Systèmes transactionnels répartis[1]

Couche 7 : OSI-TP

1.Introduction : le traitement transactionnel

Le traitement transactionnel en ligne (OLTP pour On Line Transaction Processing) est un mode d'organisation d'une application informatique qui permet à un grand nombre d'utilisateurs de soumettre, via leurs terminaux, des transactions à un système qui doit les traiter le plus rapidement possible et répercuter leurs effets sur une grande base de données.

Ce concept, apparu au début des années 70 est en cours d'évolution et doit intégrer désormais les nouvelles donnes du marché informatique à savoir :

. l'émergence des plates-formes matérielles autour des micro-ordinateurs ou de stations de travail performantes

. l'apparition de standard autour de normes OSI (Open System Interconnection) ou X/OPEN

1.1.Exemples de champs d'applications

Le premier domaine d'application concerné par le traitement transactionnel fût celui de la réservation aérienne, et par la suite, on a assisté à une croissance importante des applications "OLTP" qui couvrent aujourd'hui des domaines aussi variés que :

. la réservation dans les domaines du transport et de l'hôtellerie

- la banque et la finance pour les opérations clientèles, les ordres de bourse et la monétique

- l'assurance pour la gestion des contrats en temps réel

- les grandes administrations pour certaines applications caractéristiques comme l'interrogation de fichiers centraux

- la distribution pour les applications tournant autour de la prise de commande ou de la gestion des stocks

. l'industrie dans le cadre d'applications de pilotage de production

Cette attraction qu'exerce le traitement de transactions en ligne sur les entreprises est due aux possibilités d'accès et de modification en temps réel, permettant aux employés d'être au courant des changements constants qui affectent l'entreprise, ce qui augmente leur productivité.

On dit généralement que le système global (les gestionnaire de transactions et de base de données) doit permettre d'assurer l'intégrité transactionnelle des données modifiées. Dans ce contexte, le moniteur transactionnel s'occupe de l'acheminement de la transaction vers le processus qui doit la traiter d'une part et

[1] Nota : Ce document a été élaboré à partir de travaux de recherche de A. Eftekari, ingénieur de la société Marben (ATOS), chercheur au laboratoire Gracimp (ICTT) de l'INSA et de résultats d'un contrat de recherche entre le laboratoire LISPI de l'INSA (G. Beuchot, I. Chariot, L. Sicard), le Ministère de l'Industrie et la société Marben.

dialogue avec le gestionnaire de base de données qui doit répercuter le résultat de cette transaction dans sa base d'autre part. Le problème important à évoquer est le traitement des transactions réparties, c'est à dire celles qui impliquent l'accès et la mise à jour de plusieurs bases de données distribuées sur plusieurs machines. Pour résoudre les problèmes liés aux défaillances des systèmes ou réseau, lorsqu'une transaction répartie est en cours de traitement, le moniteur transactionnel s'appuie sur le protocole de validation en deux phases (two phases commit). Dans les pages suivantes nous donnons plus d'informations sur ce protocole.

1.2. Caractéristiques des applications transactionnelles.

Ce service est adapté à des applications présentant les caractéristiques suivantes :

- Traitements courts et répétitifs
- Grand nombre d'utilisateurs simultanés
- Nombre important de transaction à traiter simultanément, en parallèle
- Temps de réponses brefs et assez constants, de l'ordre d'une à quelques secondes
- Accès concurrent à des bases de données
- Volumes de données importants
- **Grande fiabilité et haute disponibilité**

Ces caractéristiques ne se retrouve pas seulement dans des applications du secteur tertiaire mais correspondent aussi à des besoins de coordination dans les applications de productique, pour la coordination des tâches dans les systèmes intégrés de fabrication (CIM).

Les grands constructeurs puis les organismes de normalisation du monde Unix ont introduits ces concepts dans leurs architectures distribuées. Ainsi l'X/OPEN group a développé un modèle d'architecture pour le traitement transactionnel distribue : DTP (Distributed Transaction Processing). L'OSF a aussi introduit le traitement distribué dans le modèle DCE (Distributed Computing Environment) avec le système OLTP (On Line Transaction Processing)

2. La transaction

2.1. Définition : propriétés ACID

Une **transaction** est un ensemble d'opérations apparentées caractérisée par quatre propriétés:

- Atomicité
- Consistance
- Isolation
- Durabilité

Ces propriétés sont appelées **propriétés ACID.**

L'**atomicité** implique que les opérations en rapport dans une transaction soient toutes exécutées ou qu'aucunes ne le soient.

La **consistance** implique que les effets des opérations apparentées dans une transaction soient effectuées correctement, précisément et **avec validité**, en respect avec la sémantique de l'application. Les données manipulées par une transaction passent d'un état consistant à un autre état consistant.

L'**isolation** implique qu'à part les opérations de la transaction considérée, aucune opération ou fonction d'une autre transaction ne puisse accéder aux résultats partiels obtenus. **Cette définition implique que des transactions qui travaillent sur des données communes** (à leur interface) **soient sérialisées**. Cette propriété est assurée par un contrôle de **concurrence** qui garantit qu'une action atomique ne peut être validée tant que d'autres actions atomiques qui en ont changé la valeur ne sont pas validées et qu'aucun changement n'est fait sur une données lors de son utilisation par l'action atomique, excepté par les branches de celle-ci.

La **durabilité** implique que les résultats d'une transaction terminée (complète) ne peuvent en aucune manière être altérés, par exemple en cas de défaut ou de panne. Pour annuler l'effet d'une transaction il faut exécuter une transaction de compensation. Une restauration (Recovery) assure la progression correcte d'une action atomique en cas de panne. Elle est basée sur des mécanismes de synchronisation et de **sauvegarde (log)** des données.

Des **données sûres** sont des données qui survivent à une panne d'une application et sont disponibles à l'entité d'application (du logiciel de communications transactionnel) après que des procédures de reprise locales l'ait restaurée. Des données liées sont des données sûres qui existent durant que la transaction s'exécute. Les données d'une action atomique sont des données sûres qui maintiennent:

- la connaissance qu'une action atomique est en cours.

- l'état du contrôle de concurrence.

- l'information permettant de restaurer toute données dans son état initial.

La **journalisation** (log) est un événement correspondant à l'écriture, sur un support sûr, des données spécifiques à une action atomique. Elle permet la reprise consistante et la terminaison d'une action atomique interrompue par une panne.

2.2.Transactions réparties : arbre de dialogue

Une transaction qui se déroule sur plusieurs systèmes ouverts est appelée une transaction répartie. Pour maintenir les quatre propriétés d'une transaction répartie, une coordination est nécessaire entre les entités de chaque système chargées de son traitement. Cette coordination requiert des communications entre ces entités. Dans le service transactionnel OSI, ces entités sont nommées TPSUI : Transport Processing Service User Invocation. Ce sont des instances particulières d'une entité de service (TPSU), ensemble spécifique de capacités de traitement dans un processus d'application**.**

Ces entités interagissent par des communications point-à-point dénommées dialogues. Par ces **dialogues** les entités peuvent assurer :

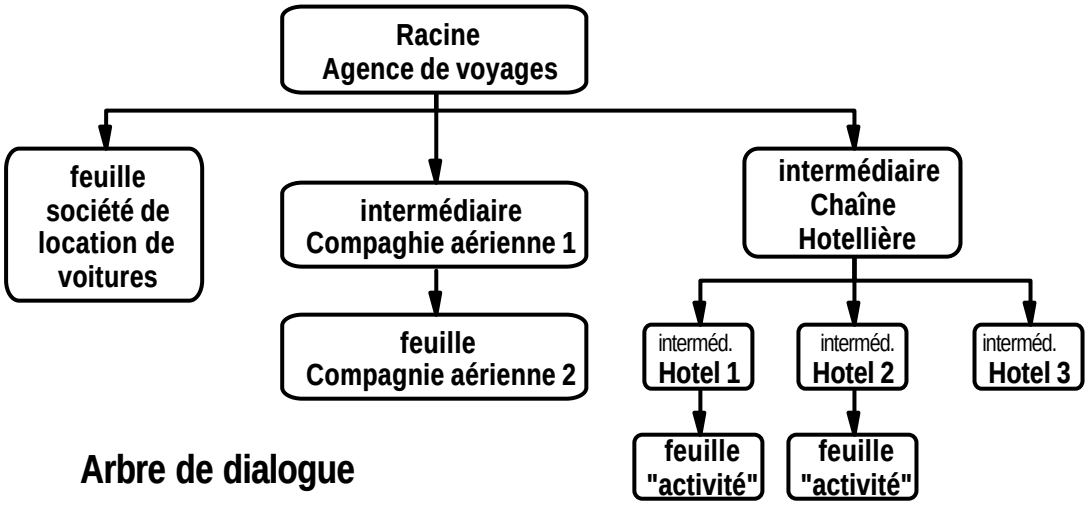
- le transfert de données
- la notification des anomalies
- l'initialisation, la validation ou l'annulation (avec remise à l'état précédent : rollback) d'une transaction
- la terminaison ordonnée ou brutale du dialogue

- La synchronisation des activités pour atteindre un point de traitement que les entités peuvent mutuellement agréer. Ceci est supporté par un service d'acquittement disponible pour la durée d'un dialogue.

Un dialogue peut se dérouler selon deux modes de commande:

- polarisé (initiateur-répondeur), dans lequel seule une entité possède le contrôle du dialogue (sauf pour la notification des anomalies, l'annulation d'une transaction ou sa terminaison brutale)
- partagé, dans lequel les deux entités commandent simultanément le dialogue.

Une entité quelconque peut créer plusieurs dialogues avec des entités distinctes. Ces dialogues sont dits adjacents. Un **arbre de dialogue** est une structure arborescente de dialogues adjacents. Dans un arbre de dialogue, l'entité qui établit un dialogue est appelée "supérieur direct" et l'entité appelée "subordonné direct". L'entité qui n'a aucun supérieur est appelée "racine". Une entité n'ayant aucun subordonné est une "feuille". Les autre entités sont des "intermédiaires".



Arbre de dialogue

La partie d'une transaction distribuée réalisée par une paire d'entités partageant un dialogue est appelée "**branche de transaction**". Le niveau de coordination détermine si un service de validation des transactions est utilisé ou non, c'est à dire qui est responsable, de l'application ou du système de gestion de transactions, du maintien des propriétés ACID. Dans un système de niveau de coordination "validation" (commitment) on peut définir un **arbre de transaction** dont la racine assure le rôle de "coordinateur de validation" ou "coordinateur d'engagement".

Parfois enfin, notamment durant les phases de reprise (recovery) les systèmes utilisants les transactions peuvent avoir à communiquer directement, sans implication des entités supportant les transactions. Ceci est réalisé par les "canaux transactionnel".

2.3.Déroulement d'une transaction

Le système qui débute la transaction est le nœud racine de l'arbre de transaction ie. le coordinateur d'engagement. Un identificateur alloué à la transaction sera propagé sur tout arbre de transaction. Il peut être

un critère d'allocation exclusive d'une ressource à la transaction. et ainsi permettre à chaque noeud de garantir l'**isolation** de la transaction.

Durant le déroulement d'une transaction distribuée, un des systèmes peut se trouver dans l'impossibilité de réaliser le travail qui lui est assigné. Cette *situation d'annulation* (Rollback) est communiquée aux autres noeuds pour qu'ils cessent leurs travaux afin de libérer les données liées à la transaction (données liées) dans leur état initial. Tant que la transaction n'est pas entrée dans sa phase terminale, une annulation peut être effectuée sans difficulté. La solution choisie (avortement présumé) entraîne la reprise à l'état initial. une solution plus sophistiquée, utilisant des points de reprise intermédiaire pourrait être ajoutée.

Pour assurer la **consistance** de la transaction un mécanisme de **validation à deux phases** (two phases commit) doit être implanté. Il se déroule de la manière suivante :

- le coordinateur envoie à ses partenaires une requête de préparation.
- chaque participant doit répondre en acceptant (*ready*) ou en refusant de poursuivre la transaction (*rollback*). Un système qui accepte la transaction est capable d'assurer l'écriture des données en mémoire stable (pour permettre la durabilité).
- Si tous les systèmes ont signalé au coordonnateur qu'ils acceptent la transaction celui-ci leur enjoint de libérer les données dans leur état final (commit).

  Si l'un des participants a refusé (rollback) ou s'il a détecté une panne, le coordonnateur répercute cette requête vers tous les participants qui remettent leurs données dans l'état initial.
- Chaque système, quand il a reçu l'ordre d'engagement (commit) met ses données dans l'état final en lieu sûr (**durabilité**) et signale au coordinateur l'achèvement de son rôle.

Suivant le rôle joué par le système les 2 phases : "prepare" et "commit" ne se situent pas au même moment. Durant la seconde phase **tout système doit stocker dans un journal** (*log*) les diverses informations concernant l'état des données, les décisions prises, les partenaires avec lesquels il coopère. Ce journal ne doit pas être affecté par une panne; enfin de cette seconde phase, les informations devenues inutiles seront éliminées.

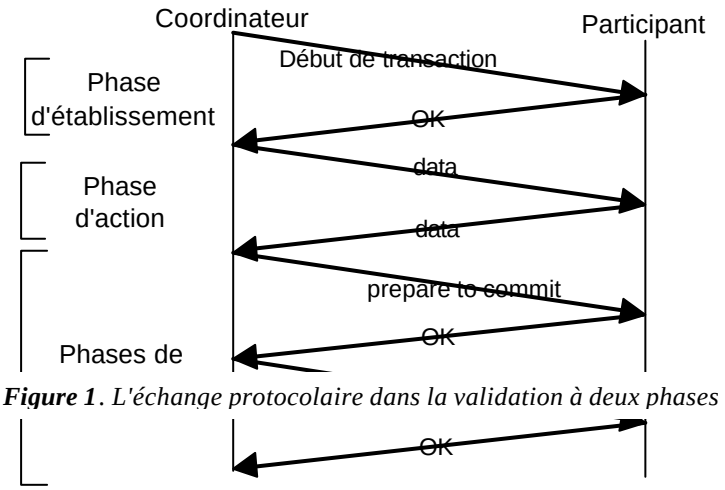


***Figure 1**. L'échange protocolaire dans la validation à deux phases*

Un participant qui a accepté la transaction (ready) ne peut plus avorter car cela pourrait entrer en collision avec la réception de la requête d'engagement venant du coordinateur. Durant la première phase, s'il détecte une rupture de la communication, il doit initialiser une reprise (nouvelle association) et répéter son acceptation (ready). Il attend alors la décision du coordonnateur.

Enfin des défaillances peuvent survenir durant la transaction, en particulier durant la (seconde) phase d'engagement.

