Le modèle client-serveur

Olivier Aubert

Sources

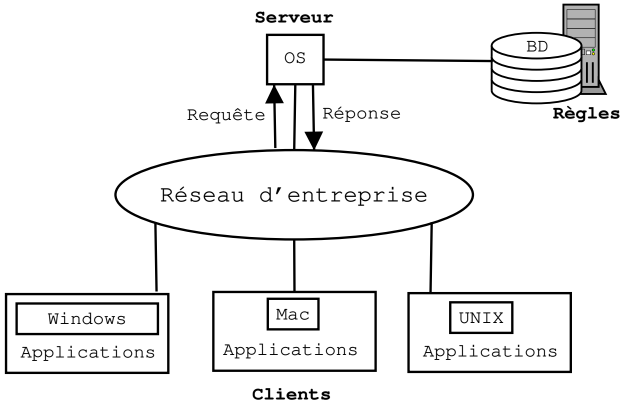
- `http://www.info.uqam.ca/~obaid/INF4481/A01/Plan.htm`

Historique

- architecture centralisée
- terminaux passifs (un seul OS, systèmes propriétaires)
- traitements au niveau du serveur

Architecture répartie

- réseaux, ordinateurs plus puissants, OS ouverts
- interfaces et API standard (RFC)
- traitement effectué en partie sur les clients

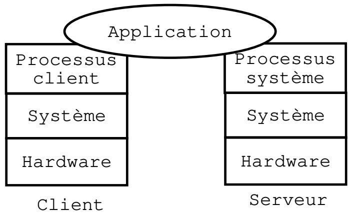


Le modèle

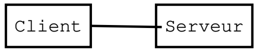
- *Client* : processus demandant l'exécution d'une opération à un autre processus par envoi de message contenant le descriptif de l'opération à exécuter et attendant la réponse de cette opération par un message en retour.
- *Serveur* : processus accomplissant une opération sur demande d'un client, et lui transmettant le résultat.
- *Requête* : message transmis par un client à un serveur décrivant l'opération à exécuter pour le compte du client.
- *Réponse* : message transmis par un serveur à un client suite à l'exécution d'une opération, contenant le résultat de l'opération.

Types d'architecture client-serveur

Le modèle :

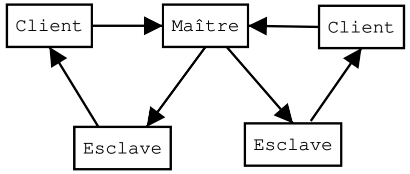


Un client, un serveur :



Un client, plusieurs serveurs :

Plusieurs clients, un serveur :



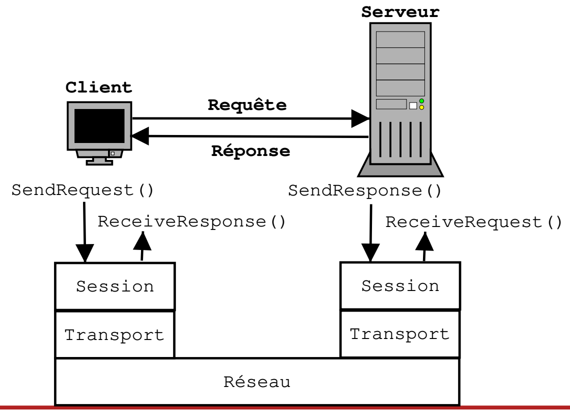
C/S orienté client ou serveur

- Client lourd
  - stocke les données et les applications localement. Le serveur stocke les fichiers mis à jour
  - Le client effectue une bonne partie du traitement
  - Le serveur est plus allégé
- Serveur lourd
  - On effectue plus de traitements sur le serveur : transactions, groupware, etc
  - Déploiement plus aisé
- Client léger
  - Client à fonctionnalité minimale (terminaux X, périphérique réseau (Network Appliance), ordinateur réseau (network computer))
  - Beaucoup de charge sur le serveur et le réseau

Dialogue client-serveur

Primitives de service:

- SendRequest()
- ReceiveResponse()
- ReceiveRequest()
- SendResponse()

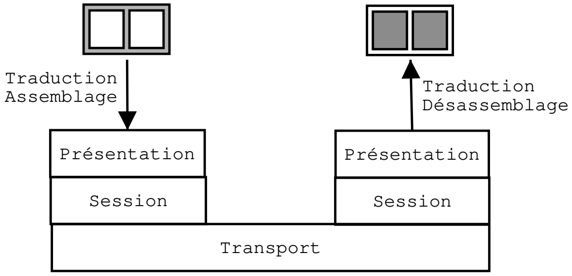


Messages client-serveur

- Trois grands types de message : REQ, REP et ACK
- Autres types possibles : AYA (Are You Alive), BUSY (ordinateur temporairement occupé), ERR (Erreur), etc.

