

# Transactionnel et transactionnel réparti

*Mars 2000*

**René J. Chevance**

- **Introduction**
- **Concept de transaction - Propriétés ACID**
- **Caractéristiques du transactionnel**
- **Rôle d'un moniteur transactionnel**
- **Composantes d'un moniteur transactionnel**
- **Modèle de moniteur transactionnel**
- **Moniteur transactionnel**
- **Threads**
- **Modèle X/Open**
- **Transactionnel réparti - commitment à deux phases**
- **Exemple de moniteur transactionnel: Tuxedo**
- **Moniteurs transactionnels et SGBDs**

- **Le transactionnel est une dimension essentielle des systèmes d'information des entreprises**
- **Un système transactionnel (OLTP On Line Transaction Processing) fournit un cadre pour les applications critiques, il est fiable et à haute performance**
- **Les besoins transactionnels ont conduit les constructeurs à développer des systèmes ou des sous-systèmes spécifiques:**
  - **Spécifique: TPF(IBM Mainframe) pour des systèmes très spécialisés (e.g. système de réservation de la TWA)**
  - **IBM: CICS sous-système transactionnel pour mainframe (environ 38000 installations) maintenant porté sur des systèmes UNIX (e.g. CICS/6000). Produits compatibles sous UNIX tels qu'UNIKIX (Integris/Bull)**
- **Bull: TDS sur Mainframe, DEC: ACMS,....**
- **Tandem: Pathway/Guardian pour ses systèmes à continuité de service**
- **USL puis Novell: Tuxedo pour systèmes UNIX, AT&T GIS: Top End pour UNIX**
- **Transarc: ENCINA pour UNIX**

# Concept de transaction - Propriétés ACID

- **Rappel, on appelle transaction une séquence d'actions sur l'état physique et logique d'une application qui respecte les propriétés suivantes dites ACID (Atomicity, Consistency, Isolation, Durability):**
- **Atomicité: Les changements opérés par une transaction sur l'état sont atomiques: ils sont tous exécutés ou bien aucun ne l'est;**
- **Consistance: Une transaction est une transformation correcte de l'état. L'ensemble des actions accomplies par la transaction ne viole pas les contraintes associées avec l'état. Ceci implique que la transaction soit un programme correct;**
- **Isolation: Bien que les transactions s'exécutent de façon concurrentes, il apparaît, à chaque transaction que les autres transactions, se sont exécutées soit avant soit après;**
- **Durabilité: Lorsqu'une transaction se termine avec succès (commitement), le changement qu'elle a provoqué sur l'état doit survivre aux défaillances.**

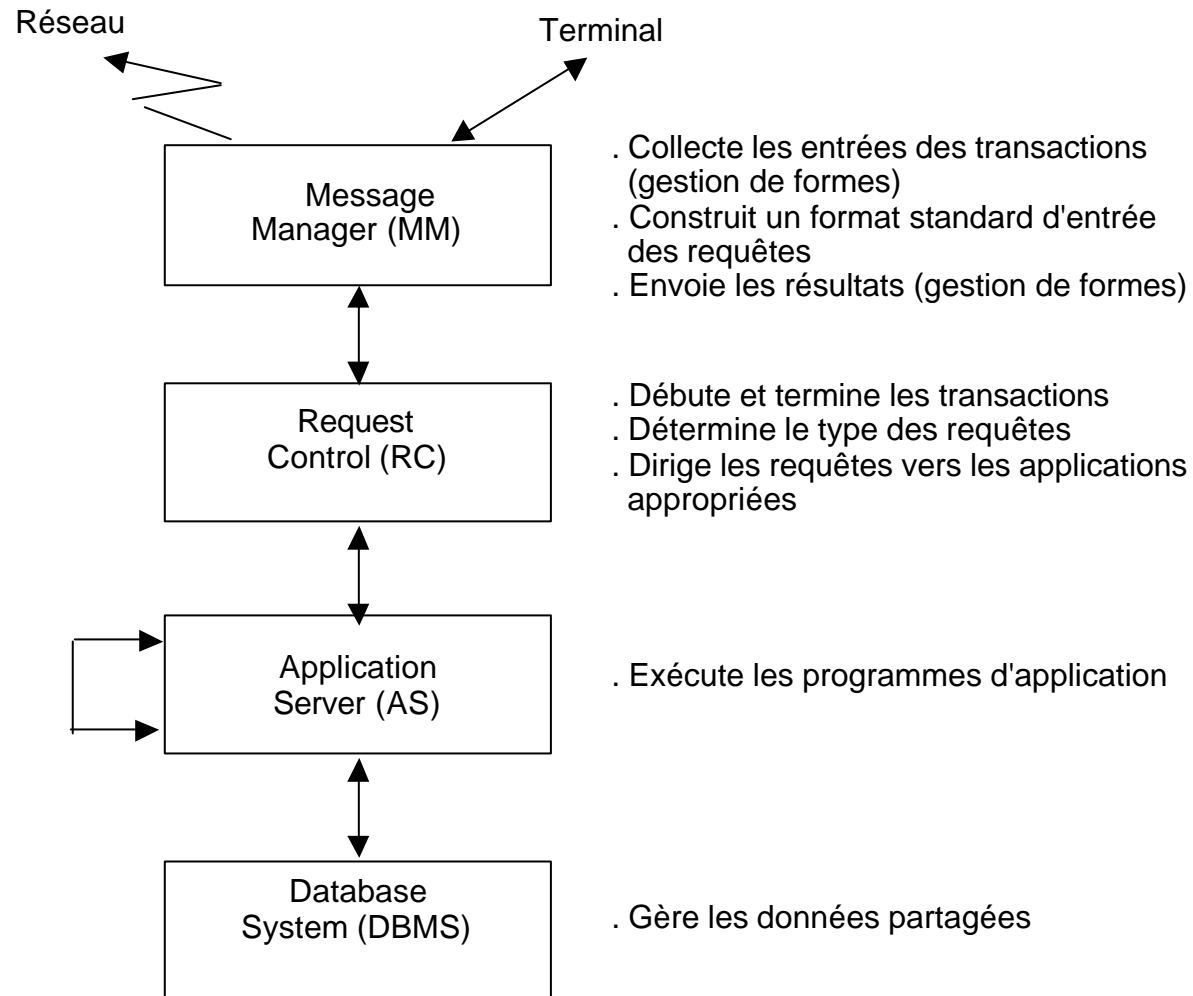
# Caractéristiques du transactionnel

- **Partage:**
  - en Lecture et Écriture
  - par l'ensemble des utilisateurs
  - Propriétés ACID
- **Flux de requêtes irrégulier**
- **Travail répétitif**
  - Répertoire de fonctions pré-défini typiquement O(100) fonctions
- **Fonctions simples**
  - Fonctions peu complexes (typiquement de  $10^5$  à  $10^7$  instructions et 10 E/S)
- **Possibilité de traitement de type batch (avec respect des propriétés ACID)**
- **Grand nombre de terminaux (1000-10000)**
- **Clients intelligents (stations, PC, autres systèmes, terminaux)**
- **Haute disponibilité requise**
  - Recouvrement effectué par le système
  - Fondé sur les propriétés ACID
- **Taille des bases de données**
  - Proportionnelle à l'activité de la Société
- **Peu de données "touchées" par une transaction**
- **Équilibrage de charge automatique**
- **Recherche de la performance au moyen du parallélisme inter-requête**
- **Performance : haut débit et temps de réponse garanti**
- **Scalabilité : exigence typique**

