Les middlewares

Introduction

Le domaine de l'intergiciel (middleware), apparu dans les années 1990, a pris une place centrale dans le développement des applications informatiques réparties. L'intergiciel joue aujourd'hui, pour celles-ci, un rôle analogue à celui d'un système d'exploitation pour les applications centralisées.

Les acteurs de l'intergiciel sont nombreux et divers :

- les organismes de normalisation
- les industriels du logiciel et des services
- les utilisateurs d'applications.

Definition

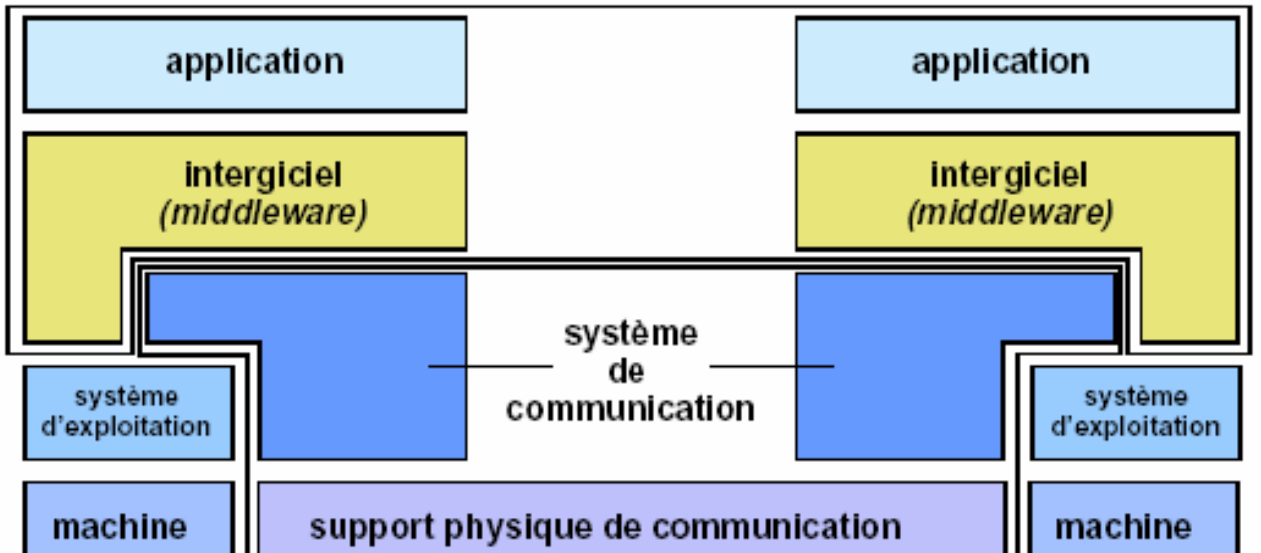
Le middleware est une couche intermédiaire entre le système d'exploitation et l'application afin d'en améliorer sa mise en oeuvre :

L'objectif principal du middleware est d'unifier, pour les applications, l'accès et la manipulation de l'ensemble des services disponibles sur le réseau, afin de rendre l'utilisation de ces derniers presque transparente.

Definition d'un middleware

Le Gartner Group définit le middleware comme **une interface** de **communication universelle** entre processus. Il représente véritablement la clef de voûte de toute application client-serveur.

L'objectif principal du middleware est d'unifier, pour les applications, l'accès et la manipulation de l'ensemble des services disponibles sur le réseau, afin de rendre l'utilisation de ces derniers presque transparente.



Organisation: importance de la normalisation

Le middleware = intergiciel est une classe de logiciels systèmes agissant en qualité d'infrastructure pour le développement et le déploiement d'applications réparties : exp. CORBA

Le développement du middleware impose une normalisation des interfaces

- Logiciel de base
- Domaine spécifiques d'applications

Objectifs de middleware

Un intergiciel vise à :

1. **cacher la répartition :** c'est à dire le fait qu'une application est constituée de parties interconnectées s'exécutant à des emplacements géographiquement répartis ;
2. **cacher l'hétérogeneité des composants matériels,** des systèmes d'exploitation et des protocoles de communication utilisés par les différentes parties d'une application ;
3. **fournir des interfaces uniformes,** normalisées, et de haut niveau aux équipes de développement et d'intégration, pour faciliter la construction, la réutilisation, la portabilité et l'interopéerabilité des applications ;
4. **fournir un ensemble de services communs** réalisant des fonctions d'intérêt général, pour éviter la duplication des efforts et faciliter la coopération entre applications.

Services rendus

Un middleware est susceptible de rendre les services suivants :

•**Conversion :** Service utilisé pour la communication entre machines mettant en oeuvre des formats de données différents .