Échange de messages

- Dans un environnement hétérogène, on doit effectuer une présentation adéquate des données.
- Traduction des données (XDR (Sun), ASN.1 (CCITT), etc)
- Assemblage des paramètres émis et des résultats (marshalling)
- Désassemblage des paramètres reçus et des résultats (unmarshalling)

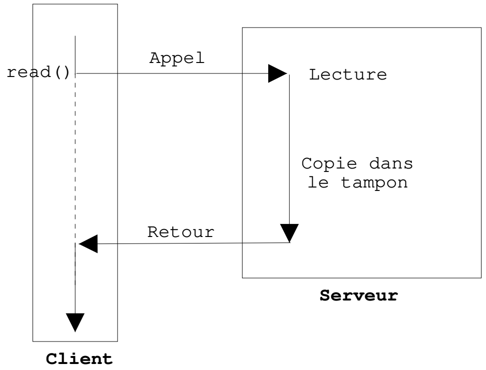


Modes de dialogue

- Mode synchrone
  - n'utilise pas de file d'attente
  - les messages sont émis aussitôt
  - mode bloquant (ex. RPC)
- Mode asynchrone
  - utilise une file d'attente
  - mode non bloquant
  - favorise le multitâche (ex. FIFO, email)

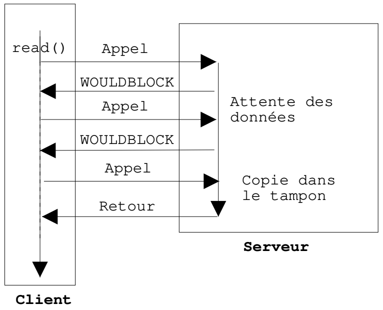
Opérations bloquantes

Le processus se bloque jusqu'à ce que l'opération se termine.

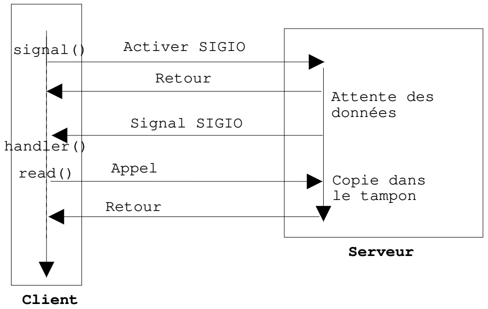


Opérations non bloquantes

Par événements



Par signaux



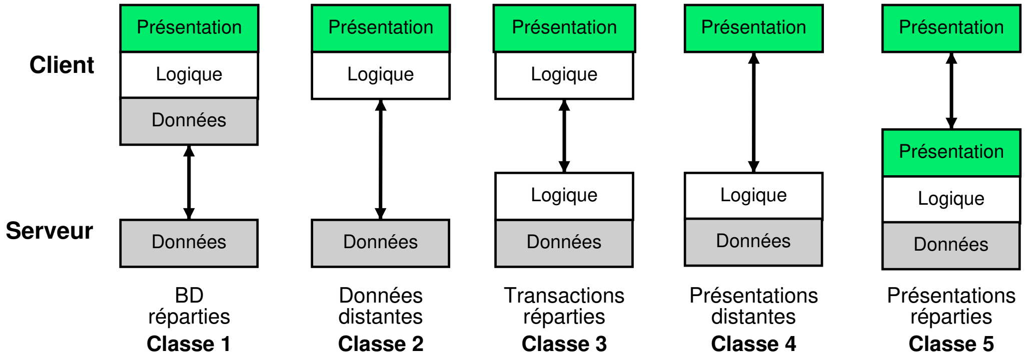
Conception d'une application c/s

Dans une application client-serveur, il faut décider de l'emplacement des composantes de :

- présentation : interfaces textuelles ou graphiques, interactions, entrée des données, validation, etc.
- logique d'application : traitements associés à l'application
- accès aux données : stockage et accès aux données (base de données, serveur web, etc)

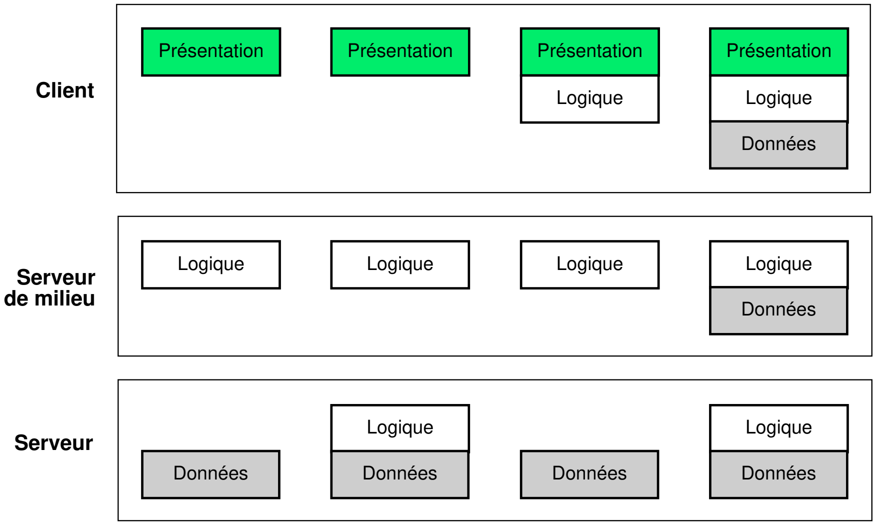
Modèle client-serveur à deux niveaux

Modèle de Gartner pour les systèmes à deux niveaux (2-tiers) :



Client-serveur à trois niveaux

Modèle de Gartner pour les systèmes à trois niveaux (3-tiers) :



Le middleware

- Traduction officielle : intersticiel
- Assure les connexions entre les serveurs de données et les outils de développement sur les postes client
- Ensemble de services logiciels construits au dessus d'un protocole de transport afin de permettre l'échange de requêtes et des réponses associées entre client et serveur *de manière transparente*.
- Les services du middleware sont un ensemble de logiciels répartis qui existe entre l'application, l'OS et les services réseaux sur un nœud du réseau.

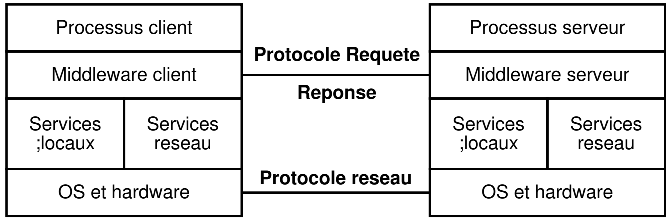
Types de middleware

- Général
  - Protocoles de communication, répertoires répartis, services d'authentification, service de temps, RPC, etc
  - Services répartis de type NOS (Networked OS) : services de fichiers, services d'impression.
- Spécifique
  - de BD : ODBC, IDAPI, EDA/SQL, etc
  - de groupware : MAPI, Lotus Notes
  - d'objets : CORBA, COM/DCOM, .NET

Composantes du middleware

- Les canaux
  - Services de communications entre composants et applications : RPC (synchrone), ORB (synchrone), MOM (Message Oriented Middleware) (asynchrone)
  - Services de support de communication : SSL, annuaires (LDAP)
- Les plate-formes
  - Serveurs d'applications qui s'exécutent du côté serveur
  - Offrent les canaux de communication
  - Assurent la répartition, l'équilibrage de charge, l'intégrité des transactions, etc
  - Exemple : architecture web-tier

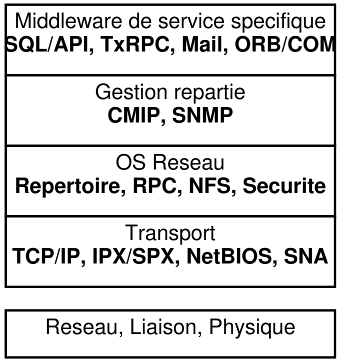
Place du middleware



Exemples :

- Les services primitifs (émulateurs de terminaux, transfert de fichier, mail, etc)
- Services de bases (RPC, etc)
- Services intégrés (DCE, OS répartis)
- Objets distribués (CORBA, COM/DCOM, etc)
- World Wide Web

Services du middleware

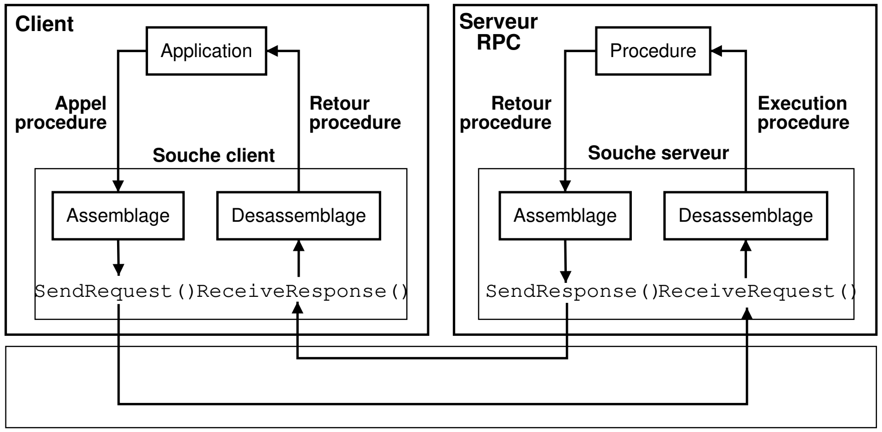


Fonctions d'un middleware

- procédure d'établissement de connexion
- exécution des requêtes
- récupération des résultats
- procédure de fermeture de connexion
- initiation des processus sur différents sites
- services de répertoire (nommage)
- accès aux données à distance
- gestion des accès concurrents
- sécurité et intégrité
- monitoring
- terminaison des processus
- mise en cache des résultats
- mise en cache des requêtes

Le modèle RPC

RPC : Remote Procedure Call



Transparence

- Transparence des réseaux : support de plusieurs types de réseaux (WAN, LAN, etc) et de protocoles (TCP/IP, SNA, etc)
- Transparence aux serveurs : support de plusieurs types de serveur (SGBD (Oracle, Sybase, SQL Server, etc))
- Transparence des données : traductions des formats de données (XDR, ASN.1, etc)
- Transparence de la localisation : utilisation d'un service de répertoire