Différentes défaillances ont été répertoriées: défaillance d'un programme d'application, défaillance des communications (rupture), faute dans le protocole, crash système, défaillance des moyens de stockage, verrous mortels.

Une défaillance se produisant durant la première phase provoque une annulation avec une clause d'avortement présumé.

Une défaillance durant la seconde phase pose des problèmes beaucoup plus graves et complexes.

On distingue 3 étapes dans la reprise:

- Lancement du processus de reprise après détection d'une défaillance

  On peut décider l'arrêt du dialogue, une annulation ou une reprise.

- Reprise de la transaction après redémarrage du composant défaillant

  On cherche à rétablir un canal avec l'entité distante et à terminer l'engagement.

- Reprise du dialogue quand la consistance est rétablie

  On cherche à rétablir le dialogue.

Il est alors possible de reprendre l'opération normale.

2.4. Validation à deux phases

Un des problèmes majeurs dans les systèmes répartis est de distribuer les données et les ressources sur l'ensemble des machines connectées et de pouvoir assurer l'intégrité de ces données. Si, accéder à une base de données unique à partir d'une machine et assurer sa cohérence est assez simple à réaliser, accéder, dans une même opération de mise à jour, à plusieurs fichiers situés dans des bases différentes, est plus délicat. La raison en est que l'opération de mise à jour doit être atomique. A ce jour, le seul algorithme reconnu qui garantit la cohérence de toutes les données, avant et après l'opération de mise à jour, même dans le cas d'arrêt brutal, panne d'une machine ou du réseau est l'algorithme de "two-phase-commit" (validation à deux phases).

Lorsque plusieurs SGBDs sont impliqués dans une transaction répartie, chaque SGBD doit assurer la cohérence des données et ceci même en cas de défaillance de la machine ou du processus qui exécutait la transaction. Pour cela, il utilise un journal de transactions où il conserve l'identifiant unique de la transaction (le numéro unique est attribué par l'initiateur de la transaction), la valeur d'item avant et après modification, les participants, le coordinateur de transaction, etc, ... En pratique, on dispose soit d'un journal "avant" soit d'un journal "après" suivant le choix qui a été fait par le concepteur. Les valeurs des items avant et après modification sont écrites seulement dans le journal qui appartient à la base où se trouvent les items (AFUU 89). A ce niveau, on introduit le concept de validation en deux phases avec les règles suivantes :

- Une transaction ne peut pas être écrite dans la base tant que le SGBD n'a pas reçu l'ordre de validation (commit).

- Une transaction ne peut pas être validée tant qu'on n'a pas enregistré toutes les modifications des items dans le journal (dans le cas d'une seule base de données, l'organisme de standardisation X/OPEN suggère pour des raisons d'optimisation, la validation en une seule phase).

Ainsi, la première phase est toujours l'écriture dans le journal et la deuxième phase est toujours l'écriture dans la base.

Cette procédure répartie requiert qu'un des participants (l'initiateur) à la transaction coordonne les échanges avec tous ses partenaires. Ce participant particulier, le coordinateur, et tous ses partenaires sont chacun doté d'un journal, comme mentionné ci-dessus. Pour y conserver des informations utiles pour la reprise éventuelle, quatre vagues successives (BARBOT 91) d'informations sont échangées :

1- Le coordinateur signale à ses partenaires le début de validation en deux phases en envoyant l'ordre de "prepare to commit" à ces derniers et appelle à voter.

2- Chaque partenaire peut voter "oui" ou "non". Un vote positif indique que le partenaire s'est assuré qu'il pourra, sur décision finale du coordinateur, annuler l'opération effectuée ou entrer en phase terminale (mise à jour effective de la base). Un vote négatif est notifié dans le cas contraire. Le coordinateur note le résultat positif du vote dans son journal.

3- Les votes sont collectés par le coordinateur. Si ses partenaires et lui-même votent "oui" à l'unanimité, le coordinateur ordonne la mise à jour effective de la base en leur envoyant l'ordre de "commit". Dans le cas de réponse positive des participants, le coordinateur note la décision dans son journal. Un seul vote négatif provoque l'annulation de la transaction et dans ce cas, le coordinateur demande à ses partenaires d'annuler les opérations effectuées jusqu'alors en leur envoyant l'ordre "abort".

4- Chacun des participants s'assure de la fin de transaction, en notifie le coordinateur, et note dans son journal le résultat. Le coordinateur, sur réception de toutes les confirmations de validation note également la fin de la transaction.

La procédure de reprise éventuelle s'applique après qu'une panne ait rompu la communication entre deux partenaires. Elle se présente différemment pour le coordinateur et ses partenaires.

Pour le coordinateur, la procédure de reprise est déclenchée sur détection de panne après la première phase. Le coordinateur trouvant dans son journal la trace du vote positif sans que la fin de transaction ait été enregistrée, entre en contact avec des participants pour lesquels il n'a pas reçu la notification finale. Le coordinateur informe ces derniers et la validation à deux phases reprend en 4).

Pour un participant, la procédure de reprise s'applique lorsqu'après avoir voté il s'aperçoit d'une rupture de communication avec le coordinateur, il trouve dans son journal la trace de son vote sans celle de la fin de transaction. Le participant prend alors contact avec le coordinateur afin que ce dernier lui communique le résultat du vote. Si le journal du coordinateur ne comporte aucune trace de cette transaction, c'est que le résultat du vote est négatif et les données ne doivent pas être mise à jour. Si non, le coordinateur et le participant, après communication du résultat du vote reprennent la validation en phase 4).

Pour que le protocole "two phase commit" s'exécute correctement dans tous les cas, il doit être associé avec le protocole "two phase locking" (verrouillage en deux temps). Le verrouillage en deux temps impose qu'aucun déverrouillage n'intervienne avant que tous les ordres de verrouillage soient exécutés après la validation de transaction. Ainsi aucune transaction n'aura accès à des données modifiées mais pas encore validées issues de transaction en cours ou annulées. Ainsi le protocole "two phase commit" garantit la cohérence des données dans tous les cas qui peuvent se présenter.

3.Modèles d'échanges entre applications

Trois grands modèles d'échanges entre applications permettent de bâtir des systèmes distribués:

- Conversationnel (APPC)
- Client-serveur (RPC)
- Queue de messages (MQ)

3.1.Modèles client-serveur

Ces modèles sont basés sur des échanges "Requête-Réponse" et sont supportées, en général, par un système d'appel de procédures distantes (RPC : Remote Procedure Call)

3.1.1.Modèle client / serveur de base

Un modèle qui peut s'intégrer dans le traitement transactionnel en ligne est celui du client/serveur. Dans cette architecture, un client demande un service qui est assuré par un serveur. Puisque le client et le serveur sont des modules logiciels, une même machine peut les traiter. La multiplication des réseaux (locaux ou étendus) permet d'envisager le développement des applications distribuées sur les différentes unités de traitement connectées.

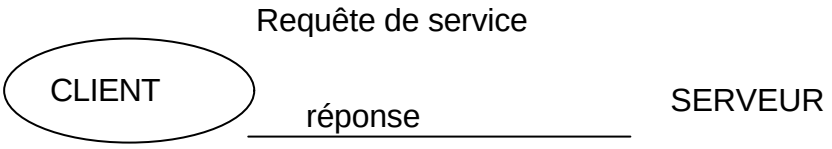


Figure 2. *. Le modèle client/serveur*

Les avantages liés à ce modèle sont multiples. Parmi ceux-ci on peut citer :

Modularité:

L'application peut avoir une forme modulaire. Dans ce cas, pour le développeur d'applications, le travail est plus simple, et la maintenance plus facile.

Facilité de distribution:

La facilité de distribution permet de placer les processus clients, le plus près des utilisateurs, et les serveurs peuvent être distribués sur les différents noeuds du réseau.

Extensibilité:

L'architecture modulaire permet d'étendre l'application. Dans ce cas, l'extension de l'application est possible sans perturber les parties existantes.

3.1.2.Le Modèle client / serveur enrichi

Le modèle client/serveur ci-dessus suggère une relation "un pour un" entre les processus client et serveur. Ceci ne représente pas une architecture optimale. Il entrave la capacité de partage des ressources.

Le processus client doit connaître les informations concernant le serveur et oblige ce dernier à avoir la capacité de satisfaire toutes les requêtes du client. Pour surmonter ces faiblesses (BERSON 94), les systèmes transactionnels proposent une version améliorée : le modèle client-serveur enrichi. Ce modèle permet de placer un serveur de noms entre le client et le serveur.

Cette nouvelle architecture présente les caractéristiques suivantes :

. un serveur est constitué d'un ensemble de services (1 à n); un service est un module écrit pour exécuter une tâche unique de l'application

. les serveurs sont des processus actifs en attente de traitement des requêtes des clients

. un serveur peut offrir plusieurs services et un service peut être offert par plusieurs serveurs

. au lieu d'activer un processus serveur, le client s'attache au serveur de noms avec une requête pour un service donné

. le serveur du nommage met en correspondance le nom logique et l'adresse physique du serveur qui peut exécuter la requête de service

. en contrôlant les services assemblés dans le serveur de nommage, l'administrateur du système peut étendre ou diminuer dynamiquement les ressources disponibles

. la séparation nette des modules clients par rapport aux modules serveurs peut poser des problèmes d'accès non autorisés, l'intégration d'un serveur d'authentification et de contrôle d'accès permet de faciliter la protection des données et des programmes.

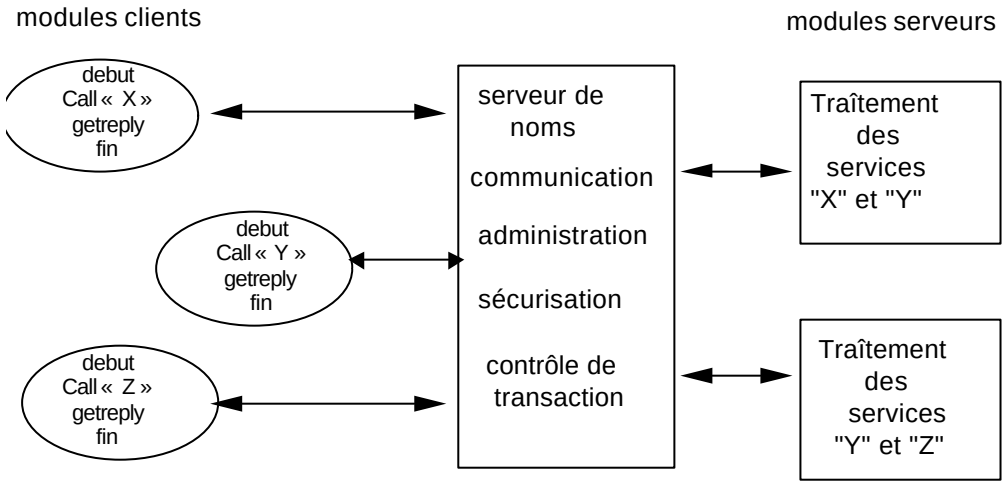


Figure 3. *. Le modèle client/serveur enrichi*

Parmi les avantages de ce modèle, on peut citer :

- **l'indépendance**: des processus clients par rapport aux processus serveur
- **la transparence de localisation**: serveurs et clients sont indépendants géographiquement
- **la performance élevée**: les serveurs sont continuellement actifs
- **la sécurité des données**: le gestionnaire de transactions assure la cohérence des bases de données
- **la sécurité d'accès**: le serveur d'authentification et de contrôle d'accès permet de contrôler l'accès aux ressources partagées
- **la souplesse**: on peut ajuster le nombre de serveurs pour équilibrer l'application

- **l'efficacité**: les ressources partagées réduisent le coût par utilisateur

- **la flexibilité** : de nouvelles fonctionalités peuvent être ajoutées sans perturber les parties existantes.

3.1.3.Le modèle DCE (Distributed Computing Environment) de l'OSF

Open Software Fondation est un consortium constitué des constructeurs informatiques tels que IBM, BULL, HP, DIGITAL et autres. Le but d'OSF est de construire un ensemble d'outils et de règles pour permettre l'interopérabilité entre les systèmes hétérogènes dans un environnement distribué (ROSENBERG 93). L'interopérabilité permet aux utilisateurs d'exploiter et de partager les possibilités et les ressources des différents systèmes connectés à travers le réseau et ainsi de fournir aux utilisateurs de meilleures performances et une utilisation plus effective des ressources informatiques. Un des choix majeurs d'OSF en matière d'architecture répartie est le DCE.

Le DCE est un ensemble de services étendus et intégrés qui supporte le développement, l'utilisation et la maintenance des applications distribuées. La possibilité d'avoir un ensemble uniforme de services sur des machines différentes, réparti dans le réseau, rend capable les applications d'exploiter réellement la puissance qui tend à rester inutilisée dans les systèmes informatiques. Cette architecture représente une opportunité pour transformer un groupe d'ordinateurs interconnectés en une ressource informatique unique et cohérente. Il est constituée d'une couche logicielle qui masque les différences entre plusieurs types d'ordinateurs.

Ainsi le Distributed Computing Environment d'OSF permet aux utilisateurs et aux concepteurs de logiciels d'exploiter les ressources informatiques réparties sur un réseau. Cet environnement logiciel et intégré fait apparaître un réseau de systèmes de différents constructeurs comme un seul système. Dès lors, les utilisateurs peuvent, de manière transparente, bénéficier d'une multitude de ressources informatiques distribuées : applications, informations stockées dans des bases de données, service d'impression et d'archivage et la puissance de traitement. Il est bien évident que l'informatique distribuée soulève un certain nombre de problèmes spécifiques de mise en oeuvre. Comment protéger les données qui doivent être partagées entre plusieurs ordinateurs? Comment synchroniser des événements qui se produisent sur des ordinateurs distincts? Comment des systèmes utilisant des formats de donnés et des structures de fichiers différents vont-ils pouvoir coopérer?

L'architecture DCE, en fournissant un ensemble d'outils et de services permet de répondre à ces différents problèmes. Cette architecture est basée sur un modèle en couche et intègre un ensemble de sept technologies (IBM 92) qui présente l'environnement comme une unité logique de système plutôt qu'une collection de services disparates.

Les services de DCE sont organisés en deux catégories :

- Les services distribués fondamentaux offrant des outils pour développer des applications distribuées comme service d'appel de procédures distantes, service de sécurité, service de Thread, service de distribution de temps et le service de nommage.

- Les services de partages de données construits sur des services fondamentaux. Il inclut le système de fichiers distribués et le support des PCs.

La plupart des constructeurs propose le produit DCE comme la base de leur architecture distribué et l'intègre dans leur environnement comme DCM de BULL et SAA d'IBM.

