Désignation dans les systèmes répartis

C. Carrez, E. Gressier-Soudan

CNAM-CEDRIC (Paris)

D. Donsez

UJF-LIG (Grenoble)

UE SMB111

Principes généraux de la Désignation

Les noms

- nom = information permettant d'identifier un objet
- adresse = information permettant d'identifier un emplacement

  adresse mémoire d'exécution

  adresse mémoire de stockage
- liaison = mécanisme qui associe un nom et un objet

Interprétation d'un nom = mécanisme permettant de passer du nom à l'objet.

On associe souvent désignation et protection, la connaissance d'un nom, c'est la moitié de l'accès. La partie protection est donnée en fin de support mais plus enseignée car elle est vue bien plus en profondeur en RSX112.

La protection

Mécanismes pour empêcher les accès non autorisés aux objets.

- Qui est le demandeur?
- Qu'a-t-il le droit de faire?
- Puis-je faire confiance à l'exécutant?

Système distribué

Changement d'échelle pour le nombre d'objets et d'acteurs

Communication au moyen d'un intermédiaire "douteux", le réseau (ex Internet)

Le nommage en général

Contexte d'interprétation des noms

L'interprétation d'un nom dépend d'un contexte plus ou moins étendu.

Nom local: n'a de sens que dans le contexte d'un processus

Nom global: indépendant de tout contexte

En conséquence, lorsque l'on transmet le nom d'un objet, il faut :

- soit être certain que le contexte est connu du récepteur
- soit transmettre la partie du contexte qui permet de l'interpréter
- soit transmettre un nom global (après transformation)

Les niveaux de noms

On peut distinguer 3 niveaux dans le système de nommage:

- Nom externe (ou symbolique) = chaîne de caractères
- Nom interne = nombre binaire

Gestion des noms assure la mise en correspondance entre les deux niveaux

Nom intermédiaire, lié à un contexte pour masquer l'accès au nom interne

Notion de nom unique

problèmes:

1. deux objets différents doivent avoir des noms globaux différents (pas d'homonymes)
2. quand un objet est détruit, ne pas avoir à rechercher toutes les références à cet objet (pas de synonymes)
3. pouvoir identifier un objet dans tout le système, indépendamment de sa localisation

Idée: un nom est un identificateur attaché à l'objet pendant toute son existence, et qui n'est pas réutilisé pour un autre objet

Information portée par un nom

Nom “pur” = configuration de bit sans signification

autres noms = configuration de bit qui porte une information

Information permettant de déduire en partie la localisation

- déplacements limités de l’objet sans changer son nom

(machine= 12, vol=43, inœud=52) localise complètement le fichier, mais interdit son déplacement

Information sur la structuration des objets

- modifications limitées de la structure sans changer son nom

/A/B/C laisse supposer l’existence d’un répertoire A qui contient un sous répertoire B

Le nommage dans les systèmes distribués

La désignation est plus complexe qu'en centralisé:

- liaisons plus complexes
  - liaison machines vers adresses réseaux
  - liaison serveurs vers machines
- il faut conserver l'aspect dynamique de la liaison
  - migration d'objets et de serveurs
  - duplication d'objets et de serveurs

Interprétation des noms

- Répertoires répartis et dupliqués (robustesse et facilité d'accès)
- Quel répertoire choisir? => accompagner les noms de suggestions de localisation "hints"

Problème de cohérence des noms

Cohérence immédiate des copies multiples de noms

- toujours la valeur la plus récente => accessibilité limitée
- accessibilité maximale => pas forcément la plus récente
  - les noms changent rarement => incohérences rares
  - un nom obsolète a un effet visible
  - si on ne s'aperçoit pas qu'un nom est obsolète, c'est sans importance.
- Attention: pas toujours vérifié!

Cohérence à long terme des copies multiples

- Si on arrête de modifier les données, toutes les copies deviendront identiques.

Facteur d’échelle

- pas de limite sur le nombre de machines ou le nombre d’objets,
- grand nombre d’administrateurs plus ou moins indépendants
- machines et systèmes hétérogènes

Exemple ORB :

IOR (**Interoperability Object Reference**) de OMG CORBA

Exemple d'IOR...

IOR:000000000000000E49444C3A48656C6C6F3A312E30000000000000010000000000000003A000100000000000F3136332E3137332E3133362E3232000013880000000000001A4F422F49442B4E554D0049444C3A48656C6C6F3A312E30003000

Transformation nom externe vers nom interne

Principe général: structure hiérarchique

- structure classique bien connue
- permet la prise en compte d'un grand nombre d'objets
- une structure graphique est possible, mais nécessite un ramasse-miettes réparti

Vision unique, ou vision dépendante de la machine

- le nom /A/B/C désignant un objet O sur une machine, est-il encore valable sur une autre machine pour désigner O
- les systèmes à montage à distance (NFS) sont dépendants

Transparence des noms

- transparence de localisation: le nom ne dit rien sur la localisation de l'objet

  /service/A/B est transparent à la localisation, car le service peut être déplacé
  /machine/A/B ne l'est pas
- indépendance de localisation: l'objet peut être déplacé n'importe où sans changer de nom

serveurs de noms spécialisés:

- assurent uniquement la traduction

  nom externe -> nom interne
- peuvent faire correspondre plusieurs noms internes à un même nom externe

  copies multiples du même objet

Localisation physique à partir du nom global

- **Par diffusion :** le nom global est envoyé à tous les sites, celui qui possède l'objet répond
- **Suivant une heuristique :** utilisation d'une suite de sites particuliers avant de faire une diffusion globale
- **Evaluation itérative :** les noms sont couplés à des hypothèses de localisation
- **Serveurs de localisation :** chaque site connaît au moins un de ces serveurs

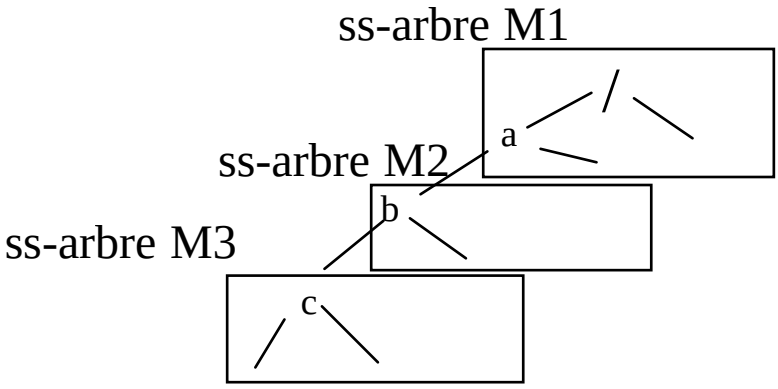
  ils sont en général dupliqués pour permettre la tolérance aux pannes

Note: en général, des caches de localisations évitent de redemander une correspondance nom-localisation de l'objet

Résolutions de noms pour SGFR (1)

1. Transformation du chemin d'accès

Principe : Chaque machine résout le morceau de chemin qui la traverse.



Un client veut accéder au fichier /a/b/c :

a. Client -> M1 qui reçoit tout le chemin

b. sur M1 :

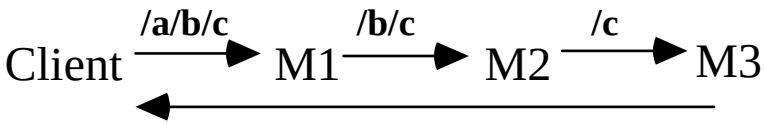
- **/ évalué sur M1** : donne le **nom interne "racine"**, on va sur le disque

- **dans le répertoire racine** on trouve le **nom interne de "a"**, on va sur le disque

- **dans le répertoire "a"**, on trouve une indication telle **"b dans ss-arbre sur M2"**, on passe **/b/c à M2**

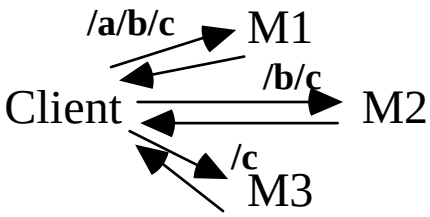
c. sur M2 : le reste du chemin **/b/c/myfile est évalué** de la même façon, **/c est passé à M3**

d. sur M3 la **fin du nom est résolu** et retour du résultat au client



Autre variante de la transformation de chemin d'accès :

. Résolution par chaque serveur et retour au client après chaque étape (image du polling).



L'effort de résolution est supporté surtout par le client.

Solution adoptée par NFS.

Résolutions de noms pour SGFR (2)

2. **Méthode des identificateurs structurés**

Pour réaliser la correspondance entre le nom d'un fichier et sa localisation on utilise un identificateur structuré, celui-ci identifie une sous-arborescence qui contient le fichier cible.

identificateur structuré (is):

< identif de ss-arbre ; identif de fichier dans le ss-arbre>

Une table contient sur chaque site la correspondance :

nom symbolique <-> id ss-arb

La table peut être gérée de différentes façons :

-> remplissage suivant le principe de résolution par transformation de chemin d'accès

-> gérée sur chaque site à l'image d'un cache ou accédée sur un serveur de noms

Exemple :

/a/b/c	<ss-arbre 3; 11>

ss-arbre 3	M3

Le nom est indépendant de la localisation, quand le fichier migre il suffit de mettre à jour la table.

Résolutions de noms pour SGFR (3)

3. **Méthode des caches de suggestion ("hints")**

. intervient pour la localisation

. améliore la performance si le cache contient une information non périmée (valide)

. si l'information est invalide, on peut utiliser les techniques précédentes pour la recharger.

Solutions utilisées lors d'une résolution de chemin d'accès:

- accès cache de noms côté client sinon requête au serveur de fichiers (ANDREW)
- accès cache de noms (partie préfixe du chemin) sinon diffusion de la demande (SPRITE)
- accès cache de noms sinon emploi d'une heuristique (APOLLO-DOMAIN)
- accès cache de noms sinon interrogation d'un serveur de noms (GRAPEVINE)

Résolutions de noms pour SGFR (4)

4. **Points d'attachement**

Un point d'attachement est l'association d'une sous-arborescence à une feuille d'une autre sous arborescence.

Une table mémorise l'association

<feuille ; sous-arbre>

On parcourt la table à chaque fois qu'un chemin d'accès traverse un point d'attachement.

Résolution de noms dans l'Internet

Plusieurs types de noms à résoudre :

- adresse physique d'une machine
- adresse réseau d'une machine
- nom d'une machine
- adresse d'un service
- nom d'un service
- un utilisateur de mail
- une ressource WEB -> URL : Uniform Resource Locator

Solutions basées sur :

- Serveurs de noms : Domain Name System (DNS), NIS, Hesiod, LDAP

- Convention de désignation

Les Yellow Pages - NIS

Principes :

- Les “Yellow Pages” forment une base de données dupliquée qui fait correspondre une “map” à un fichier[1] :

Fichier des utilisateurs	/etc/passwd
Fichier des sites connectés	/etc/hosts
Fichier des services	/etc/services
.....	

Elles permettent d’avoir une définition unique de certains paramètres de configuration du réseau de machines. Un réseau de machines qui utilisent la même copie de la base YP, définit un domaine d’administration YP.

[1] Il est possible, pour certaines “map”, de consulter le fichier local avant de faire une requête à un serveur YP. C’est le cas, pour le fichier des utilisateurs.