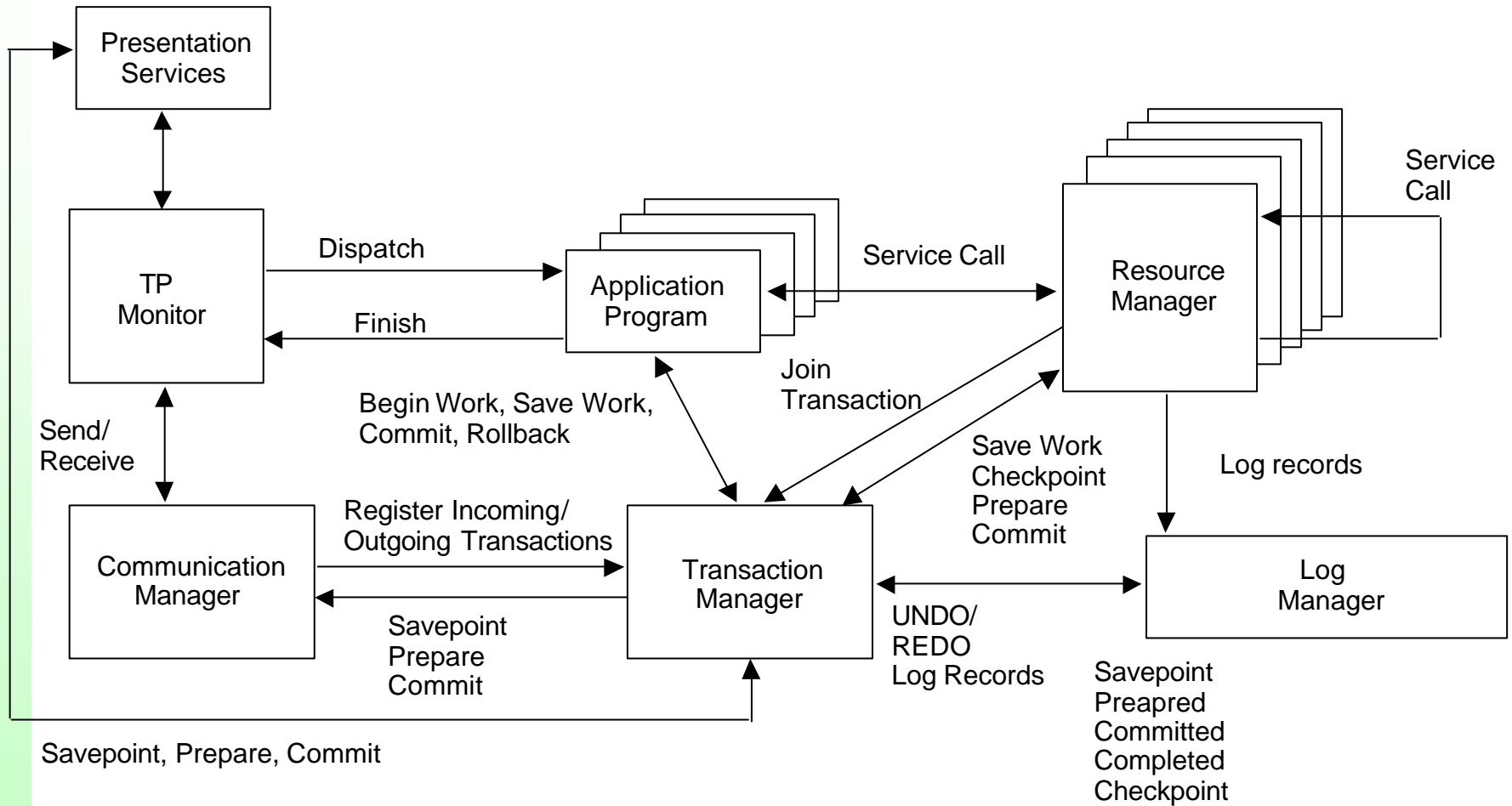
# Rôle d'un moniteur transactionnel

- **Peu de systèmes d'exploitation ont été conçus dans l'optique du transactionnel**
- **Le support d'un grand nombre d'utilisateurs et d'un flux important de transactions (plusieurs milliers par seconde) provoque un effondrement des systèmes**
- **Le rôle d'un moniteur transactionnel est de :**
  - **Gérer des processus comprenant le lancement des applications, le contrôle de leur déroulement et l'équilibrage de charge (on peut parler de multiplexage des requêtes sur les ressources du système)**
  - **Gérer des transactions (respect des propriétés ACID) dans un contexte, éventuellement distribué, mettant en jeu plusieurs gestionnaires de données)**