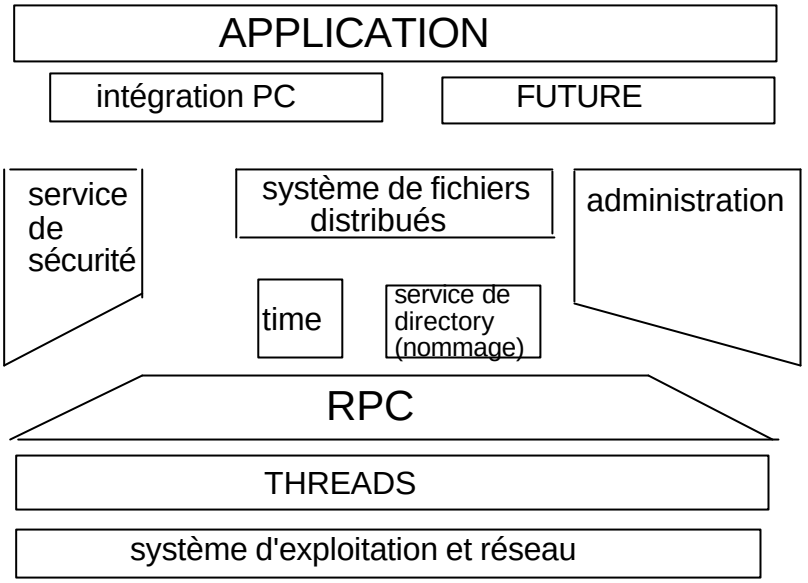


Figure 4. *. Les composants de DCE*

En résumé, le DCE essaie d'apporter une solution aux besoins des entreprises qui portent essentiellement sur les points ci-dessous :

- protéger l'investissement informatique
- élargir le choix des solutions
- optimiser l'utilisation des ressources
- améliorer la productivité des utilisateurs

Présentation des composants de DCE

L'appel de procédures distantes (RPC pour Remote Procedure Call)

Ce mécanisme permet de distribuer une application sur plusieurs machines interconnectées. Il représente le traitement d'une requête d'un programme appelant pour utiliser une procédure placée dans un programme séparé dans l'une des machines .

Le service d'annuaire (Directory Service)

Au coeur de DCE se trouve le service d'annuaire, le composant qui permet aux utilisateurs ainsi qu'aux applications de décrire et localiser les objets (personne, application, service, ...) participant à l'environnement distribué. Ce service de nommage ou localisation est très important, car il aide les usagers à gérer ou utiliser les informations disponibles dans le réseau.

Le service de sécurité (Security Service)

Un environnement informatique distribué peut avoir à supporter des centaines d'utilisateurs accèdent aux ressources situées sur n'importe quel ordinateur dans le réseau. Comme il s'avère difficile de maintenir les informations de sécurité de chaque utilisateur sur chaque machine de l'environnement, DCE rassemble ces données dans une base de données qui est logiquement centralisée, mais physiquement dupliquée sur les noeuds du réseau.

Les threads

Le service des threads de DCE permet aux programmeurs d'exploiter la puissance de calcul et le parallélisme inhérent disponible à travers le réseau. Pour atteindre ce but, le DCE fournit des facilités pour le support de la programmation concurrente, permettant à une application d'exécuter plusieurs actions de façon simultanée.

Pendant qu'un thread exécute une procédure distante, un autre peut permettre la saisie des données de l'utilisateur. Le service de thread est utilisé par la plupart des composants de DCE tels que RPC, service de sécurité, service d'annuaire, service de temps et le système de gestion de fichiers distribués.

Les services de synchronisation d'horloge (Time Service)

La synchronisation du temps est un problème majeur dans l'environnement distribué. Cette synchronisation permet à des applications distribuées de déterminer l'enchaînement, la durée et le calendrier prévisionnel des événements, quelque soit l'endroit où ils se produisent. Aussi longtemps que les horloges qui contrôlent les événements sont correctement synchronisées soit avec une base de temps, soit avec d'autres événements, les applications peuvent s'exécuter sans problème.

Le Distributed Time Service de DCE essaie d'apporter une cohérence au niveau de temps en synchronisant les horloges des ordinateurs interconnectés par des réseaux locaux ou à grande distance.

Le service des fichiers distribués (DFS)

Le DFS permet d'avoir une vue globale sur les fichiers manipulés par les systèmes hétérogènes. En fournissant une interface cohérente, le DFS permet des accès globaux aux fichiers distribués sur les différents nœuds du réseau aussi facilement qu'en local. Pour réaliser cette tâche, le DFS d'OSF utilise le modèle commun client/serveur basé sur le RPC.

Le service d'intégration des PCs

Le service d'intégration des PCs d'OSF donne aux utilisateurs de mini-ordinateurs, "mainframes" et PC la capacité de partage de fichiers, périphériques et application dans un environnement distribué. En utilisant ce service, les utilisateurs des PCs sous DOS ou OS/2 peuvent accéder aux services offerts par DCE sur d'autres machines, visualiser, copier et transformer des fichiers vers ou à partir du monde UNIX. Ils peuvent partager des serveurs d'impression pour imprimer les fichiers stockés localement ou sur la machine serveur.

Le service d'intégration de PC utilise le service de NFS de SUN et le Server Message Bloc standardisé par X/OPEN et s'appuie sur TCP/IP ainsi que le protocole OSI pour la communication au niveau transport.

3.2.Le modèle Distributed Transaction Processing (DTP) de l'X/OPEN

Historiquement, le système de gestion de TP (Transaction Processing) a démarré à partir des besoins spécifiques et pour des traitements particuliers des clients. Il était implémenté uniquement sur des

"mainframes" propriétaires. De cette manière, les applications développées pour l'utilisation d'un moniteur transactionnel ne pouvaient en aucun cas tourner sous un autre moniteur transactionnel.

Pour pallier cet inconvénient majeur, les clients et fournisseurs ont essayé de travailler ensemble pour assurer la pérennité des applications développées et réduire le coût de ces développements. Ces efforts ont été manifestés sous forme de standard "national" ou "international" par les organisme tels que ANSI (American National Standards Institute), ISO (International Standard Organization) et le groupe X/OPEN. Ces organisations ont essayé de casser le mur des systèmes transactionnels propriétaires (entre autre) en proposant des solutions standards sous forme de protocole et API (Application Program Interface) pour accès aux services des moniteurs transactionnels. Un système de gestion des transactions est nécessaire pour gérer une large variétés de ressources à travers des plates-formes différentes.

Pour répondre aux besoins des clients et pour permettre la pérennité de leur investissement, le groupe X/OPEN a proposé un modèle transactionnel distribué en définissant le protocole et les APIs correspondants.

3.2.1.Le groupe X/OPEN

X/OPEN est une organisation indépendante, mondiale et acceptée par bon nombre de fournisseurs de système, utilisateurs et les sociétés de logiciels. Sa mission est d'apporter une valeur ajoutée aux utilisateurs à travers des implémentations pratiques des systèmes ouverts. La stratégie de l'X/OPEN pour atteindre ce but est de combiner l'existant et fusionner les standards dans un environnement intégré (DTP 91), utilisable et compréhensif appelé "Common Application Environment".

Les APIs définis par X/OPEN permettent d'améliorer de façon significative la portabilité des programmes d'applications au niveau de code source.

Une des plus importantes contributions de l'X/OPEN a été de spécifier un modèle pour le traitement transactionnel distribué (Distributed Transaction Processing). Ce modèle a l'avantage d'offrir une découpe organique et modulaire des différentes entités impliquées dans le traitement transactionnel et ainsi de proposer une implémentation simple dans un environnement local ou distribué selon le principe de modèle client/serveur.

3.2.2.Le modèle DTP

Le Distributed Transaction Processing (DTP 91) définit un modèle pour le traitement transactionnel dans un environnement distribué. Ce modèle prévoit trois composants logiciels :

- Un programme d'application (AP) définit les limites de transaction et spécifie les actions que constitue une transaction

- Un gestionnaire de ressources (Resource Manager comme un gestionnaire de base de données ou un système de gestion de fichiers ou toute autre ressource) fournit l'accès aux ressources partagées.

- Un gestionnaire de transactions (Transaction Manager) prend la responsabilité de routage, validation de transaction et recouvrement des données en cas de panne.

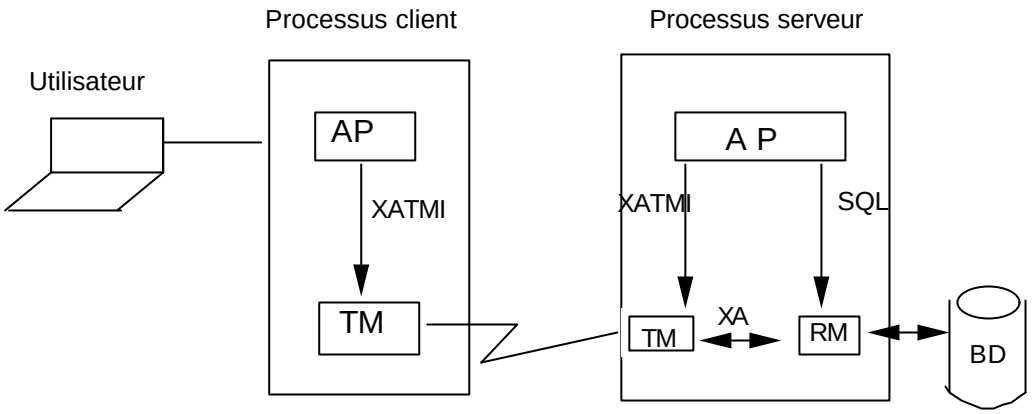


Figure 5. *. Le modèle DTP de l'X/OPEN*

La figure ci-dessus illustre une instance locale d'un système DTP où AP appelle un TM pour structurer une transaction. Il peut exister plusieurs systèmes DTP sur le même processeur. Les interfaces utilisées par les composants de DTP sont l'interface XATMI (Application Transaction Manager Interface) entre AP et TM et permettant à AP de dialoguer avec TM, L'interface bidirectionnel entre TM et RM permet de gérer les transactions réparties en se basant sur le protocole de validation à deux phases et enfin l'interface entre AP et RM est en général ISO - SQL. Il faut noter que le protocole de validation en deux phases d'X/OPEN s'appuie sur la définition des services et protocole de CCR (Commitment, Concurrency and Recovery) d'ISO dans le cadre d'OSITP (voir ci-dessous).

Ce modèle s'appuie sur l'architecture client/serveur enrichi et ses composants sont les processus client, serveur et le gestionnaire de transactions.

Le processus client

C'est la partie de l'application qui s'interface avec l'utilisateur final (via un terminal, un lecteur de code à barres, ...). Comme celui-ci est le processus "frontal", il est responsable de :

- L'acquisition des données en entrée
- Le lancement de(s) requête(s) de service
- La réception de(s) réponse(s)
- La gestion de l'affichage des résultats

Le processus serveur

C'est la partie de l'application qui exécute des services spécifiques. Souvent, elle interface un gestionnaire de ressources (RM), typiquement un SGBD mais pouvant aussi bien être un autre type de gestionnaire de ressources, un spooler d'imprimante par exemple. Selon le modèle DTP, ce processus comme le processus client est formé d'une partie applicative (AP) et d'un gestionnaire de transactions (TM) et souvent, il comprend un gestionnaire de ressources.

Le gestionnaire de transactions (TM)

Il assure le routage de requêtes et la réception de réponses pour des services nommés et prend en charge la gestion de la communication TM à TM, quand elle se fait à travers un réseau. Il a pour principales tâches de :

- Router les transactions vers le processus serveur concerné où qu'il soit dans le réseau à partir de requêtes de service par nom, ce qui assure une indépendance totale du client par rapport au serveur.

- Assurer l'atomicité des transactions réparties (une transaction répartie correspond à plusieurs transactions locales au niveau de plusieurs serveurs et ce par le biais d'une interface/protocole de "pilotage" du gestionnaire de ressources, c'est à dire l'interface XA).

- Permettre à l'administrateur de contrôler et d'adapter l'application aux conditions d'exploitation.

L'interface XA

C'est l'interface entre le TM et RM dans le modèle DTP de l'X/OPEN. Dans le cadre des transactions réparties, cette interface garantit l'intégrité transactionnelle. Le protocole XA se base essentiellement sur le protocole de validation à deux phases pour réaliser cette tâche. Cette interface est complémentaire aux services de CCR (Commitement, Concurency and Recovery) d'ISO puisqu'elle gère la relation entre le gestionnaire de transactions et le gestionnaire de ressources dans le cadre de ce protocole. Dans un système transactionnel basé sur le modèle DTP, le Moniteur transactionnel fournit une partie des fonctions de cette interface, pour permettre au gestionnaire de ressources de récupérer des informations concernant la transaction répartie. La deuxième partie des fonctions est fournie par le gestionnaire de ressources qui permet au TM de manipuler la transaction (initialisation, validation, annulation,...).

3.3. Echanges inter-applications par files de messages : modèle MQ

Le **modèle MQ, Message Queueing**, est le plus récent et sans doute le moins connu alors qu'il offre une grande souplesse et une facilité d'utilisation qui en font tout l'intérêt. Fonctionnant typiquement sur un mode asynchrone, il permet de désynchroniser les applications et n'exige même pas que les deux applications qui interagissent s'exécutent simultanément. Par ailleurs il peut être aussi bien implanté entre des applications s'exécutant sur un seul systèmes ou réparties. Enfin il peut , en utilisant un système de double files, répondre aussi aux besoins d'échanges Client/serveur de type Requête/Réponse.

Ce système de communication **repose sur des transactions réparties**; il est en cours de normalisation à l'OSI comme 7ème partie du standard OSI TP (OSI Transaction Processing) avec la référence ISO/IEC CD 10026-7.

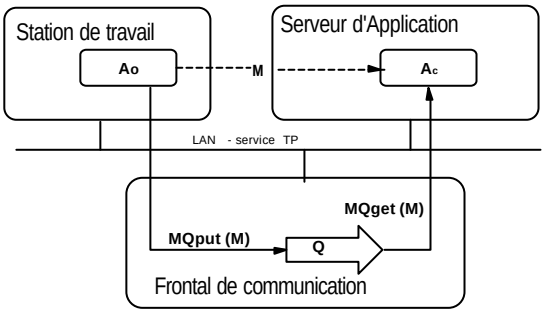


Figure 6

3.3.1. Principes

Le modèle Message/Queue MQ repose sur le **transfert de messages entre deux applications à travers des files d'attentes sécurisées**.

Une application origine A_O envoie un message M à une application collecteur A_C par l'intermédiaire d'une file d'attente Q. Ce message est déposé par une commande MQput par A_O et retiré de la file par une commande MQget de A_C.

