# Systèmes transactionnels

## TP : Transaction processing

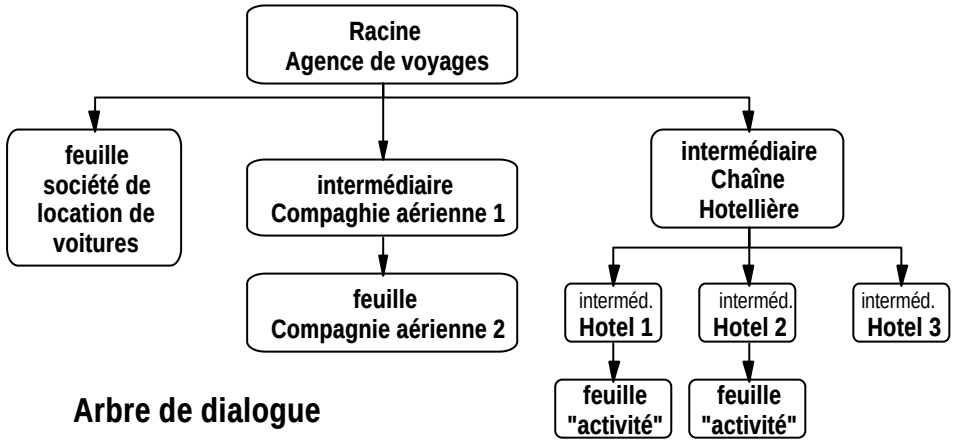
- **Traitement correspondant à une opération élémentaire caractérisée par ses propriétés ACID**
  - **Atomicité**
    - **entièrement ou pas du tout**
    - **peut correspondre à plusieurs transactions physiques**
  - **Consistance**
    - **Effectuée avec validité**
    - **Données passent d 'un état consistant à un autre état consistant**
  - **Isolation**
    - **aucune opération d 'une autre transaction n 'accède aux résultats partiels obtenus durant la transaction**
    - **Si données communes : sérialisation**
  - **Durabilité**
    - **Le résultat d 'une transaction complète ne peut être altérée (par panne, défaut)**
    - **Annulation ou restauration**
    - **Synchronisation - sauvegardes (log)**

# Caractéristiques des applications transactionnelles

- Traitements courts et répétitifs
- Grand nombre d'utilisateurs simultanés
- Nombre importants de transactions à traiter en parallèle
  - temps de réponse faibles et assez constants : 1 à qq secondes
- Accès concurrent à des bases de données
- volumes de données importants
- Grande fiabilité
- Haute disponibilité

- Architectures distribués
  - X/Open Group : DTP 5distributed Transaction Processing)
  - OSF : OLTP (On Line Transaction Processing) dans DCE

- **Des données sûres**
  - survivent à une panne d 'une application
  - sont disponibles après reprise locale : restauration
- **Données liées :**
  - données sûres qui existent durant exécution de la transaction
- **Données d'une action atomique : données sûres qui maintiennent**
  - Connaissance de l'action atomique en cours
  - Etat du contrôle de concurrence
  - Information pour restaurer données à état initial
- **Journalisation (log)**
  - Evénement correspondant à écriture des données correspondant à action atomique sur un support sûr.
- **Transaction répartie**
  - se déroule sur plusieurs systèmes ouverts
  - Coordination est nécessaire entre entités chargées du traitement
    - TPSUI : Transaction Processing Service User Invocation



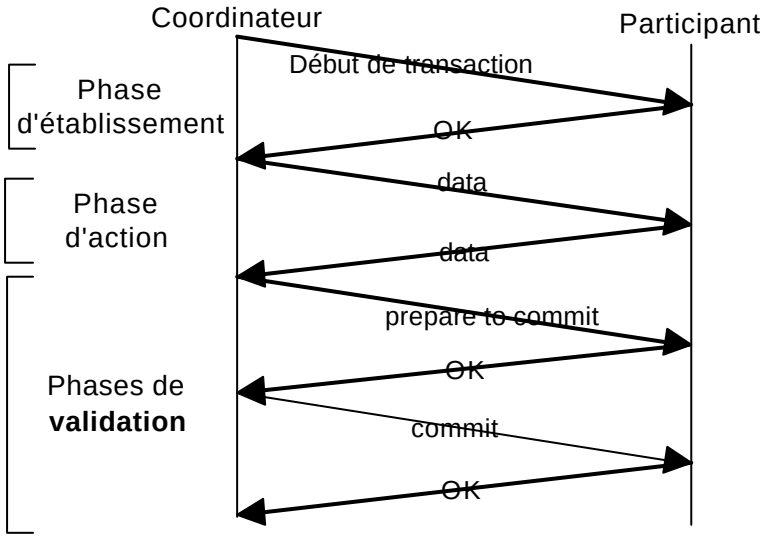

Arbre de dialogue

- **Objet:**
  - transfert de données
  - notification anomalies
  - initialisation, validation, annulation de transaction
  - terminaison
    - négociée
    - brutale
  - synchronisation des activités
- **types**
  - Polarisé
    - Initiateur-répondeur
  - Partagé
    - commande simultanée par 2 systèmes

## Structure des dialogues

- à la demande d ’une entité
- dialogue adjacents entre entités distinctes
- Arbre de dialogue
  - supérieur -subordonné
  - branche de dialogue

- Lancée par «  noeud racine  » : coordinateur d’engagement
- Identificateur propagé dans tout l  ’arbre
  - allocation de ressource - isolation
- Un système peut être dans l’impossibilité d’assurer la transaction
  - Annulation - Rollback
  - peut être effectuée tant que l’on est pas en phase terminale
- Pour consistance : validation à 2 phases
  - Première phase : préparation
  - Seconde phase : validation
    - ➠ stockage des informations sur état des données : log



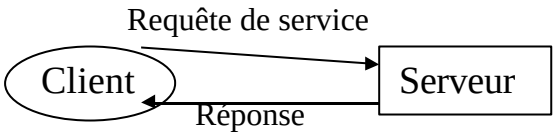

- **Règles**
  - validation AVANT écriture dans SGBD
  - validation APRES enregistrement de toutes les modifications dans journal
- **Phase 1 : journalisation**
  - coordinateur → Préparation à valider (prepare to commit)
  - partenaire vote : oui ou non. Note vote dans son journal
  - coordinateur collecte les votes. Si unanimité de oui → ordre de valider (commit), sinon annulation (abort). coordinateur note dans son journal

    participant finit transaction, notifie le coordinateur, note fin dans son journal
  - coordinateur note fin de transaction dans son journal
- **Phase 2 : écriture dans la base**

  **ou annulation.**

- En cas de panne : rupture de communication
- Coordinateur
  - Déclenche après la première phase
    - vote positif mais pas de fin de transaction
    - rappel des participants dans ce cas et reprise de validation
- Participant
  - Voit rupture de communication après vote
  - trouve trace vote dans son journal sans fin de transaction
  - reprend contact avec coordonnateur pour connaître résultat du vote
  - si pas de trace dans journal coordonnateur: vote négatif
    - ne pas mettre à jour les données
    - sinon reprendre fin de validation
- nécessité d ’un verrouillage en 2 temps (2 phase looking)
  - aucun accès aux données modifiées non validées

- **3 grands modèles :**
  - **Conversationnel**
    - **basé sur APPC (IBM)**
  - **Client-Serveur**
    - **basé sur usage du RPC**
  - **Queue de messages**
    - **basé sur OSI-TP**

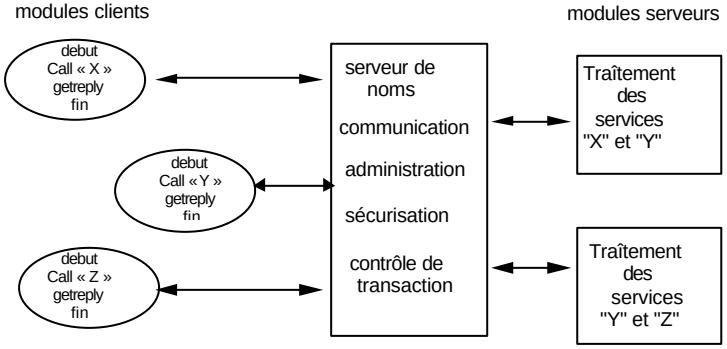



## Modèle de base

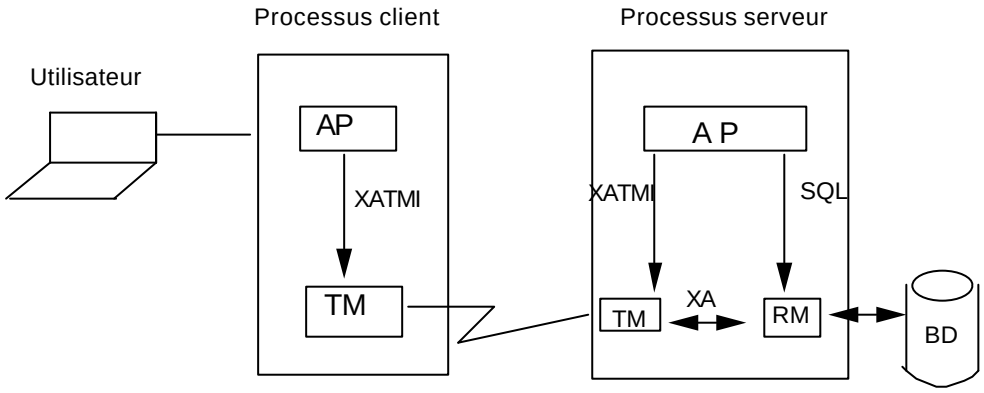
- sur même système
- sur systèmes distants par RPC par exemple
  - modularité
  - facilité de distribution
  - extensibilité

## Modèle enrichi

- utilisation d'un serveur de noms
  - correspondance nom logique-nom physique
- Sécurisation par serveur d'authentification
  - Indépendance clients-serverus
  - transparence de localisation
  - performances
  - sécurité des données , d 'accès
  - souplesse (ajustement des serveurs)
  - efficacité (partage de resooures réduit côut)
  - Flexibilité

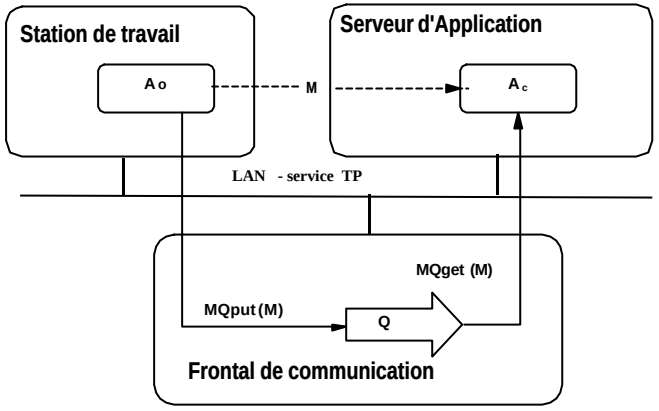


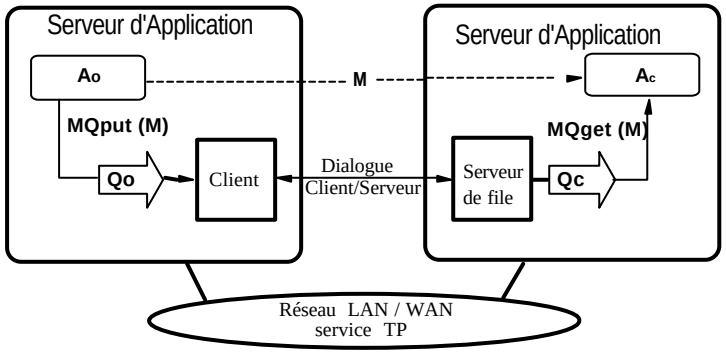
# Modèle DTP de l 'X/Open




- **Programme d'application (AP)**
  - définit les limites de la transaction
  - spécifie les actions
- **Gestionnaire de transaction (TM)**
  - routage
  - validation
  - recouvrement
- **Gestionnaire de ressources (RM)**
  - accès aux données sur serveur
  - utilise SQL
- **Interface XATMI**
  - entre AP et TM sur clients et serveurs
- **Interface XA**
  - entre TM et RM
  - validation à 2 phases
  - s'appuie sur CCR

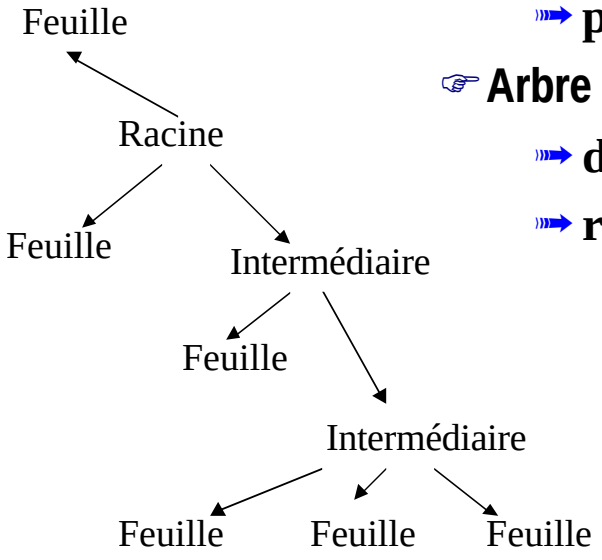
# Modèle MQ : message queueing




- Plus récent et moins connu
- mode asynchrone
  - les 2 applications peuvent être décalées dans le temps
  - même système ou réparti
  - repose sur OSI-TP (7ème partie)
- Transfert de messages entre applications à travers de files sécurisées
- Queue apparaît comme un service dans architecture client-serveur
  - gestionnaire de queue est un serveur
- message considéré comme écrit dans queue après validation seulement
- Possibilité d'insérer des serveurs de communication entre serveur d'application

- **Base : propriétés ACID**
- **S'appuie sur CCR (commitment, concurrency, recovery)**
  - validation à 2 phases
- **Transactions distribuée sur plusieurs systèmes**
  - dialogue entre entités distantes associées
    - plusieurs dialogues possible pour un système
  - Arbre de transaction
    - dialogue = branche de l 'arbre
    - racine - nœuds intermédiaires - feuilles




- **Architecture**
- **TPSUI établit branche de transaction**
- **TPPM assure les propriétés ACID (validation)**
- **ACSE : association - Rupture**

# Entités fonctionnelles

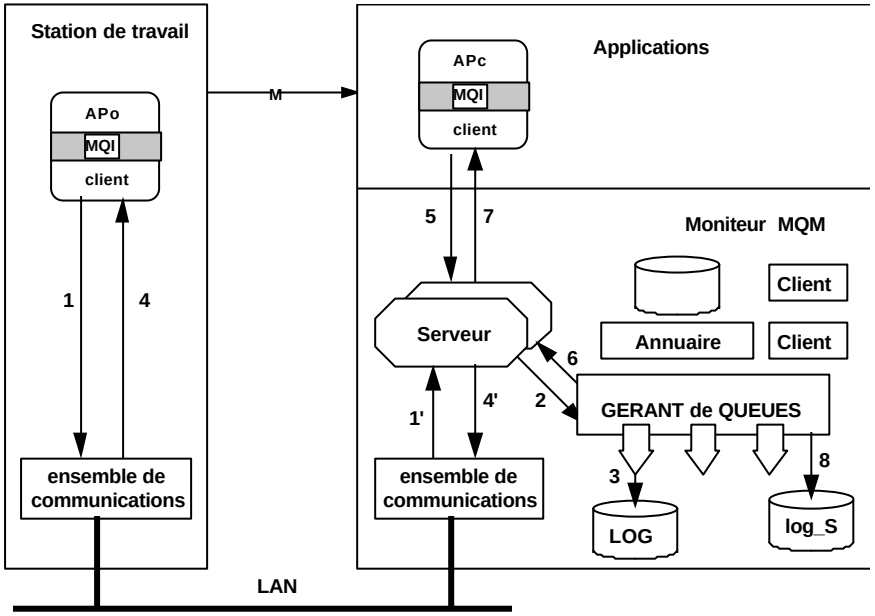
- Noyau
  - Établissement -rupture s ’association via ACSE
- Contrôle partagé
  - 2 TPSUI ont le contrôle du dialogue (doivent se coordonner…)
- Contrôle polarisé
  - 1 TPSUI a le contrôle du dialogue à un moment donné
  - Elle peut le transférer à l ’autre tPSUI
- Synchronisation
  - Handshake : synchronisation des traitements entre TPSUI
- Commit
  - Validation fiable
  - Retour en arrière (rollback)
    - coordination permet de déclencher 2ème phase de validation ou reprise
    - journalisation - gestion des données liées
    - Utilisation de CCR
- Chaînage -déchaînage des transactions

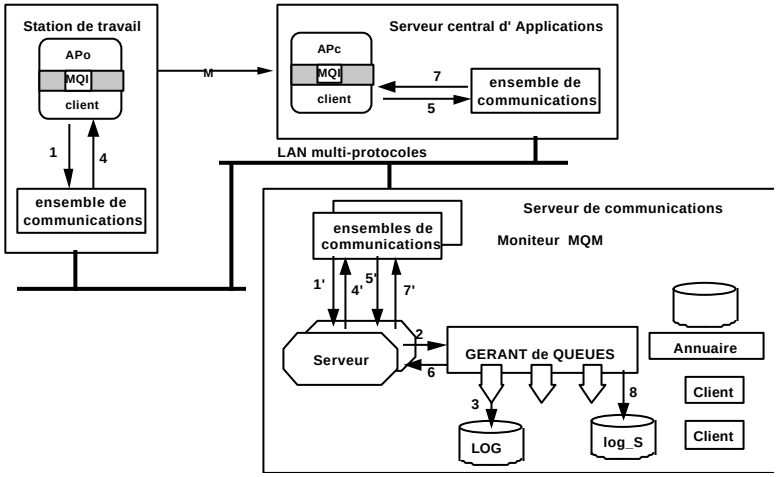
| unités fonctionnelles | primitives de service |
|---|---|
| noyau | TP-BEGIN-DIALOGE<br>TP-P-REJECT<br>TP-END-DIALOGUE<br>TP-DATA<br>TP-U-EROR<br>TP-P-ERROR<br>TP-U-ABORT<br>TP-P-ABORT |
| contrôle partage | pas de primitive |
| contrôle polarisé | TP-GRANT-CONTROLE<br>TP-REQUEST-CONTROLE |
| synchronisation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

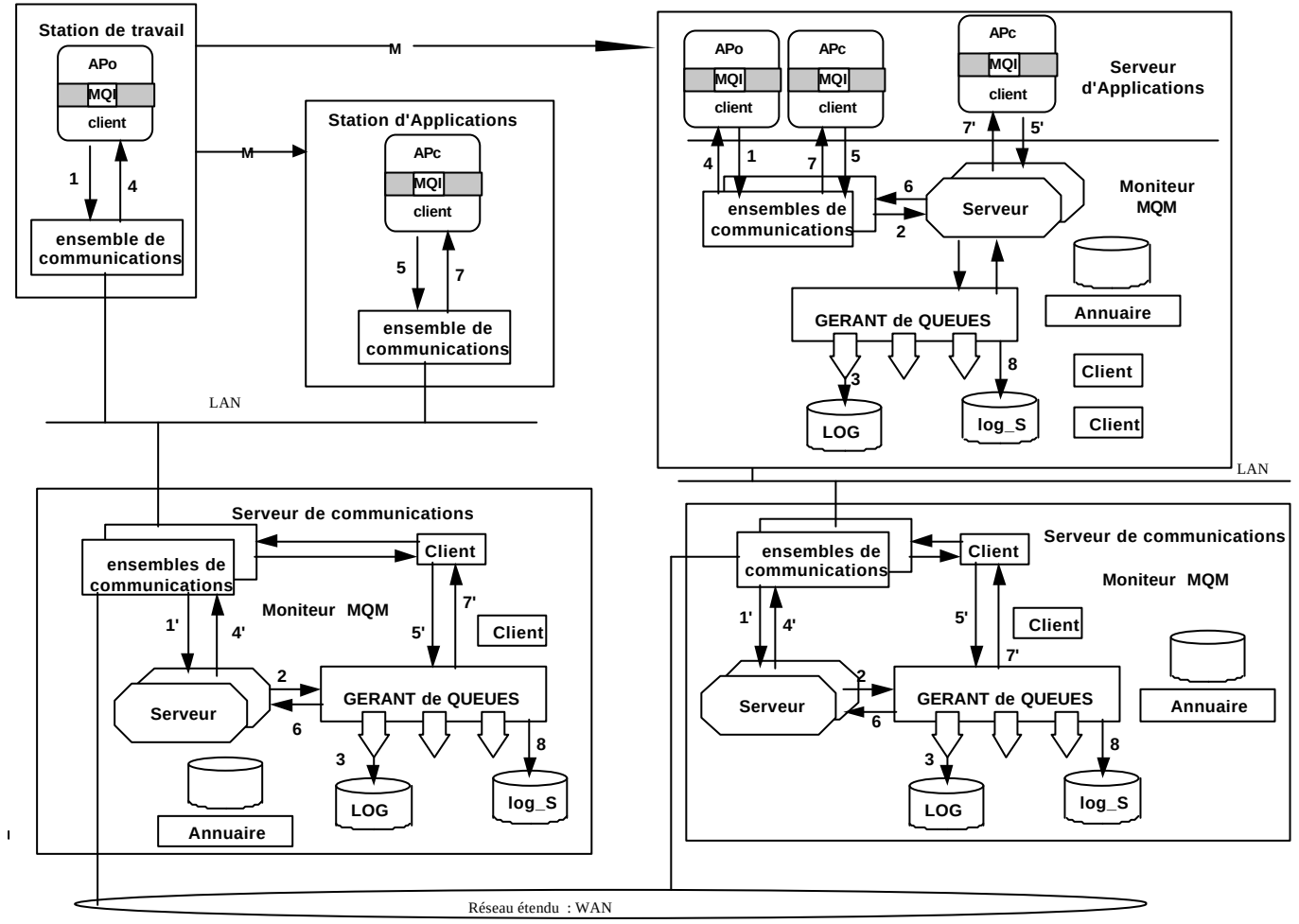
- **En pratique une partie importante des fonctions de coordination ou pour les reprises doit être prise en compte dans l'application**

- **Tuxedo (USL)**
  - suit le modèle DTP de l 'X/Open
- **Encina (Transarc)**
  - Répond complètement aux concepts d 'OLTP. Conforme à DTP
  - Extension fonctionnelle de DCE
  - s 'appuie sur RPC et CDS (annuaire distribué de DCE)
- **CICS/6000 (IBM)**
  - Respecte le modèle DTP
  - utilise les API de CICS (APPC)
  - Réutilise les briques de base de ENCINA (Transarc filiale de IBM ...)
  - Existe sous MVS, VSE, OS/400, OS/2
  - sous Unix environnement DCE (RPC, CDS, Kerberos)
- **Moniteur distribué selon modèle MQ (Marben)**

# Moniteur MQM




Station de travail
APo
MQI
client
1
4
ensemble de communications
M
M
Station d'Applications
APc
MQI
client
5
7
ensemble de communications
LAN
APo
MQI
client
APc
MQI
client
APc
MQI
client
Serveur d'Applications
7'
5'
4
1
7
5
ensembles de communications
6
2
Serveur
Moniteur MQM
GERANT de QUEUES
Annuaire
3
8
Client
LOG
log_S
Client
LAN
Serveur de communications
ensembles de communications
Client
1'
4'
Moniteur MQM
5'
7'
Client
2
Serveur
GERANT de QUEUES
6
3
8
Annuaire
LOG
log_S
Serveur de communications
ensembles de communications
Client
Moniteur MQM
1'
4'
5'
Client
7'
2
Serveur
GERANT de QUEUES
6
Annuaire
3
8
LOG
log_S
Réseau étendu : WAN