# Modèle de moniteur transactionnel



# Composants d'un moniteur transactionnel

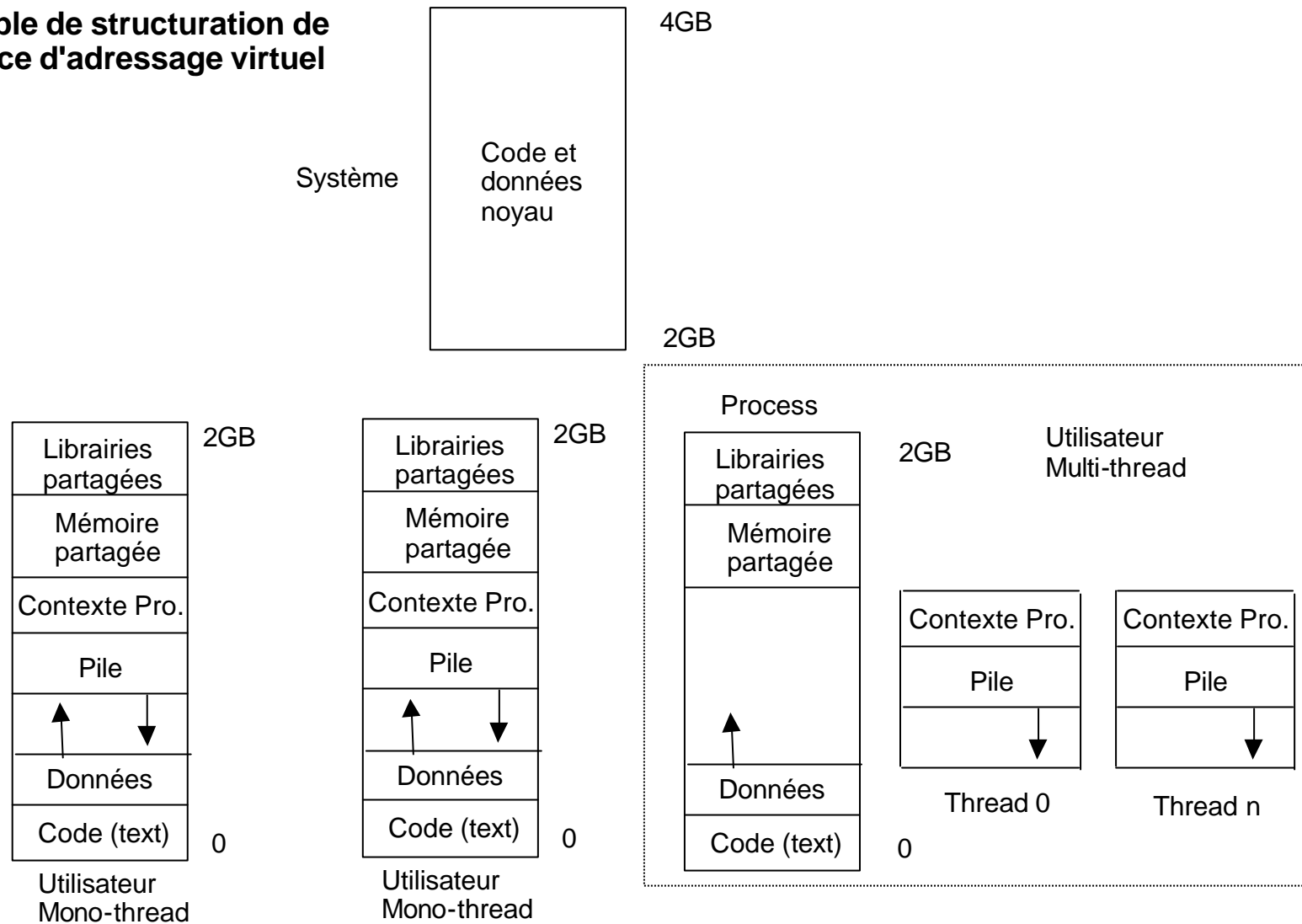


# Concept de threads (processus légers)

- La gestion d'un grand nombre d'utilisateurs connectés et d'un grand nombre de transactions actives dépasse, bien souvent, les capacités de traitement des systèmes d'exploitation et du matériel les supportant
- Il convient d'alléger la gestion des contextes des utilisateurs, c'est-à-dire d'éviter d'avoir un processus par utilisateur (trop de processus conduit à un effondrement du système)
- Solution: les "threads", ou chemins d'exécution indépendants au sein d'un même processus. Ceci correspond au multiplexage de processus "légers" au sein d'un processus. Une commutation de thread au sein d'un processus coûte environ 10 fois moins de temps qu'une commutation de processus
- Le processus est l'unité d'allocation de ressources du point de vue du système: protection, espace mémoire, fichiers, connections réseau,.....Les threads partagent les ressources au sein du processus. Bien évidemment, l'accès aux ressources partagées nécessite une synchronisation
- Les threads sont supportés soit au niveau du système d'exploitation (e.g. implémentation de la norme POSIX 1003.4a) soit au sein des sous-systèmes eux-mêmes (e.g. moniteurs transactionnels, systèmes de gestion de bases de données,..)

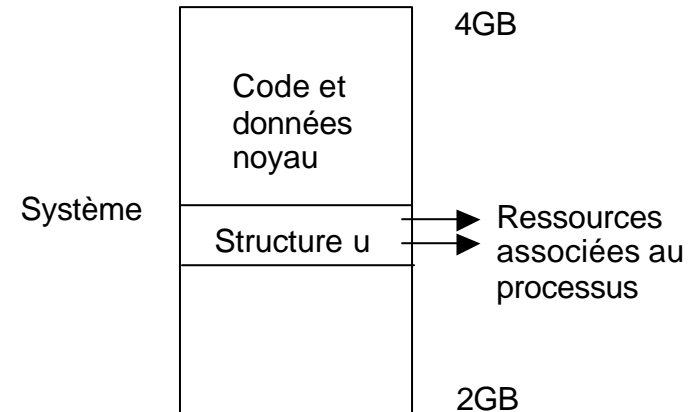
# Notion de processus (Unix) et de thread

**Exemple de structuration de l'espace d'adressage virtuel (Unix)**



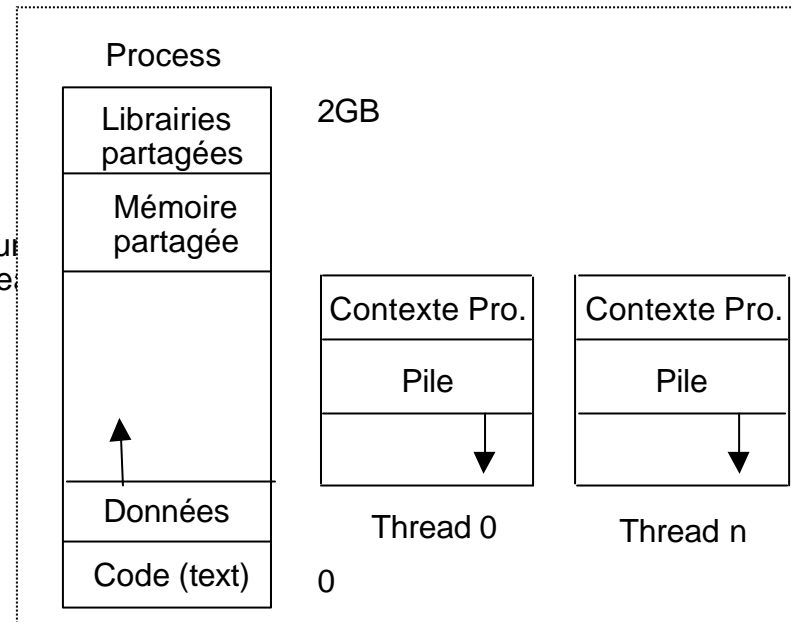
# Notion de thread (Unix)

**Partage des ressources "système »  
entre les threads d'un processus**

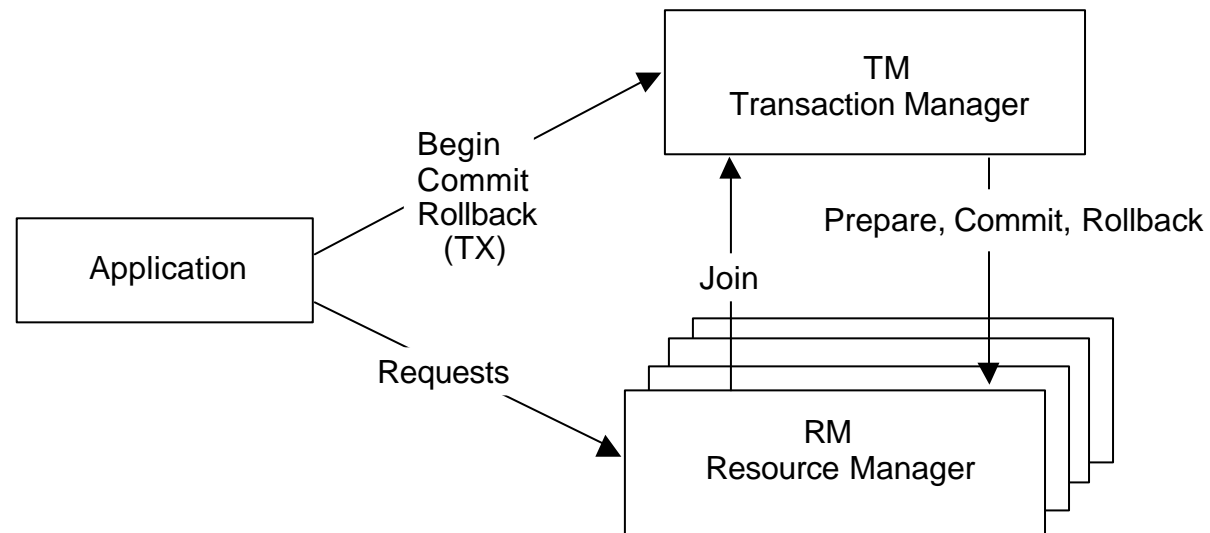


Utilisateur  
Multi-thre

**Notes:**  
 - Deux threads "système" sont automatiquement associés à un processus multi-thread pour la gestion des threads "utilisateur » (exemple scheduling, signaux,...)  
 - L'accès aux données communes au niveau du processus nécessite une synchronisation entre les threads

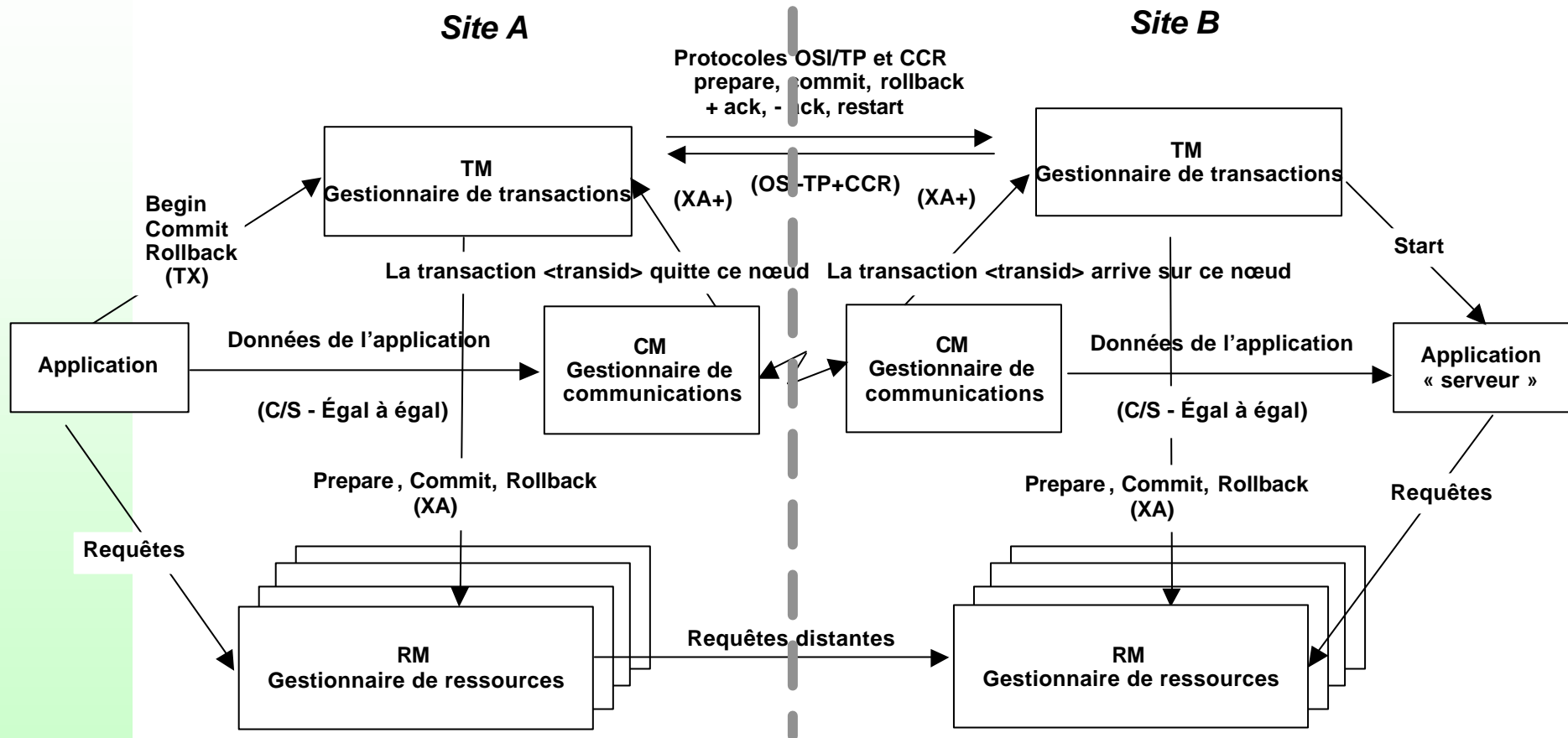


## ■ Transactionnel centralisé



***Le modèle X/open suppose que les Resource Managers ont leurs propres services de log et de verrouillage (lock) et que ces Resource Managers réalisent leurs propres reprises (rollback) à la demande du Transaction Manager***

## ■ Transactionnel distribué (DTP)



Note: TM et CM peuvent être intégrés

## ■ Standards

Eléments participant	Protocole/API	Organisme
Application:TM	TX	X/Open DTP
Application:RM	spécifique du RM	fournisseurs des RMs
Application:Serveur	C/S ou Peer to Peer	OSI + application
TM:RM	XA	X/Open DTP
TM:CM	XA+	X/Open DTP
TM-TM	OSI-TP + CCR	OSI

- OSI définit des protocoles et des formats (FAP - Format And Protocols). Ceci est nécessaire pour l'interopérabilité.

\_ X/Open définit des interfaces de programmation API (Application Programming Interface). Ceci est nécessaire pour la portabilité.

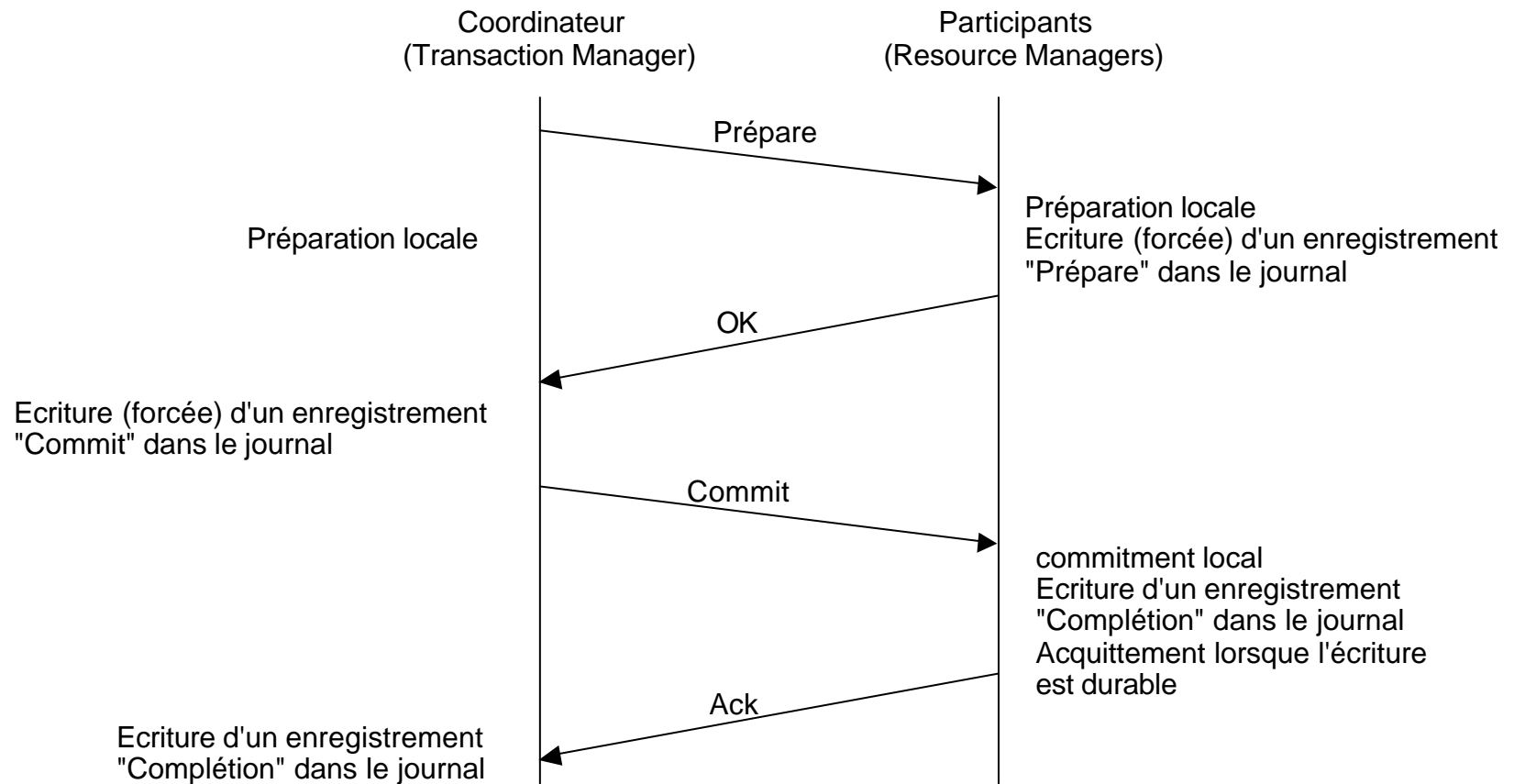
- C/S: Client/Seveur qui utilise souvent un RPC (Remote Procedure Call) spécifique

- Peer to Peer: dialogue d'égal à égal

- CR: Commit, Concurrency Control and Recovery

Note: OSI-TP est similaire à LU6.2 qui est un standard (FAP) de fait relatif aux interactions entre clients et serveurs dans un environnement transactionnel.

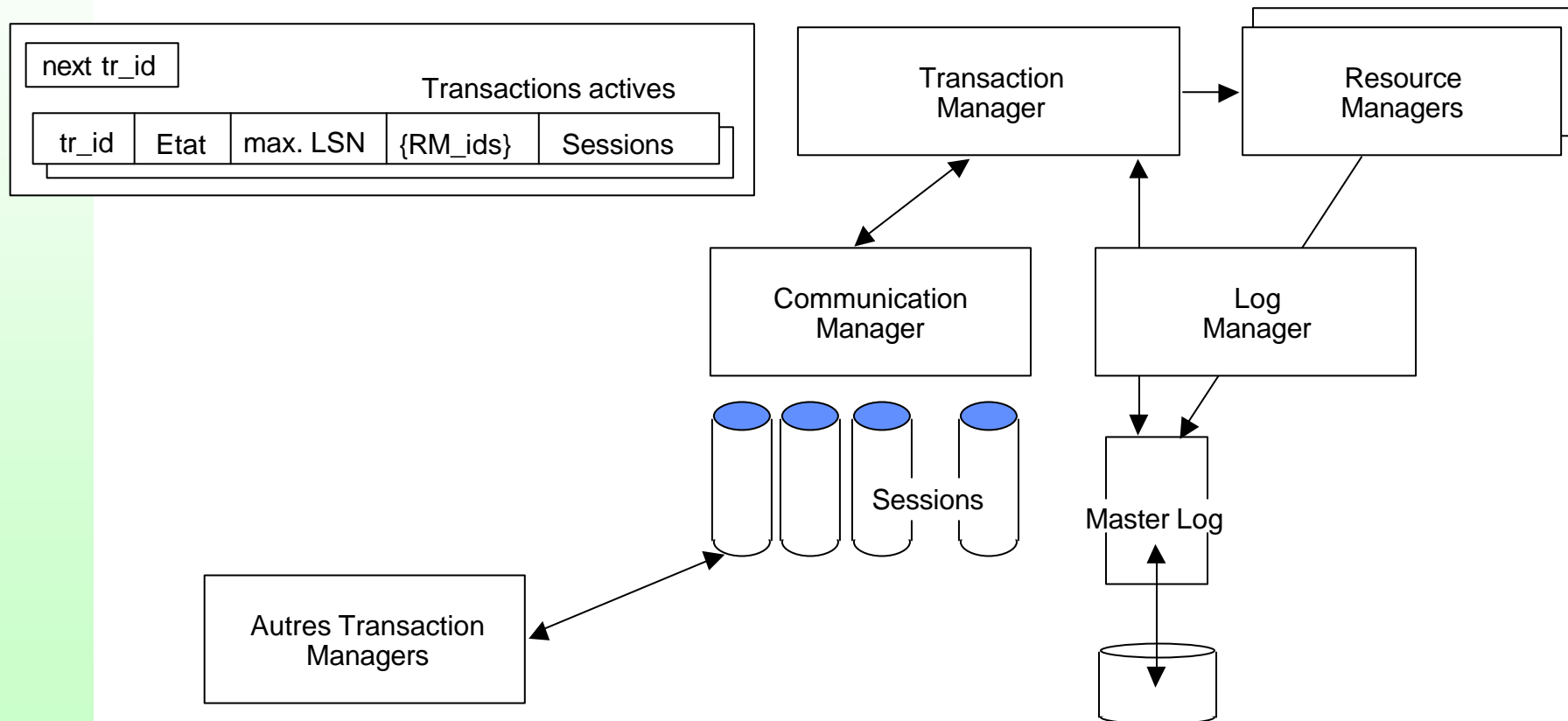
## ■ Cas centralisé [GRA93]



# Validation à deux phases(2)

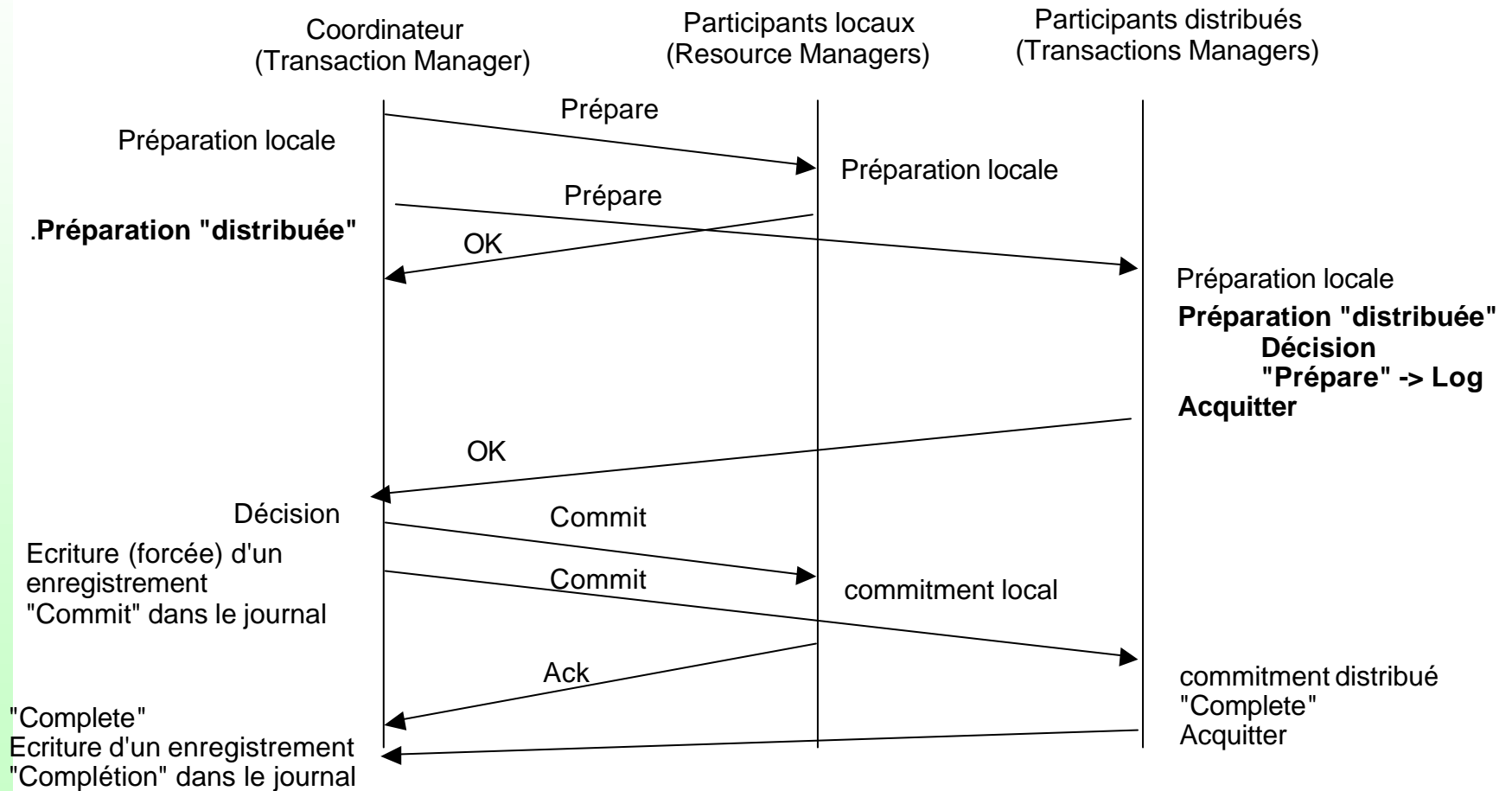
## ■ Cas distribué [GRA93]

Etat courant



# Validation à deux phases(3)

## ■ Cas distribué (suite)



## ■ Cas distribué (suite)

### Transaction Manager "Coordinateur":

#### Commit

- . **Préparation locale** : "prépare" envoyé à chaque RM local
- . Préparation "distribuée" : "prépare" envoyé à chacune des sessions impliquées dans la transaction (en fait des TMs)
- . **Décision** : Si tous les RM locaux et les sessions impliquées répondent OK, écriture d'un enregistrement "commit" avec toutes ces informations dans le journal)
- . **Commit** : Envoi de l'ordre "commit" à tous les RM locaux et aux TM des sessions concernées
- . **"Complète"** : Si tous les RM locaux et les toutes les sessions impliquées (les TMs) répondent positivement écriture (forcée) d'un enregistrement "complétion" dans le journal. Après la fin d'écriture, mise à jour de l'état de la transaction ("finie")

#### Abort

- . **"Broadcast Abort"** : envoyer le message "abort" à toutes les sessions concernées
- . **Défaire** : défaire la transaction à l'aide des informations du journal
- . **"Complète"** : Ecriture d'un enregistrement dans le journal et mise à jour de l'état de la transaction

### Transaction Manager "Participant":

#### Prépare()

- . **Préparation locale** : "prépare" envoyé à chaque RM local
- . **Préparation "distribuée"** : "prépare" envoyé à chacune des sessions impliquées dans la transaction (en fait des TMs)
  - **Décision** : Si tous les RM locaux et les sessions impliquées répondent OK, le TM est quasi prêt
  - **Préparé** : écriture d'un enregistrement "commit » avec les informations (RMs et TMs participants ainsi que le TM "parent" dans le journal)
- . **Réponse** : répondre positivement au demandeur
- . **Attente** : attente d'un ordre "commit" en provenance du coordinateur.