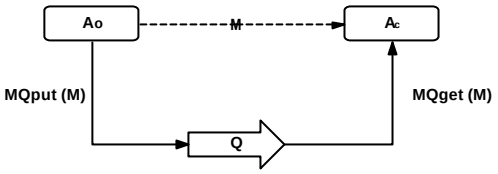


Figure 7

Les entités A_O,, A_C et Q peuvent être situées sur le même serveur d'applications ou être réparties entre un serveur d'application, une station de travail et éventuellement un frontal de communication qui ne supporte que les files. Ces architectures sont illustrées sur les schémas ci-dessous.

La queue apparaît comme un service dans une architecture client-serveur.

Les applications en sont des **clients source ou collecteur** des messages qui y sont déposés.

Le gestionnaire de file d'attente est un **serveur** qui rend, à distances, les services suivants:

- établissement du dialogue (avec le serveur)
- création et/ou ouverture des queues
- écriture ou lecture d'un message dans une queue
- fermeture et/ou destruction d'une queue
- terminaison du dialogue

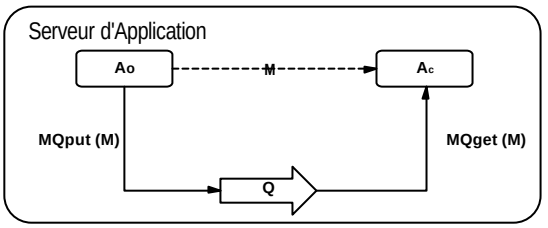


Figure 8

Les communications entre les applications clientes et le serveur de files d'attentes s'appuient sur un service (de communications) transactionnel qui assure l'intégrité et la durabilité des messages. Un **message** n'est considéré **écrit dans une queue** qu'après qu'une **validation à deux phases** (commit) ait été réalisée. De même, à la suite d'une lecture par un client, un message n'est effacé d'une queue qu'après qu'une validation à deux phases ait prouvé la fin du traitement.

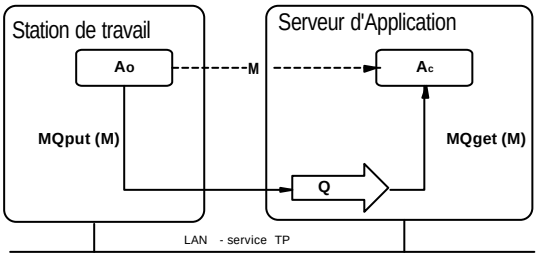


Figure 9

Le serveur (gestionnaire) de files d'attente peut être physiquement installé sur un frontal de communications comme dans le schéma ci-dessus soit réparti sur les systèmes qui supportent les applications comme illustré par le schéma ci-contre :

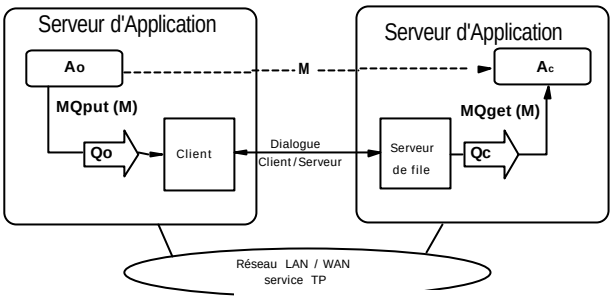


Figure 10

Le programme Client de l'application source lit un message dans la file Qo selon le modèle MQ et l'écrit dans la file Mc de l'application collecteur selon le même modèle. Dans chaque serveur d'application on peut aussi utiliser ce modèle pour écrire (resp. lire) les messages dans les files Qo et Qc.

On peut aussi ajouter des fonctions permettant de consulter sélectivement les files, de les purger sélectivement ou d'y modifier l'ordre des messages.

Enfin le serveur de files est identifié et chaque file qu'il contient est aussi identifiées par un nom logique et un type caractéristique des objets contenus.

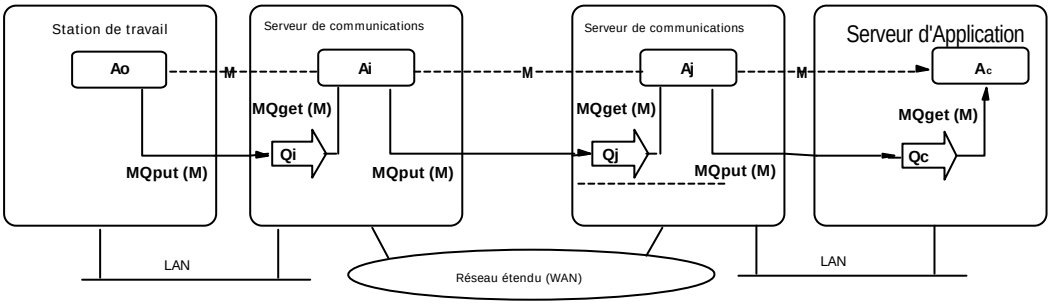


Figure 11

Ce système de files réparties peut être généralisé en utilisant des systèmes de communications intermédiaires qui supportent ces queues comme illustré ci-dessous. L'utilisation des systèmes relais peut être rendue complètement transparente grâce aux noms logiques et à des fonctions d'annuaires. Cette technique permet aussi d'utiliser des protocoles de communications différents entre les systèmes par exemple un ensemble ISO pour le réseau étendu et des ensembles Inet (TCP/IP) sur les réseaux locaux (ou toute autre architecture spécifique comme SNA). Des fonctions supplémentaires d'archivage, de diffusion, de chiffrement, de compression peuvent être placées sur ces systèmes intermédiaires.

3.3.2. Communications asynchrones et synchrones

Le modèle MQ est spécialement bien adapté aux communications asynchrones; il ne demande pas aux applications de s'exécuter simultanément, les files permettant de stocker les messages temporairement de manière sûre.

Un système de communication symétrique avec des files permettant des échanges dans les deux sens permet de travailler en mode asynchrone avec réponse ou en mode synchrone (attente de la réponse pour continuer l'application cliente) si les messages sont correctement identifiés pour pouvoir mettre en correspondance une réponse dans la file de retour avec une requête de la file de commande. Ce service de réponse peut permettre de sécuriser les échanges sans écriture systématique des messages sur disque; dans ce cas une réponse de l'application serveur (positive ou erreur) permet de connaître le sort de la requête (une non-réponse dans un délai donné doit être alors considéré comme un incident qui demande une reprise ...). La charge système pourrait en être allégée. Toutefois cette technique n'est pas régulière vis à vis d'une application asynchrone et demande un système spécifique pour traiter les anomalies. L'utilisation d'un seul service et d'une seule interface constitue un avantage non négligeable.

4.La normalisation du traitement transaction réparti : OSITP)

Dans le cadre de l'interconnexion des systèmes ouverts, L'OSI a entrepris la normalisation d'un protocole de communication pour les applications transactionnelles distribuées. Dans le contexte de l'Interconnection des Systèmes Ouverts, l'objectif d'un tel effort a été de définir un modèle à partir duquel les services et le protocole pourraient être dérivés pour supporter les transactions distribuées à travers une collection de systèmes ouverts. Ce protocole doit permettre le traitement des donnés réparties de façon fiable et cohérent. Les travaux de normalisation ont débuté en Avril 1986 et enregistré sous forme de DIS (Draft International Specification) en novembre 1989 en définissant les services d'ISO attendus ainsi que les spécifications de protocole. Dans ce cadre, de 12 à 14 pays membres de l'OSI, CCITT et l'ECMA ont suivi les travaux , soit environ une cinquantaine d'experts. Au niveau français, l'AFNOR s'est montré particulièrement actif dès le début des travaux et a proposé le modèle, services et protocole dont de très larges part sont retenues dans la version de DIS (BARBOT 91).

La norme définitive(IS) a été adoptée en avril 1991 et les documents détaillés ont été publiés.

4.1.Présentation de la norme TP (Transaction Processing)

Dans le contexte d'ISO, le traitement transactionnel est pris en charge par un ensemble de systèmes ouverts exécutant la transaction. Deux ou plusieurs systèmes coopèrent pour réaliser une tâche donnée appelée transaction. L'objectif premier de la norme est de spécifier un protocole qui par encapsulation de la communication entre processus d'application confère à cette communication des caractéristiques transactionnelles. Ces caractéristiques résident principalement dans le traitement résistant aux pannes et le traitement réparti hiérarchisé. Aux côtés des buts strictement transactionnels, on retrouve les exigences de performances, conditions essentielles à la pratique des transaction réparties. Le cahier des charges souligne le besoin d'exécuter des séquences de transaction, avec un temps de réponse minimum et d'un traitement efficace de transactions courtes ou longues, et cela même dans une situation de trafic élevé.

Vis-à-vis du Modèle de Référence ISO, la norme TP se place dans la couche application (DRAFT-ISO 90), c'est à dire au niveau des entités d'application rendues visibles au monde des communications OSI.

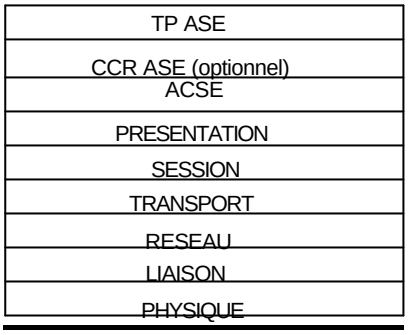


Figure 12. *. Le modèle architectural d'OSITP*

La norme TP répond également aux principes d'intéropérabilité et d'économie développés par la normalisation OSI. Pour capitaliser sur l'investissement des normes OSI déjà développées, les services définis par la norme TP sont construits aussi souvent que possible, à l'aide des services offerts par la couche présentation, ACSE et CCR.

4.2.Terminologie et concepts de base

Le terme "transaction" utilisé correspond à une unité de travail caractérisée par les propriétés ACID. Dans le cadre de l'OSI, ce terme recouvre les notions de "donnés liées" et propriétés ACID. Les données liées (bound data) sont ainsi appelées car leur sort est lié à la destinée de la transaction. Une transaction OSI acquiert les donnés liées, les traite et enfin les libère.

Du point de vue de la transaction, les donnés sont acquises dans l'état initial et sont libérées dans l'état final (leur valeur après traitement) ou l'initial, ce dernier cas est appliqué en cas d'échec.

La transaction OSI se définit au moyen de quatre propriétés, dites "ACID", vis à vis des données liées.

- la transaction doit être considérée comme une unité indécomposable de traitement ou encore atomique (le "A")

- La transaction OSI a pour objectif d'assurer que les données liées sont libérées dans le même état sur tous les sites de traitement, dans l'état final si la transaction se termine avec succès, sinon dans l'état initial ("C").

- Les données liées sont acquises par une transaction et une seule. Il ne peut y avoir ni partage de données entre transactions OSI, ni accès aux résultats intermédiaires d'une transaction OSI par une autre transaction OSI ("I").

- les effets de la transaction OSI au niveau des données liées sont rendus durables vis-à-vis des situations de panne, bien que la transaction soit terminée ("D").

En résumé, la transaction OSI transforme les données liées d'un état cohérent à un autre état cohérent (GHERNAOUTI 90), même en cas de panne. C'est sur cette base que la norme OSITP a été établie. Ce protocole transactionnel vise à garantir les quatres propriétés ACID.

Pour garantir l'intégrité transactionnelle, OSITP se base sur le protocole de validation à deux phases assuré par les services de CCR (Commitment, Concurrency and Recovery).

Pour atteindre cet objectif, l'OSITP définit un modèle de communication dans lequel les différentes entités sont organisées en arbre de dialogue et transaction répartie.

4.3.Dialogue

Lorsque deux systèmes ouverts veulent communiquer, ils le font en établissant une relation entre eux. En terme d'OSI, cette relation est établie à différents niveaux. Par exemple, une connexion réseau doit être établie pour permettre une connexion transport, etc . . . jusqu'à la couche application, dans laquelle selon la terminologie d'architecture OSI, on établit une association entre les processus d'application.

Pour permettre la relation entre processus de traitement des transactions, et réduire également l'"overhead" de traitement, OSITP en établissant une association, définit un niveau abstrait de relation entre systèmes ouverts appelé "dialogue".

Lorsqu'après l'établissement d'association entre deux systèmes, une invocation de processus transactionnel (TPSUI pour Transaction Processing Service User Invocation) veut demander un service particulier de transaction d'un autre processus (TPSU), si la TPSUI demandée est active, un dialogue va être établi entre ces deux TPSUIs.

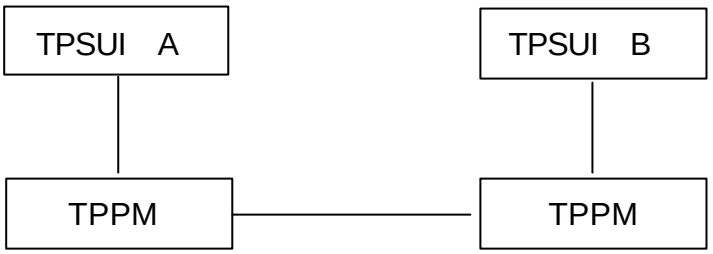


Figure 13. *. Le modèle de dialogue dans OSITP*

En d'autre terme, le dialogue se définit comme la relation de communication entre deux TPSUIs appartenant à deux processus d'applications différentes. Pour la transmission des éléments sémantiques, le dialogue utilise une association qui peut être établie et gérée dans un "pool" commun.

4.3.1.Arbre de transaction

Une TPSUI peut établir un ou plusieurs dialogues, chacun avec une TPSUI distante, ce qui peut signifier la création d'une nouvelle instance de TPSU demandé. A son tour, une TPSUI distante peut établir un ou plusieurs dialogues avec d'autres TPSUIs, etc...

Cette possibilité pour une TPSUI de créer une instance d'un TPSU donne naissance à une structure dont la nature est un arbre (arbre de dialogue).

OSITP fournit deux modes pour assurer le dialogue entre TPSUIs. Dans un mode, il fournit le support minimal à ces derniers (GHERNAOUTI 90) et leur permet la coordination entre systèmes ouverts pour garantir la bonne exécution de transaction distribuée. Le second mode est plus riche et fournit aux utilisateurs de services OSITP le support complet de maintenance des propriétés ACID de la transaction répartie (le choix de ces modes s'effectue suivant l'attribut "coordination" d'un dialogue).