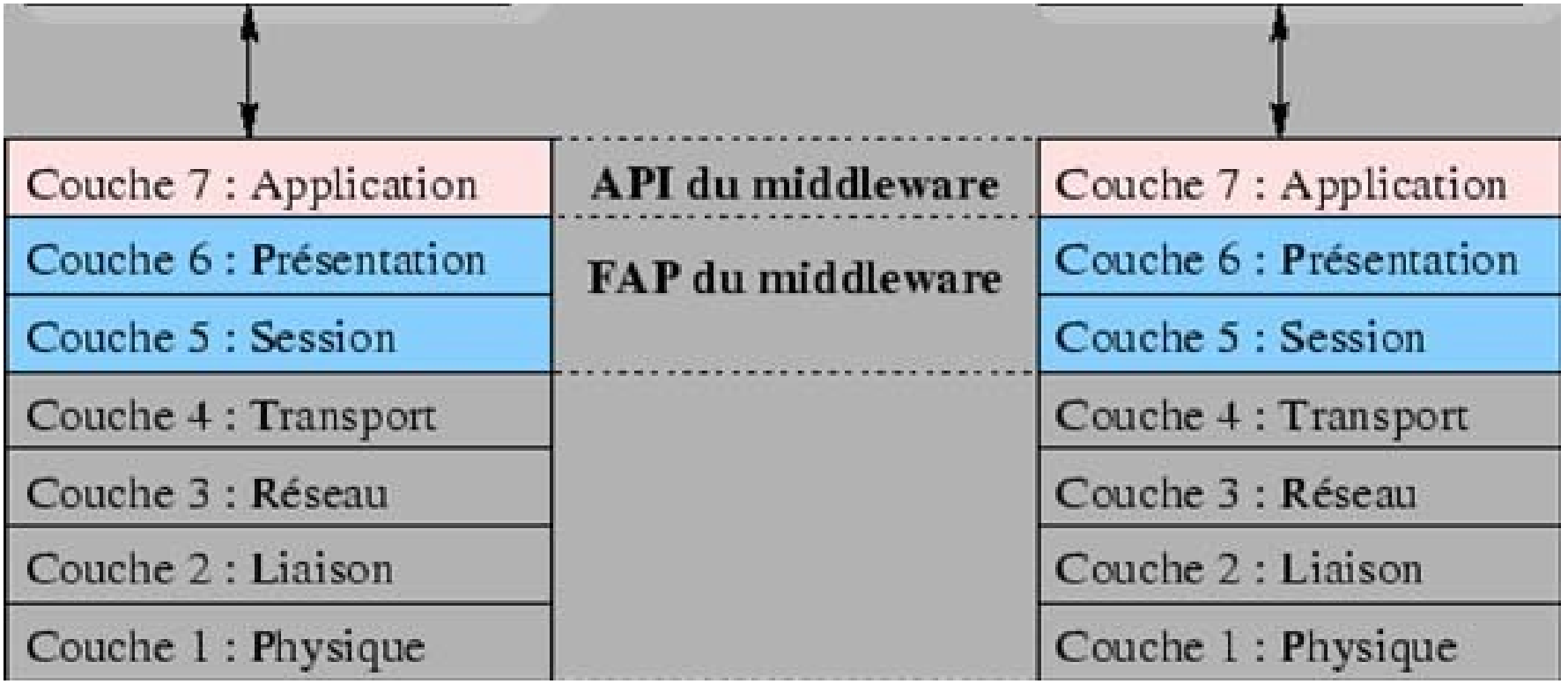
•**Adressage :** Permet d'identifier la machine serveur sur laquelle est localisé le service demandé afin d'en déduire le chemin d'accès. Dans la mesure du possible, cette fonction doit faire appel aux services d'un annuaire.

•***Sécurité :*** Permet de garantir la confidentialité et la sécurité des données à l'aide de mécanismes d'authentification et de cryptage des informations.

•**Communication** : Permet la transmission des messages entre les deux systèmes sans altération. Ce service doit gérer la connexion au serveur, la préparation de l'exécution des requêtes, la récupération des résultats et la déconnexion de l'utilisateur.

Fonctionnement d'un middleware

Le middleware masque la complexité des échanges inter-applications et permet ainsi d'élever le niveau des API utilisées par les programmes. Sans ce mécanisme, la programmation d'une application client-serveur serait extrêmement complexe et rigide. Le middleware se situe dans le modèle OSI au dessus de la couche de transport (couches 5, 6 et 7).



Fonctionnement d'un middleware

La double mission d'interfaçage du middleware est :

•**La communication entre le processus client et serveur** : la gestion des appels de fonctions de l'application ou la gestion du renvoi des résultats.

•**La mise en forme des données** en vue de leur prise en charge par la couche transport.

Les deux missions sont assurées par deux composants distincts :

• **Le Protocole d'Accès Formaté** (Format And Protocol, FAP) met en forme les différentes données au niveau du réseau.

•**L'interface de programmation (Application Programming Interface, API) se charge :**

- des connexions et déconnexions avec le serveur ;
- de la définition de l'environnement de la connexion (variables de contexte, zones tampon) ; et
- du transfert des requêtes et de la réception des résultats.