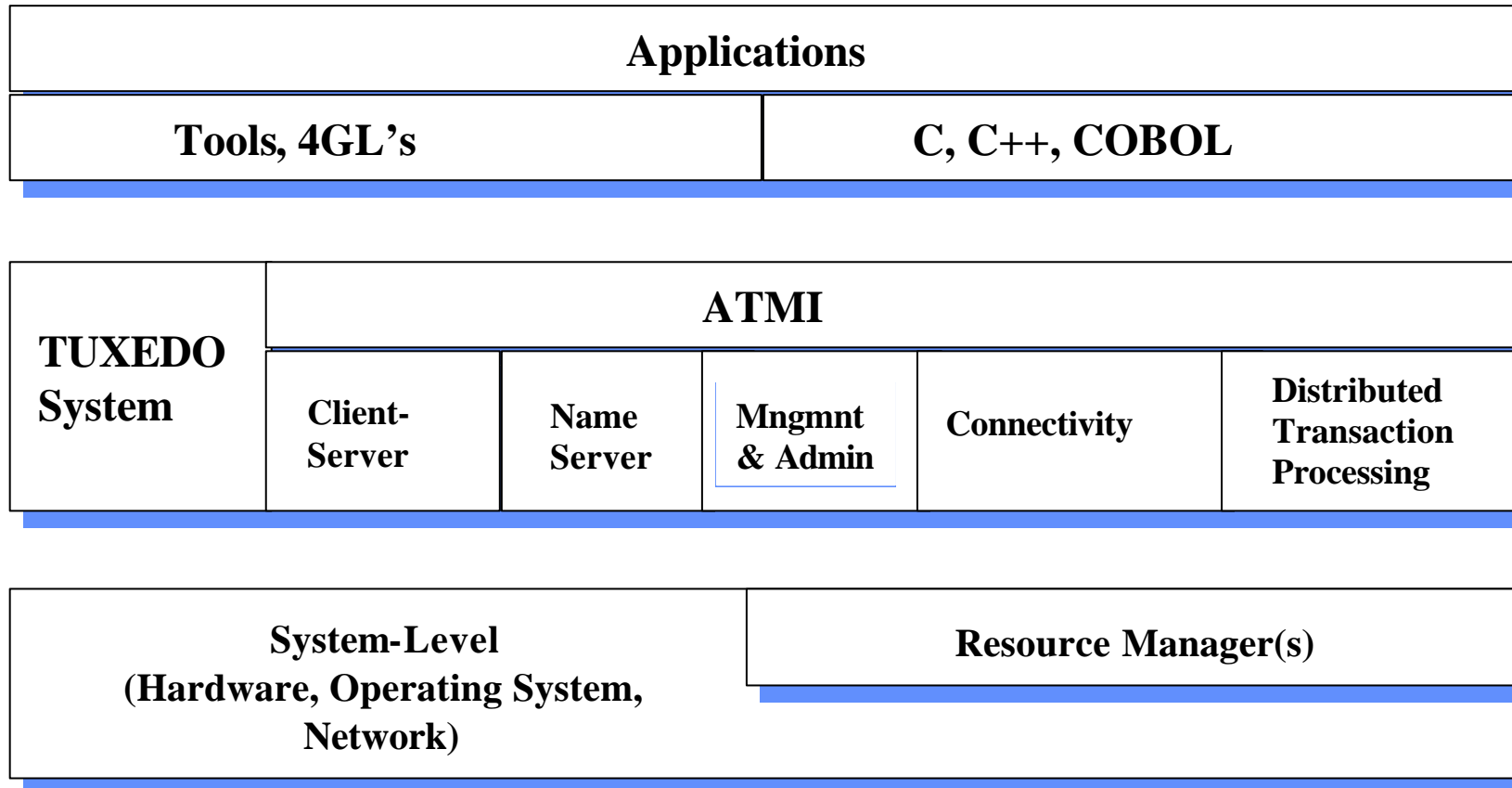
#### Commit()

- . **Commit** : Envoi de l'ordre "commit" à tous les RM locaux et aux TM des sessions concernées
- . **"Complète"** : Si tous les RM locaux et les toutes les sessions impliquées (les TMs) répondent positivement écriture (forcée) d'un enregistrement "complétion" dans le journal. Après la fin d'écriture, mise à jour de l'état de la transaction ("phase 2 terminée")
- . **Acquittement** : après l'écriture dans le journal, envoyer un acquittement du commit au coordinateur et mettre à jour l'état (local) de la transaction

# Exemple de moniteur transactionnel : Tuxedo

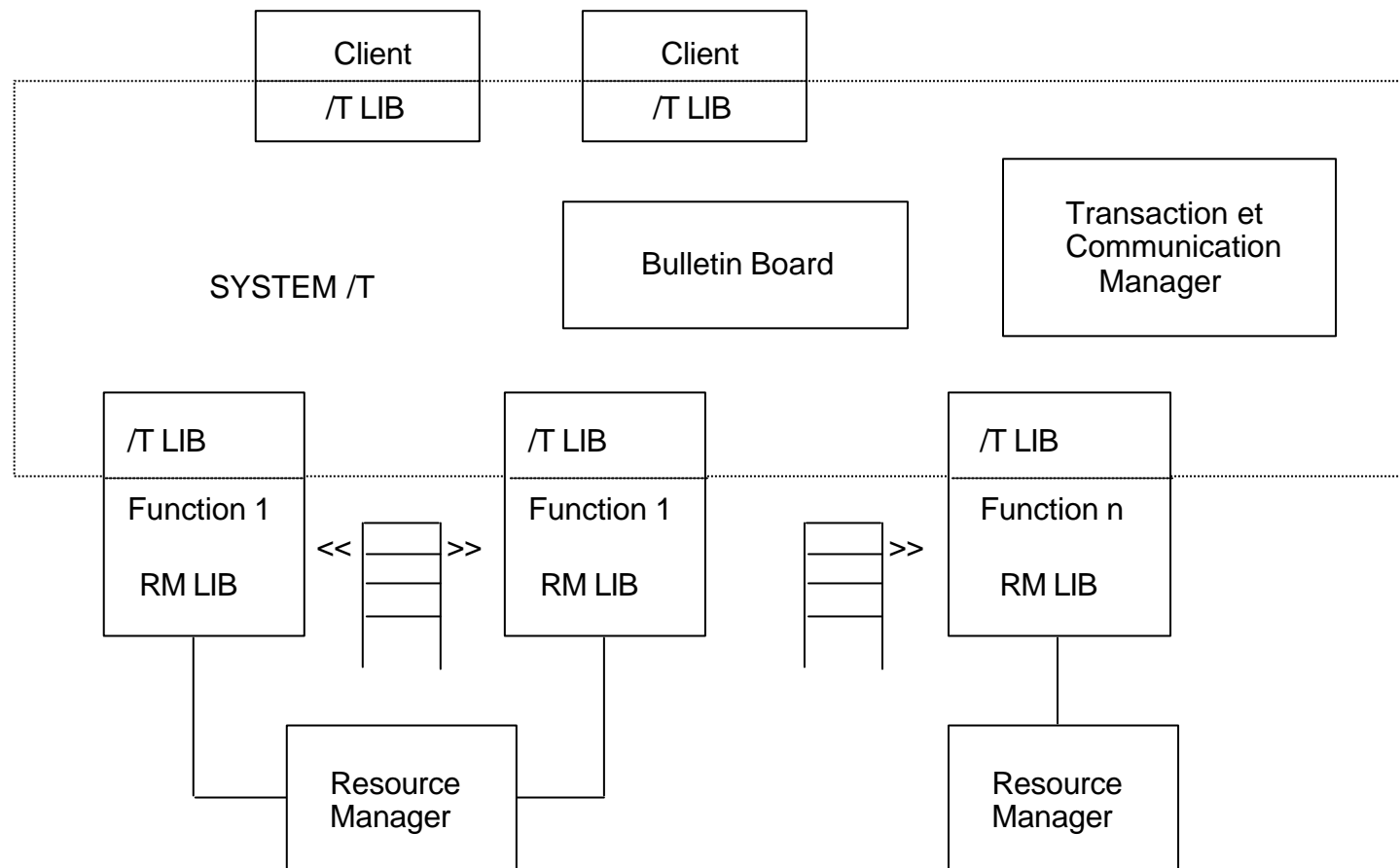
- Initialement développé par AT&T pour ses propres applications transactionnelles sous Unix, repris ensuite par USL (Unix System Laboratories), puis Novell et maintenant possession de BEA
- Début de commercialisation en 1989. Plusieurs milliers de systèmes installés
- Disponible sur un grand nombre de systèmes
- Caractéristiques
  - Conforme au modèle X/Open DTP (Distributed Transaction Processing)
  - Portabilité
  - Support au niveau des langages de programmation (e.g. Visual Basic, Cobol)
  - Architecture Client/Serveur
  - Management du système
  - Multiplexage Clients - Serveurs
  - Mécanisme de gestion de files d'attente de messages
  - Transactions distribuées
  - Sécurité

## ■ Architecture d'application

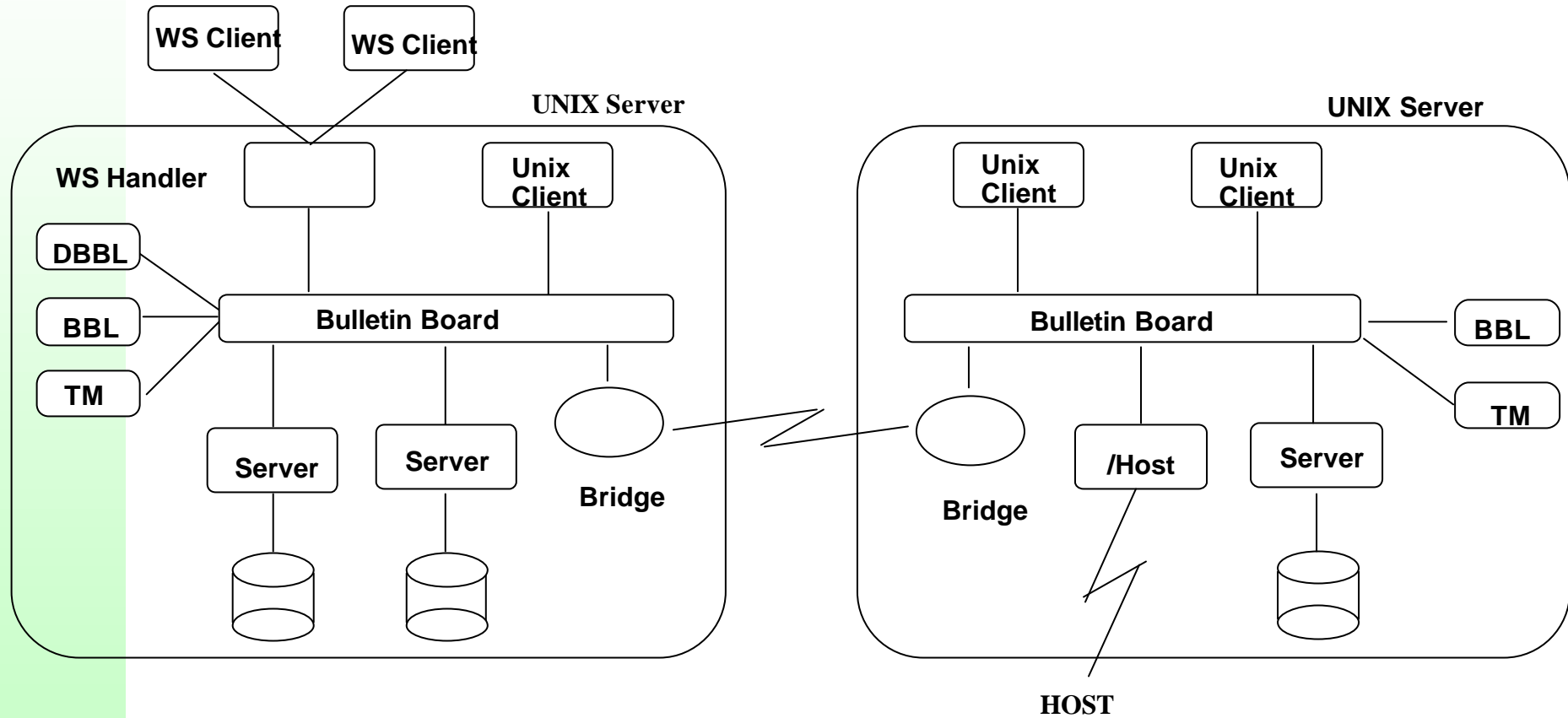


**ATMI : Application - Transaction Manager Interface**

## ■ Architecture - Cas centralisé



## ■ Architecture - Cas distribué



## ■ Composants

### □ Tuxedo System/T

- Composant principal de Tuxedo (fonctionne sous Unix)
- Serveur de Nom et Gestionnaire de Transactions (TM)

### □ Tuxedo System/WS

- Partie Client, fonctionne sous DOS/Windows, Unix et OS/2

### □ Tuxedo System/Host

- Permet à des services de Tuxedo de fonctionner sur des systèmes propriétaires

### □ Tuxedo System/Q

- Mécanisme de mise en queue de messages respectant les propriétés transactionnelles (soumission et achèvement garantis)
- Gestionnaire de ressources (RM) conforme à XA

### □ Tuxedo System/TDomain

- Requêtes transactionnelles entre des domaines d'administration séparés de Tuxedo

# Moniteurs transactionnels et SGBD

- **Il y a deux possibilités pour la programmation d'un système transactionnel:**
  - **Programmation Client/Serveur, sans moniteur transactionnel, en relation avec les possibilités offertes par les SGBDs. Ceci est appelé "TP Lite" ou transactionnel léger**
  - **Utilisation d'un moniteur transactionnel qui fournit le cadre architectural des applications et utilisation des services fournis par différents composants logiciels (e.g. SGBDs). Ceci est appelé "TP Heavy" ou transactionnel lourd**
- **Pour le choix entre ces deux approches, différents éléments rentrent en ligne tels que:**
  - **Transactionnel léger : dépendance vis à vis du fournisseur de SGBD, limitations vis à vis de la programmation des transactions (la validation est faite au niveau du SGBD), problèmes de performance,...**
  - **Pour TP Heavy: limitation potentielle dans les progiciels que l'on peut intégrer, pérennité du moniteur, complexité de la programmation,....**

- [BER97] Philip A. Bernstein, Eric Newcomer « Principles of Transaction Processing »  
Morgan Kaufmann, San Mateo, 1997**
- [BES97] Jérôme Besancenot, Michèle Cari, Jean Ferrié, Rachid Guerraoui, Philippe Pucheral, Bruno Traverson, " Les systèmes transactionnels : concepts, normes et produits "  
Hermès Science, 1997**
- [GRA93] Jim Gray, Andreas Reuter « Transaction Processing: Concepts and, Techniques »  
Morgan Kaufmann, San Mateo, 1993**