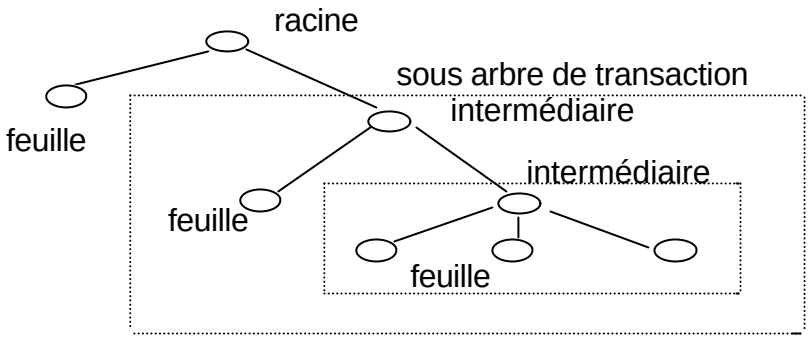


Figure 14. *. L'arbre de transaction dans OSITP*

Dans le cas où ces deux modes sont disponibles, une TPSUI peut établir une branche de transaction avec son partenaire dans le cadre d'un dialogue. Cette branche peut être ouverte à l'ouverture de dialogue ou plus tard. La fin de la branche de transaction se produit soit par la fin de la validation à deux phases soit par un retour arrière; c'est à dire la libération des données liées dans l'état initial. En d'autres termes, à l'intérieur d'un arbre de dialogue, on peut avoir un ou plusieurs sous arbres ayant demandé le support complet des propriétés ACID à OSITP. Comme ces sous-ensembles de l'arbre ont également une structure en arbre, ils sont appelés arbres de transaction.

Un arbre de transaction consiste en une racine, des noeuds intermédiaires et des feuilles comme arbre de dialogue.

4.4.Services et protocole d'OSITP

La machine protocolaire OSITP utilise l'ASE (Application Service Element) de Association Control Service Element (ACSE) pour établir et libérer les associations d'application.

Lorsqu'une TPSUI demande l'établissement d'un dialogue, il spécifie si le TPPM devra fournir les services pour assurer les propriétés ACID.

Les services d'OSITP sont regroupés sous formes d'unités fonctionnelles (GHERNAOUTI 90). L'unité fonctionnelle noyau (Kernel) supporte le service de base de traitement transactionnel permettant d'ouvrir un dialogue entre 2 TPSUIs. D'autres unités fonctionnelles sont définies pour assurer les modes de contrôle polarisés et partagés pour fournir les services de synchronisation, ceux de coordination et de chaînage ou de déchaînage.

4.4.1.Noyau

L'unité fonctionnelle noyau supporte les services pour établir un dialogue entre deux TPSUIs dans lesquels les U-ASEs peuvent être invoquées. Cette unité fonctionnelle comprend les services pour ouvrir un dialogue, le terminer, l'interrompre, d'en refuser l'ouverture et de traiter les erreurs.

4.4.2.Contrôle partagé

Cette unité fonctionnelle permet à deux TPSUIs d'avoir le contrôle de dialogue, charge à ces derniers de s'entendre pour exercer leur contrôle de façon cohérente. Par exemple, les données peuvent être transférées simultanément par les deux invocations de TPSUs.

4.4.3.Contrôle polarisé

Avec ce mode, seule une TPSUI possède le contrôle de dialogue à un instant donné. Les données sont transmises par la TPSUI qui possède le contrôle de dialogue.

A noter que les unités fonctionnelles de contrôle partagées et polarisées sont mutuellement exclusives. Pour un dialogue, une seule doit être choisie.

4.4.4.Synchronisation

L'unité fonctionnelle "handshake" permet aux TPSUIs de synchroniser leur traitement. La synchronisation correspond à la possibilité pour les TPSUIs de vérifier qu'ils ont atteint un point de traitement convenu à l'avance. La sémantique d'une telle synchronisation est totalement définie par ces derniers.

4.4.5.Commit

L'unité fonctionnelle commit (coordination) rend possible la validation (comittment) fiable ou le retour en arrière (rollback) des transactions. La coordination correspond à la possibilité de déclencher la phase terminale de la transaction avec validation à deux phases et reprise possible, sous contrôle de fournisseur de service TP. C'est cette possibilité de coordination qui permet de garantir les propriétés ACID. Si le dialogue n'est pas établi avec cet attribut, la garantie des propriétés ACID incombe à l'application. La validation à deux phases comprend la coordination de l'arbre de transaction, la journalisation, la gestion des données liées et le pilotage de CCR.

4.4.6.Le chaînage et le déchaînage des transactions

Ces modes permettent le regroupement ou l'exclusion des TPSUIs de même branche dans une transaction.

unités fonctionnelles	primitives de service
noyau	TP-BEGIN-DIALOGE TP-P-REJECT TP-END-DIALOGUE TP-DATA TP-U-EROR TP-P-ERROR TP-U-ABORT TP-P-ABORT
contrôle partage	pas de primitive
contrôle polarisé	TP-GRANT-CONTROLE TP-REQUEST-CONTROLE
synchronisation	TP-HANDSHAKE TP-HANDSHAKE-AND-END TP-HANDSHAKE-AND-GRANT-CONTROL
COMMIT	TP-DEFERRED-END-DIALOGUE TP-DEFERRED-GRANT-CONTROL TP-COMMIT TP-CONTINUE-COMMIT TP-COMMIT-RESULT
	TP-DONE TP-COMMIT-COMPLETE TP-PREPARE TP-READY TP-ROLLBACK TP-ROLLBACK-COMPLETE
Transactions non chainées	TP-DEFER-NEXT-TRANSACTION TP-BEGIN-TRANSACTION

Figure 15. *. Les unités fonctionnelles et primitives de services d'OSITP*

5.Les moniteurs transactionnels

Le rôle d'un moniteur transactionnel est de permettre la communication et la mise en relation des processus clients et serveurs. Cette communication est complètement transparent pour ces processus quant à leur localisation et le média de communication (réseau) utilisé. En offrant une interface programmatique et des outils de supervision, il simplifie le rôle des développeurs d'application ainsi que celui de l'administrateur.

Comme il a été évoqué dans les pages précédentes, le moniteur transactionnel et le SGBD sont les deux composants indissociables d'un vrai système OLTP.

L'importance de ce composant dans les offres actuelles ou futures de tous les grands constructeurs (PIMONT 92) comme IMS/CICS d'IBM, TDS et TP8 chez Bull, ACMS de Digital,... montre à l'évidence le caractère stratégique du moniteur TP dans les architectures des systèmes distribués.

Dans le monde UNIX, et en voyant son importance comme système ouvert, le manque d'un moniteur TP a conduit les fournisseurs à proposer des produits pour répondre à ce besoin.

En effet, si pendant de nombreuses années le traitement transactionnel a été pratiquement le domaine réservé des grands systèmes propriétaires (IBM/MVS, BULL/GCOS, UNISYS,...), la situation est en train de changer pour différentes raisons, parmi lesquelles :

- le rapport prix/performance du système UNIX par rapport à celui des grands systèmes
- la montée en puissance des processeurs RISC et des systèmes UNIX construits autour de ceux-ci
- l'arrivée de nouvelles architectures de systèmes ouverts telles que les machines multiprocesseurs et à tolérance de panne
- la baisse des coûts des réseaux locaux et des stations de travail
- l'effet d'entraînement des standards et des systèmes ouverts
- l'amélioration de la fiabilité des systèmes UNIX
- la disponibilité de nombreux SGBD comme ORACLE, INGRES, INFORMIX, SYBASE,...
- les interfaces graphiques évoluées offertes par les stations de travail
- l'existence aujourd'hui d'une offre UNIX chez tous les constructeurs

Actuellement, on assiste à l'émergence sur le marché des moniteurs transactionnels sous UNIX des produits tels que:

- TUXEDO d'USL
- ENCINA de TRANSARC
- CICS/6000 d'IBM

Nous donnons dans les pages suivantes une description générale de produit TUXEDO de Unix System Laboratories ainsi qu'une présentation succincte des produits ENCINA et CICS/6000. Nous décrirons aussi un moniteur distribué utilisant le Modèle MQ.

5.1.TUXEDO

5.2.Présentation

Ce produit a été développé et commercialisé par Unix System Laboratories d'ATT. TUXEDO a été conçu pour fournir un environnement transactionnel efficace sous UNIX avec une séparation nette des modules clients et serveurs sur les principes décrits précédemment. Ce produit a été porté sur de nombreuses plates-formes UNIX (USL 95) et également sous d'autres systèmes d'exploitation (VMS, OS2,...).

Il constitue la base de l'"offre transactionnelle" de plusieurs constructeurs comme BOS/TP de BULL ou OPEN OLTP d'UNISYS, mais est commercialisé également sur les plates-formes UNIX de DEC, HP, SUN, IBM,...

Il est sans aucun doute le moniteur le plus utilisé aujourd'hui. TUXEDO est conforme au modèle général DTP et correspond au composant TM tel que défini par l'X/OPEN. Il dispose de l'interface XA et peut donc communiquer avec les principaux SGBD du marché.

Son interface programmatique ATMI entre les applications et le moniteur a servi de base (USL 95) a l'X/OPEN pour définir XATMI (qui reste très proche d'ATMI). L'interface ATMI offre aux programmeurs d'application un accès transparent aux services offerts par TUXEDO, tels que les services de dialogues entre applications, conversion et manipulation des données et des opérations de contrôle d'une transaction. ATMI contient notamment les interfaces Tx, TxRPC et XATMI. L'interface Tx permet de délimiter la notion de transaction. L'interface TxRPC permet d'utiliser des appels de procédure disatnte (RPC) tel que défini par X/OPEN (compatible DCE) et enfin l'interface XATMI contient les primitives permettant de prendre en charge le dialogue client/serveur.

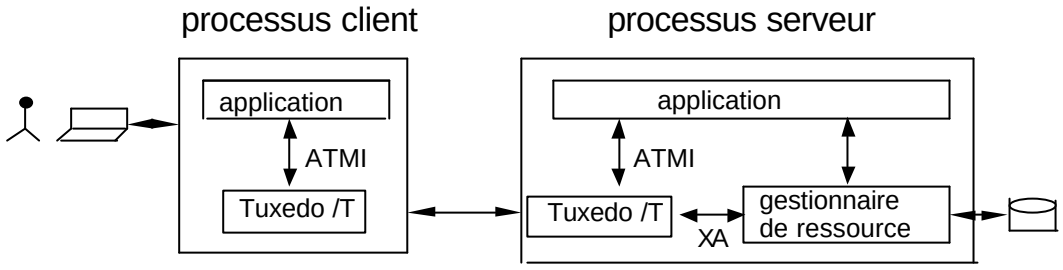


Figure 16. *. L'architecture et les interfaces de TUXEDO*

TUXEDO définit un intermédiaire logiciel entre les applications et les ressources en général. Il est capable de fédérer à l'aide d'interfaces standards (définis dans le modèle DTP de l'X/open) des plates-formes matérielles, des systèmes d'exploitation (DOS, UNIX, . . .), des différents SGBDs (Informix, Oracle, Sybase), le tout dans un ensemble cohérent et transparent.

Le modèle OLTP trouve dans TUXEDO le moniteur transactionnel qui assure le lien entre client et serveur; le modèle client/serveur est ainsi enrichi car TUXEDO introduit un serveur de nom (Name server) entre les deux, pour jouer le rôle de "dispatcher" entre n clients qui demandent m services de p serveurs. Ce produit est écrit en "C" ANSI et est conforme à SVID (System V Interface Definition), POSIX et XPG3 (guides de portabilité de l'X/OPEN).

Le moniteur TUXEDO comporte deux composants bien séparés :

- le SYSTEM /T transaction manager : ce module est basé sur l'architecture client/serveur enrichi et gère toutes les communications client et serveur

- le system /D DBMS : C'est le module de gestion de base de données à haute performance et conforme à l'interface XA. Il offre un ensemble d'outils pour construire et administrer des applications transactionnelles.

La figure précédente montre les processus clients connectés aux serveurs à travers le "Tuxedo /T" et le "Front End". Les avantages de cette interface sont :

- la transparence de la localisation du client par rapport au system /T

- la transparence du terminal : le type du terminal utilisé est sans importance pour le system /T. Il n'y a pas de restriction sur le type de client dans une application donnée

- la transparence de l'échec d'un client : si un client tombe en échec, cela n'affecte pas le reste de l'application

- l'entrée multiplexée : on peut combiner l'entrée de plusieurs clients en une seule entrée si le processus de Front End est programmé pour la manipuler.

La figure montre également les processus serveurs connectés à Tuxedo /T à travers l'interface offerte par le "Transaction manager". Cette architecture offre les fonctionnalités suivantes :

- la localisation du processus serveur est transparent pour l'utilisateur ou le processus client

- un serveur peut offrir plusieurs services et un même service peut être offert par plusieurs serveurs

- on peut attribuer des niveaux de priorités aux services. Les serveurs traitent en premier les requêtes avec les plus grandes priorités

- un groupe de serveurs peut être activé (déplacé) sur une autre machine en cas d'échec d'un noeud et les applications continuent à tourner. Le recouvrement après un échec est accompli sans perte de transactions

- les serveurs d'application sont linkés avec les librairies de TUXEDO, ils sont donc accessibles pour le contrôle par le system /T

- on peut équilibrer les charges du système (Load balancing) en attribuant un facteur de charge à chaque service. Dans ce cas, le system /T choisit le serveur le moins chargé pour satisfaire la requête, ou bien attribuer une même queue à plusieurs serveurs et dans ce cas, la requête est prise en compte par le premier serveur disponible. En fonction de la configuration, ce choix peut améliorer la performance globale du système

- administration de TUXEDO est centralisée à travers un fichier de configuration. Ainsi, toutes les parties de l'application comme les ressources, machines, serveurs, services, etc. sont complètement définies dans ce fichier central

- les ressources de l'application sont protégées contre les accès non autorisés. Les mécanismes de sécurité offerts par TUXEDO utilisent les services d'authentification de Kerberos et ainsi permettent de contrôler les permissions et les droits d'accès aux ressources de l'application.

5.2.1.Les composants de TUXEDO

TUXEDO comprend différents composants parmi lesquels le moniteur transactionnel proprement dit appelé /T ainsi qu'un gestionnaire de données /D. Le système /T de TUXEDO offre une extension pour les PC (/WS) sous DOS ainsi que des extensions (/HOST, /CONNECT et /CONNECT REVERSE) pour les ordinateurs centraux (USL 95) en s'appuyant sur le protocole LU 6.2 (SNA). Les autres composants de Tuxedo sont /DOMAIN, /QUEUE, /RPC et /OSITP.

Le module NET/T de TUXEDO s'appuie sur l'interface socket pour TCP/IP et l'interface TLI (Xti) pour accéder aux fonctions de la couche transport du Modèle OSI pour l'utilisation des services du réseau.

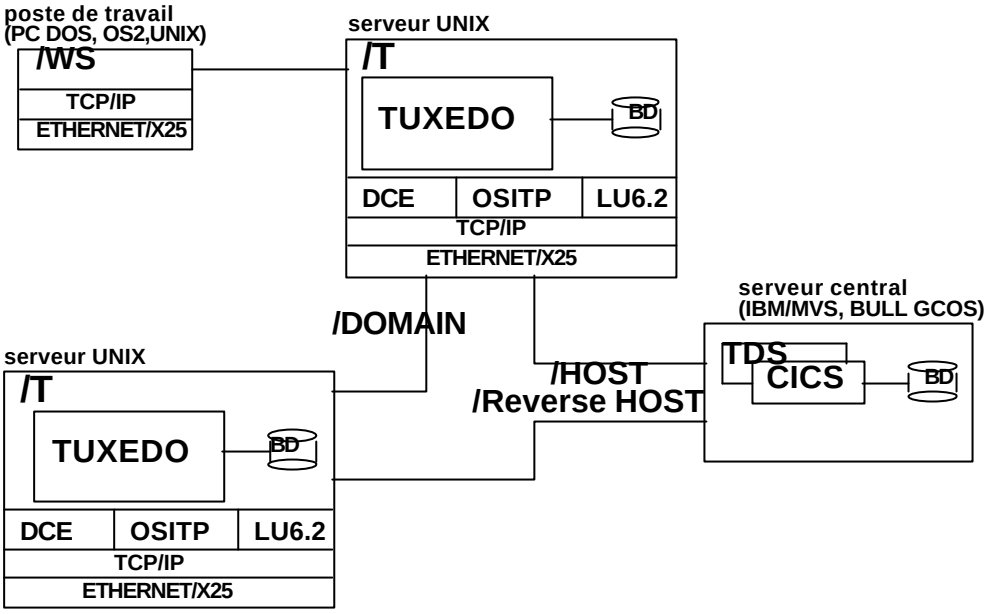


Figure 17. *. Les différents composants de TUXEDO*

- /WS

Cette extension de TUXEDO permet côté client à une application transactionnelle de s'exécuter entièrement ou partiellement à partir d'une station du travail. De cette manière, les modules clients peuvent être déplacés sur des machines intelligentes tournant sous UNIX, MS-DOS, OS2, Windows et Macintosh et réduire la surcharge du système. Cette facilité permet également d'utiliser les possibilités des gestionnaires d'écran existant pour les stations de travail et les PCs.

Le TUXEDO /WS fournit les fonctionnalités nécessaires à l'établissement d'une connexion réseau entre le client et le serveur (interface TLI et socket) et réalise l'encodage/décodage des messages.

- /HOST, /CONNECT et /Reverse HOST

Ces modules étendent le modèle client/serveur dans l'environnement hôte et permettent aussi bien à un client TUXEDO d'accéder aux services et aux données d'un gestionnaire de transactions non TUXEDO en l'occurrence CICS d'IBM qu'à un module s'exécutant dans l'environnement mainframe d'appeler des services applicatifs de TUXEDO..

Ce logiciel fournit une passerelle qui permet la communication entre un noeud du SYSTEM /T et une application tournant sous CICS via le protocole de communication LU 6.2 de SNA en utilisant l'interface normalisée CPI-C.

- /QUEUE

Ce module permet à un client ou à un serveur TUXEDO de poster un message dans une file d'attente sécurisée (sauvegarde sur le disque) pour un traitement ultérieur (traitement asynchrone). Un serveur particulier retire les requêtes de service de ces files d'attente et effectue la demande de service.

- /DOMAIN

Ce module permet de décomposer une application TUXEDO en un ensemble d'applications à travers différents lieux géographiques et de les gérer sous forme de domaines de point de vue administratif.

- /DCE

S'appuie principalement sur TxRPC qui est une extension des RPCs de DCE en mode transactionnel. Dans la version 6, TUXEDO annonce l'intégration des autres services de DCE.

- /OSITP

TUXEDO dans sa dernière version permet l'utilisation du protocole OSITP par son extension /OSITP. De cette manière, TUXEDO se conforme aux spécifications d'ISO dans le cadre du système transactionnel normalisé.

5.3.ENCINA

Le moniteur TP ENCINA de la société TRANSARC est un produit conçu spécifiquement pour la mise en place d'applications transactionnelles distribuées entre des systèmes ouverts.

Il répond complètement aux concepts d'"open OLTP" et conforme au modèle DTP de l'X/OPEN (ENCINA 92).

Transarc a été créé en 1989, par d'anciens chercheurs de l'université de Carnegie Mellon et des anciens ingénieurs d'IBM, et a annoncé son premier produit ENCINA en 1991.

L'importance de produit d'ENCINA, en plus de sa conformité au modèle DTP de l'X/open est le fait qu'il est construit sur la technologie DCE de l'OSF (Transarc est par ailleurs à l'origine du composant DFS de DCE) et de ce fait, lui confère un rôle important dans les années à venir au sein des systèmes ouverts. Les RPC de DCE et le service d'annuaire distribué CDS sont les deux briques de base, permettant d'assurer l'interopérabilité entre les composants d'ENCINA supportant une transaction distribuée entre plusieurs systèmes ouverts.

La caractéristique architecturale majeure du moniteur TP de Transarc est donc son intégration profonde avec DCE, dont il constitue une extension fonctionnelle.

Ce produit est modulaire et simple à mettre en oeuvre et a été choisi par différents constructeurs comme base de leur offre transactionnelle sous UNIX, en particulier IBM, HP et STRATUS.

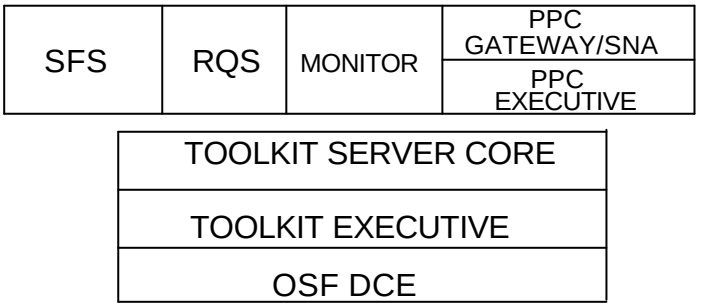


Figure 18. *. L'architecture générale d'ENCINA*

Les composants d'ENCINA

Encina est basé sur deux concepts stratégiques :

1- il étend le concept de DCE pour inclure les services de traitement transactionnel réparti, et la gestion de recouvrement des données.

2- il propose un produit conçu sur un modèle en couches fournissant dans chacune d'elles, des services utilisables par les couches supérieures à travers des APIs bien définies. L'architecture générale d'Encina comprend deux composants appelés respectivement Encina Toolkit Executive et Encina Toolkit Server Core.

Encina TOOLKIT executive étend les services de DCE d'OSF avec l'univers des technologies qui utilisent l'architecture client/serveur.

Il inclut transactional-C, une API qui fournit l'accès aux services TP et résout les problèmes liés aux accès concurrents, enrichit la technologie RPC de DCE pour y inclure la notion d'intégrité transactionnelle, et pour la gestion des données réparties sur plusieurs sites s'appuie sur le protocole de validation à deux phases. Ensemble DCE d'OSF et Encina, permettent à une station de travail d'initialiser une transaction, de localiser les services (à travers le service d'annuaire CDS de DCE et par le mécanisme de RPC) et valider les transactions réparties. Les programmes serveurs sont activés sous forme de "thread" DCE et accèdent à des bases de données externes via SQL.

Encina Toolkit server core étend les services de l'Executive pour supporter le stockage et la maintenance des donnés recouvrables. Il inclut une librairie d'accès aux données, un système de journalisation basé sur des fichiers log et l'interface XA de l'X/OPEN, pour permettre l'interopérabilité des bases de données impliquées dans la transaction.

Transarc a développé sa propre famille de produits TP construit sur DCE de base et Encina Toolkit. Ces produits incluent :

- **Encina Monitor** représente le moniteur TP fournissant un environnement d'administration, de développement et d'intégration d'applications réparties TP.

Encina Monitor fournit également un environnement d'exécution fiable qui gère l'équilibrage des charges (load balancing) ainsi que des services d'ordonnancement (scheduling) à travers des ordinateurs hétérogènes dans le but d'assurer la haute performance et l'intégrité transactionnelle. Il utilise les services d'authentification de DCE pour gérer les problèmes de sécurité dans un environnement réparti et enfin fournit un ensemble d'outils pour la configuration et la gestion de tous les objets (client, serveur, machine, ...) d'un système réparti comme une unité cohésive.

- Structured File Server (SFS) est un système de gestion de fichier orienté "record". Il fournit un accès rapide et sécurisé aux données et permet de respecter la cohérence de celles-ci en cas de problème. SFS est conçu spécifiquement pour le transactionnel sous UNIX et permet de pallier les faiblesses de ce système d'exploitation dans ce domaine en offrant le mécanisme de verrouillage des enregistrements dans le cas d'accès multiples par les serveurs par exemple. En outre, SFS se base sur les mécanismes d'indexation, clustérisation, Btree pour accéder de façon performante aux données et fournit également les interfaces de type ISAM de l'X/OPEN et VSAM.

- Peer - to - Peer communication service (PPC)

Ce module offre la possibilité de communication programme à programme au-dessus de TCP/IP ou transport SNA par le biais de l'interface CPIC de LU 6.2 (unité logique "conversationnelle" dans l'architecture SNA). De cette manière, un module client se trouvant sur une station de travail peut avoir un dialogue de type programme à programme avec un serveur tournant sous CICS sur une machine host d'IBM.

- **Recoverable Queuing Service (RQS)** fournit un mécanisme d'enqueuing/dequeuing des données des transactions. Il permet de mettre en file d'attente les transactions pour être traitées ultérieurement.

5.4.CICS/6000

CICS/6000 d'IBM comme TUXEDO d'USL est conçu pour réaliser et exploiter des applications transactionnelles à deux ou à trois niveaux (PIMONT 95) dans un environnement de systèmes ouverts interconnectés par un réseau hétérogène.

Ce produit offre un environnement relativement complet permettant le développement, l'exécution et l'administration d'applications réparties basées sur le modèle client/serveur.

Il est conforme aux principaux standards du marché et interopère avec les autres produits nécessaires pour mettre en oeuvre ce type d'architecture distribuées (SGBD, gestionnaires d'interface homme/machine,...).

CICS/6000 est un produit assez récent et basé sur des technologies moderne comme DCE.

Ce produit a été développé à l'origine sur les systèmes AIX (RISC 6000) et a été porté par la suite sur des plates-formes HP-UX, ULTRIX et SNI (Simens-Nixdorf) et récemment sur SOLARIS de SUN (PIMONT 95).

Ce moniteur transactionnel respecte le modèle DTP de l'X/OPEN mais a choisi les APIs de la "famille CICS" et vise en priorité les clients d'IBM ayant déjà ce produit sur leurs systèmes centraux. De cette manière, ce produit permet d'enrichir la famille CICS en fonctionnant dans l'environnement RISC 6000 (existe déjà sur MVS, VSE , OS/2 et OS/400).

Toutefois, la spécification du produit montre qu'il s'appuie sur les briques de base du moniteur transactionnel ENCINA de TRANSARC (dont IBM est par ailleurs propriétaire). Ce choix s'explique largement par la stratégie "OSF/DCE" d'IBM qui a annoncé cette infrastructure sur toutes les plates-formes majeures (MVS, AIX, OS/400 et OS/2).

En proposant CICS/6000 dans l'environnement UNIX, IBM a pu commercialiser un moniteur transactionnel bénéficiant de l'ensemble des services de DCE (RPC, annuaire CDS, sécurité KERBEROS,...) et présentant les mêmes interfaces programmatiques que les autres produits de la famille CICS.

Par le fait que le CICS/6000 dispose des mêmes APIs que les autres moniteurs de la famille CICS (ce qui rend les applications portables d'une plate-forme MVS à une plate-forme UNIX) et qu'il interopère avec tous ceux-ci (en protocole SNA/LU6.2), IBM espère convaincre ses grands clients de l'intérêt de son propre moniteur par rapport à des produits comme TUXEDO ou ENCINA.

5.5. Moniteur MQM pour mise en œuvre du modèle MQ

Etude réalisée dans le cadre d'une convention de recherche "Marben" - INSA de lyon - Ministère de l'Industrie.

5.5.1. Propriétés

Un moniteur MQM est un logiciel permettant la mise en œuvre du Modèle Message/Queue dans un environnement hétérogène. Il permet les échanges inter-applications en mode asynchrone à travers une interface programmatique (API) simple d'emploi. Cette interface peut, par exemple être l'interface MQI (Message Queue Interface) définie par IBM.

Un moniteur MQM permet de réaliser simplement des applications transactionnelles ou Client/Serveur et en facilite la portabilité. Il doit disposer d'une fonction d'annuaire ou avoir accès à un tel service pour assurer la transparence de localisation des applications (identifiées par des noms logiques).

Il doit être utilisable sur le système serveur qui le supporte ou accessible à distance grâce à des modules clients sur des stations de travail.

Un moniteur MQM doit pouvoir être utilisé par des applications situées sur le même systèmes ou sur des systèmes distants. Dans ce cas il utilise un protocole transactionnel OSI/TP pour sécuriser les échanges. Pour

une implantation purement locale une sécurisation par un module local spécifique est utilisé pour sécuriser les messages dans les files.

Le moniteur doit assurer **l'intégrité complète et la durabilité de tous les messages** placés dans ses files d'attente. Pour cela ils doivent être écrits sur des mémoire non volatiles (disques en général) et cette écriture est contrôlée par un mécanisme de **journalisation** (logging) qui permet de connaître leur état et d'effectuer les reprises en cas d'incident sur les systèmes (crash machine par exemple).

5.5.2. Architecture

Un moniteur MQM doit comporter:

- Un gérant de files d'attentes
- Une fonction d'annuaire
- Un (ou des) modules Serveur
- Des modules clients
- Une fonction de journalisation/reprise

Ces composants logiciels sont indépendants et peuvent s'exécuter de manière parallèles. Ils seront réalisés par des processus ou de threads.

Le **module de gestion des files d'attente** constitue le noyau du moniteur. Ces files sont placées en mémoire partagées. Ils permet de créer, d'ouvrir, de fermer, de supprimer ces files ainsi que d'y lire ou y écrire des messages. Il dispose pour cela de mécanismes de verrouillages (sémaphores) permettant des accès concurrents à ces files des la part des modules serveurs, clients ou de journalisation.

La **fonction d'annuaire** permet de déterminer l'adresse à utiliser et le protocole nécessaire pour établir un dialogue avec le gérant de la file d'attente ainsi que le nom de la queue à lire ou à écrire dans celui-ci. Pour cela on utilise des identificateurs associés à chaque application et chaque file d'attente. L'annuaire peut aussi fournir des paramètres additionnels sur la qualité du service requis. On peut utiliser une base de données spécifique ou consulter un annuaire externe (X500 ou CDS dans l'architecture DCE de l'OSF).

Le moniteur MQM attend les requêtes de service (lecture ou écriture) des applications clientes. Pour cela il comporte un ou plusieurs modules **Serveurs** qui sont chargés de l'établissement des dialogues et de l'accueil de ces requêtes. Ces serveurs sont spécifiques d'un support de communications (OSI, Inet, SNA, ...).

Pour supporter les échanges directs entre des moniteurs, pour des applications en mode Requête/Réponse ou pour utiliser des systèmes intermédiaires, le moniteur MQM doit disposer de **modules Clients** chargés d'établir les communications avec les autres moniteurs et d'écrire ou lire dans leurs files.

Pour sécuriser les messages et en assurer l'intégrité, ils sont écrits, à la suite d'une requête MQput, dans un fichier journal et identifiés par un nom unique. Lorsqu'ils sont extraits de la file (Requête MQget) un enregistrement de suppression, comportant l'identificateur du message, est écrit dans un fichier journal secondaire. Une fonction de réorganisation, indépendante des fonctions précédentes, permet d'éliminer du fichier journal les messages obsolètes, en utilisant les informations du fichier journal secondaire. Ces fonctions sont réalisées par le composant de **journalisation (logging)**. A la suite d'un incident, après relance du moniteur, la lecture des fichiers journaux permet de reconstruire en mémoire partagée les différentes files d'attente et d'y réinscrire les messages qu'elles comportaient lors de l'incident; c'est la fonction de **reprise** (**recovery**) du composant de journalisation.

5.5.3. Communications entre composants

5.5.3.1.Echanges au sein d'un même système

Les communications entre modules passent par les moyens d'échange offert par le système d'exploitation : tubes, tubes nommés, streams, mémoire partagée, etc..

Chaque application est munie d'une interface MQI et d'un module client. Le moniteur dispose d'un module Serveur chargé de l'accueil des requêtes des clients (et des autres composants décrits ci-dessus)

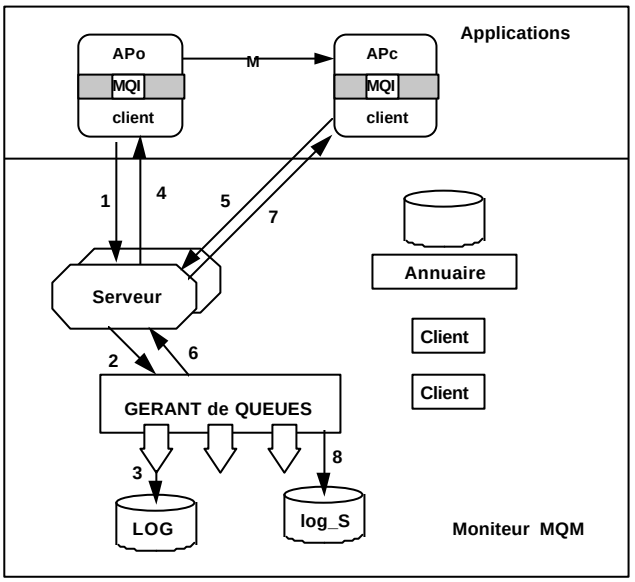


Figure 19

L'application cliente (initiatrice) ouvre un dialogue avec l'un des serveurs du moniteur MCM en utilisant le module client le mieux adapté à ses besoins(tube, tube nommé, socket, stream, etc.) Puis elle **ouvre une file d'attente** par un appel de primitive MQ-OPEN et **écrit un message dans cette queue** par à une primitive MQ-ENQUEUE -Message (MQPUT avec l'API MQI) **(1)**. Si la queue ne préexiste pas elle est créée par les serveur (queue dynamique).

Pour effectuer ces opérations le serveur utilise des primitives internes **(2)** iMQOPEN ou iMQPUT vers le gérant de queues.

Si la qualité "messages sécurisés" est demandée, le message est enregistré dans le fichier journal principal LOG **(3)**. Dans ce cas, l'activité MQ-ENQUEUE se termine **(4)** seulement après écriture du message dans le fichier journal.

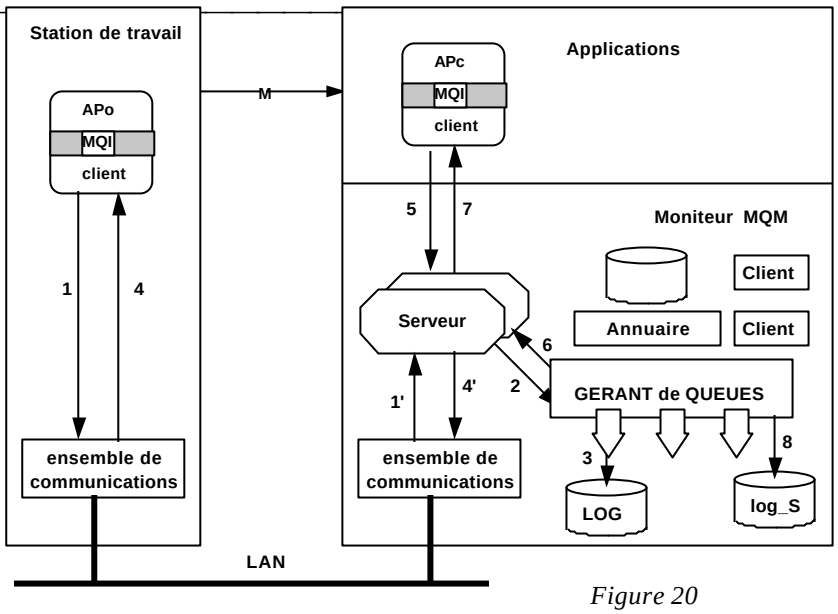


Figure 20

L'application serveur (répondeur) émet une primitive MQ-OPEN-DIALOGUE pour ouvrir un dialogue avec le serveur puis une primitive MQ-OPEN-QUEUE pour ouvrir la file d'attente à travers laquelle elle traite ses échanges. Elle demande la lecture d'un message par un appel de la primitive MQ-DEQUEUE-Message **(5)** (MQGET avec l'API MQI). Si un message est stocké dans la queue il est lu grâce à une primitive interne iMQGET adressée au gérant de queues **(6)**. Ce message est alors transmis au client **(7)** au moyen du dialogue établit auparavant. Si le message est sécurisé, une demande de suppression **(8)** est enregistrée dans le fichier journal secondaire log_S.

Si il n'y a pas de message en attente, le comportement du moniteur MQM dépend de l'option choisie lors de l'appel de la primitive de lecture MQ-DEQUEUE-Message : synchrone (attente bloquante) ou asynchrone (attente non bloquante). Dans le premier cas le serveur ne fait rien d'autre que de noter la demande de lecture en attente. Dans le second il envoie à l'application une réponse signalant que la queue est vide.

5.5.3.2.Echanges entre plusieurs systèmes

D'autres architectures sont susceptibles d'être utilisées. Le schéma ci-dessus montre le déport de l'application cliente initiatrice sur un terminal connecté par un réseau local.

Le schéma suivant montre l'installation du serveur de queues et de son moniteur MQM sur un frontal de communication. Les applications client et serveur sont reliées à ce frontal pal un réseau. Les protocoles de communications vers ces systèmes peuvent être différents (par exemple OSI et TCP/IP)

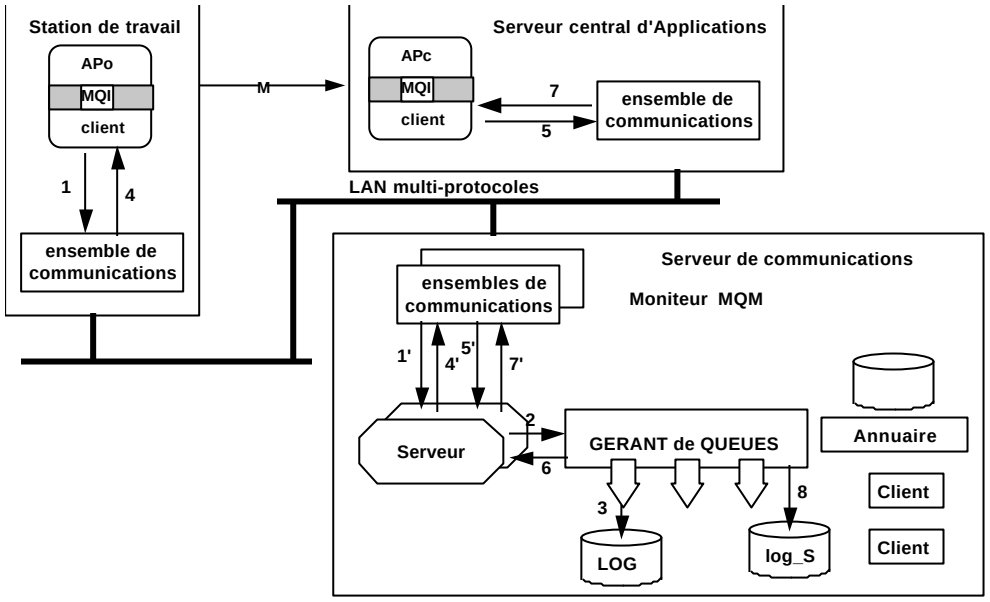


Figure 21

Le dernier schéma illustre une architecture plus générale. Elle met en jeu des systèmes qui peuvent être seulement client ou serveur, des systèmes intermédiaires qui ne sont que serveur de queue et des systèmes qui peuvent être serveur de queue et serveur d'application (en utilisant soit leur propre service de queue soit un service de queues distant.

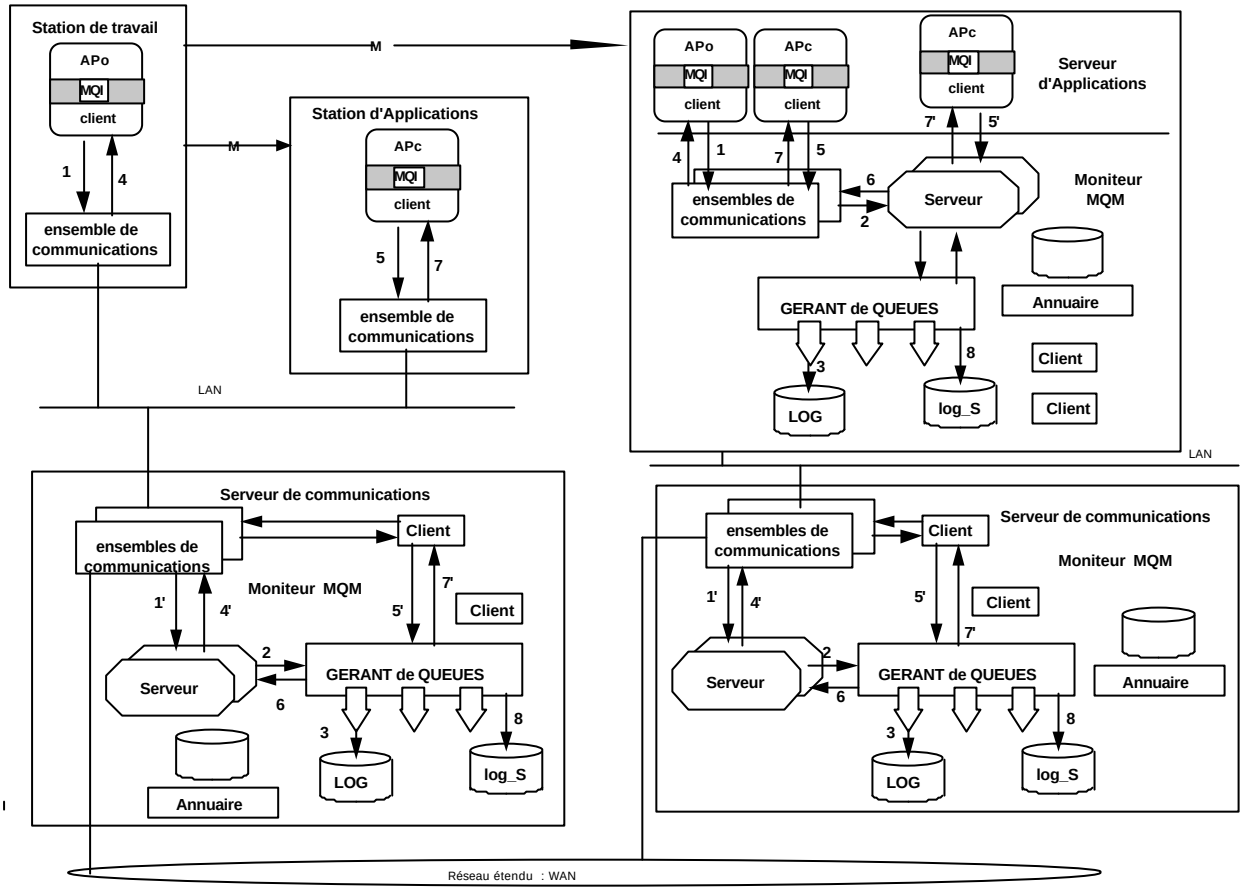


Figure 22

6.Conclusion

La décennie 90 a vu l'apparition et la généralisation du modèle client/serveur. Ce modèle adapté aux réseaux informatiques d'entreprise favorise la fédération des réseaux et des systèmes (avec leur applications) et l'accès par l'intermédiaire d'une station de travail ou d'un PC aux informations de différents systèmes.

Les modèles d'environnement distribué proposés par les grands constructeurs comme SAA d'IBM, NAS de DIGITAL ou DCM de BULL mettent en évidence la généralisation de l'architecture client/serveur dans un environnement ouvert et communicant basée sur les normes et standards. Ce modèle s'oriente vers une interface commune d'accès et de programmation et permettent aux utilisateurs de travailler dans un monde de continuité, de pérennité et d'efficacité.

La tendance actuelle des constructeurs et des utilisateurs par le biais des organismes de normalisation comme OSI ou X/OPEN est d'atteindre des objectifs de "transparence" du système d'information pour l'utilisateur. Cette "transparence" ne peut être réalisée qu'à travers un poste de travail donnant accès parallèlement à un grand nombre d'applications et de bases de données locales ou éloignées.

L'implémentation par les constructeurs des protocoles OSI (FTAM, X.400, OSITP,...) ainsi que les interfaces proposées par l'X/OPEN à un rythme soutenu ainsi que l'interopérabilité de ces environnements hétérogènes montrent la volonté de ces constructeurs de l'ouverture vers le monde extérieur.

Dans ce contexte, du fait des contraintes administratives et géographiques, les données et les traitements sont répartis sur de nombreux systèmes de tous types (personnels, intermédiaires et centraux), dans des lieux souvent différents.

Le défi, par conséquent, est d'être capable de fournir des solutions applicatives (basées sur le modèle client/serveur et/ou coopératif) pouvant être mises en oeuvre dans un grand nombre d'environnement, tout en les rendant accessibles à tout le personnel de l'entreprise.