Fonctionnement d'un middleware

Les deux missions du miidleware se déroulent comme suit :

- L'interface de programmation transmet au FAP les requêtes destinées au serveur qui va se charger de conditionner les données au transport par le réseau.
- Le FAP est propre à chaque protocole réseau.
- Le FAP du client reçoit la requête et la plie dans une trame destinée au transport sur le réseau.
- Le FAP du serveur reçoit la trame, la déplie et transmet la requête à l'interface. Après traitement, le serveur renvoie le résultat de la requête à l'interface qui transmettra au client via les FAP du serveur, puis du

Un intergiciel peut être à usage général ou dédié à une classe particulière d'applications.

exemples de middleware

On cite plusieurs exemples de middleware, à savoir :

- **SQL*Net :** Interface propriétaire permettant de faire dialoguer une application cliente avec une base de données Oracle. Ce dialogue peut aussi bien être le passage de requêtes SQL que l'appel de procédures stockées.
- **ODBC :** Interface standardisée isolant le client du serveur de données. C'est l'implémentation par Microsoft du standard CLI défini par le SQL Access Group. Elle se compose d'un gestionnaire de driver standardisé, d'une API s'interfaçant avec l'application cliente (sous Ms Windows) et d'un driver correspondant au SGBD utilisé.
- **DCE :** Permet l'appel à des procédures distantes depuis une application.

Principales catégories de middlewares

On distingue cinq catégories de middlwares à savoir :

- Intergiciels à messages (MOM)
- Intergiciels de bases de données (ODBC)
- Intergiciels à appels de procédure distante (DCE)
- Intergiciels à objets répartis (CORBA, JAVA RMI)
- Intergiciels à composants (EJB, CCM, Web Services)

Intergiciels à messages (MOM : Message-Oriented Middleware)

L’intergiciel à messages sont basés sur la réalisation d’une file d'attente permettent une communication entre différents systèmes informatiques, connectant plusieurs applications. Ce système de synchronisation de messages fournit typiquement une fonctionnalité de persistance pour s'assurer que les messages ne soient pas perdus en cas d'échec du système.

Les files d'attente de message fournissent des liaisons asynchrones normalisées. Elles ont pour but que l'expéditeur et le récepteur du message ne soient pas contraints de s'attendre l'un l'autre. Des messages placés dans la file d'attente sont stockés, jusqu'à ce que le destinataire les recherche. L'expéditeur n'a pas à attendre que le récepteur commence à traiter son message, il poste son information et peut passer à autre chose.

Ex :

MQ Series d’IBM et MSMQ (Microsoft Message Queues Server) de Microsoft

Intergiciels de bases de données (ODBC)

Les middlewares proposés par les fournisseurs de SGBD sont très performants et permettent d tirer profit de l'ensemble des fonctionnalités du serveur de données pour lequel ils ont été conçus. Par contre, ils ne permettent pas, le plus souvent, l'accès à d'autres sources de données.

Pour certaines applications devant accéder à des services hétérogènes, il est parfois nécessaire de combiner plusieurs middlewares. Dans ce cas, le poste client doit connaître et mettre en oeuvre plusieurs IPC, on en vient à la notion de client lourd.

Intergiciels à appels de procédure distante (DCE)

L'appel de procédure à distance que l'on appel **RPC** **(Remote Procedure Call)**, est un mécanisme indispensable au développement d'applications réparties. En effet, celui-ci permet d'appeler des procédures situées sur des machines différentes.

Il se base lui aussi sur le schéma fondamental du **Client/Serveur.** Lorsqu'un programme client fait appel à une procédure distante, l'exécution de celle-ci se poursuit dans le **programme serveur** où se trouve l'implémentation.

Le résultat est ensuite retourné au client.

La gestion des appels et les transferts de données entre le client et le serveur doivent être invisibles.

Intergiciels à objets répartis

Pour permettre la répartition **d'objets** entre machines et l'intégration des systèmes **non objets,** il doit être possible d'instaurer une **communication entre tous ces éléments.**

Ces middlewares sont constitués d'une série de mécanismes permettant à **un ensemble de programmes d'interopérer de façon transparente**. Les services offerts par les applications serveurs sont présentés aux clients sous **la forme d'objets**. **La localisation et les mécanismes mis en oeuvre pour** cette interaction sont cachés par le middleware.

Ex:

- Le système CORBA (Common Object Request Broker Architecture)

- (D)COM / OLE / Active X (Distributed Component Object Model / Object Linking and Embedding) de Microsoft pour les plate-formes de type Windows.

- JAVA RMI

Intergiciels à composants

Un **composant** est un **module logiciel autonome et réutilisable.** On peut représenter un composant comme un type de boîte noire (la connaissance de son implémentation n'est pas nécessaire) constitué d'un ou plusieurs objets, où il suffit de connaître les services rendus et les quelques règles d'interconnexion. En effet, comme nous l'illustre la figure ci dessous, un composant exporte les interfaces qu'il fournit et les interfaces qu'il requiert.

De plus, le composant est interconnectable avec d'autres composants d'origines diverses, configurable, auto descriptif (introspection), mais aussi, il est diffusable de manière unitaire et prêt à l'emploi. La plupart de ces fonctionnalités sont dues à ses interfaces.

EX:

Java Beans

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction