Thème n°4 : Les Applicatifs Réseaux

L'Environnement Distribué

DCE

(Distributed Computing Environment)

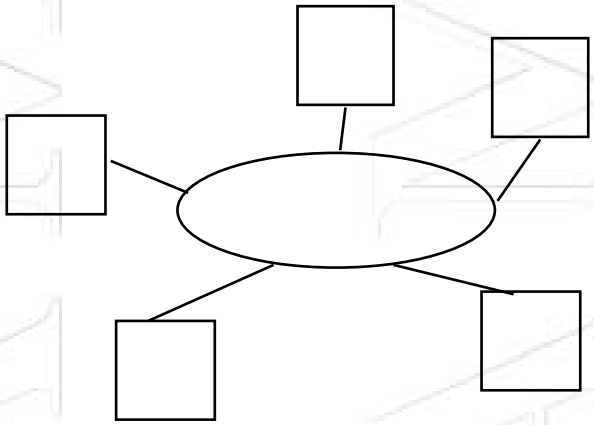
UV B : Complément Réseaux de Transport et Applications

Année 95/96

Laurence Duchien
CNAM-Cedric, 292, rue st Martin
75141 Paris Cedex 03
tel : 40 27 25 83
e_mail : duchien@cnam.fr
http://tulipe.cnam.fr/personne/duchien/poly.html

Programmation distribuée

Coopération entre machines qui communiquent à travers un réseau*



Besoin :

- Partage des données et des ressources
- Partage des traitements
- Intégration des SI

=> **Offre :**

- Services de distribution sur des plates-formes hétérogènes,
- Services applicatifs

=> **Résultat :**

- Cohérence des systèmes,
- Évolution et réaction possibles

* : ne pas confondre **Systèmes distribués** (système d'exploitation distribué : Chorus, Mach, ...) et **Architectures distribuées** (système d'exploitation centralisé + ajouts d'outils)

1. L'environnement Distribué DCE

Environnement de programmation intégré :

=> Pour développer des applications distribuées sur des plates formes hétérogènes

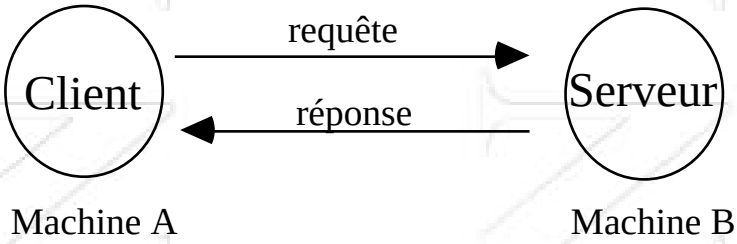
=> Masquent aux applications distribuées la spécifité des systèmes d'exploitation et l'existence des réseaux.

Comprend :

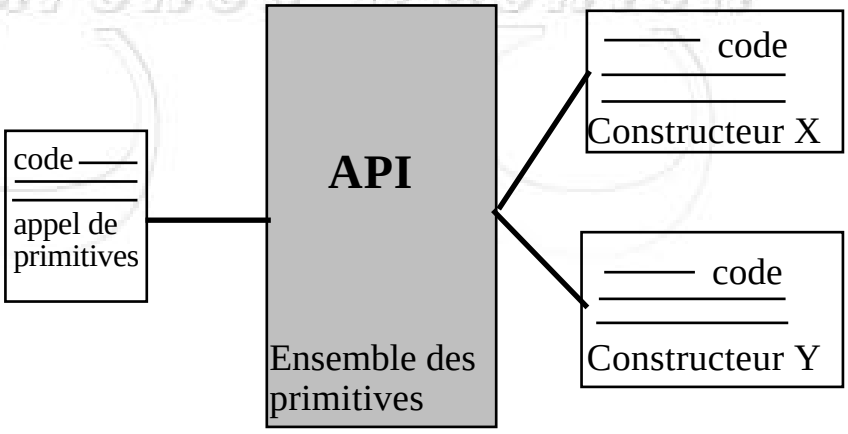
- outils de développement
- services

Fondé sur :

- Modèle Client/Serveur :



- API



1.1 Introduction

OSF (Open Software Foundation) :

- association de plusieurs constructeurs
  (IBM, DEC, BULL, Apollo, HP, Siemens&Nixdorf,...)
- groupe à l'origine d'autres standards (Motif,...)
- Version 1.0 et version 1.1 (nov 94)

=> L'OSF a identifié les technologies de base nécessaires à la construction et au bon fonctionnement d'une architecture distribuée.

=> Après appel d'offres pour le développement, il a choisi les implanteurs des différentes briques, a vérifié l'intégration et a diffusé le code aux constructeurs intéressés pour portage sur leur plates-formes.

Différents constructeurs ont annoncé la disponibilité de DCE sur leurs machines:

- Sous différents Unix (Ultrix, AIX, HP-UX, DG-UX, SunOS, ...)
- Sur des plates-formes non-Unix (OpenVMS, Microsoft Windows, Macintosh, Windows NT, OS/2, HP3000, MVS, Bull/DPX, IBM AS/400, Gcos sur Bull DPS 9xxx/7xxx,...)

1.2 Modèle d'architecture distribuée

DCE est fondé sur trois modèles de distribution :

- Le modèle client/serveur
- L'appel de procédure distante
- Le modèle de partage des données

1.2.1 Modèle Client/Serveur(1)

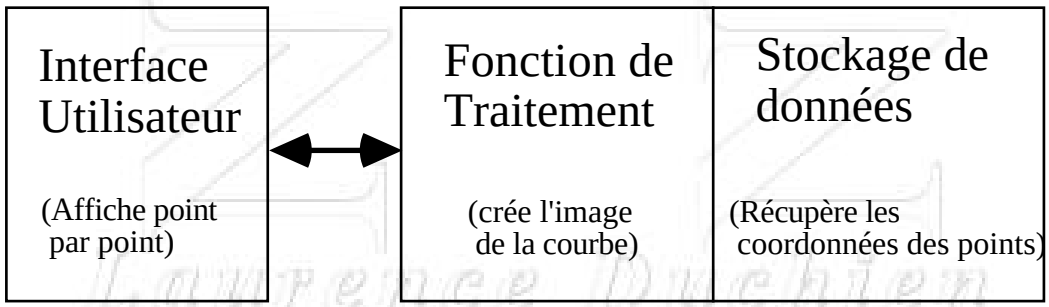
- Un modèle d'informatique non-distribuée, les composants suivants sont souvent intégrés :

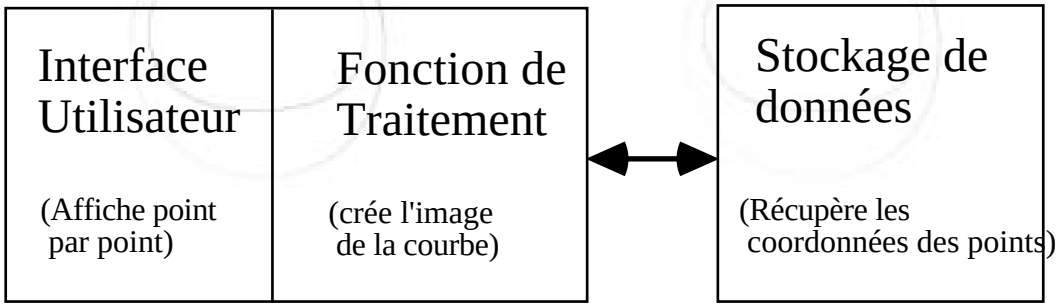
Interface utilisateur	Fonction de Traitement	Stockage de données

- Des modèles d'informatique distribuée :

=> L'approche la plus simple : décomposer en modules et répartir l'ensemble

Exemple :



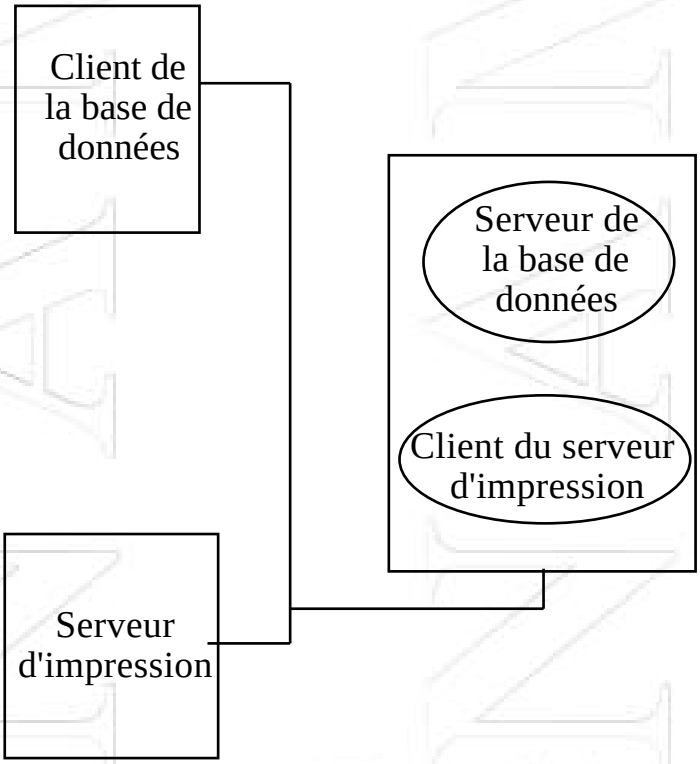


1.2.1 Modèle Client/Serveur(2)

- Relativité des rôles client/serveur

=> Les termes client et serveur font référence à des composants logiciels plutôt qu'à du matériel

Exemple de configuration :



- DCE est basé sur le modèle client/serveur.
- Les services de DCE sont eux-mêmes des exemples de programmes distribués avec un côté client et un côté serveur.

1.2.2 L'appel de procédure distante

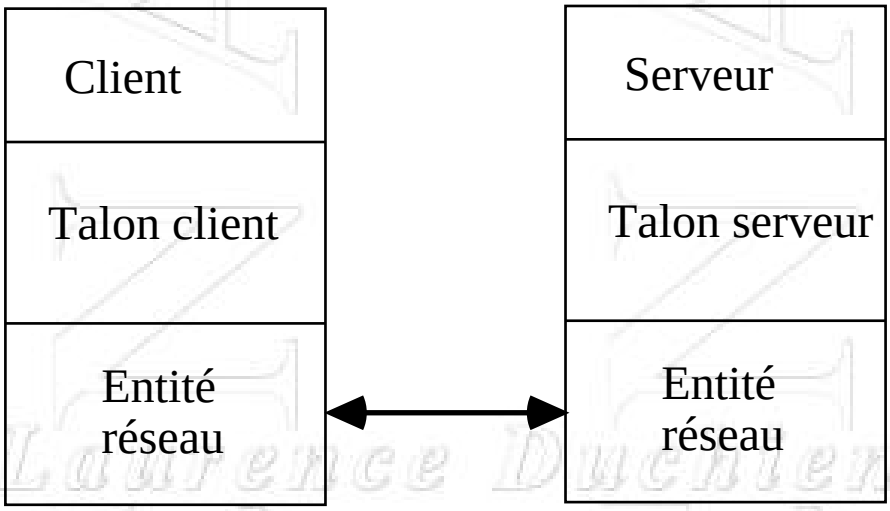
- Permet **le partage des traitements**
- Les moyens de communication associés au modèle Client/Serveur

  => être indépendant du réseau

  => accepter tous les formats (milieu hétérogène)

  => utilisation simple

  Solution proposée : l'appel de procédure à distance (RPC)



- DCE propose une implantation de ce modèle de communication

1.2.3 Le partage de données

- Dans ce modèle de distribution, les données sont envoyés vers le client par le serveur.

Exemple : si un client souhaite accéder à un fichier, une copie du fichier lui est envoyé par le serveur.

- Cela implique plusieurs copies des mêmes données.

  Une copie maître sur le serveur,

  Plusieurs copies sur les clients.

=> les copies peuvent diverger selon les mises à jour des clients.

=> les services de distribution doivent fournir des services permettant de garder les copies cohérentes.

=> il faut aussi synchroniser les accès multiples aux données

- Deux services de DCE sont construits sur ce modèle :

=> le service d'annuaire et le service de fichiers distribués

1.3 L'architecture de DCE

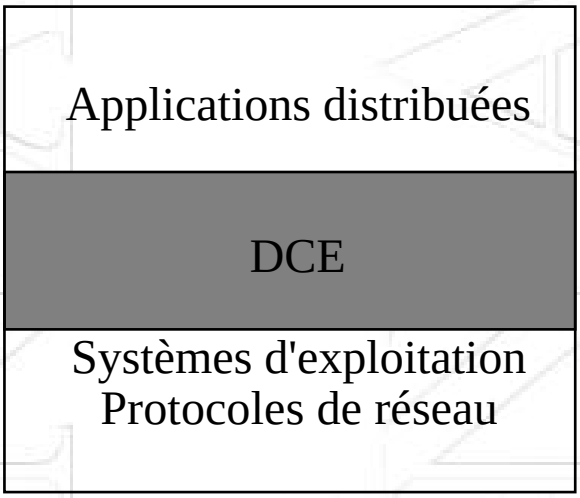
- DCE:

  - ensemble complet et intégré de services pour le développement, l'utilisation et l'administration d'applications distribuées.

  - boîte à outils de développement pour la programmation distribuée en milieu hétérogène.

  - accès aux outils et aux services par l'intermédiaire d'interfaces (API)

- DCE par rapport au système d'exploitation et au réseau:



- On parle de **middleware** ou de **bus logiciel**
- Les composants sont de deux types :

  - Les outils de développements (DCE Threads et DCE RPC),

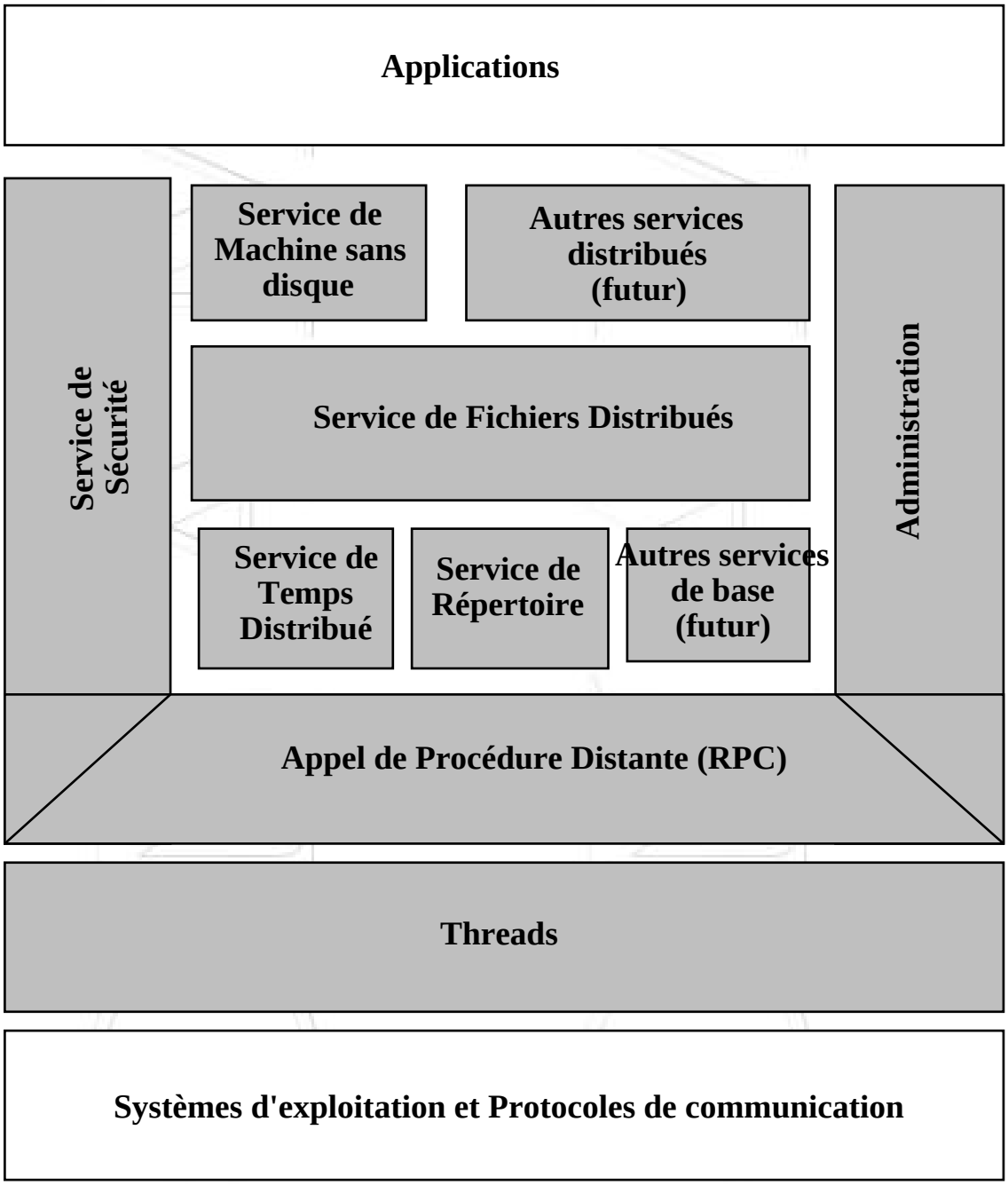
  - Les services :

    . Les services de base : service de répertoire, de sécurité, de temps distribué,

    . Les services étendus : service de fichiers distribués, de support de machine sans disque,

    . Un service d'administration.

1.3 L'architecture de DCE(2)



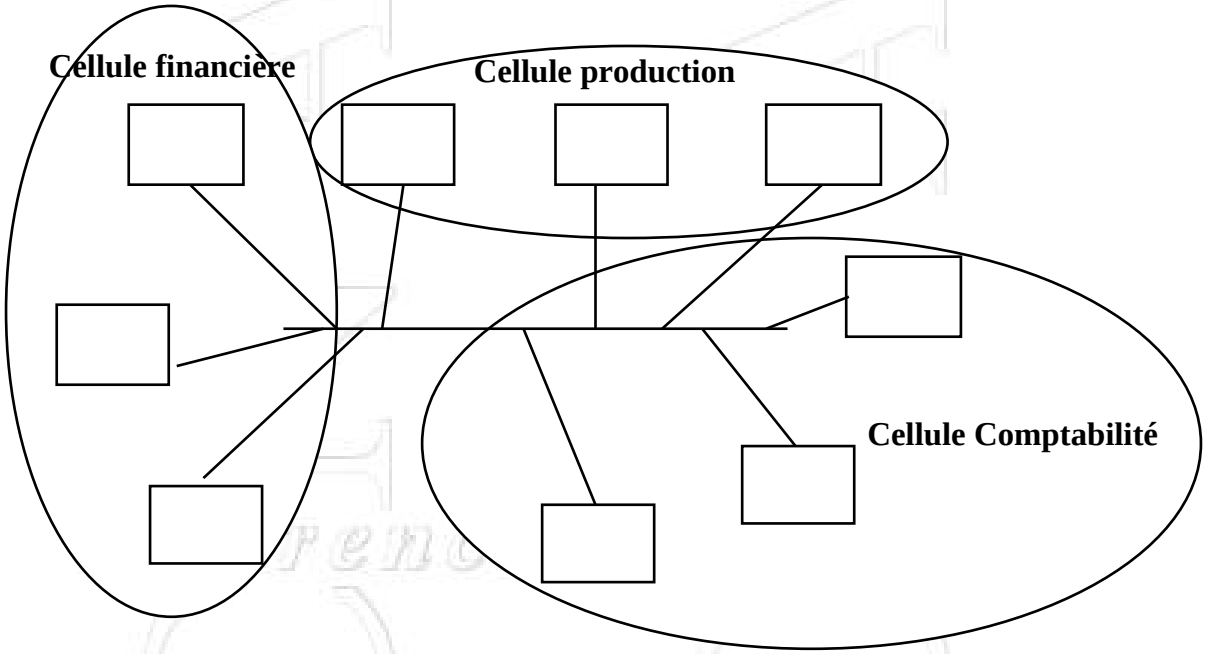
1.3.1 Une description rapide des composants

- **DCE Threads :** supporte la création, la gestion et la synchronisation des fils de contrôle (ou processus légers).
- **DCE RPC :** correspond à un outil de développement et à un "run-time". L'outil est composé d'un langage (et son compilateur). Il génère un code qui transforme les appels de procédure en messages pour le réseau.
- **Service de répertoire ou d'annuaire DCE (CDS ou GDS):** établit la correspondance entre l'identifiant logique et la localisation physique des objets manipulés. C'est le service de base d'une architecture distribuée.
- **Service de temps distribué DCE (DTS):** fournit à toutes les machines d'une entité DCE une référence temporelle commune.
- **Service de sécurité DCE (SS) :** permet d'avoir des communications sécurisées sur le réseau et d'avoir des contrôles d'accès à l'environnement distribué.
- **Service de fichiers distribués (DFS) :** permet d'accéder et de partager des fichiers n'importe où dans le réseau sans se préoccuper de leur localisation physique.
- **Service de machines sans disque (DSS) :** donne la possibilité à une machine du réseau sans disque de participer à l'environnement DCE.

1.4 L'organisation d'un environnement DCE : La notion de cellule

- **Cellule** = ensemble de machines interconnectées et administrées comme une unité.
- Il s'agit d'un regroupement fonctionnel de machines susceptibles de communiquer fréquemment.
- L'espace de noms, la sécurité et l'administration sont fondés sur cette notion de cellule.

Exemple d'organisation dans une entreprise:



- Les différentes cellules peuvent interagir entre elles pour des besoins ponctuels.
- Les limites d'une cellule sont le reflet de préoccupations organisationnels.

1.4 L'organisation d'un environnement DCE : La notion de cellule (2)

- Comment décider du regroupement des machines en cellule?

  - Déterminer l'objectif : les machines dans une cellule doivent travailler ensemble dans un but commun (un projet, réaliser une fonction,...). Les utilisateurs doivent se connaître et avoir plus de contacts entre eux qu'avec d'autres.

  - La sécurité : les utilisateurs de la cellule échangent des informations entre eux. Les accès intra-cellules sont facilités. Par contre, les limites de la cellule fonctionnent comme des portes coupe-feu. L'accès à des ressources externes dans d'autres cellules amènent des négociations inter-cellules plus complexes.

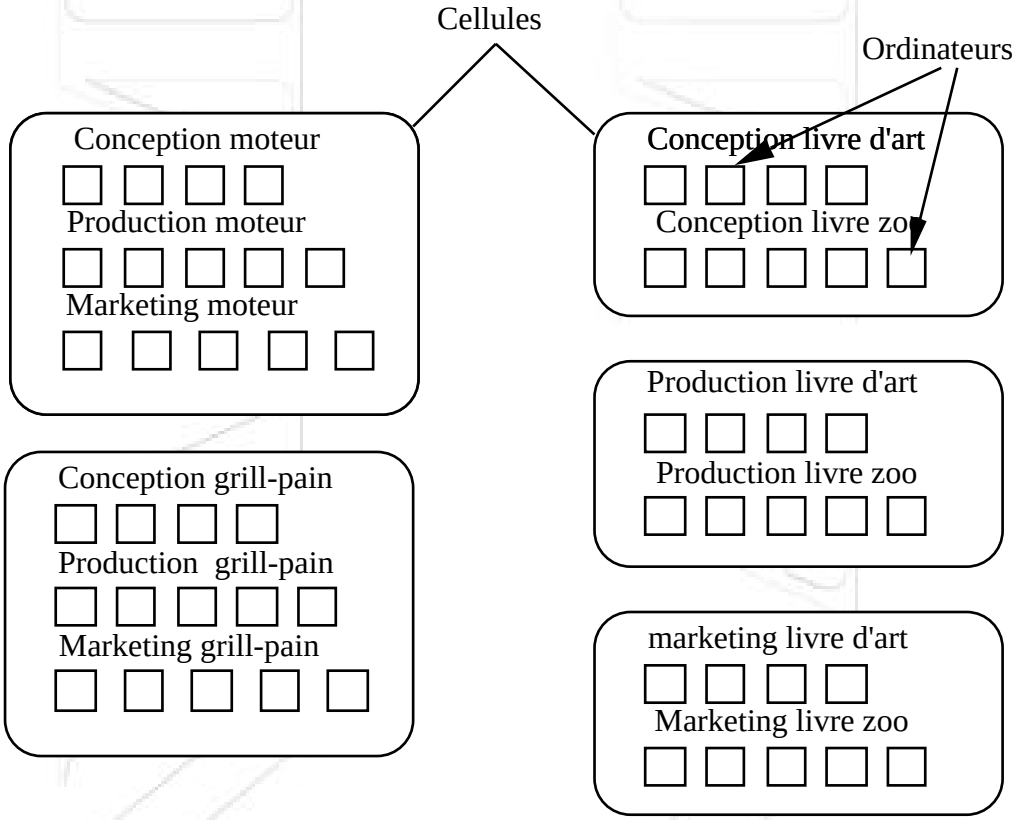
  - Le temps de gestion (overhead) : les échanges sont optimisés pour les communications intra-cellules. Le facteur géographique peut être un facteur important à prendre en compte. Il peut être pénalisant de mettre ensemble des utilisateurs devant communiquer à travers un réseau grande distance.

  - L'administration : Une cellule a un seul administrateur. Si une cellule est répartie sur deux entités (deux entreprises par exemple), il faut penser au partage de cette tâche.

- Il est souhaitable d'avoir un minimum de cellules pour minimiser le nombre d'opérations inter-cellules.

1.4 L'organisation d'un environnement DCE : La notion de cellule(3)

- Exemples :



(a) : organisation par produit (b) : organisation par fonction

- la détermination des limites des cellules doit prendre en compte des considérations de gestion de l'entreprise et non des propriétés techniques de DCE.
- la taille d'une cellule peut varier, mais elle contient toujours :

  - un serveur de temps
  - un serveur de répertoire
  - un serveur de sécurité

1.5 L'intégration des composants de la technologie DCE

- L'un des atouts de DCE est sa cohérence
- Les modules et leur interface sont bien définis et bien intégrés

  - Les ≠ composants de DCE utilisent chacun les services des autres lorsque c'est possible.

Exemple :

Le service de RPC : - utilise le service de répertoire pour rechercher les serveurs de RPC ainsi que leur caractéristiques.

- utilise aussi les services de sécurité pour assurer l'intégrité des communications.

- utilise les threads pour gérer les exécutions concurrentes des RPC multiples.

Le service de fichiers distribués :

- utilise les threads, le RPC le service d'annuaire, le serveur de temps distribué, le service de sécurité.

- En général, les composants "hauts" de DCE utilisent les composants "bas" de l'architecture.

Exemple :

Les threads sont utilisés par la plupart des autres composants DCE, mais n'utilise pas lui-même d'autres composants.

- Cet ordre n'est pas strictement hiérarchique

Exemple :

Deux services peuvent dépendre l'un de l'autre : sécurité et répertoire.

1.6 Les relations de DCE avec les services du réseau et du système(1)

- **Services réseaux :**

  DCE est posé sur une couche Transport => UDP, TCP, ou ISO TP0-TP4.

  - Cette couche est accédée via une interface Socket ou XTI (X/open Transport Interface).

  - Le réseau doit être disponible, il peut s'agir d'un réseau local, d'un réseau grande distance ou un mélange des deux.

  - Il est possible de faire communiquer des cellules utilisant des piles de protocoles différentes (IP ou OSI), il faut pour cela avoir une pile en commun.

  DCE offre une possibilité :

  - d'identification d'un noeud (une machine) avec une adresse unique dans le réseau

  - d'identification d'un processus avec une adresse de "connexion" (un port, une porte, un T-selector,...)

1.6 Les relations de DCE avec les services du réseau et du système (2)

- **Services du système d'exploitation**

  - le multitâche

  - la gestion de délai (timer)

  - les communications locales inter-processus

  - les opérations de base du système de fichiers (couche VFS)

  - la gestion de la mémoire

  - les mécanismes de sécurité locale

  - les threads (ou la possibilité d'utiliser les threads de DCE)

  - divers utilitaires d'un système

- Les dépendances d'implantations :

  - l'utilisation d'IP et services sockets

  - les appels systèmes

2. La configuration DCE

- Les différentes configurations des machines d'une cellule DCE:

- Machine utilisateur DCE : contient le logiciel client d'un environnement DCE

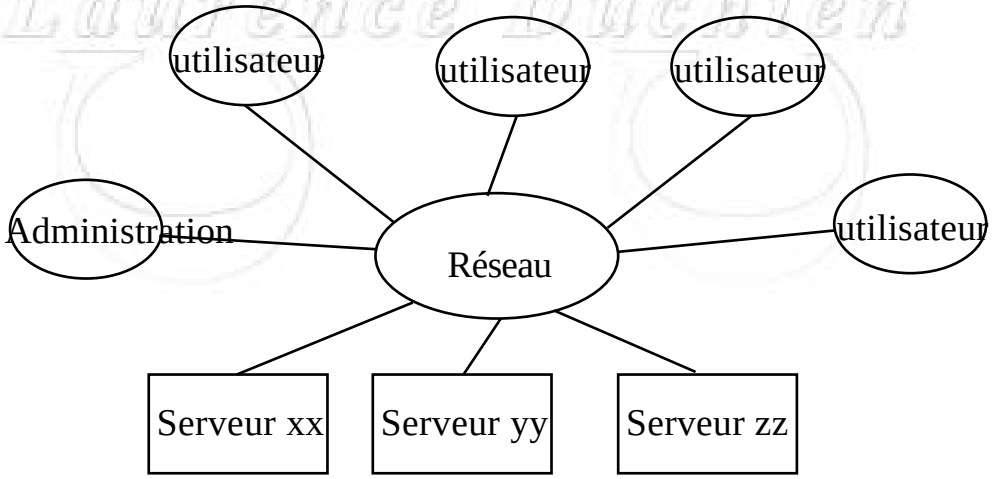
(ex : une station de travail d'un utilisateur)

- Machine d'administration DCE : contient le logiciel qui permet de contrôler les serveurs fonctionnant dans l'environnement

(ex : la station de l'administrateur du système DCE)

- Machine serveur DCE : implémente un ou plusieurs services de DCE.

(ex : serveur de fichiers, de sécurité,...)



2.1 Les composants logiciels d'une configuration de base

- Le logiciel DCE est divisé en plusieurs composants :

  => Chaque service de DCE est divisé en deux types de fonctionnalités :

  Les fonctionnalités utilisateurs (lire un fichier, accéder dans une base de données)

  Les fonctionnalités d'administration (démarrer ou arrêter les programmes serveurs, sauvegarder des données)

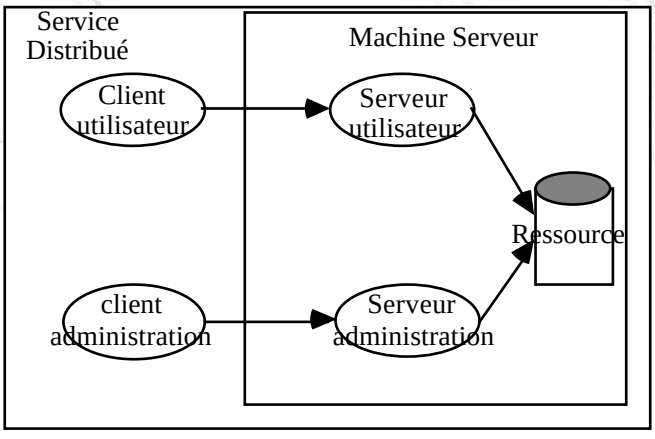
  => Tous les services de DCE sont fondés sur le modèle client/serveur, les fonctions utilisateurs et administrations sont divisés en 2 :

  Le client

  Le serveur

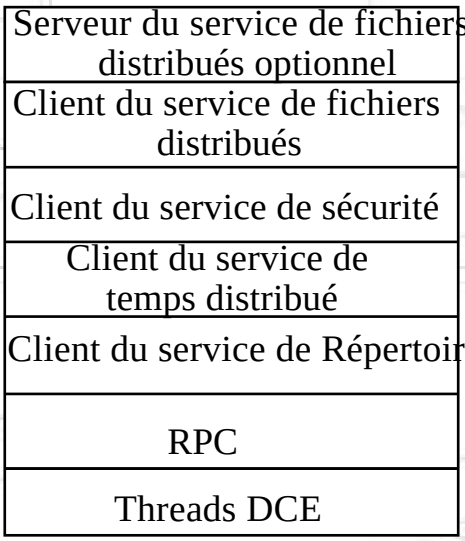
  => Au total, chaque composant DCE peut être conceptuellement divisé en 4 composants de configuration:

  - Client utilisateur
  - Serveur utilisateur
  - Client administration
  - Serveur administration



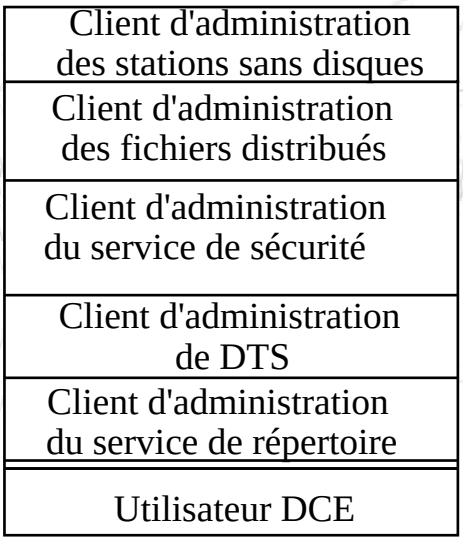
2.2 Exemples de configuration de machines DCE

- Configuration d'une machine client utilisateur DCE



Le serveur de fichiers distribués est optionnel : cela permet d'exporter les propres fichiers de la machine vers les autres utilisateurs.

- Configuration de la machine de l'administrateur



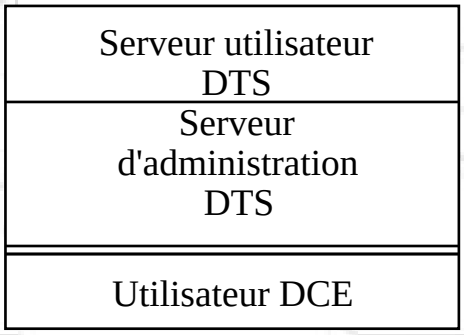
2.2 Exemples de configuration de machines DCE(2)

- Les machines serveurs

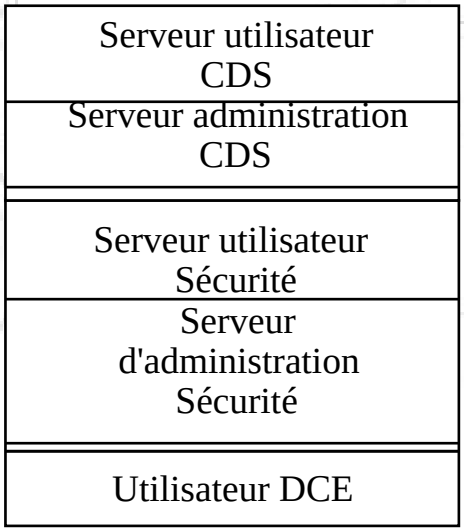
  - sont configurées avec les parties serveur d'utilisateur et d'administration

  - peuvent aussi contenir la partie client d'un composant lorsqu'elles agissent en tant que client d'un autre service

Exemple du serveur de temps distribué :



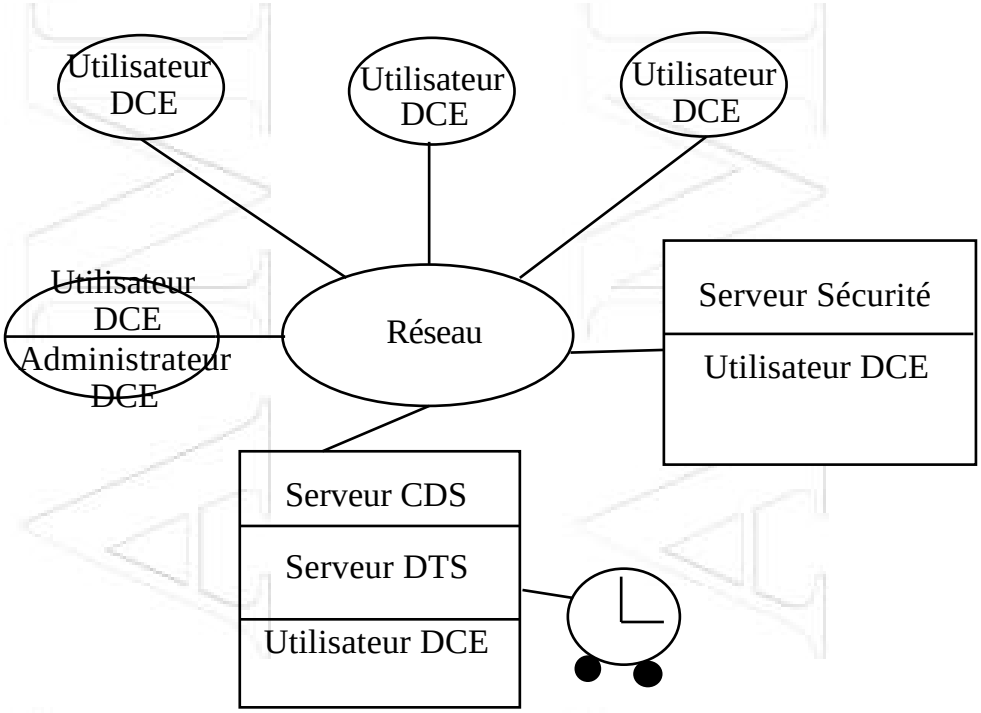
Exemple du serveur de sécurité et CDS (Cellule Directory Service)



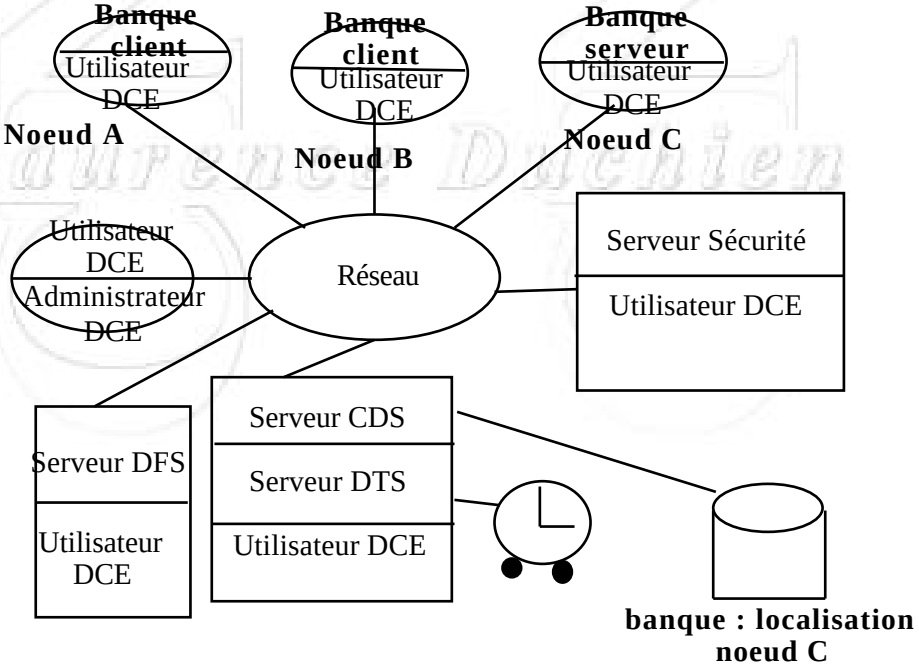
* à partir de maintenant, lorsque l'on parle de serveur, on entend la partie utilisateur et la partie administration.

2.3 Exemples de configuration de cellules(1)

- Une cellule simple

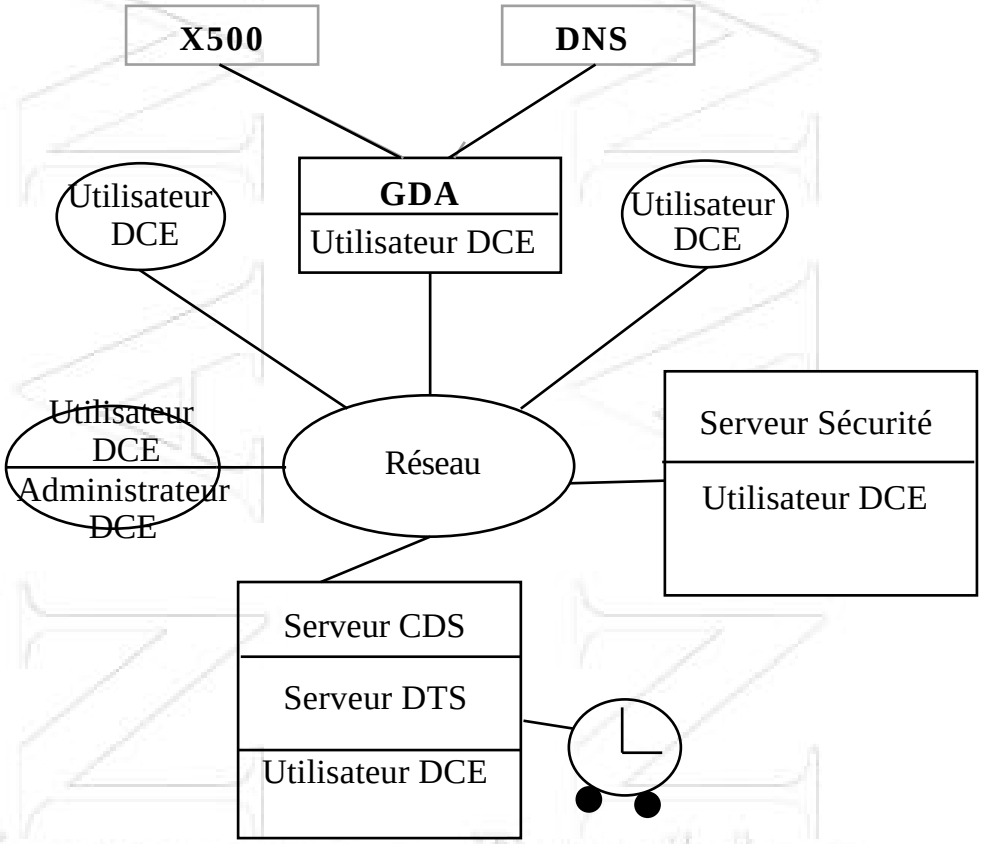


- Une cellule avec un serveur de fichiers DFS



2.3 Exemples de configuration de cellules(2)

•Une cellule connectée



3. Présentation de l'Architecture DCE

- Les threads
- Le RPC
- Le système de temps distribué (DTS)
- Le système de répertoires (CDS et GDS)
- Le système de fichiers distribués
- Le système de démarrage de stations sans disque

3.1 Les Threads

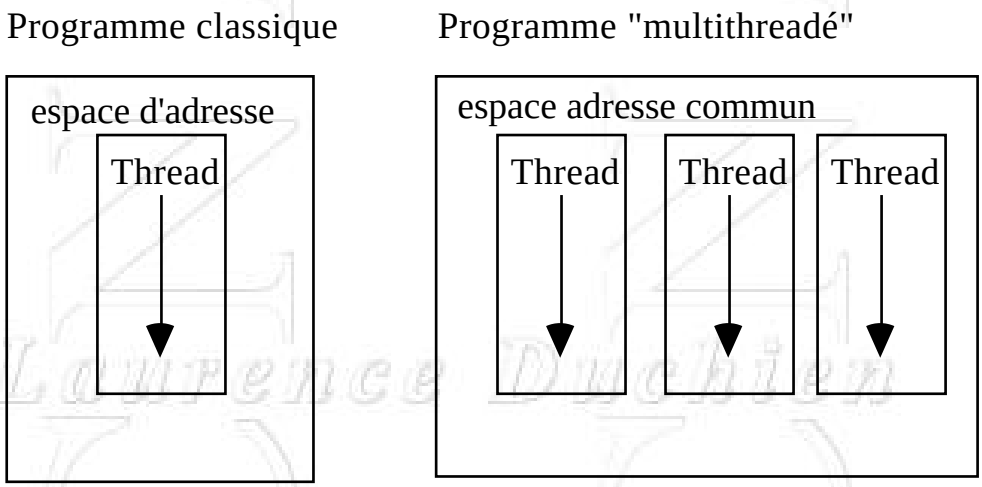
Thread : - flot de contrôle séquentiel indépendant associé à un processus.

- permet de développer des programmes dans lesquels plusieurs flots de contrôle peuvent être spécifiés.

(exemple : lorsqu'une activité exécute une entrée/sortie et est en attente de la fin de cette opération, une autre activité peut exécuter un autre traitement).

=> améliore la performance

=> permet une meilleure structuration de l'application



3.1.1 Quelques exemples de conceptions d'applications avec les Threads

- Modèle "Chef/travailleur" (Boss/worker)

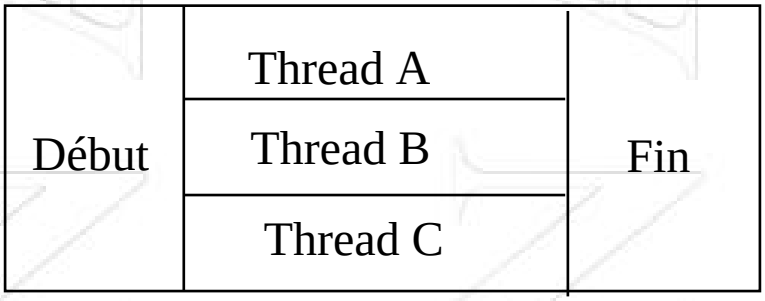
  La fonction du Boss est de donner des tâches à effectuer aux threads travailleurs.

  Lorsqu'un thread travailleur a terminé, il interrompe le chef pour lui indiquer qu'il a terminé.

  Le chef interroge régulièrement les travailleurs pour savoir où ils en sont.

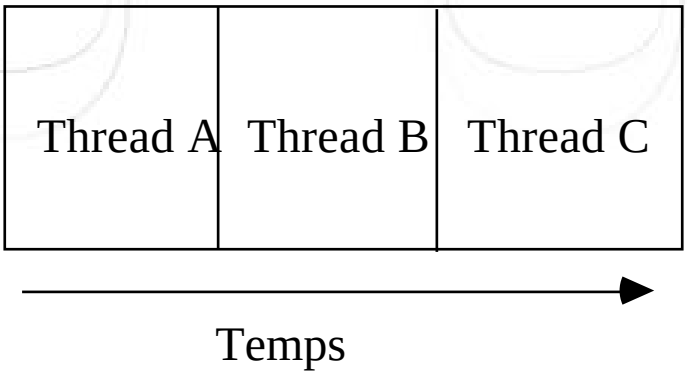
- Modèle "équipe" (crew)

  La tâche à réaliser est divisée en plusieurs threads qui exécutent en parallèle une partie du travail.



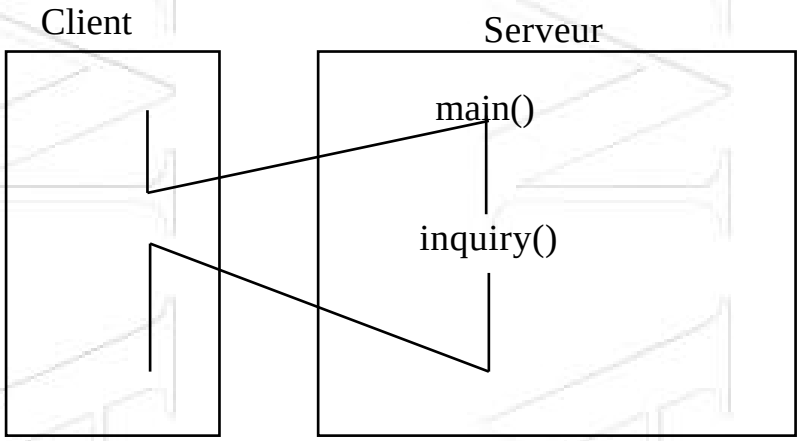
- Le modèle "pipeline"

Une tâche est divisée en étapes. Les étapes sont effectuées en séquence, le résultat d'une étape correspondant aux données en entrée de l'étape suivante:



3.1.1 Quelques exemples de conceptions d'applications avec les Threads(2)

- Exemple de serveur "mono-thread" :

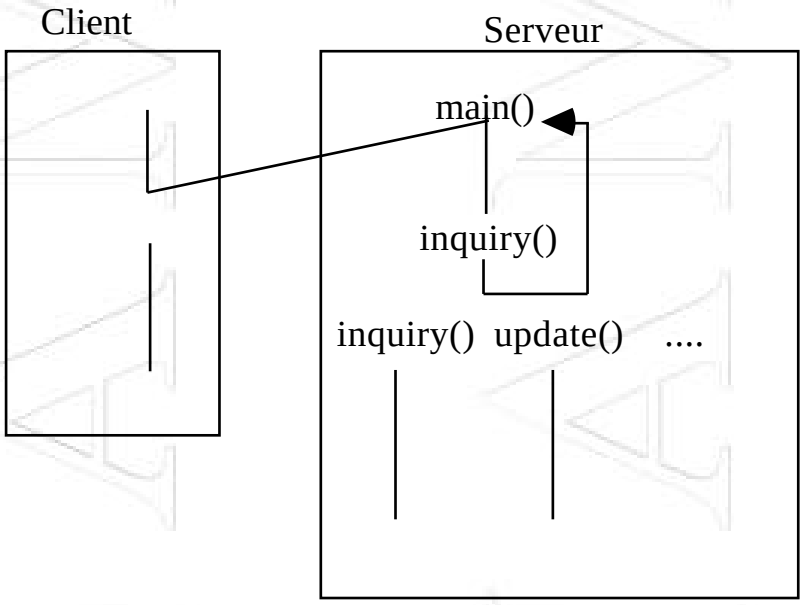



```
void main()
{
    do {
        select();
        switch (msg.req) {
        case inq:
            inquiry(state_entry);
            break;
        case update :
            ....
        }
    } until cows_come_home;
}
}

void inquiry (state_table_pt   state entry)
{ switch (state_entry->state) {
    case initialize :
        .....
    case guts :
        .....
    case final :
        ....
    }
}
```

3.1.1 Quelques exemples de conceptions d'applications avec les Threads (3)

Exemple de serveur "multi-thread" :




```
void main()
{
     do {
     select ();
     switch (msg.req){
          case inq :
               create(...,inquiry,...);
               break;
          case update :
               ...
          }
     } until cows_come_home;
}

void inquiry ()
{
allocate_this;
....
}
```

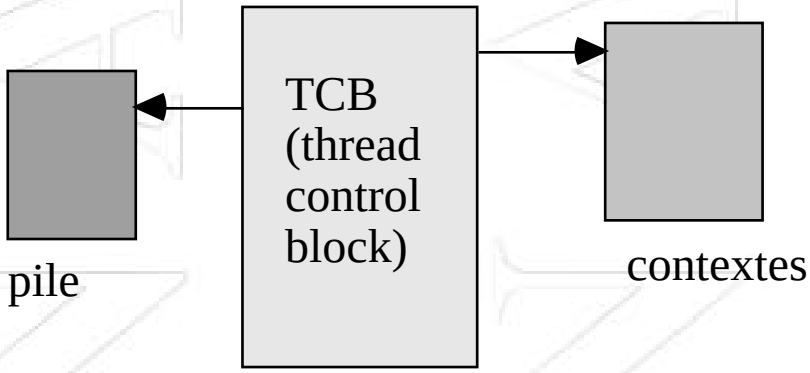
3.1.2 Structure d'un thread

Un thread a : - son propre identificateur

- une politique et une priorité d'ordonnancement

- une partie privée : pile d'exécution, valeur de registre, mémoire locale, variable errno

- une partie partagée : texte du programme, variables statiques, fichiers ouverts, sockets, etc....



- Le package DCE Thread est fondé sur le standard POSIX P1003.4a. Il a été fourni par DEC.
- Il s'agit d'une collection de procédures d'une librairie de niveau utilisateur qui permet **de créer, détruire et manipuler** des threads.
- Ce composant "Thread" fait conceptuellement partie de différents systèmes d'exploitation. S'il existe dans le système sur lequel il est porté, ce composant n'est pas nécessaire.

3.1.3 Les états d'un thread

- Un thread est dans l'un des quatre états suivants :

État	Description
Actif	le thread est actif utilisant le CPU
Prêt	le thread veut occuper le CPU
En attente	le thread est bloqué attendant un événement
Terminé	le thread a existé, mais n'est pas encore détruit

- Sur une machine monoprocesseur, un seul thread est actif à la fois. Sur une machine multiprocesseur, plusieurs threads peuvent être actifs (parallélisme réel).

3.1.4 Quelques problèmes peuvent se présenter dans trois domaines:

- **Au niveau du traitement des signaux :**

  Un signal peut être capturé, ignoré ou mis en attente

  Certains signaux sont synchrones : ils sont causés par le thread actif suite à une erreur (une violation de segmentation , une erreur de calcul en virgule flottante)

  Quand un signal synchrone arrive, il est géré par le thread courant, le processus entier est alors tué, sauf si le signal est ignoré ou capturé.

  D'autres sont asynchrones : ils sont causés par des processus externes (l'appui par l'utilisateur sur la touche DEL pour interrompre le processus en cours)

  Quand un signal asynchrone arrive, le système de thread vérifie qu'aucun thread n'est en attente de ce signal. Si c'est le cas il est passé à tous les threads pouvant le gérer.

- **La plupart des procédures de la librairie standard ne sont pas réentrantes:**

  exemple : il se peut qu'un thread soit occupé à faire de l'allocation mémoire lorsqu'il y a une commutation de contexte (changement de thread). A ce moment là les structures internes de l'allocateur de mémoire sont incohérentes ... ce qui pose des problèmes au nouveau thread actif qui essaie de faire de l'allocation mémoire....

  On résout ce problème en ajoutant à quelques procédures (surtout E/S) des enveloppes qui garantissent un accès en exclusion mutuelle.

  Pour les autres procédures, une exclusion mutuelle globale rend certain que seul un thread à un instant est actif dans la librairie.

  Les procédures comme le read, le fork,... sont des procédures enveloppées.

- **La variable d'erreur *errno***

  les appels système d'Unix retournent leur état d'erreur dans une variable globale *errno*.

  Si l'un des threads fait un appel système, mais, juste après que celui-ci soit terminé, un autre thread est exécuté et fait également un appel système, l'original de la variable **errno** est perdu.

  Une interface de gestion des variables d'erreur est ajoutée.

  C'est une macro qui permet au programmeur de consulter la version spécifique chaque thread de la variable *errno*.

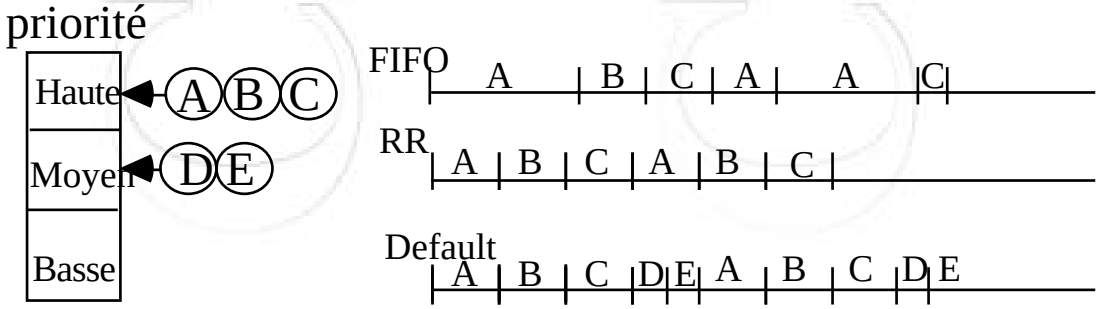
3.1.5 L'ordonnancement des threads

Ordonnancement : fournir une portion de temps CPU

- Rôle de l'algorithme d'ordonnancement :
  - => combien de temps un thread peut s'exécuter ?
  - => quel est le thread suivant ?
- On attribue un priorité à chaque thread.

- Par rapport à la gestion de processus, la gestion des threads est visible par l'application.

- DCE offre trois algorithmes d'ordonnancement : FIFO, Round Robin, default :

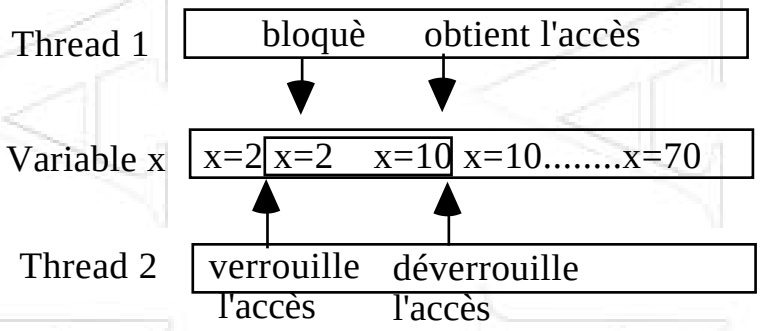


3.1.6. La synchronisation

=> Deux moyens de synchroniser des threads:

- l'exclusion mutuelle
- les variables de condition

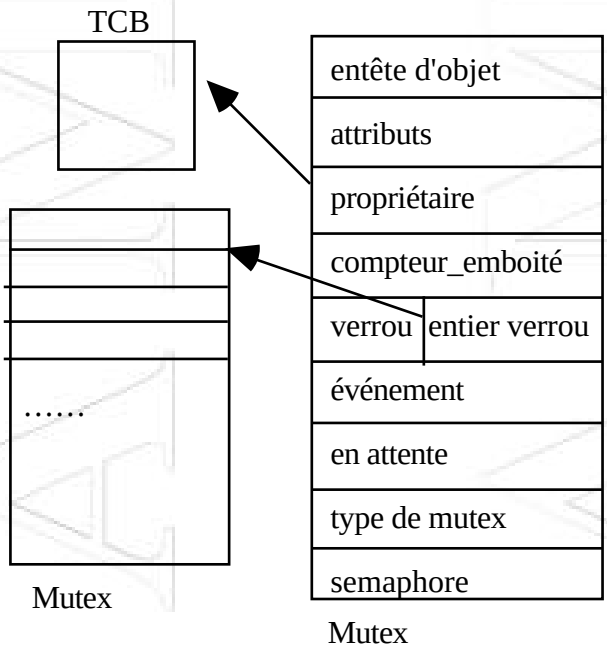
Exemple d'accès partagé à une variable globale :



Trois types d'exclusion mutuelle sont disponibles :

Type d'exclusion mutuelle	Propriétés
Rapide	bloqué si il y accède une seconde fois
Récursif	permet à un thread d'accéder une seconde fois à un verrou
Non récursif	verrouillé une seconde fois provoque une erreur

La représentation d'une variable d'exclusion mutuelle



Les variables de conditions :

- sont utilisées en conjonction avec les exclusions mutuelles:

=>permet d'attendre un événement, de libérer un ressource

=>permet de recevoir une notification d'événement, d'acquérir une ressource

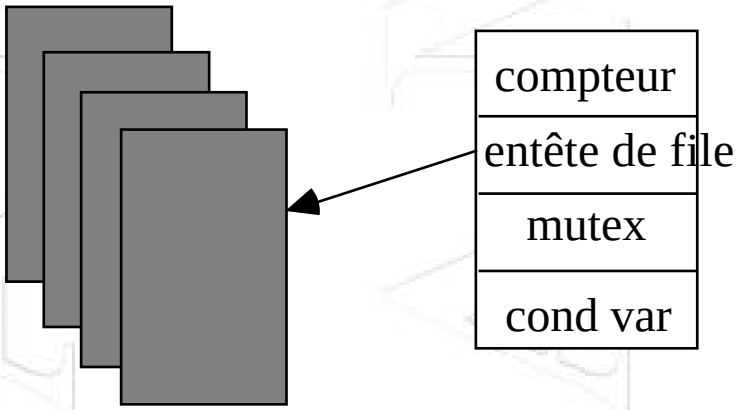
- il s'agit de bloquer une exécution jusqu'à ce que des variables partagées atteignent un état pré-défini.

- les threads testent les données partagées et restent en attente que la variable de condition soit réalisée.

- contiennent : une entête d'objet, un pointeur vers les attributs de description et un sémaphore représentant la file des threads en attente.

exemple d'utilisation : la gestion d'une file d'éléments

- un thread produit des éléments dans une liste qui sont consommés par un autre thread. Si la file est vide, le second thread reste en attente sur une variable de condition indiquant s'il y a un élément à traiter.




```
acquire_resource(resource)
{
ret=pthread_mutex_lock(resource.mutex);
while (resource.count==0)
     ret=pthread_cond_wait(resource.cond,
resource.mutex);
my_resource=dequeue(resource.queue);
resource.count--;
ret=phread_mutex_unlock(resource.mutex);
return(my_resource);
}

release_resource(resource)
{
ret=phthread_mutex_lock(resource.mutex);
enqueue(resource.queue,resource);
resource.count++;
ret=pthread_cond_signal(resource.cond);
ret=pthread_mutex_unlock(resource.mutex);
}
```

3.1.7. Les primitives de gestion des threads

- Au total 54 primitives permettent la gestion des threads.
- Quelques appels de primitives pour la gestion de threads:

Appel	Description
Create	Création d'un nouveau thread
Exit	Appelé par un thread lorsqu'il est terminé
Join	Comme l'appel système Unix WAIT
Detach	indique qu'il n'est pas nécessaire pour le parent d'attendre que l'appelé se termine.

- Quelques appels de primitives pour la gestion des exclusions mutuelles:

Appel	**Description**
Mutex_init	Création d'une variable d'exclusion mutuelle
Mutex_destroy	Destruction d'une variable d'exclusion mutuelle
Mutex_lock	essai de verrouiller la variable d'exclusion mutuelle. Si elle est déjà verrouillée, il y a blocage
Mutex_trylock	essai de verrouiller la variable d'exclusion mutuelle. Si elle est déjà verrouillée, il y a abandon
Mutex_unlock	déverrouille la variable d'exclusion mutuelle

- Quelques appels de primitives pour la gestion des variables de condition:

Appel	Description
Cond_init	Création d'une variable de condition
Cond_destroy	Détruire la variable de condition
Cond_wait	attendre sur une variable de condition jusqu'à ce qu'un signal ou un événement diffusé arrive
Cond_signal	réveille au moins un thread en attente sur une variable de condition
Cond_broadcast	réveille tous les threads en attente sur la variable de condition

- Quelques appels de primitives pour la gestion de variables globales:

Appel	Description
Keycreate	Création d'une variable globale pour ce thread
Setspecific	affecter une valeur de pointeur à une variable globale par thread
Getspecific	lire une valeur de pointeur d'une variable globale

- Quelques appels de primitives pour la destruction de thread:

Appel	Description
Cancel	essaye de tuer un autre thread
Setcancel	donne la possibilité ou non à d'autres threads de tuer ce thread

- Quelques appels de primitives pour la gestion de l'ordonnancement de threads:

Appel	**Description**
Setscheduler	règle l'algorithme d'ordonnancement
Getscheduler	lit l'algorithme courant d'ordonnancement
Setprio	règle la priorité d'ordonnancement
Getprio	donne la priorité courant d'ordonnancement

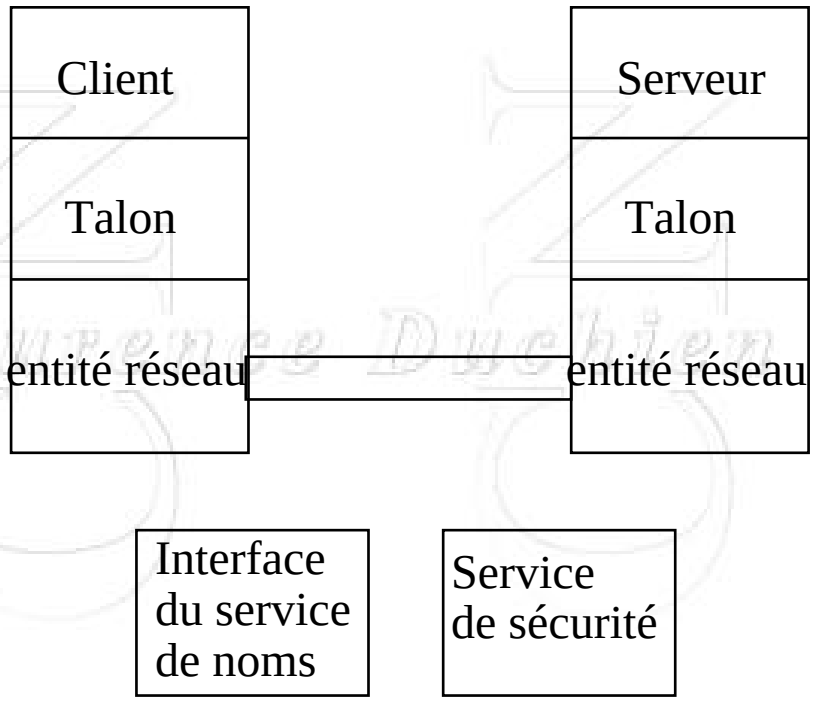
3.2 Le RPC- Appel de Procédure distante

=> le concept de base : étendre l'appel de procédure local à un appel distant

intérêt :

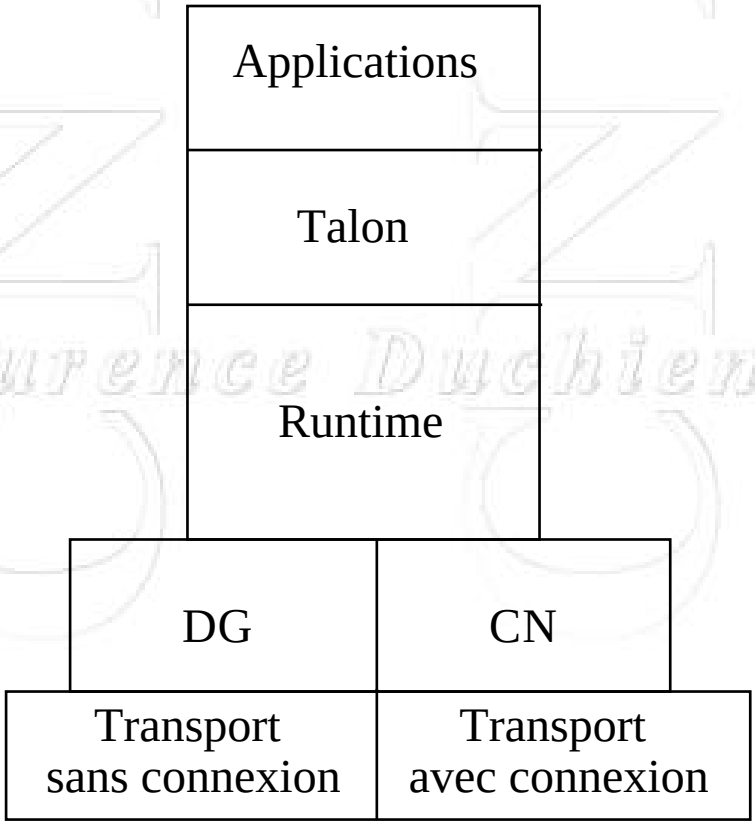
- changer le moins possible les habitudes du programmeur
- réutiliser du code existant avec peu de changement
- essayer de cacher les détails de la programmation distribuée

Schéma classique :

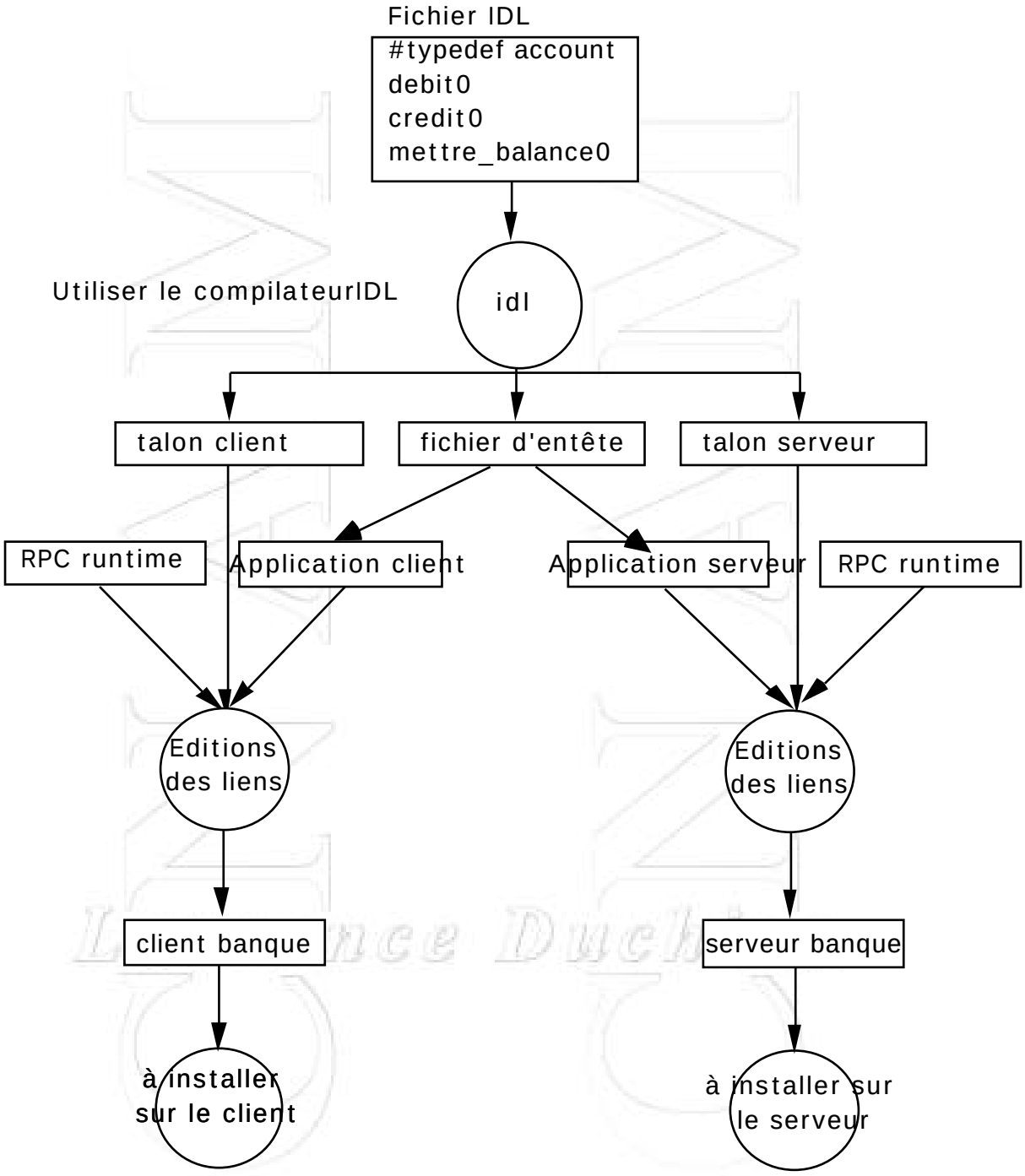


3.2.1 Les différents composants du RPC de DCE:

- le langage de définition d'interface (IDL) et son compilateur
- La librairie du Runtime RPC
- le RPC authentifié
- les API du service de désignation (NSI)
- le démon RPC
- le programme de contrôle RPC
- des possibilités de génération d'identificateurs uniques universels (UUIDs)



3.2.2 Écrire un client et un serveur



- **Le fichier IDL :**

Le programmeur d'application définit l'interface RPC et les types des données associées en utilisant le langage de définition d'interface (IDL).

Une interface est un groupe d'opérations qu'un serveur peut exécuter (similaire à un module ou à une librairie).

exemple : l'ensemble des services bancaires

Après compilation, on a un talon client et un talon serveur et un fichier d'en-tête.

Les talons sont les primitives nécessaires pour gérer les appels distants.

- **Côté client :**

le programmeur écrit le code qui fait l'appel des opérations. Le code du talon client est lié au code de l'application.

- **Côté serveur** :

Le programmeur écrit les primitives d'application qui implémente les opérations définies dans l'IDL.

exemple : dans l'application bancaire, une recherche dans la base de données peut implémenter l'opération de lecture.

Le talon serveur, généré par le compilateur d'IDL est lié au code applicatif du serveur.

3.2.3 La localisation du serveur et du client

Avant d'appeler un serveur, un client doit le localiser et se "lier" avec lui (se faire reconnaître).

- un utilisateur naïf peut ignorer ce processus de liaison et laisser les talons faire ce travail.
- un utilisateur averti peut contrôler la démarche et demander la connexion avec un processus particulier.

=> Le problème essentiel : comment le client fait-il pour localiser le bon serveur ?

Une possibilité : diffuser un message contenant un identificateur unique vers tous les processus de toutes les cellules demandant aux serveurs s'ils sont prêts à donner un coup de main..... trop coûteux!

La localisation se fait en deux étapes :

- localiser la machine du serveur
- localiser le bon processus sur cette machine

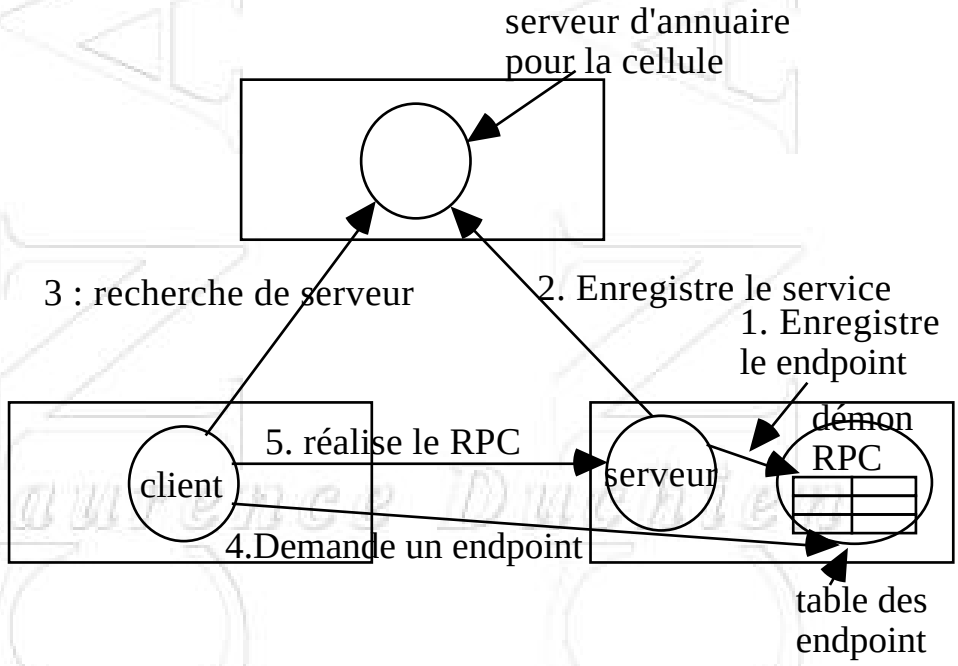
On utilise ≠ mécanismes pour ces étapes:

. Besoin de connexions entre client et serveur pour communiquer de manière fiable et sécurisée

. Besoin d'une adresse numérique (endpoint) sur la machine du serveur pour les connexions et l'envoi de messages.

. Les adresses numériques doivent être dynamiques et maintenues par une base de données des entrées (serveur, endpoint) est maintenue sur chaque machine serveur par un processus appelé RPC daemon (rpcd).

- Les différentes étapes nécessaires à la liaison sont :



3.2.4 Les sémantiques du RPC

Plusieurs types de pannes peuvent se produire:

- le client
- le réseau
- le serveur

Le RPC de DCE propose plusieurs sémantiques:

- au plus une fois
- au moins une fois (opérations idempotentes)
- la diffusion à tous du RPC

3.2.5 L'indépendance vis à vis du système

- indépendance du langage:

  Les talons générés par les paramètres de l'IDL peuvent être appelés par des programmes écrits dans n'importe quel langage.

  Le compilateur d'IDL génère des talons qui utilise les conventions du langage C, mais un client écrit en Fortran peut appeler un talon de la même manière que s'il appelait une fonction locale en C.

- indépendance de la représentation des données

  Par défaut, la représentation des données est la syntaxe de Transfert DCE : Network Data Representation (NDR).

  Elle permet la représentation des types classiques mais aussi de types construits.

  Un avantage : si les deux machines sont de même type, on conserve la représentation interne des machine.

  L'architecture de DCE permet l'utilisation d'autres formats de présentation comme l'ASN.1/BER

- indépendance du protocole

  Le RPC repose sur deux 2 protocoles, l'un en mode connecté, l'autre sans connexion. Le programmeur a la possibilité d'indiquer dans ses appels au RPC le type de protocole utilisé.

  L'annuaire qui identifie les serveurs note les protocoles utilisés par le serveur. Il met en relation client et serveur ayant le même type de protocole.

- indépendance de la machine

  Par la syntaxe de transfert, les applications distribuées sont indépendantes de la machine.

- indépendance du système d'exploitation

  Le programmeur n'utilise pas directement les appels systèmes du réseau et du système d'exploitation local.

3.2.6 Un exemple

Le problème :

Le client de l'application envoie un message de bienvenue au serveur de l'application.

Le serveur imprime le message et envoie une réponse au message de bienvenue au client. Le client imprime la réponse du serveur et se termine.

Les différentes étapes du développement de l'application :

1. Générer un squelette IDL

=> utiliser le programme **uuidgen** qui crée un identificateur unique afin d'identifier de manière unique l'interface de l'application :

uuidgen -i > greet.idl

cela crée le fichier greet.idl qui contient :

```
[uuid(3d6ead56-06e3-11ca-8dd1),
version(1.0)
]
interface INTERFACENAME
{
}
```

2. Donner un nom à l'interface:

```
[uuid(3d6ead56-06e3-11ca-8dd1),
version(1.0)
]
interface greetif
{
}
```

3. Définir les opérations de l'interface:

```
/*
* greet.idl  The "greet interface
*/

[uuid(3d6ead56-06e3-11ca-8dd1),
version(1.0)
]
interface greetif
{
const long int REPLY_SIZE=100;
void greet (
     [in]     handle_t h,
     [in, string] char client_greeting[],
     [out,string] char server_reply[REPLY_SIZE]
     );
}
```

4. faire fonctionner le compilateur d'IDL :

idl greet.idl

ce qui donne les fichiers suivants :

greet.h

greet_cstub.o

greet_sstub.o

5. Ecrire le code de l'application client : greet_client.c

```
/*
* greet_client.c
*/
#include <stdio.h>
#include <dce/nbase.h>
#include <dce/rpc.h>
#include <greet.h>
#include <util.h>

int
main(
    int argc,
    char *argv[]
)
{
rpc_ns_handle_t import_context;
handle_t binding_h;
error_status_t status;
idl_char reply[REPLY_SIZE];

if (argc<2){
    fprintf(stderr,"usage: greet_client <chemin
CDS>\n");
    exit(1);
}

/* debut de l'importation des serveurs utilisant le
 * nom utilisé sur la ligne de commande
 */
rpc_ns_binding_import_begin(
    rpc_c_ns_syntax_default, (unsigned_char_p_t)
argv[1], greetif_v1_0_c_ifspec, NULL, &import_context,
&status);
ERROR_CHECK(status,"ne peut pas être importe");

/* importe le premier serveur
 */
rpc_ns_binding_import_next(import_context,
&binding_h, &status);
ERROR_CHECK(status,"ne peut pas être importe");

/* Lance l'appel distant
 */
greet(binding_h,(idl_char*)"hello, server", reply);
printf("le server greet a repondu : %s\n", reply);
}
```

6. Ecrire le code de l'application serveur : greet_serveur.c

```
/*
 * greet_server.c
 */
#include <stdio.h>
#include <dce/dce_error.h>
#include <dce/rpc.h>

#include "greet.h"
#include "util.h"

int
main ( int argc,
     char * argv[]
     )
{
     unsigned32 status;
     rpc_binding_vector_t *binding_vector;

     if (argc<2) {
          fprintf(stderr,"usage : greet_server
<chemin CDS>\n");
          exit(1);
          }

     /* enregistre l'interface avec le runtime RPC
      */

     rpc_server_register_if(greet_if_v1_0_s_ifspec,
NULL, NULL, &status);
     ERROR_CHECK(status, "ne peut pas enregistre
l'interface");

     /*
      * utilise tous les protocoles qui sont
      * disponibles
      */
     rpc_server_use_all_protseqs(
rpc_c_protseq_max_reqs_default, &status);
     ERROR_CHECK(status, "ne peut pas utiliser tous
les protocoles");
     /*
      * gère les lins avec le runtime
      */
     rpc_server_inq_bindings(&binding_vector,
&staus);
```

```
        ERROR_CHECK(status, "ne peut pas faire le lien
pour le serveur");
        /*
         * enregistre les points d'accès affectés dans
         * le "mapper" de point d'acces (rpcd)
         */
        rpc_ep_register(greetif_v1_0_s_ifspec,
binding_vector, NULL, (unsigned_char_p_t) "serveur greet
version 1.0", &status);
        ERROR_CHECK(status, "ne peut enregistrer le
point d'acces dans le mapper");
        /*
         * s'exporter dans l'espace de noms CDS
         */
        rpc_ns_binding_export (rpc_c_ns_syntax_default,
(unsigned_char_p_t) argv[1],greetif_v1_0_s_ifspec,
binding_vector, NULL, &status);
        ERROR_CHECK(status, "ne peut s'exporter dans
l'espace de noms CDS");
        /*
         * commence à écouter les appels
         */
        printf (" ecoutons.... \n");

        rpc_server_listen(
    rpc_c_listen_max_calls_default, &status);
        ERROR_CHECK(status, " n'arrive pas a demarrer
        l'ecoute des appels");
        /*
         * on enleve un point d'entree dans le mapper
         */
        rpc_ep_unregister(greetif_v1_0_s_ifspec,
binding_vector, NULL, &status);
        ERROR_CHECK(status, "ne peut pas enlever un
point d'entrée dans le mapper");

}
```

7. Ecriture du code de l'opération sur le serveur : **greet_manager.c**

```
/*
 * Greet_manager.c
 * implementation de l'interface greet
 */
# include <stdio.h>
# include "greet.h"

void greet(
    handle_t h,
    idl_char *client_greeting,
    idl_char *server_reply);
{
    printf(" Le client a dit : %s\n", client_greeting);
    strcpy(server_reply, "Hi, client");
}
```

8. Ecriture des divers utilitaires

9. Compilation des programmes serveurs et clients

cc -o greet_client greet_client.c greet_cstub.o util.o -ldce

cc -o grret_server greet_server.c greet_manager.c greet_sstub.o util.o -ldce

10. Mise en place de l'application sur le client et le serveur.

3.3 Le service de temps distribué (DTS)

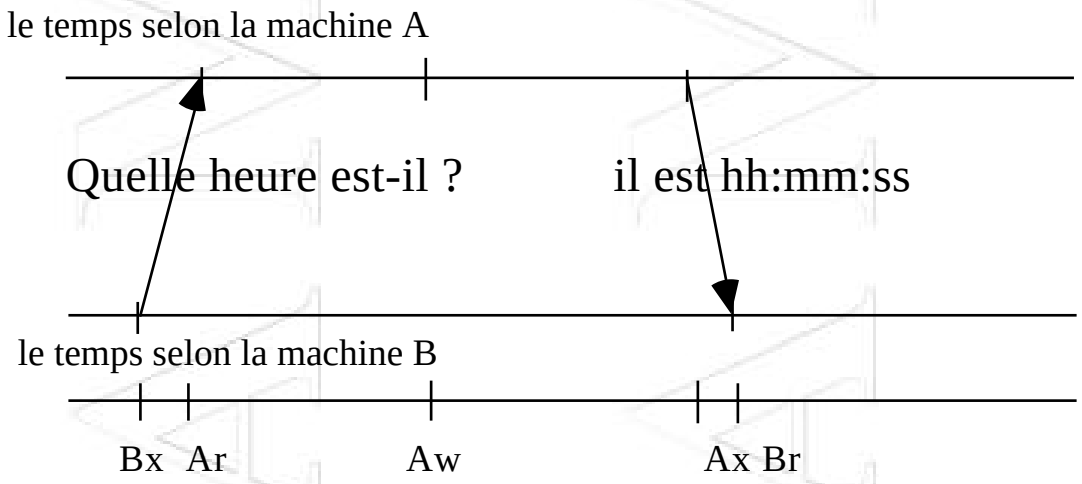
L'un des problèmes d'une architecture distribuée est de garder **ses horloges synchronisées**.

Quelques exemples des problèmes que l'on peut rencontrer dans des applications distribuées sans synchronisation d'horloge:

- capture de résultats sur différentes machines et essai de remise dans l'ordre temporel des événements.
- vente distribuée aux enchères,
- …

3.3.1 Le problème de la synchronisation

Le problème de synchronisation n'est pas simple :



- On peut mesurer le temps total écoulé entre Bx et Br:
- soit AW(x) l'heure da la machine A à l'instant Bx.
- on connait AW(Aw),
- et on souhaite connaître AW(Br).

Si l'on suppose que Aw a pu être donné n'importe quand dans l'intervalle [Bx,Br], l'estimation de AW(Br) est quelque part dans l'intervalle [AW(Aw), AW(Aw)+(Br-Bx)]

- Une horloge n'utilise pas un temps continu.

Elle donne les "tops" d'horloge .

on note r la longueur d'un top d'horloge.

- Les quartz des horloges dérivent de quelques millisecondes toutes les heures.

on note d la constante de dérive de l'horloge.

On peut améliorer notre formule précédente :

[AW(Aw), AW(Aw)+(Br-Bx + r)(1-d)]

Quelques valeurs caractéristiques d'un système distribué :

r a pour valeur a 10 millisecondes

Le délai de communication est de 20 millisecondes

et la dérive d'horloge peut être de 0,0004

Que faire ?

- Consulter plusieurs serveurs de temps et faire une moyenne des résultats obtenus.

- Consulter plusieurs serveurs qui répondent en donnant l'heure avec une borne sur l'exactitude de leur estimation du temps.

- Le client obtient plusieurs intervalles à partir desquels il peut déterminer un intervalle de temps qui est plus étroit que celui obtenu par l'interrogation d'un seul serveur.

=> essayer d'arriver à un consensus....

Autre problème : les menteurs

3.3.2. Ce qu'offre DCE

DTS gère les horloges de DCE par des serveur de temps

DTS connaît la vitesse de dérive de chaque horloge

DTS fait fonctionner un protocole de mise à l'heure des horloges relativement souvent afin d'atteindre la synchronisation souhaitée.

exemple : pour des horloges qui dérive de 1 pour 10^6, si aucune horloge ne doit dériver de plus de 10 ms, il faut resynchroniser toutes les 3 heures.

DTS offre :

- la possibilité de garder les horloges cohérentes,
- la possibilité de garder les horloges en contact avec la réalité

ce qui permet :

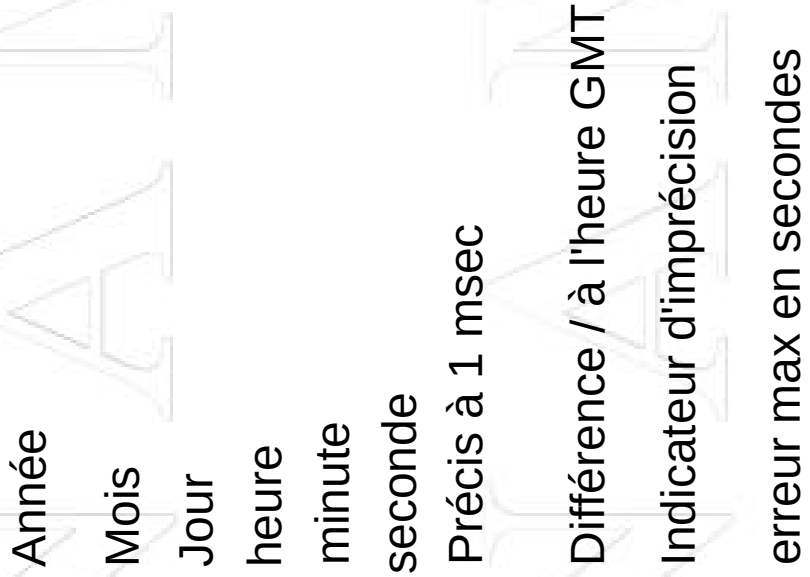
- d'être sûr que toutes les horloges retournent la même valeur lorsqu'on leur demande l'heure
- et même si les horloges retournent la même valeur, il s'agit bien de l'heure correspondant à l'heure en vigueur à travers le monde.

3.3.3 Le modèle

DTS fournit l'heure avec un intervalle de confiance.

L'heure est stockée de manière interne sur un nombre binaire de 64-bits.

Mais l'affichage se fait de la manière suivante :



1789-07-14-15:30:00.000-05:00-I-005.000

L'enregistrement des temps sous forme d'intervalle introduit un pb supplémentaire : il n'est pas possible de dire si un temps est avant un autre.

exemple : un make file Unix

- un fichier source a un temps de dernier enregistrement compris entre 10:35:10 et 10:35:15

- et le fichier binaire correspondant 10:35:14 et 10:35:19.

- le binaire est-il plus récent ?

Lorsqu'un programme DTS compare deux temps, il y a trois réponses possibles :

- le premier temps est le plus vieux
- le second est plus vieux
- DTS ne peut répondre

DTS supporte 33 primitives permettant de gérer le temps en 6 groupes :

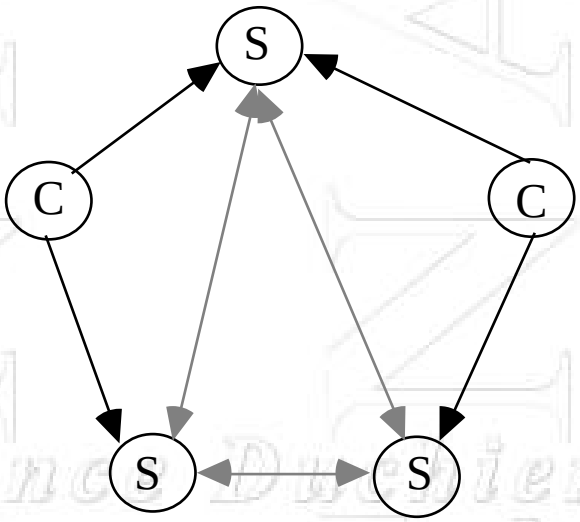
Groupe	**nb d'appels**	**Description**
Retrouver l'heure	2	donner l'heure
Convertir l'heure	18	conversion binaire-ascii
manipuler l'heure	3	intervalle arithmétique
comparer l'heure	2	comparer deux temps
calculer l'heure	5	opérations arithmétiques sur le temps
utiliser les zones de temps	3	gestion des zones de temps

3.3.4 Les différents composants du DTS

Le logiciel DTS contient les composants suivants:

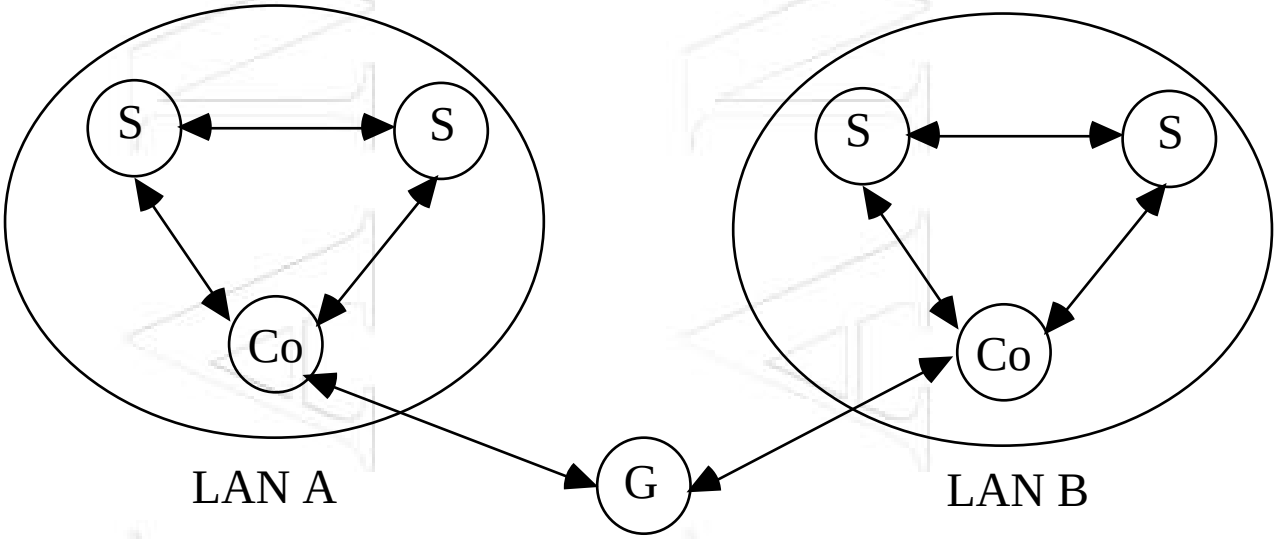
- le time clerck (employé du temps) : fonctionne côté client, synchronise l'horloge locale en demandant l'heure corrigée au serveur de temps.

- le serveur de temps : il s'agit d'un noeud désigné pour répondre aux questions à propos du temps. Le nombre de serveurs de temps dans une cellule est configurable. 3 par LAN est un nombre classique.



On distingue parmi les serveurs de temps:

- serveur de temps local :
- serveur de temps global
- serveur de temps messager (courier)
- serveur de temps messager de secours

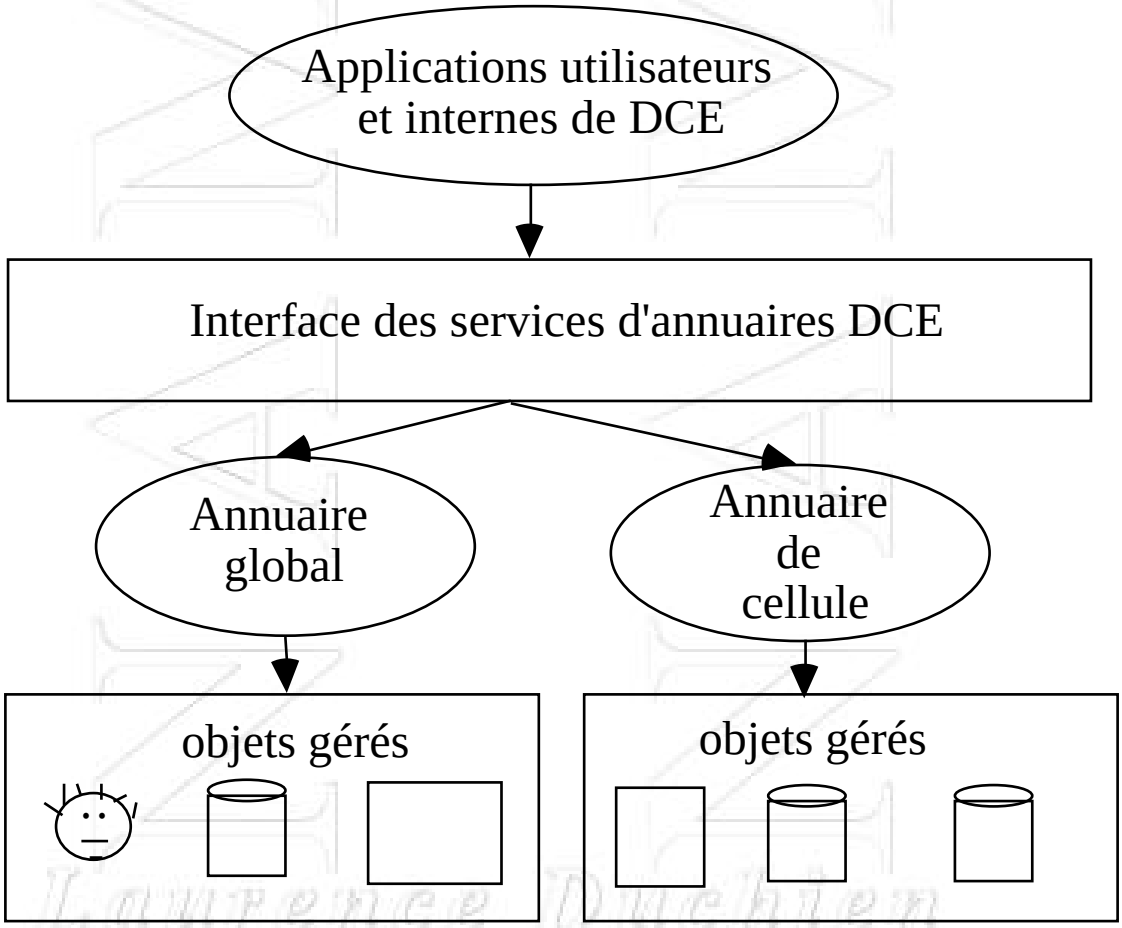


Ces serveurs de temps doivent pouvoir se synchroniser avec un temps externe. Cela se fait par l'intermédiaire d'un périphérique externe (radio, téléphone ou administrateur)

L'interaction avec NTP (Network Time Protocol) - standard Internet- est également un moyen de synchronisation possible pour un système DTS.

3.4 La localisation des ressources

- La vue utilisateur du système d'annuaires de DCE



- L'architecture du serveur d'annuaire est un service de base de données

  - dupliquée : => grande disponibilité, fiabilité
  - distribuée : => grande disponibilité

- La base de données du service d'annuaire consiste en un ensemble de noms organisés hiérarchiquement.
- Exemple de requêtes :

  recherche de la localisation d'un fichier, d'un service, d'un utilisateur,...

3.4.1. Les différents composants du service d'annuaire

- **Le service d'annuaire (ou de répertoire) de cellule (CDS)**

  - gère les noms et les attributs des ressources localisées dans une cellule DCE.

  - est optimisé pour les accès locaux

  - est dupliqué

  - contient les informations sur les ressources du système, le système de fichiers, la sécurité, les services disponibles.

- **Le service d'annuaire global (GDS)**

  - fondé sur le standard international X500 ou le DNS

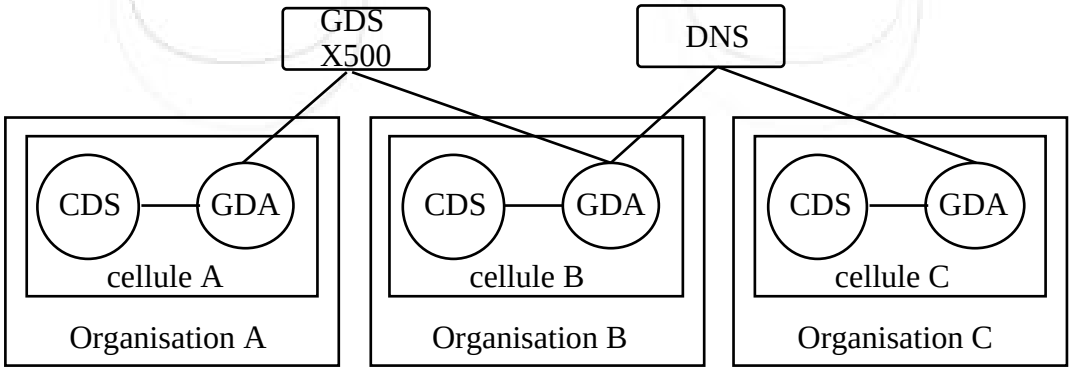
  - est utilisé pour rechercher un nom hors de la cellule locale

  - permet l'indépendance des cellules et leur interactions

  - contient des informations utilisateurs, des no de fax, des entrées non DCE.

- **L'Agent d'annuaire global (GDA)**

  - c'est l'intermédiaire entre l'annuaire d'une cellule et le reste du monde.



3.4.2 L'espace des noms de DCE

- ensemble de noms utilisés par le serveur d'annuaire de DCE
- Cet espace est organisé de manière hiérarchique, similaire au système de fichier Unix
- Les noms sont typés ou non, peuvent être de différents formats (X500, DNS,...)

Préfixe	Nom de la cellule	Nom	Jonction	Application
/...	/C=US/O=IBM		/sec	/principals/linda
/...	/lcs/mit/edu	/subsys		/file

- **Préfixe** : indique si le nom est global pour l'espace global de noms ou local à la cellule.

  /... indique que le nom est global- le nom doit contenir le nom de la cellule.

  /.- indique que le nom est local

  Lorsqu'une requête arrive sur le CDS, il est possible de dire si la requête doit être passée au GDA pour être traitée par le GDS.

- **Le nom des cellules :** peuvent être au format X500 ou DNS

  X500 : chaque entité nommée dispose d'une collection d'attributs permettant de le décrire. (exemple : pays, l'organisation, le département, le nom de famille, le no de bureau, le no de téléphone, ...).

  un nom X500 est une liste d'items *attribute=value* séparée par des /

/C=US/O=YALE/OU=CS/TITLE=PROF/TELEPHONE=3456/OFFICE=210/SURNAME=LIN

  DNS (Domain Name System) est le schéma de désignation des hôtes et des autres ressources de l'Internet.

  Il divise lemonde en des somaines de niveau haut. ce qui correspond aux pays et aux organisations.

  puis ensuite en sous domaines,...

3.4.3 Le service d'annuaire de la cellule

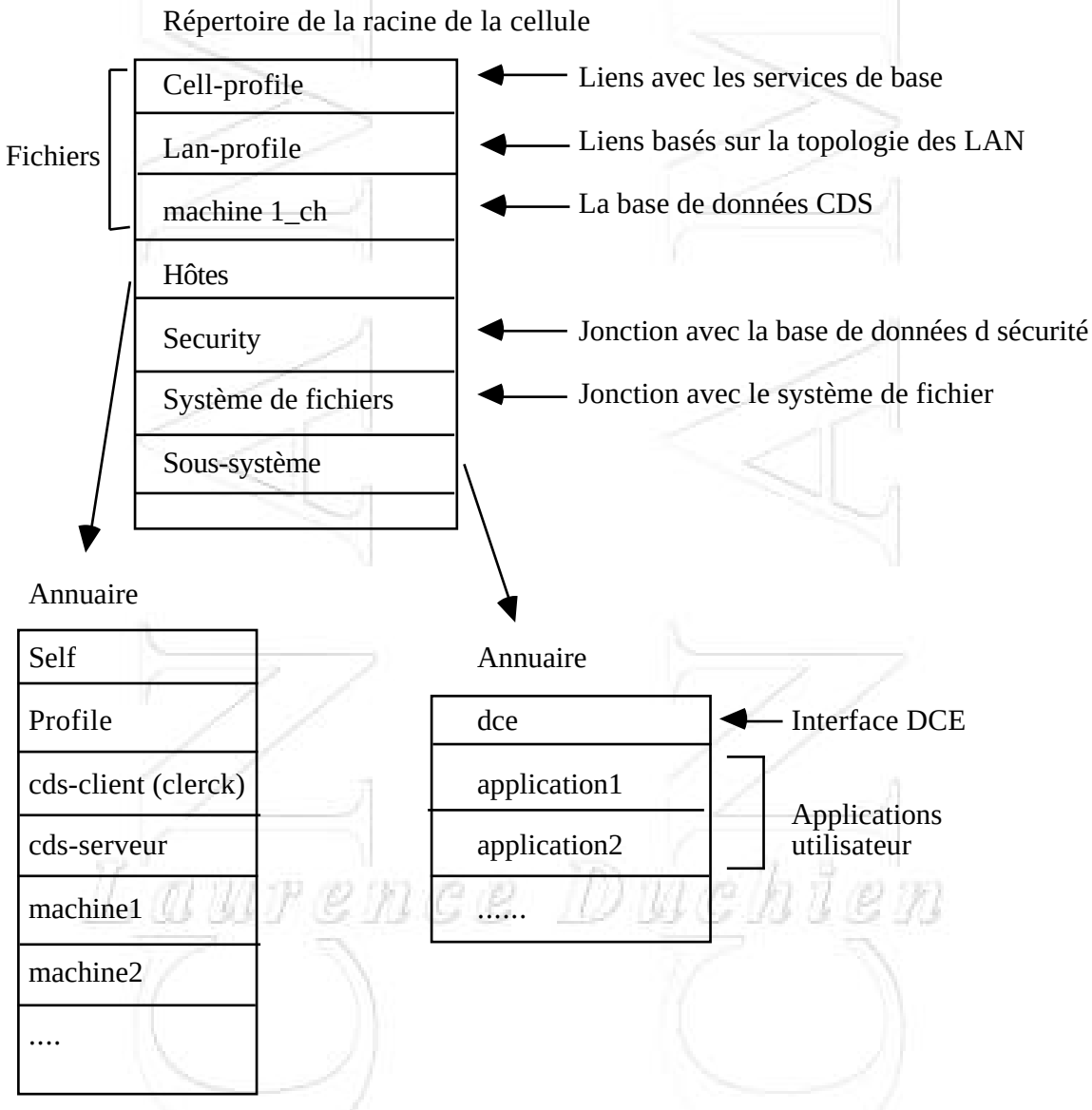
- Nom : identifie de façon propre l'utilisateur final, désigne une entité de façon unique dans un domaine d'appellation.

- Adresse : définit la position du destinataire dans le système réparti, désigne l'emplacement géographique de l'entité.

Pour le service d'annuaire, **un nom est une construction qui identifie de manière non ambiguë un objet particulier au sein d'un ensemble**.

- CDS gère les noms pour une cellule.
- Ceux-ci sont gérés hiérarchiquement.
- Des liens symboliques peuvent être utilisés.

La structure de l'annuaire



Les copies multiples

CDS permet aux entrées d'être dupliquées pour fournir un disponibilité importante et une meilleure tolérance aux pannes:

Ce qui veut dire que la création et la suppression d'entrées sont des opérations d'administration du système.

La base de données physique d'un annuaire est appelée "clearinghouse".

Une cellule peut avoir plusieurs bases de données, donc plusieurs serveurs mais la recherche d'un nom a été conçue pour qu'elle puisse être effectuée par n'importe quel serveur.

-> S'il s'agit d'un nom global, on passe la requête au GDA,

-> S'il s'agit d'un nom local, la racine de l'annuaire contient les premiers composants des noms. La racine est dupliquée sur chaque serveur CDS. Les annuaires pointés sont alors soit local à ce serveur, soit sur d'autres serveurs. Il est alors toujours possible de continuer la recherche et de localiser un nom.

Le problème de la mise à jour des copies multiples:

CDS a choisi le moyen le plus simple :

- une copie maître (recherche et mise à jour)
- une copie esclave (recherche)

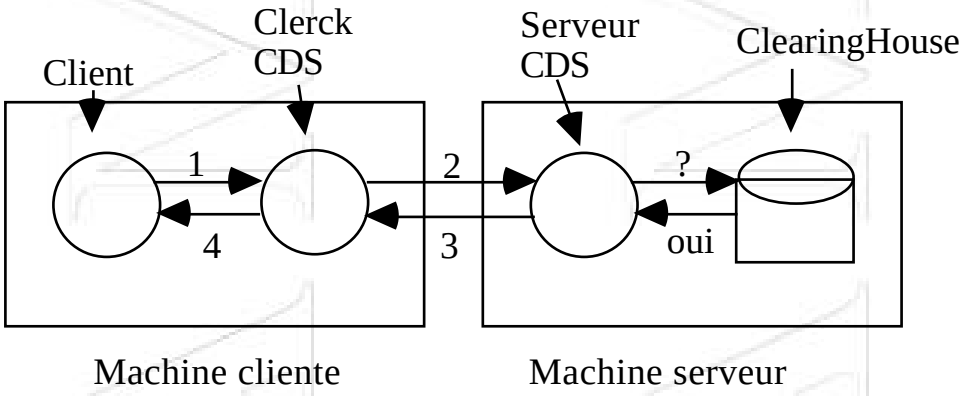
2 options possibles pour effectuer les mises à jour des copies esclaves :

- Pour que les données soient cohérentes à tout moment on diffuse immédiatement vers tous les esclaves les mises à jour

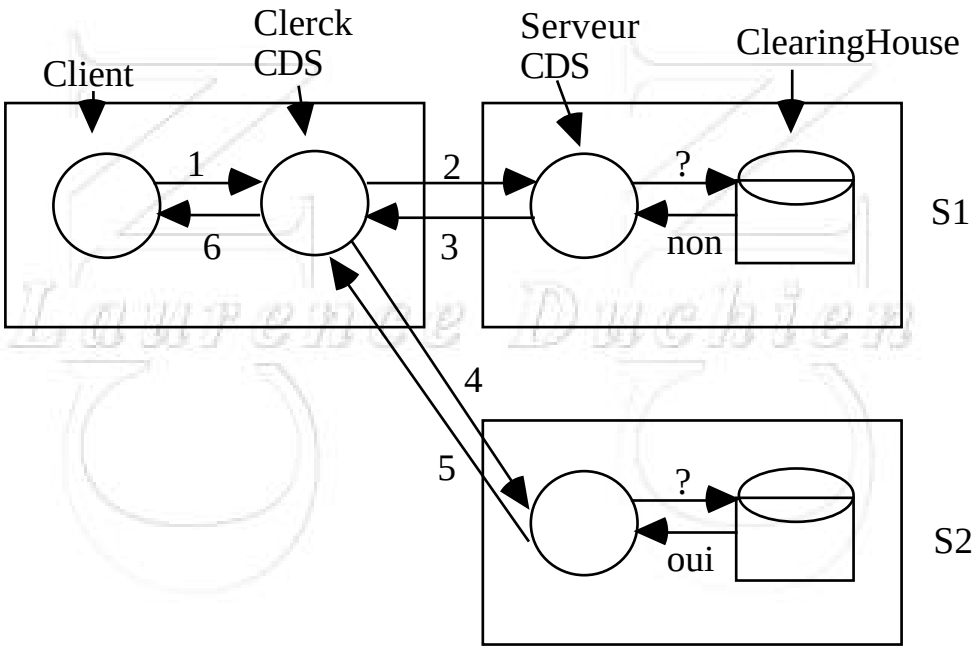
- On groupe les mises à jour et on les envoie par groupe

Exemples de recherche d'un nom

- Le premier serveur contacté détient le nom dans sa base

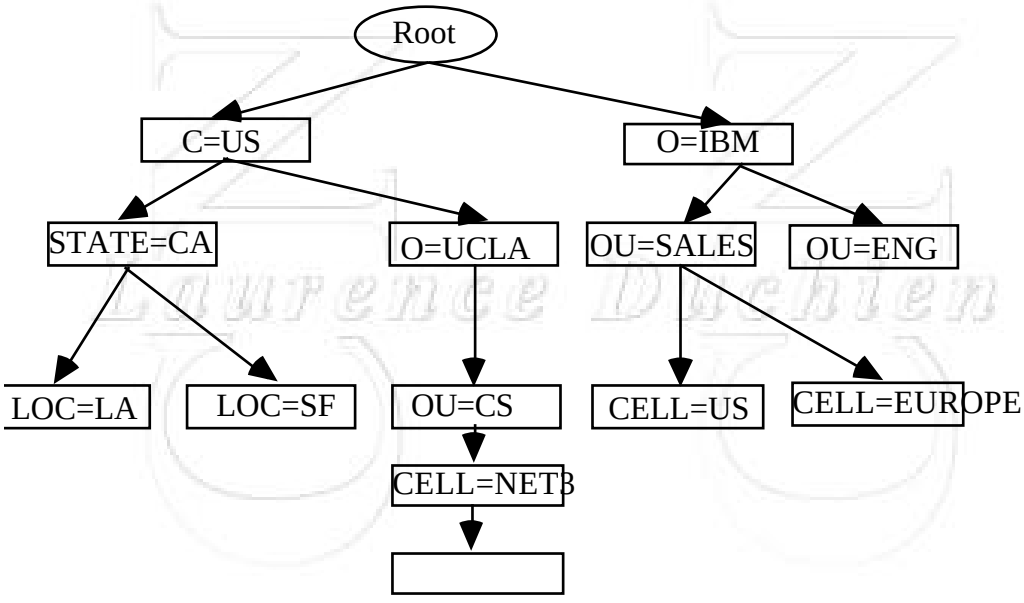


- Le premier serveur ne contient pas le nom demandé



3.4.4 Le service d'annuaire global-GDS

- X500 est un modèle d'information orienté-objet.
- Chaque enregistrement de l'annuaire est un objet.
- Un objet peut être un pays, une compagnie, une ville, une personne, une cellule, un serveur, ...
- Un objet a un ou plusieurs attributs. Un attribut est typé et a une valeur. La syntaxe est la suivante : type=value
- Les objets sont groupés en classe pour référencer un même objet.
- Un classe peut avoir des attributs obligatoires (code postal pour une adresse postale) ou optionnels (numéro de fax pour un particulier)
- Exemple d'arbre d'information d'annuaire X500

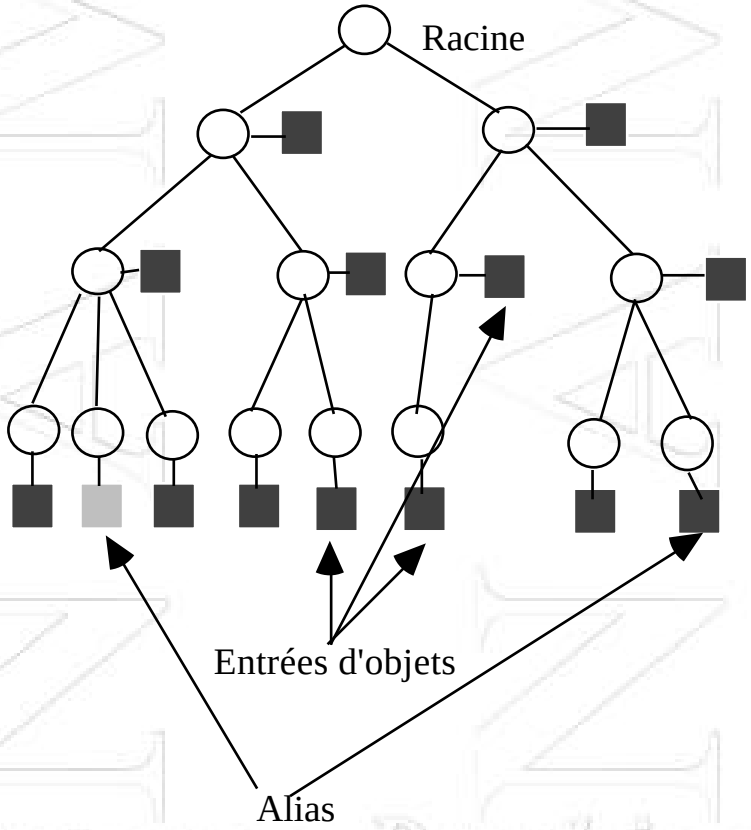


Les chemins de la racine aux feuilles permettent d'identifier une entité. On sépare par des / les attributs.

ex : /C=US/STATE=CA/LOC=SF/O=JOE'S-DELI/CN=JOE

p La base de données correspondant à l'arbre est composée d'entrées d'annuaire. Une entrée peut être **une entrée d'objet** (nom complet) ou une entrée *"alias"*. Un arc représente une relation hiérarchique entre deux objets.

p Chaque sommet de l'arbre contient les informations décrivant un objet particulier. Un sommet est dit *"entrée d'objet"*. La racine de l'arbre est un sommet particulier. Elle ne décrit aucun objet. Elle constitue le point d'entrée de l'annuaire.



p les entrées près de la racine représentent les informations telles que les organisations, les pays. Les objets les plus éloignés de la racine représentent des personnes, des processus d'applications, des ressources matérielles.

Plusieurs types d'objets sont normalisés : adresse OSI PSAP, adresse O/R, numéro de téléphone, numéro de fax.

Le RDN (Relative Distinguished name) est un type d'attribut particulier dans une entrée d'objet suivi de sa valeur. Il constitue l'information élémentaire caractérisant l'objet décrit.

On obtient un nom spécifique DN d'une entrée d'objet par adjonction successive des noms spécifiques relatifs des différentes entrées parcourues dans le chemin menant à cet objet.

3.4.5 La structure de l'annuaire

La structure et les propriétés de X500 sont décrits par le schéma de l'annuaire. Ce schéma est décrit par un ensemble de règles de structure, de classes d'objets et d'attributs pour l'arbre d'information.

On définit trois tables :

- La table des règles de structures: où chaque objet appartient à l'arbre
- La table de classe d'objet : les relations d'héritage à travers les classes
- la table d'attributs : spécifie la taille et le type de chaque attribut

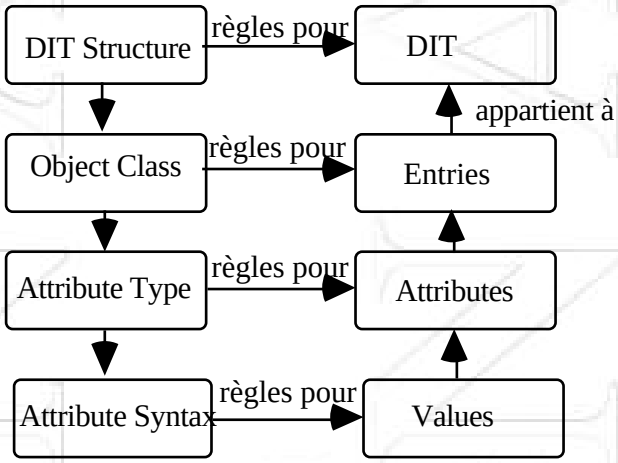
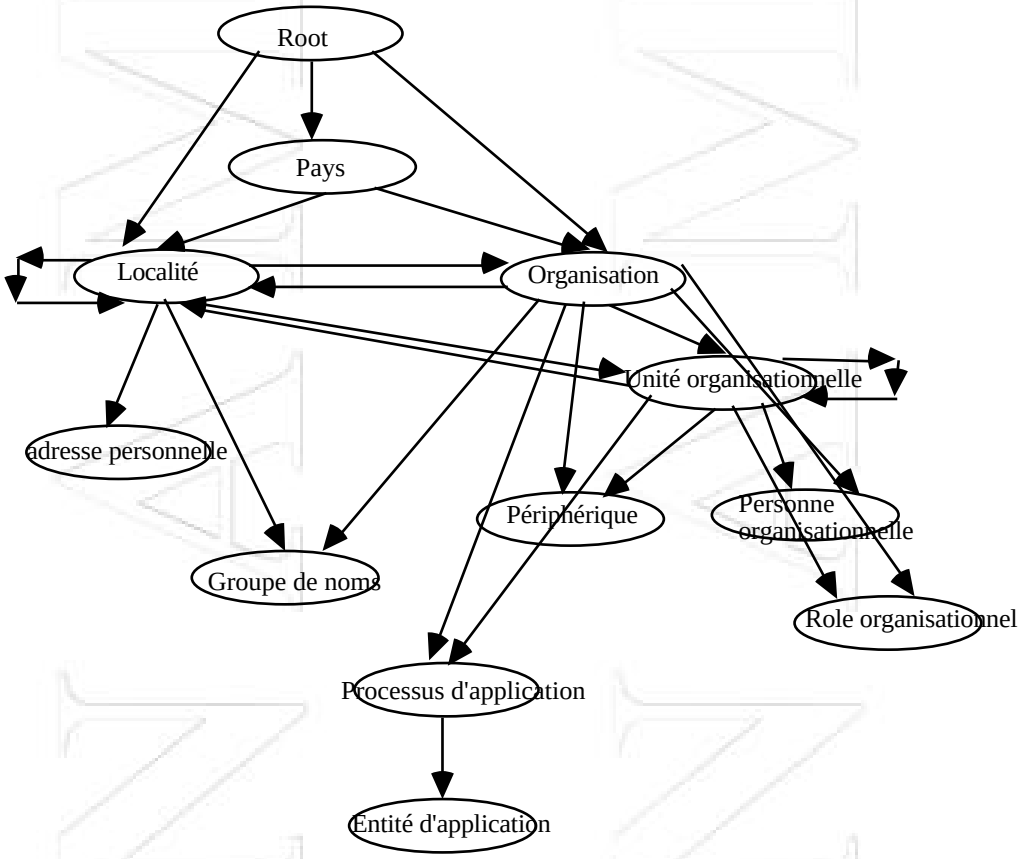


schéma d'annuaire

Structure de l'arbre d'information proposé par X500



Structure du DIT proposé par le Standard X500

3.4.6 Les services d'accès à l'annuaire

pL'interface standard d'X500 est XOM (X/Open object Management). Cependant l'accès DCE utilise XDS (X/Open Directory Server).

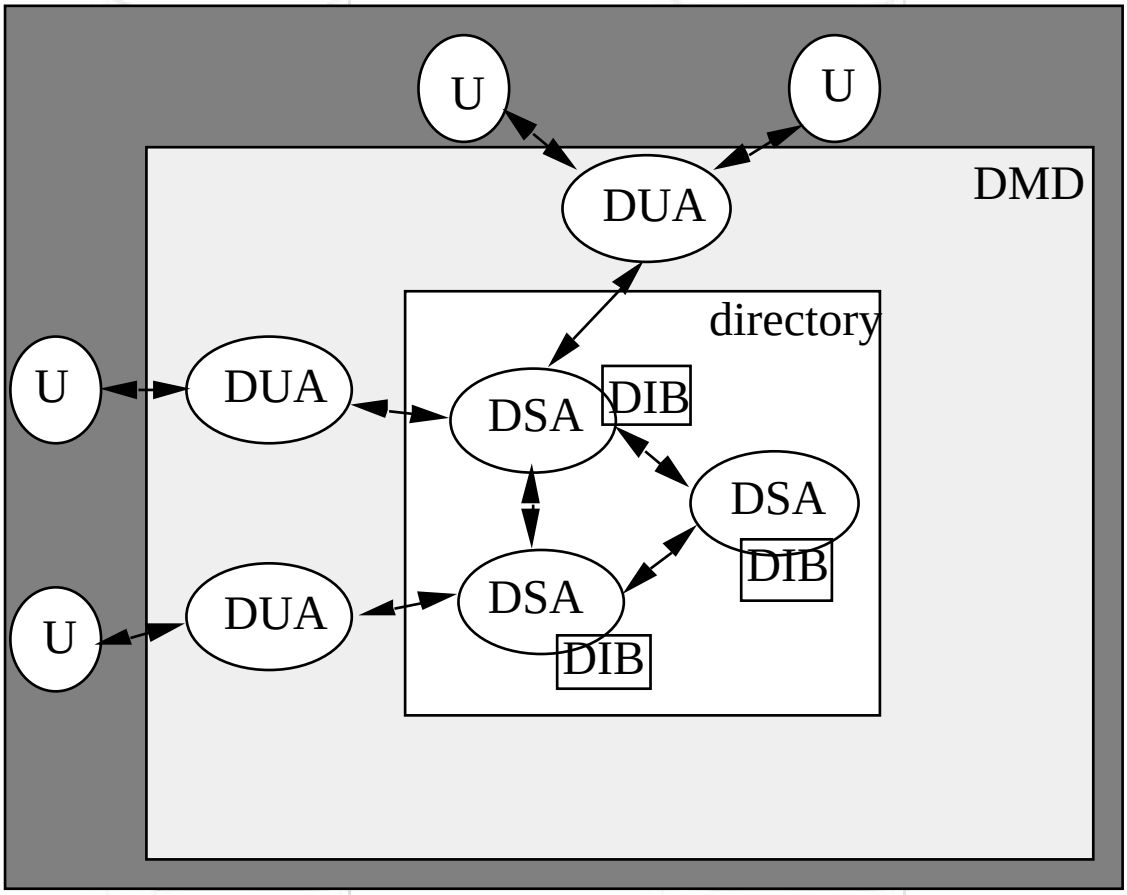
pLorsqu'un appel est réalisé avec l'une des procédures XDS, on contrôle si l'entrée manipulée est une entrée CDS ou GDS. S'il s'agit d'une entrée GDS, il y a transformation en appel XOM.

L'interface XDS est assez simple. Elle contient 13 appels (comparée aux 101 appels de RPC...).

Les huit appels les plus utilisés sont les suivants :

Appel	Description
Add_entry	Ajout d'un objet ou d'un alias dans l'annuaire
Remove_entry	Retrait d'un objet ou d'un alias
List	Liste tous les objets concernant directement un objet
Read	lit les attributs d'un objet spécifié
Modify_entry	Change de manière atomique les attributs d'un objet spécifié
Compare	compare la valeur d'un attribut avec un autre
Modify_rdn	renome un objet
Search	recherche une portion dans l'arbre pour un objet

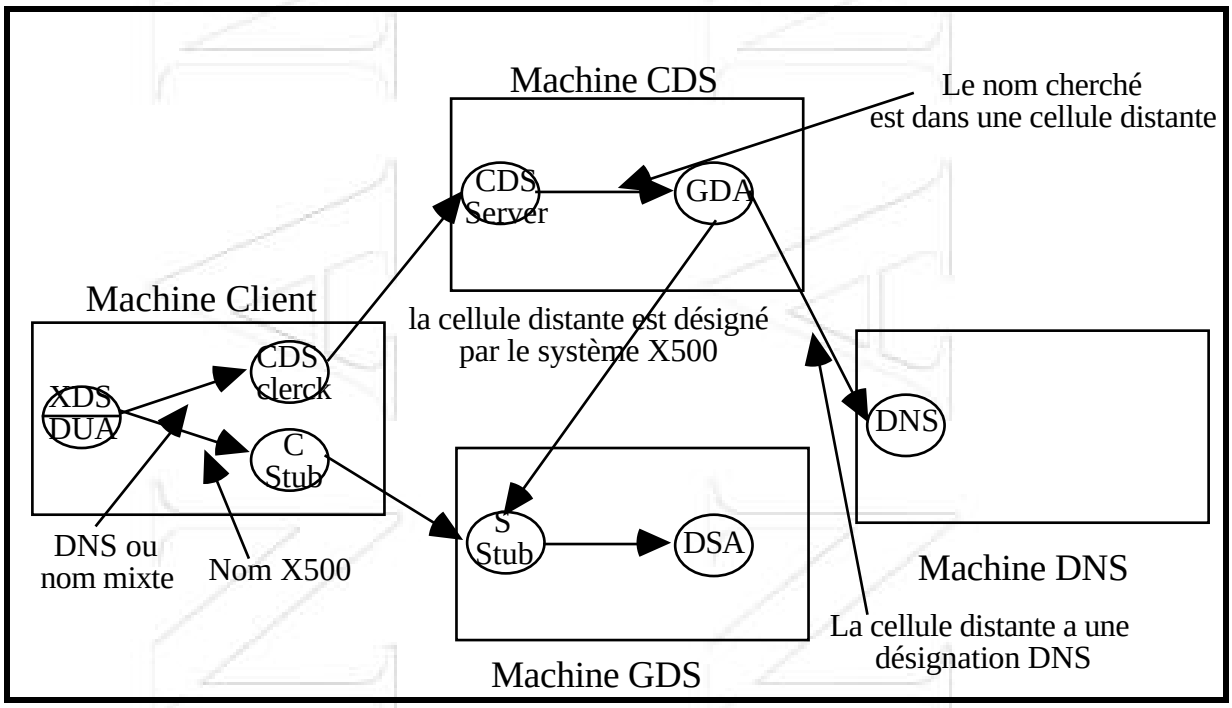
3.4.7 Le modèle fonctionnel du système d'annuaire X500



DUA : Agent utilisateur d'annuaire
DSA: Agent système d'annuaire
DMD Domaine de gestion d'annuaire
DIB : Base d'information d'annuaire

3.4.8.Correspondance entre nom et adresse réelle dans le système complet

p Pour effectuer la correspondance entre nom et adresse réelle, la librairie XDS appelle un DUA, une librairie liée au code du client. Ceci gère les appels à GDS de la même manière que pour le clerck de CDS.