L'évolution dont nous venons de retracer les grandes lignes, va profondément modifier les conditions d'utilisation des grands ordinateurs traditionnels ainsi que leur bases de données centralisées.

Ces ordinateurs vont devoir se comporter désormais comme de puissants serveurs, mis au service de projets dont ils ne constituent plus le coeur. De la même manière, avec la répartition des traitements et des données obligeant à éclater les informations sur les différents systèmes, les bases de données centralisées (DECISION 96) seront de moins en moins en mesure de répondre aux exigences transactionnelles des entreprises, et cèderont donc la place aux SGBD distribués ou répartis.

7.Annexes :

7.1.TERMINOLOGIE

ACID	Atomicité, Consistance , Isolation et Durabilité, propriétés attribuées à une transaction
ACSE	Association Control Service Element, protocole défini par ISO pour l'établissement d'une association
ANSI	American National Standards Institute
AP	Application Program, un des "composants" d'un processus client ou serveur
API	Application Program Interface, interface de programmation
ASE	Application Service Element, éléments de service définis par ISO au niveau de la couche application
ATMI	Application Transaction Manager Interface, interface programmatique offerte par le moniteur transactionnel TUXEDO
CCR	Committement, Concurrency and Recovery, protocole de validation et reprise de transactions défini par l'ISO
DCE	Distributed Computing Environment, architecture d'environnement distribué définie par l'OSF
DDA	Distributed Data Access, accès aux bases de données distribuées
DFS	Distributed File Service, service de fichiers distribués
DIS	Draft International Specification
DTP	Distributed Transaction Processing, modèle de référence de l'organisme X/OPEN pour le traitement transactionnel réparti
EDI	Exchange Data Interchange, Echange de Données Informatiques
FTAM	File Transfer, Access and Management, la norme pour le transfert de fichiers définie par l'ISO
IPM	Inter Personal Message, messagerie interpersonnelle
IS	International Specification (norme)

ISO International Standard Organization, organisme de normalisation

MHS Message Handling System, système de messagerie

OLTP On Line Transaction Processing, traitement transactionnel en ligne

OSF Open Software Foundation, consortium crée par les constructeurs comme IBM, BULL, HP,...

OSITP OSI Transaction Processing, norme pour le transactionnel réparti définie par ISO

RM Resources Manager, gestionnaire de ressources (gestionnaire d'une base de données par exemple)

RPC Remote Procedure Call, procédure d'appel distant

SC Serveur Central

SGBD Système de Gestion de Base de Données

SL Serveur Local

SNA System Network Architecture, architecture de réseau IBM

SQL Structured Query Language, le langage standard d'accès à une base de données

ST Station de Travail

SVID System V Interface Definition, l'interface définie par AT&T

TWO PHASE COMMIT protocole de validation en deux phases

TM Transaction Manager, gestionnaire de transactions

TP Transaction Processing, traitement transactionnel

TPPM Transaction Processing Protocol Machine, machine protocolaire d'un système transactionnel

TPS Transaction Par Seconde, unité de mesure de la performance d'un système transactionnel

Two Phase Locking verrouillage en deux temps

TPSU Transaction Processing Service User, processus offrant un service transactionnel

TPSUI Transaction Processing Service User Invocation, une invocation de processus transactionnel

XA interface/protocole définie par X/OPEN pour le dialogue entre le gestionnaire de transactions et le gestionnaire de ressources

XATMI Application Transaction Manager Interface, interface programmatique proposée par X/OPEN dans le cadre d'un système transactionnel

X/OPEN organisme de standardisation crée par les utilisateurs, les éditeurs de logiciels et les constructeurs

XPG3 guides de portabilité définis par l'X/OPEN

7.2.BIBLIOGRAPHIE

- (UNISYS 89) Traitement transactionnel en ligne sur systèmes ouverts (UNISYS France 1989)

- (AFUU 89) Tribunix Bulletin de liaison de L'AFUU no 24 Fevrier 1989

- (GHERNAOUTI 90) S. Ghernaouti : Réseaux : applications réparties normalisées 1990 Eyrolles

- (DRAFT-ISO 90) Distributed Transaction Processing (DRAFT ISO /IEC 1990)

- (ATT 90) TUXEDO system transaction manager (document AT&T 1990)

- (DTP 91) Distributed Transaction Processing (X/OPEN 1991)

- (DESAINTQUENTIN 91) J.M DESAINTQUENTIN : L'informatique éclatée 1991 MASSON

- (OSF 91) Documentation technique de DCE (document OSF 1991)

- (BARBOT 91) The OSI - TP protocol : H. Barbot (BULL), G. Lacoste (IBM),S. sedillot (INRIA)

- (MARTIN 92) Daniel Martin : La performance transactionnelle 1992 Masson

- (CLAYBROOK 92) B. Claybrook : On Line Transaction Processing Systems 1992 Wiley

- (PIMONT 92) Systèmes distribués et traitement transactionnel J. L. Pimont, MARBEN 1992

- (IBM 92) The OSF DCE for The IBM MVS /ESA SYSTEM (document IBM 1992)

- (BULL 92) Distributed Computing Model (document Bull 1992)

- (ENCINA 92) ENCINA from TRANSARC (document Transarc corporation 1992)

- (GRAY 93) J. GRAY : Transaction Processing:concepts and techniques 1993 M.Kaufmann

- (ROSENBERG 93) W. Rosenberg, D. Kenney : Comprendre DCE 1993 Addison - vesleg

- (MARBEN 93) OSI Transaction Processing (document Marben 1993)

- (BERSON 94) A. BERSON : Client/Serveur Architecture 1994 McGraw-Hill

- (USL 95) TUXEDO system 5, Description technique d'ensemble document USL 1995)

- (PIMONT 95) Le client/serveur avec les technologies Internet J. L. Pimont MARBEN 1995

- (DECISION 96) Decision Micro & Réseaux no 266 9 septembre 1996

Articles sur Internet

- client-serveur goes business critical, Standish group int., webmaster@beasys.com

- On Line Transaction Processing, Simens Nixdorf & informations system, franz.dielmann@mch.sni.de, /public/mr/oltp2/oltphome.htm, 29 March 96

- Oracle powers world's largest OLTP client/serveur for Europcar, http://www.uk.oracle.com, /europe/emea/info/news/europcar.html, Jul 1996

- OLTP : Tuxedo conference 96 Overview, BEA systems Inc. http://www.beasys.com, /Oltp/tux96e3.htm, 23 May 96

- OLTP : Tuxedo conference 96, BEA systems Inc. http://www.beasys.com, /Oltp/tux96e2.htm, 21 June 96

- Transarc revises ENCINA OLTP monitor, Unix News Int. no 8, 5 octobre 1995, http://apt.usa.globalnews.com, /UNI/iss8/news2.htm

- On Line Transaction Processing, Data Management & Transaction Processing, http://www.mch.sni.de, /public/mr/oltp2/oltphome.htm, 10 June 96

- Encina Base Services, Introduction the Open Software Foundation (OSF) Distributed Computing Environment (DCE) provides a set of powerful, http://www.transarc.com, /afs/transarc.com /public/www/public/ProdServ/Product/Encina/base_s, 6 June 96

∞

7.3.L'environnement distribué

7.3.1.Le Modèle client/serveur généralisé

L'étude comparative des différents modèles proposés par les grands constructeurs (DCM chez BULL, NAS chez DEC, SAA d'IBM , NEWWAVE chez HP,...), pour faciliter la mise en oeuvre des systèmes distribués, fait apparaître un modèle commun appelé "**modèle client - serveur généralisé**" (PIMONT 92) et qui se caractérise de la manière suivante :

Au niveau le plus bas, le réseau d'entreprise permet d'assurer l'interconnectivité physique des différents systèmes (en mettant en oeuvre différentes technologies : 802.3, 802.5, FDDI pour les réseaux locaux ou X25, RNIS, Frame Relay,... pour les réseaux étendus).

Chaque système, (c'est-à-dire une station de travail ou un serveur) comporte un service de communication supportant différents protocoles (principalement OSI, SNA ou TCP/IP) qui lui permet de communiquer à travers le réseau d'entreprise avec la couche homologue d'un système distant.

Au dessus de cette "**couche protocole**", une "**couche service**" (couche au sens du terme informatique) permet d'offrir à travers des interfaces programmatiques standards (API) les différents services dont ont besoin les applications distribuées. Ces services, appelés globalement "services de distribution et d'intégration d'application" sont principalement :

- le partage de fichier dans un réseau local (DFS : Distributed File Service)
- l'accès transparent à des bases de données (DDA : Distributed Data Access)
- le traitement transactionnel coopératif (DTP : Distributed Transaction Processing)
- la messagerie et l'échange électronique de données (MHS/EDI:Message Handling Service/Electronic Data Interchange)

- les transferts de fichiers (FT : File Transfer)
- l'émulation des terminaux (TE : Terminal Emulation)

Ces services de distribution et d'intégration utilisent bien entendu les services fournis par la couche de protocole pour faire communiquer, à travers le réseau physique, une application répartie entre un client et un serveur et plus généralement entre un client et plusieurs serveurs.

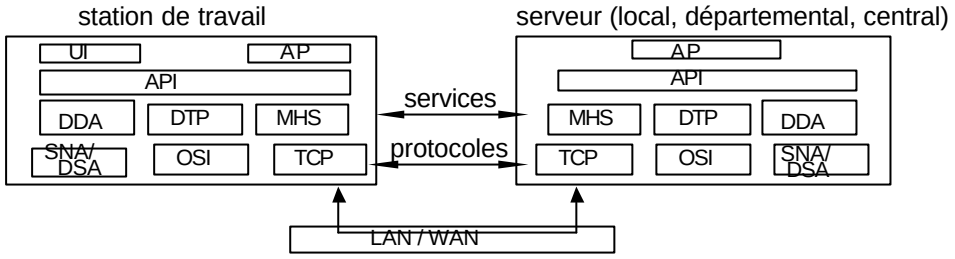


Figure 9. *. Le modèle client/serveur généralisé*

7.3.2.Le modèle conceptuel du réseau informatique

La conception de l'architecture générale rend nécessaire la définition d'un modèle simple permettant de schématiser l'ensemble du réseau informatique et de faire apparaître d'une part les principaux composants, d'autre part les relations existantes entre ceux-ci. Le réseau d'entreprise permet les types d'échanges suivants :

- une station de travail (ST) d'un réseau local peut communiquer avec :
  - le serveur local (SL de son réseau d'établissement)
  - le serveur local (SL d'un autre établissement)
  - le serveur départemental
  - le serveur central (SC)

Ces échanges peuvent s'effectuer selon les besoins des applications en mode :

- émulation de terminal
- transfert de fichiers
- communication de programme à programme (DDA, DTP)
- messagerie (MHS) de personne à personne (IPM) ou entre applications(EDI)

Ainsi, une station de travail (ST) peut être utilisée :

• en émulation de terminal 3270 pour accéder à une application CICS d'un serveur central écrit pour ce type de terminal

• comme poste de travail pour accéder à une application transactionnelle distribuée (DTP) ou à une base de données distantes (DDA) ou encore à la messagerie interpersonnelle (MHS IPM).

Un serveur peut communiquer avec un autre serveur en utilisant des services de :

• transfert de fichiers

• communication d'applications à applications en mode

• transactionnel coopératif synchrone (DTP) ou

• messagerie (transactionnelle asynchrone)

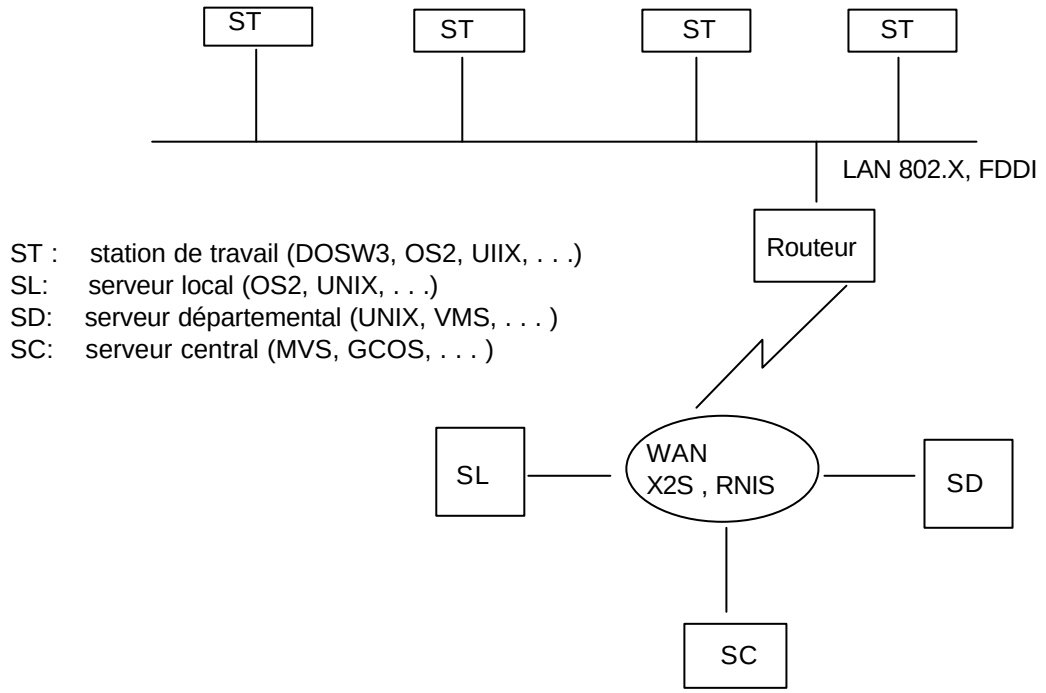


Figure 20. *. Le modèle architectural à trois niveaux*

En fonction du modèle cité ci-dessus, aux architectures transactionnelles centralisées héritées des années 70/80, se substituent déjà depuis quelques années des architectures TP "client/serveur" à 2 ou à 3 niveaux (PIMONT 95) mettant en oeuvre :

- soit des postes de travail et des serveurs centraux UNIX ou propriétaires (TP à 2 niveaux),

- soit des postes de travail, des serveurs locaux/ et/ou départementaux et des systèmes centraux (TP à 3 niveaux).

Dans ces nouvelles architectures transactionnelles, les fonctions applicatives sont réparties entre les plates-formes:

- les postes de travail intelligents assurent l'interface homme/machine et une partie de traitement,

- les serveurs applicatifs effectuent le reste des traitements et l'accès aux bases de données.

Les postes de travail sont le plus souvent des "PC" (Windows ou OS2) connectés par un réseau local (Ethernet ou Token Ring) à un serveur d'établissement (NT, OS2 ou UNIX) et ce serveur local est lui-même relié à un système central UNIX ou propriétaire (IBM/MVS, BULL/GCOS),..) par un réseau généralement longue distance.