L'architecture d'un environnement informatique ou d'un réseau est distribuée lorsque toutes les ressources ne se trouvent pas au même endroit ou sur la même machine. Ce concept s'oppose à celui d'architecture centralisée dont une version est l'architecture client-serveur que nous allons aussi voir dans cette partie.

En effet, la programmation orientée objet a permis le développement des architectures distribuées en fournissant des bibliothèques de haut-niveau pour faire dialoguer des objets répartis sur des machines différentes entre eux. Les objets distribués sur le réseau communiquent par messages en s'appuyant sur l'une des technologies telles que CORBA, RMI, les services web XML, .Net Remoting, Windows Communication Foundatation, etc.

Les architectures distribuées reposent sur la possibilité d'utiliser des objets qui s'exécutent sur des machines réparties sur le réseau et communiquent par messages au travers du réseau.

1. Avantages des architectures distribuées :

· *Augmentation des ressources* : la distribution des traitements sur les ordinateurs d'un réseau augmente les ressources disponibles;

· *Répartition des données et des services* : (cas de l'architecture 3-tiers à la base de la plupart des applications distribuées de commerce électronique permettant d'interroger et de mettre à jour des sources de données réparties)

2. Types d'architecture distribuée :

2.1. Architecture client-serveur

Le client server est avant tout un mode de dialogue entre deux processus. Le premier appelé client, demande l'exécution de services au second appelé serveur. Le serveur accomplit les services et envoi en retour des réponses. En général, un serveur est capable de traiter les requêtes de plusieurs clients. Un serveur permet donc de partager des ressources entre plusieurs clients qui s'adressent à lui par des requêtes envoyées sous forme de messages.

Par extension, le client désigne également l'ordinateur sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur.

Le client serveur étant un mode de dialogue, il peut être utilisé pour réaliser de multiples fonctions.

Parlant de l'architecture client-serveur, nous distinguons trois types d'acteurs principaux:

1) Client :

Un client est un processus demandant l'exécution d'une opération à un autre processus (fournisseur des services) par l'envoi d'un message contenant le descriptif de l'opération à exécuter et attendant la réponse à cette opération par un message en retour.

Caractéristiques d'un client :

· Il est actif le premier (ou maître) ;

· Il envoie des requêtes au serveur ;

· Il attend et reçoit les réponses du serveur.

Parlant aussi de client, nous en distinguons trois types :

- *Client léger* : est une application accessible via une interface web consultable à l'aide d'un navigateur web.
- *Client lourd* : est une application cliente graphique exécuté sur le système d'exploitation de l'utilisateur possédant les capacités de traitement évoluées.
- *Client riche* : est une combinaison du client léger et client lourd dans lequel l'interface graphique est décrite avec une grammaire basée sur la syntaxe XML.

2) Serveur :

On appelle serveur un processus accomplissant une opération sur demande d'un client et lui transmettant le résultat. Il est la partie de l'application qui offre un service, il reste à l'écoute des requêtes du client et répond au service demandé par lui.

En effet, un serveur est généralement capable de servir plusieurs clients simultanément.

Caractéristiques d'un serveur :

· Il est initialement passif (ou esclave, en attente d'une requête) ;

· Il est à l'écoute, prêt à répondre aux requêtes envoyées par des clients ;

· Dès qu'une requête lui parvient, il la traite et envoie une réponse.

Nous distinguons plusieurs types de serveur en fonction des services rendus : Serveur d'application, serveur de base de données, serveur des fichiers, etc.

3) Middleware :

Le middleware est l'ensemble des services logiciels qui assurent l'intermédiaire entre les applications et le transport de données dans le réseau afin de permettre les échanges des requêtes et des réponses entre client et serveur de manière transparente.

Le client serveur étant un mode de dialogue, il peut être utilisé pour réaliser de multiples fonctions. Il existe donc différents types de client-serveur qui ont été définis : le client serveur de présentation, le client serveur de données et le client serveur de procédures.

2.2. Architecture pair-à-pair (peer-to-peer ou P2P) :

Le pair-à-pair est un modèle de réseau informatique proche du modèle client-serveur mais où chaque ordinateur connecté au réseau est susceptible de jouer tour à tour le rôle de client et celui de serveur.

P2P est une architecture pouvant être centralisée (les connexions passant par un serveur central intermédiaire) ou décentralisée (les connexions se faisant directement). Le pair-à-pair peut servir au partage de fichiers en pair à pair, au calcul distribué ou à la communication entre noeuds ayant la même responsabilité dans le système.

La particularité des architectures pair-à-pair réside dans le fait que les données peuvent être transférées directement entre deux postes connectés au réseau, sans transiter par un serveur central. Cela permet ainsi à chaque ordinateur d'être à la fois serveur de données et client des autres. On appelle souvent *noeud* les postes connectés par un protocole réseau pair-à-pair.

Outre, les systèmes de partage de fichiers pair-à-pair permettent de rendre les ressources d'autant disponibles qu'elles sont populaires, et donc répliquées sur un grand nombre de noeuds. Cela permet

alors de diminuer la charge (en nombre de requêtes) imposée aux noeuds partageant les fichiers dans le réseau. C'est ce qu'on appelle le *passage à l'échelle*. Cette architecture permet donc de faciliter le partage des ressources. Elle rend aussi la censure ou les attaques légales ou pirates plus difficiles.

Figure 1 : Architecture pair-à-pair

Ces atouts font des systèmes pair-à-pair des outils de choix pour décentraliser des services qui doivent assurer une haute disponibilité tout en permettant de faibles coûts d'entretien. Toutefois, ces systèmes sont plus complexes à concevoir que les systèmes client-serveur.