En conclusion : la recherche dans les annuaires est très complexe !

3.5 Le service de sécurité

Dans un système centralisé, le problème de la sécurité est réglé à l'aide de protection des comptes, des fichiers,... géré par l'administrateur système.

Dans un environnement réparti, il faut déterminer les ressources partagées, les utilisateurs autorisés, mais aussi tenir compte de la traversée d'un réseau non sécurisé.

- Le vocabulaire :

  Un principal : un utilisateur, un processus,(humain, serveur, application)... qui a besoin de communiquer de manière sécurisée.

  Chaque principal est défini par un UUID avec un nombre binaire associé.

  L'authentification: C'est le processus qui permet de déterminer si un principal est bien celui qu'il prétend être.

  Dans un système centralisé, l'utilisateur s'identifie au moyen de son nom de compte et du mot de passe associé.

  Dans DCE, la procédure d'authentification est différente. On utilise un serveur d'authentification en prenant soin de ne pas faire transiter le mot de passe en clair sur le réseau.

Les autorisations : Lorsqu'un utilisateur a été authentifié, il faut lui donner des droits et des moyens d'accès sur les ressources du système.

Dans DCE, on associe un ACL (Access Control List) à chaque ressource. Cette liste indique quels sont les utilisateurs, les groupes, les organisations qui peuvent accéder à cette ressource et leurs droits (R,W,E,...)

Le grain de protection peut être "grossier" comme la notion de fichier ou plus fin comme la notion d'enregistrement ou d'entrée dans une base de données.

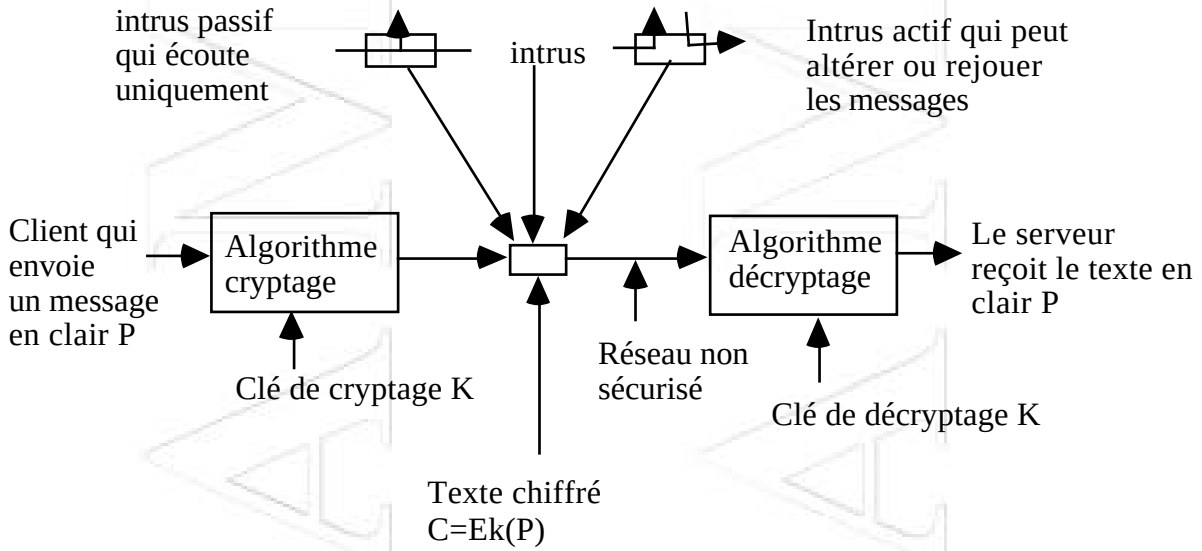
Les limites de la cellule permettent de déterminer un périmètre de sécurité.

Chaque cellule a un service de sécurité avec des principaux locaux à protéger, un service d'authentification avec les clés, les mots de passe et d'autres infos maintenus dans une base de données sécurisée appelée **registry**.

Des cellules de différentes organisations peuvent s'échanger des informations sécurisées selon un protocole plus complexe qui demande le partage entre cellules de clés.

3.5.1 Le modèle de sécurité

- Principe de base de la cryptographie:



- Quelques hypothèses sur le système

  - le réseau n'est pas sécurisé

  - un intrus peut capturer un message et le rejouer ou le transformer

  - le serveur de sécurité est modérément sécurisé. Il est dans une pièce fermée à clé et pour y accéder il faut utiliser Kerberos.

  - Le serveur de sécurité possède les mots de passe des utilisateurs, même s'ils ne circulent pas sur le réseau.

  - un utilisateur n'oublie pas son mot de passe et ne quitte pas une station sans se déloger.

  - On assure que les horloges des différentes machines du système sont synchronisées.

- Quelques attendus du système à mettre en place:

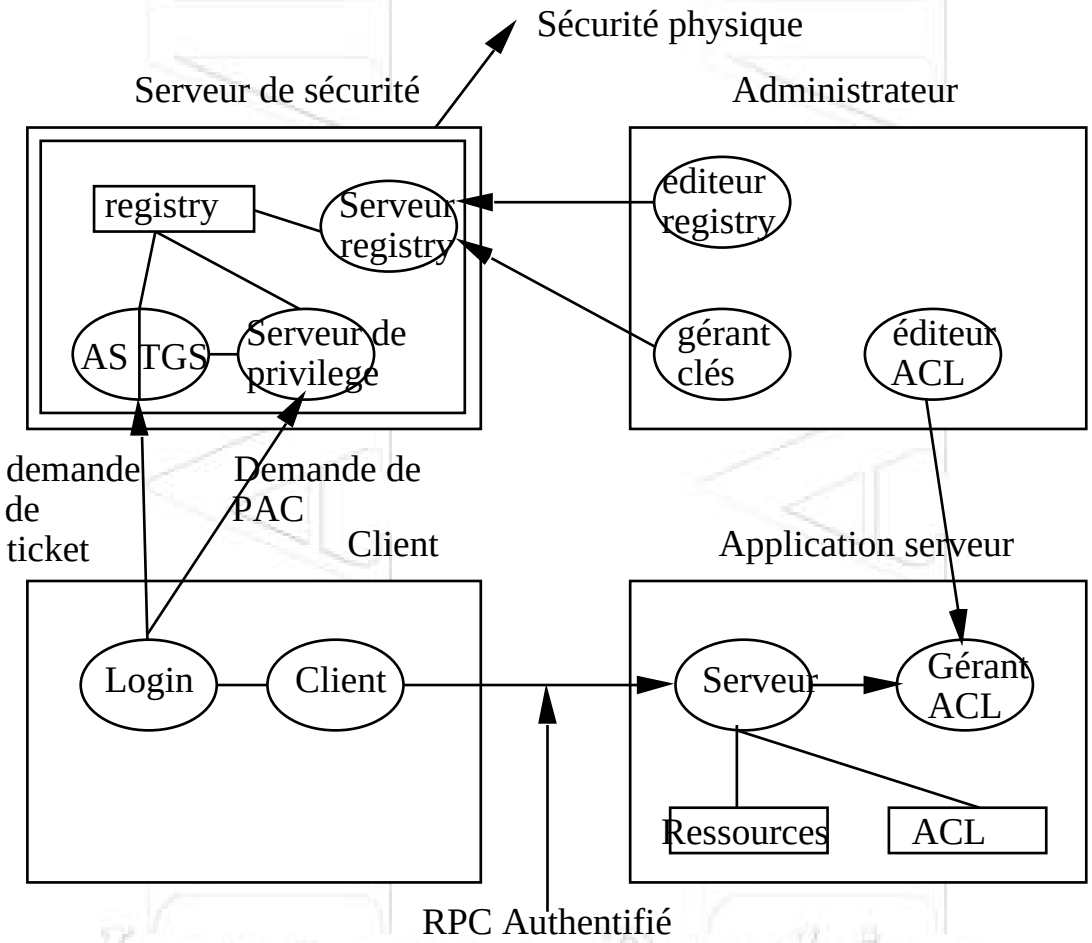
  - les mots de passe ne circulent jamais en clair sur le réseau et ne sont jamais gardés sur des serveurs normaux.

  - Les mots de passe des utilisateurs ne peuvent pas être gardés plus de quelques microsecondes sur les machines clientes => pour être exposé le moins possible à un "dump" de la mémoire.

  - L'authentification doit fonctionner dans les deux sens : client -serveur, mais aussi serveur- client.

  - Le système doit avoir des "portes coupe-feu". Si une clé a été dérobée, les dommages doivent être limités. On utilisera alors des clés de session, des clés à temps limité,....

3.5.2. Les composants de sécurité



- **Le serveur registry (d'enregistrement) :** gère la base de données se sécurité (registry)
- La base de données de sécurité : contient le nom de tous les principaux, des groupes, des organisations.
- Pour chaque principal, on a des informations concernant son appartenance à un groupe, une organisation, si c'est un client ou un serveur et d'autres infos.
- La base de données contient aussi des informations de gestion de la cellule, incluant la longueur, le format et le temps de vie des mots de passe. Elle peut être vue comme un successeur du fichier /etc/passwd d'Unix.
- Est éditée par un éditeur utilisé uniquement par l'administrateur du système.
- **Le serveur d'authentification:** est utilisé quand un utilisateur se loge ou lorsqu'un serveur démarre.
- permet la vérification de l'identité donnée du principal et donne une sorte de ticket qui permet au principal de s'authentifier ensuite sans avoir à donner à nouveau son mot de passe.
- Il est aussi connu comme serveur de ticket de permission (ticket granted server) lorsqu'il authentifie des serveurs.
- **Le serveur de privilège :** donne des documents appelés PAC (Certificat d'attribut de privilège) pour authentifier des utilisateurs. Ce sont des messages cryptés qui contiennent l'identité du principal, son appartenance à un groupe, et à une organisation dans le but de convaincre instantanément les serveurs.

=> Ces trois serveurs fonctionnent sur la machine du serveur de sécurité avec le système Kerberos pour garder le tout !

- **Le login** est un programme qui demande à l'utilisateur son nom et son mot de passe.

  Il utilise les serveurs de privilège et d'authentification

  Quand un utilisateur est logé, il peut démarrer un processus client qui peut communiquer de manière sécurisée avec un processus serveur à l'aide d'un RPC authentifié.

  Quand une requête RPC authentifiée arrive sur le serveur, celui-ci utilise le PAC pour déterminer l'identité de l'utilisateur et vérifier par l'ACL si celui-ci a les droits requis.

- Chaque serveur a son propre **gestionnaire d'ACL** gardant ses propres objets.

  Les utilisateurs peuvent être ajouter ou retirer d'une ACL en utilisant un éditeur d'ACL (géré par l'administrateur système).

3.5.3 Tickets et authentifications

=> Comment fonctionner de manière sécurisée dans un univers non sécurisé :

- Chaque utilisateur a une **clé secrète** connue uniquement de lui seul et du "registry".

  - Cette clé est fabriquée à partir du mot de passe et d'une fonction à un seuls sens (one-way)(non réversible)

  - Les serveurs ont aussi des clés secrètes

  - Ces clés sont utilisées uniquement lorsqu'un utilisateur se loge ou lorsqu'un serveur démarre. Ensuite on utilise les tickets et PACs.

- **Un ticket :** est une structure de données cryptée venant du serveur d'authentification ou du serveur de ticket pour prouver à un serveur spécifique que le porteur est bien un client avec une identité propre.

  Ticket=S,{clé_de_session, client, Délai_expiration, identificateur_du_message}K_S

  S= serveur

  {}K_S = donnée crypté avec la clé privée K_S du serveur

  Clé_de_session = correspond à la clé qui sera utilisé entre le client et le serveur dans la suite des échanges

  - Dans quelques situations ticket et PAC sont utilisés ensemble

- **Un authentificateur :** est une donnée structurée contenant les informations suivantes :

  Authentificateur = { émetteur, Somme de contrôle MD5, vecteur d'estampille}K

  MD5 (Message Disgest 5) est algorithme de contrôle qui a la propriété que étant donné une somme de contrôle MD5 sur 128 bits, il n'est pas possible de modifier le message.

  Le vecteur d'estampille est nécessaire pour que le récepteur puisse détecter que le message est rejoué avec un vieil authentificateur.

3.5.4 Le RPC Authentifié

- La séquence du login jusqu'au premier RPC authentifié demande 5 étapes.
- Chaque étape est composée d'un message d'un client vers un serveur suivi de sa réponse

Principals :

A : Serveur d'authentification (gérant l'authentification)

C : Client (utilisateur)

P : Serveur de privilège (dont viennent les PAC)

S : Serveur de l'application (qui fait le "vrai" travail)

T : Serveur de distribution de ticket (dont viennent les tickets)

Étape 1 : Le client souhaite obtenir un ticket pour le serveur de ticket.
C ->A : C (le nom du client en clair)
C<-A : {K1,{K1,C}KA}KC

Étape 2 : Le client utilise le ticket précédent pour avoir un ticket pour le serveur de privilège:
C->T : {K1,C}KA,{C, somme de contrôle MD_5, vecteur d'estampille}K1
C<-T : {K2, {C,K2}Kp}K1

Étape 3 : Le client demande au serveur de privilège pour le PAC initial
C->P : {C,K2}Kp,{C, somme de contrôle MD_5, vecteur d'estampille}K2
C<-P : {K3, {PAC,K3}KA}K2

Étape 4 : Le client demande au serveur de ticket un PAC utilisable par S
C->T : {K3,{PAC,K3}KA}K3,{C,somme de contrôle MD_5, vecteur d'estampille}K3
C<-T : {K4,{PAC,K4}Ks}K3

Étape 5 : Le client établit une clé avec le serveur d'application
C->S:{PAC, K4}Ks, {C,somme de contrôle MD_5, vecteur d'estampille}K4
C<-S:{K5,Vecteur d'estampille}K4

3.5.5 ACL : Liste de contrôle d'accès

- Toute ressource peut être protégée par une liste de contrôle d'accès
- Une liste de contrôle d'accès indique qui a le droit d'accéder à la ressource et comment il peut le faire.
- Les ACLs sont gérés par des gestionnaires d'ACLs qui sont des procédures ajoutées à chaque serveur.
- Quand une requête arrive sur un serveur contrôlé par un ACL, il décrypte le PAC du client pour examiner son identificateur et son groupe, l'opération souhaitée.
- A partir de ces informations le gestionnaire d'ACL prend une décision concernant l'accord ou non de l'accès au serveur.
- La plupart des serveurs utilise un gestionnaire standard d'ACL.
- Il divise en deux les ressources :

  - ressources simples : fichiers, enregistrement de base de données

  - les contenants : les répertoires, les tables des bases de données qui contiennent des ressources simples.

- Il distingue les utilisateurs

  - les utilisateurs locaux, ceux qui vivent dans la cellule,

  - les étrangers

- Chaque type d'utilisateur peut être

  - propriétaire,

  - appartenant au groupe

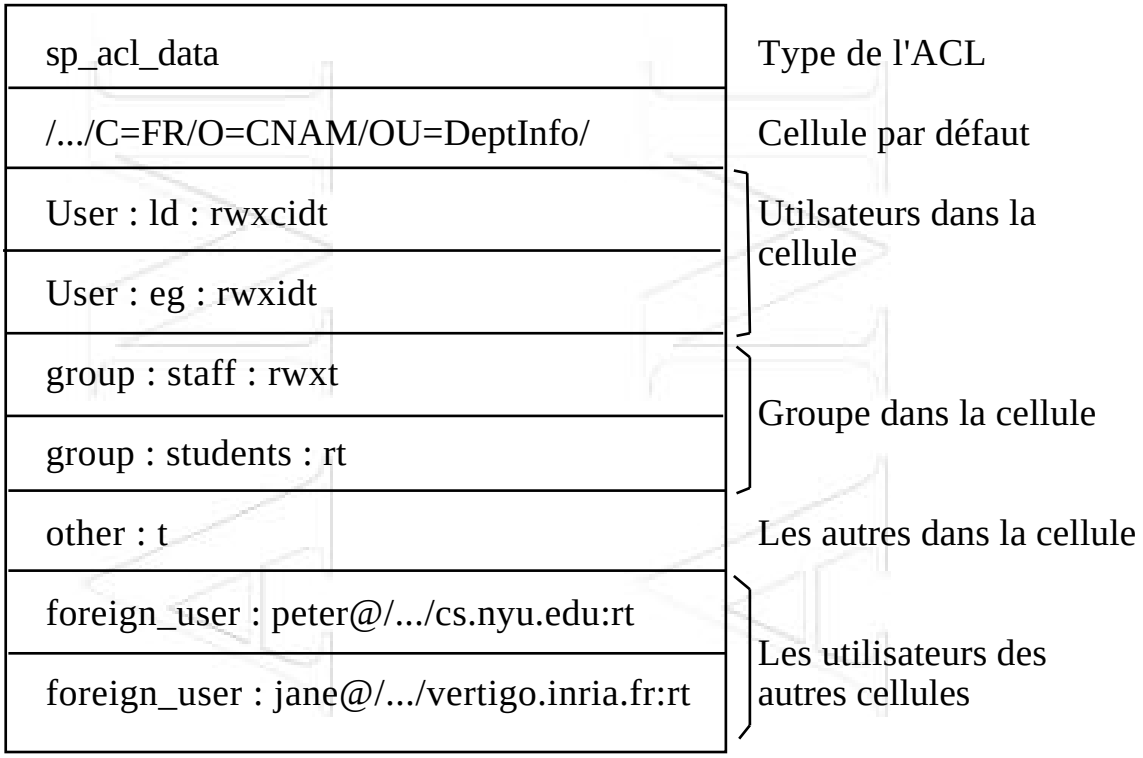
  - les autres.

Par exemple on peut donner tous les droits au propriétaire, les membres locaux du groupe peuvent lire et écrire et les extérieurs uniquement lire.

- Les droits :

  - lire, écrire, exécuter, changer l'ACL, insérer un contenant, effacer un contenant, et tester.

Exemple d'ACL



3.6 Le service de fichiers distribués (DFS)

=> Système de fichiers distribué global

=> Permet l'accès à des fichiers locaux, mais aussi à des fichiers situés n'importe où dans le monde dans une cellule DCE.

=> contient deux parties :

- L'environnement local : appelé **Episode** analogue au système de fichier standard d'Unix.

- L'environnement global : il s'agit d'un ajout qui permet à tous les systèmes de fichiers individuels de former un système global recouvrant un ensemble de cellule. Ce système est dérivé du système AFS de CMU.

=> Les avantages :

- haute disponibilité des données et des ressources

- possibilité de partager des information à travers un très grand système.

- un nommage uniforme intégré avec CDS - les noms de fichiers sont indépendant de leur localisation.

- les administrateurs peuvent bouger les fichiers d'un serveur sur un autre dans une même cellule sans changement dans le programme utilisateur.

- les fichiers peuvent être dupliqués pour améliorer la charge et maintenir la disponibilité de l'information lorsqu'un serveur tombe en panne.

- Il existe des possibilités de distribution automatique de nouvelles versions de programmes binaires et d'autres fichiers utilisés en lecture vers un groupe de serveurs.

DFS est un exemple d'application distribuée:

C'est une application client/serveur qui utilise les threads, le RPC, DTS, le service d'annuaire, les serveurs d'authentification et de privilége.

Episode est une réécriture du système de fichier UNIX et peut le remplacer sur n'importe quelle machine DCE avec Disque.

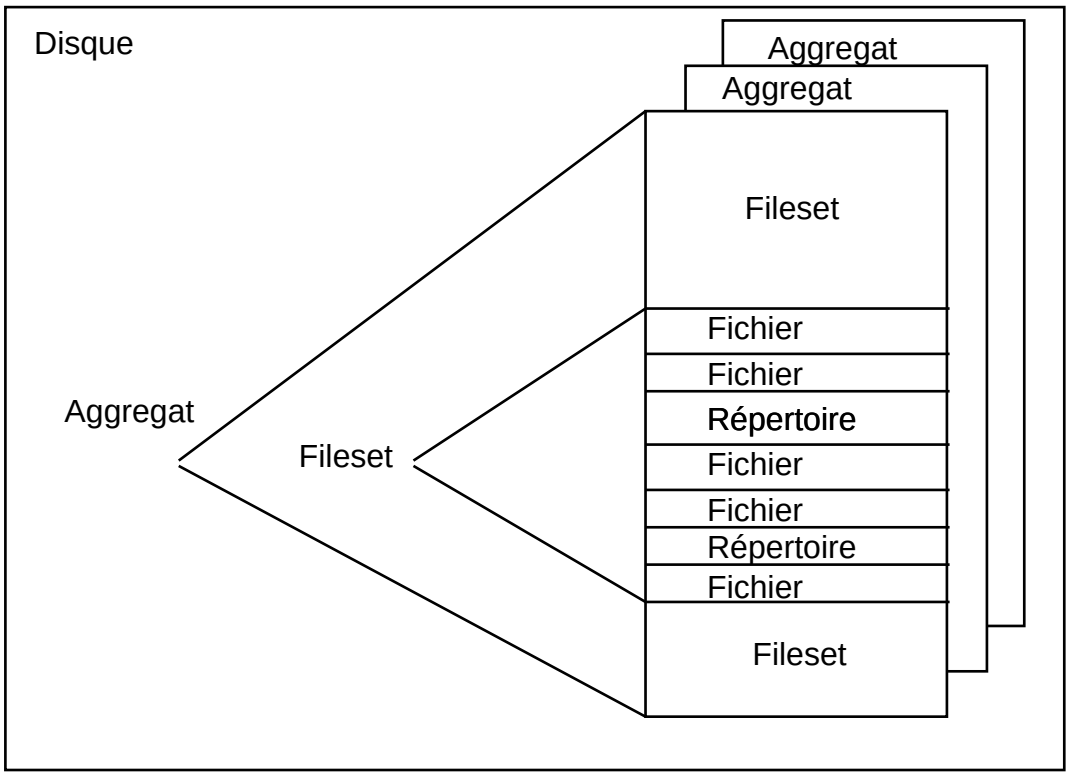
=> Il supporte une sémantique de fichier tel qu'une écriture faite dans un fichier, suivie immédiatement d'une lecture, donne à la lecture la valeur qui vient d'être écrite.

=> Supporte le contrôle d'accès avec ACLs

=> Supporte également un recouvrement rapide après crash (sans programme fsck d'Unix)

=> De plus peut être intégré à côté de systèmes existants (Unix BSD 4.3, System V, NFS, ...)

3.6.1 L'organisation des données



Fichiers et répertoires :

Un fichier est l'unité de données d'un utilisateur.

Les répertoires organisent les fichiers sous forme de structure d'arbre hiérarchique.

Fileset :

c'est une unité d'administration. Il s'agit d'un sous-arbre de fichiers et de répertoires, pas plus grands qu'un disque ou une partition on un volume logique.

Le fileset est un regroupement de fichiers dans un but d'administration. (exemple : le sous-arbre des fichiers d'un projet ou département particulier)

Aggregat :

C'est l'unité de stockage correspondant à une partition disque. Il contient un ou plusieurs filesets.

3.6.2 L'interface DFS

Très similaire à Unix

Les fichiers peuvent être lus, écrits, ouverts,...

La plupart des logiciels existants peuvent être recompilés avec les librairies DFS sans grand changement.

Le montage de fichier distant est aussi disponible.

Une ACL est associé aux fichiers et aux répertoires.

Le catalogue / correspond à la racine locale et les directories /bin, /lib/usr se référent aussi aux directories binaire, librairie et utilisateur.

Une nouvelle entrée est /… dans la directory racine correspond à la racine globale.

Chaque fichier dans DFS a un nom unique à partir de la racine globale avec pour point d'entrée le nom de la cellule, puis le chemin local.

Exemple:

un nom de fichier global (format Internet)

/.../deptinfo.cnam.fr/fs/usr/ann/exams/janvier

un nom de fichier global au format X500

/.../C=FR/O=CNAM/OU=DeptInfo/fs/usr/ann/exams/janvier

un nom de fichier global (par rapport à la cellule)

/.:/fs/usr/ann/exams/janvier

un nom de fichier global (par rapport au système de fichier)

/:/fs/usr/ann/exams/janvier

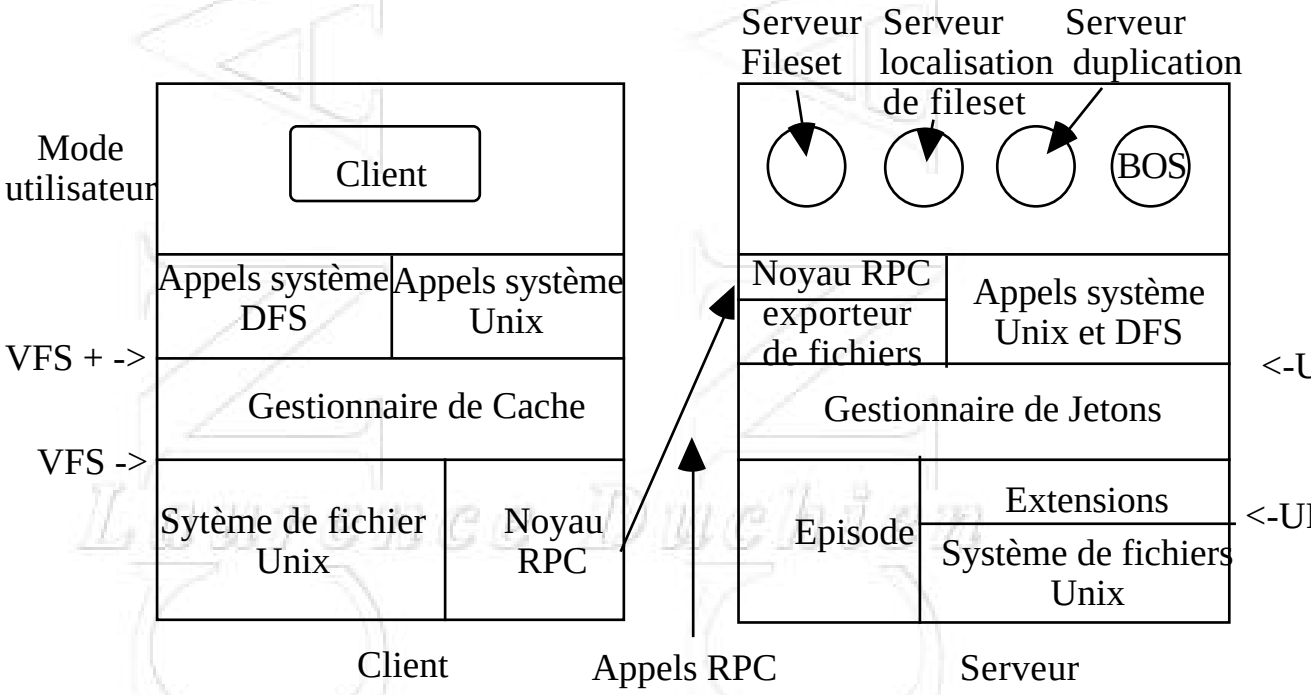
3.6.3 Les composants de DFS

On décompose les composants en 3 groupes :

- Le client avec le gestionnaire de cache

- Le serveur avec l'exporteur de fichier et, le LFS et le gestionnaire de jeton.

- Le serveur administratif avec le serveur de fileset, le serveur d'observation (BOS), le serveur de duplication, le serveur de mise à jour, le serveur de reprise, le serveur de localisation des fileset.



3.6.3.1 Les composants des clients DFS

Le **gestionnaire de cache** correspond au côté client de DFS qui fonctionne sur n'importe quelle machine agissant comme un client DFS

Il traite les requêtes du lient vers le système de fichier et regarde dans un cache si une copie du fichier n'est pas déjà présente.

Si ce n'est pas le cas, la requête est envoyée au serveur de fichier.

=> le trafic du réseau diminue

3.6.3.2 Les composants des serveurs DFS

- **L'exporter de fichier** (file exporter) correspond au côté serveur de DFS. Il fonctionne sur un serveur de fichier qui gère les requêtes des clients.

  Il reçoit les appels RPC et accède à son propre système de fichier local (soit DCE soit un autre système de fichiers Unix par exemple).

  En utilisant le système de gestion de jeton, il gère la synchronisation des différents clients qui demandent des accès concurrents sur le même fichier.

  Il retourne l'information au client.

- **Le système de fichier local** DCE: C'est le système de fichier physique fourni par DCE. Il gère le stockage des fichiers dur le disque. Il est analogue au système de fichier UNIX en plus puissant.

  Il gère les possibilités d'accès par l'ACL.

  Il gère la possibilité de duplication, de sauvegarde et d'autres éléments de service dans interruption de service.

  Il gère un recouvrement rapide après un crash, en laissant la possibilité de se loger.

  Possibilité d'utiliser le système de fichier d'Unix à la place de DCE avec certaines restrictions (1 fileset par système de gestion de fichier, pas de duplication)

- **Le gestionnaire de jeton :** synchronise les accès aux fichiers pour plusieurs clients simultanés.

  Il distribue des jetons qui représente la possibilité d'exécuter des opérations.

  Un jeton définit différents droits d'accès tel que l'écriture ou la lecture

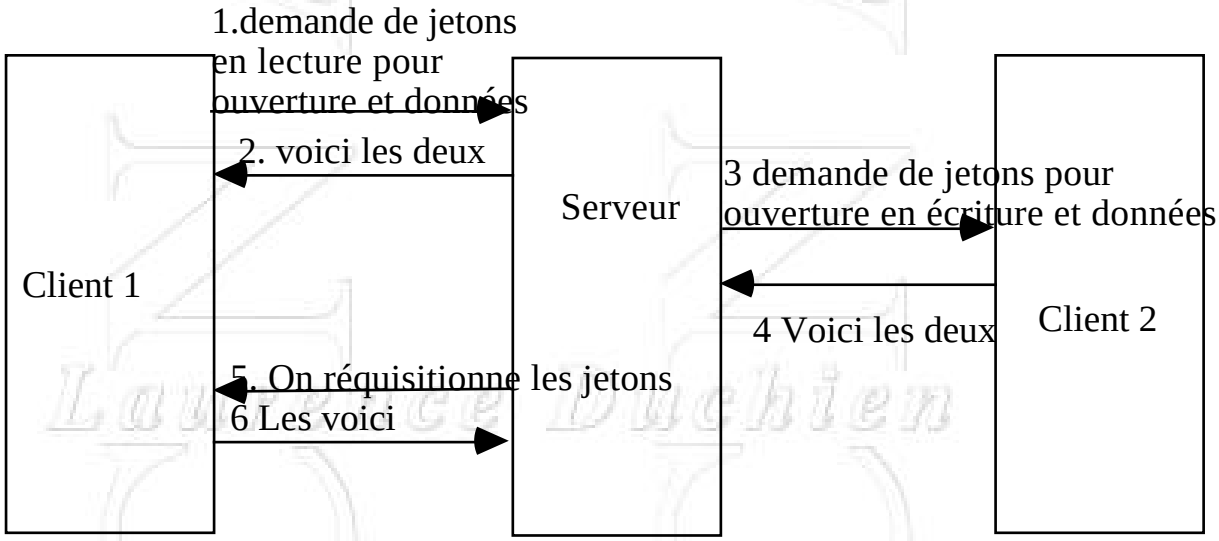
  Il y a 4 types de jetons :

  - jeton de données pour accéder à un fichier ou à une directory

  - jeton d'état : pour à l'état d'un fichier ou d'un répertoire

  - jeton verrou : pour verrouiller une partie de fichier

  - jeton d'ouverture : pour ouvrir un fichier



3.6.3.3 Les composants d'administration DFS

- **Le serveur de Fileset :** permet aux administrateurs de créer, effacer, déplacer et exécuter des opérations sur les filesets.

  Exemple : un administrateur peut utiliser le serveur de fileset pour déplacer un fichier d'un serveur de fichier vers un autres serveur de fichiers pour gérer la charge.

- **Le serveur d'observation (Basic OverSeer Server)** gère les processus DFS qui fonctionne sur un machine et les redémarre au besoin.

  Il maintient l'information concernant ces processus et répond aux requêtes d'administration.

- **Le serveur de duplication :** Gère la duplication de filesets.

  Exemple : un administrateur peut créer une copie d'un fileset sur une seconde machine serveur de fichier.

  Le serveur de duplication devra mettre à jour la copie à intervalles réguliers (toutes les 30 mn par exemple).

  Ce qui veut dire que même si le serveur maître tombe, une copie du fichier reste disponible.

- **Le serveur de mise à jour :** fournit la possibilité de distribuer des fichiers binaires ou de l'information d'administration aux noeuds dans le système DFS.

  Il est constitué de deux parties :

  - upclient : tourne sur une machine qui a besoin de recevoir de nouvelles versions de fichiers binaires.

- **upserver** : tourne sur une machine maître et propage automatiquement les changements des binaires vers les machines tournant le upclient.

- **Scout** (éclaireur) : Outil administratif de collecte et d'affichage d'informations sur les exporteurs de fichiers fonctionnant sur la machine du serveur de fichier.
- **Le serveur de localisation de fileset :** C'est une service d'annuaire dupliqué qui garde une trace des filesets que l'on peut trouver et sur quelle machine.

  Il fournit un service analogue au service CDS, mais spécialisé pour le DFS.

  Il supporte la gestion de la transparence de localisation.

  On accède simplement à un fileset en connaissant son nom sans avoir à connaître sa localisation.

3.7 La gestion de station sans disque

DCE permet à des stations sans disque de participer à l'environnement distribué.

Une station sans disque ne peut pas stocker un système de fichiers local.

Un support de stations sans disque veut dire qu'il faut fournir des fonctionnalités à travers le réseau pour remplacer le disque local.

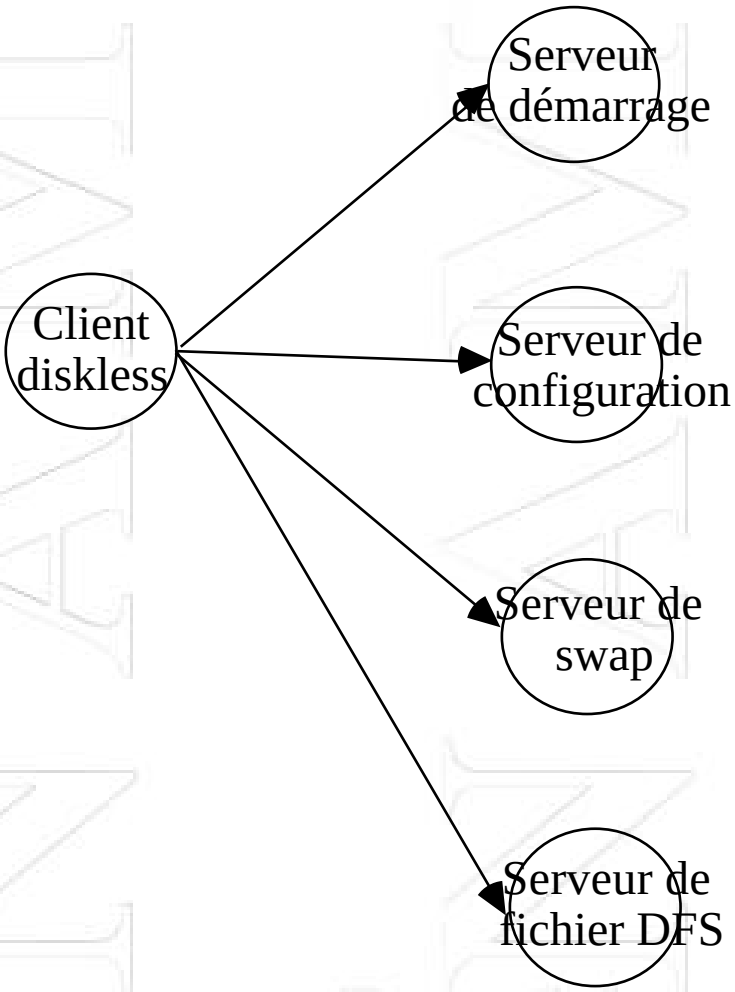
4 points doivent être couverts :

- Le démarrage du système
- L'obtention d'informations de configuration
- Le support d'un système de fichiers distants
- Le support d'un système de swap distant.

Dans un système centralisé ou traditionnel, une machine cherche au démarrage une image binaire de son système d'exploitation dans un fichier "bien connu" sur son disque local (vmunix par exemple).

Avec une station sans disque, il faut réussir à obtenir une copie du système à travers le réseau.

3.7.1 Les services proposés par DS



le démarrage

- le client Diskless (clients BOOTP et TFTP) et
- Le serveur de démarrage (bootpd et tftpd)

bootp et tftp correspondent à un code résidant dans la station sans disque dans une ROM. Correspond au seul stickage permanent de la station.

Pendant la phase de démarrage, la station utilise le code de bootp pour obtenir sa propre adresse et l'adresse du serveur de démarrage; elle utilise ensuite le code tftp pour obtenir le système.

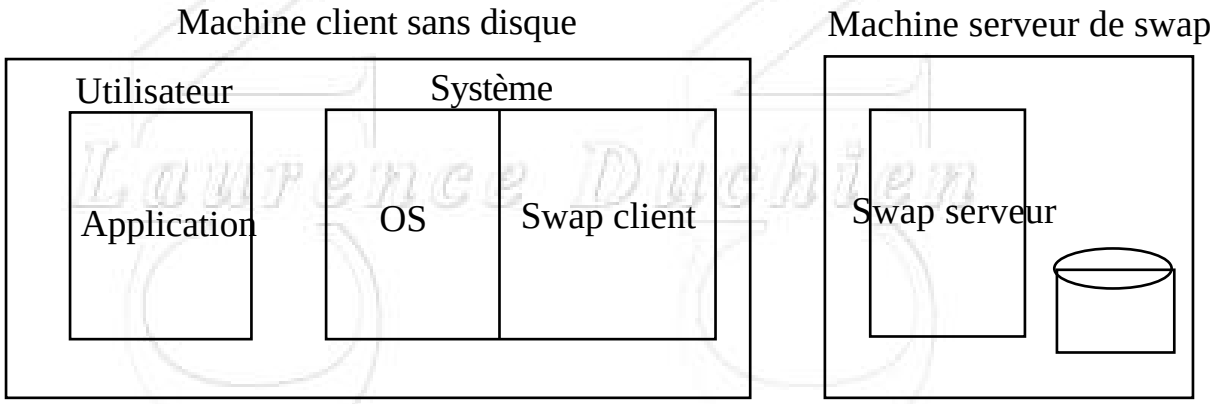
la configuration :

le serveur de configuraion est le lieu de stockage des informations de configuration nécessaire à la mise en place du système sur la station. (exemple d'informations : le server de swap du client, la localisation du système de fichier et de son serveur, les paramétres de configuration du gestionnaire de cache de DFS)

Le chargement et déchargement de la mémoire (swap)

Il permet de gérer le chargement et le déchargement de la mémoire sur disque. Il se fait par trois éléments (un client, un serveur et un administrateur).

Le serveur gère une base d'information à propos des clients, des fichiers swapés et accepte les requêtes de l'administrateur.



Le système de fichiers

On utilise un système de fichier distant basé sur DFS. Plusieurs aspects sont mis en jeu :

- Le gestionnaire de cache de la sation sans disque. Avec une machine avec disque, on utilise un système de cache stocké sur le disque local. Avec ce type de machine on utilise un morceau de la mémoire.

- Les fichiers spécifiques de la machine : DCE pouvant fonctionner sur des plate-formes hétérogènes, il faut être capable de distinguer le système de fichier nécessaire.

Le mécanisme de fichier spécifique à la machine est constitué de deux fichiers :

- @host : qui est remplacé par le nom de l'hôte du client (equivalent de /etc/rc)

- @sys : remplacé par le nom du système du client. Une station pouvant avoir plusieurs fichers executables pour différentes plates-formes, on remplace @sys par le nom de la plate-forme spécifique que l'on veut/

- Les fichiers de périphérique

/dev/* correspond des périphériques. Ils correspondent à des périphériques de la machine client

4. Intégration des composants DCE

Quels composants DCE sont utilisés par les autres composants DCE ?

- colonne de gauche : les composants consommateurs de technologie
- ligne du haut : les composants producteurs de technologie
- X utilise le composant yyy pour la technologie zzz
- NA non applicable

	Thread	RPC	CDS	DTS	Sécur	GDS	DFS	Ss Dk
Thread	NA							
RPC	X	NA	X		X			
CDS	X	X	NA	X	X	X		
DTS	X	X	X	NA	X			
Secur.	X	X	X	X	NA			
GDS						NA		
DFS	X	X	X	X	X		NA	
Ss Dk	X	X	X				X	NA

5. Conclusion

- La disponibilité des composants
- L'administration
- la documentation
- Les performances
- Un support de programmation objet
- Un support des applications transactionnelles
- La compatibilité avec les normes
- L'internationalisation

Bibliographie

L'Internet :

http://web1.osf.org:8001/dce/faq_mauney.html

http://www.osf.org/dce/dce-rfc/

gopher ://info.pilgrim.umass.edu/77/lib/.wais-sources/osf-dce-rfc.txt

Les bouquins :

OSF/DCE, Introduction to OSF/DCE, Prentice Hall.

OSF/DCE, Administration reference, Prentice Hall.

OSF/DCE, User's guide and reference, Prentice Hall.

OSF/DCE, Application development reference, Prentice Hall.

OSF/DCE, Application development guide, Prentice Hall.

Rosenbarry, Kenney & fisher, understanding DCE, O'Reuilly

Rapport de stage de P.Blanc, l'Environnement distribué DCE, Ratp, 1993.

A. Tanenbaum, Distributed Operating Systems, Prentice Hall, 1995.

Versions disponibles du domaine public(uniquement RPC):

ftp://gatekeeper.dec.com/archive/.2/DEC/DCE/PD-DCE-RPC.tar.Z

http://www.pilgrim.umass.edu/pub/osf/_dce/contrib/PD-DCE-RPC.tar.Z