6.21.10.Site officiel

http://www.oracle.com

6.22.STORMMQ

6.22.1.Historique

Développé en Java mais supporté par d'autres langages, StormMQ a l'avantage d'être adaptable selon le type de structure dans laquelle il est utilisé. En effet, la plate-forme peut être utilisée selon différentes configurations :

- Sur le cloud pour une petite structure désireuse de minimiser les coûts
- Sur des serveurs dédiés à l'extérieur ou directement sur site

6.22.2.Version actuelle

La version actuelle est la 2010.05.20.1859.RC, sortie le 20 mai 2010.

6.22.3.Date de sortie

2010

6.22.4.Licence

Mozilla Public License

6.22.5.Protocole

AMQP

6.22.6.Fonctionnalités

StormMQ est un service de Message Queuing (une forme de middleware orientée messagerie qui fournit un message broker), en utilisant la norme Advanced Message Queuing Protocol (AMQP). StormMQ fournit une plateforme ouverte, sûre et les protections nécessaires pour les données qui transitent par le cloud. La plate-forme est livrée en deux formats, soit comme un service cloud basé sur les utilisateurs à faible volume, soit un service de colocalisation ou hébergé pour les organisations ayant des exigences opérationnelles strictes.

Message Queuing (MQ) est un mécanisme qui permet de distribuer efficacement et en toute sécurité de grandes quantités de données entre plusieurs ordinateurs, sans surcharge d'un serveur.

6.22.7.Domaines métiers principaux

- Automobile
- Energie

6.22.8.Couche OSI

Couche 7 –Applicative

6.22.9.Architecture interne

L'architecture interne de StormMQ n'est pas disponible au grand public il est propriétaire. Cependant, on peut déduire la structure de ce dernier grâce à l'architecture commune au middleware basé sur les protocoles AMQP.

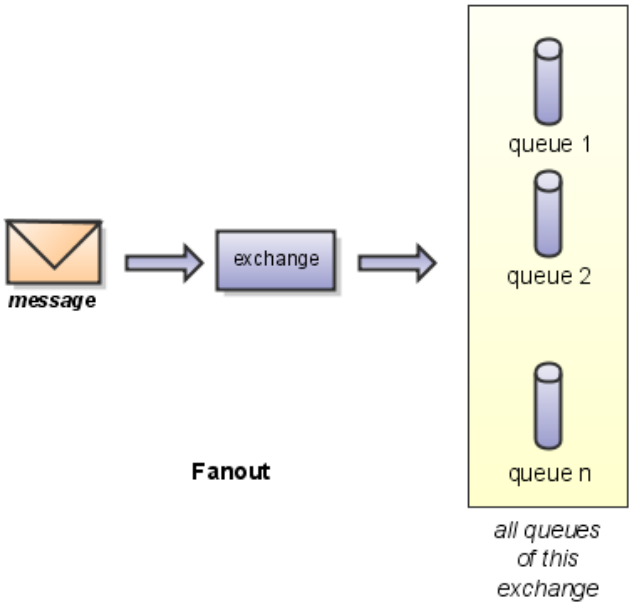

Figure 21 : Architecture AMQP

6.22.10.Site officiel

http://stormmq.com/

7. CONCLUSION

Ce document avait pour but de lister et répertorier les middlewares et les protocoles spécifiés dans ceux-ci. Nous avons pu voir que le choix d'un middleware doit se faire selon différents besoins, comme le domaine d'activité dans lequel il sera employé, le support physique utilisé et d'autres spécifications tels que l'architecture du réseau à concevoir et sa complexité. De plus, une taxinomie a été réalisée suivant différents critères pour en déduire certaines interopérabilités qui pouvaient exister.

FIN DU DOCUMENT