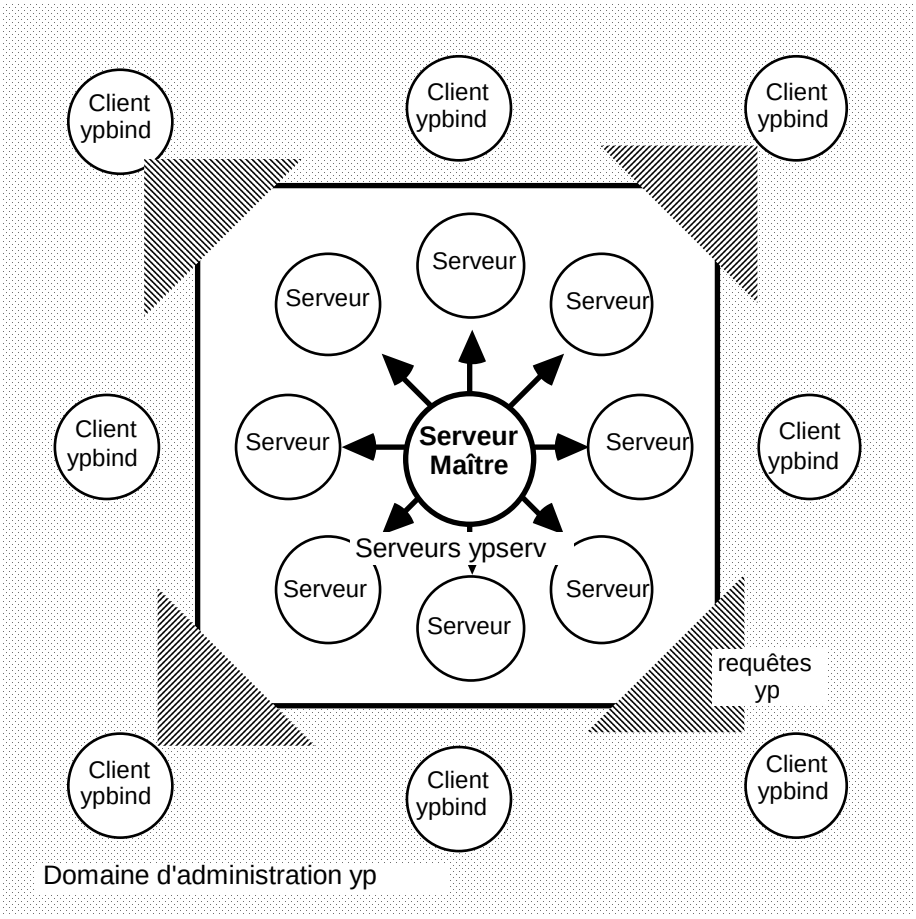
- Deux types de machines :

Les serveurs qui possèdent une copie de la base de données (ypserv) :

. Le serveur maître sur lesquelles sont effectuées les mises à jour et qui effectue la propagation de ces mises à jour

. Les serveurs esclaves => résistance aux pannes, récupération des mises à jours

Les clients qui interrogent un serveur pour accéder aux informations de la base de données (ypbind), pour connaître le serveur associé, utiliser la commande ypwhich.



- Les YP sont construites au-dessus de RPC/XDR

- Cohérence faible des données entre serveur maître et serveurs esclaves, la mise à jour d’un mot de passe par exemple n’est prise en compte sur tous les serveurs qu’au moment de la prochaine mise à jour des "map" sur les serveurs esclaves.

- Lors de la panne d’un serveur, un client joint un nouveau serveur par une requête de diffusion, le premier qui a répondu devient son nouveau serveur.

Quand le serveur maître tombe en panne, plus aucune mise à jour n’est possible....

- Fonctionne bien entre machines homogènes, mal entre machines hétérogènes:

“tables” différentes, protocoles différents ???

Exemple de résolution de noms par substitution

Le cas de Chorus

ChorusOs a été choisi car il met complètement en œuvre ce mécanisme de substitution de noms. Pour simplifier, on peut voir ChorusOS comme un système d'exploitation dont la vocation particulière est d'être une plateforme pour faire des systèmes distribués.

UID

Noms Internes

Noms Uniques et Globaux : **UID**

- Unicité dans le temps : date + nombre aléatoire

- Unicité dans l'espace : contient le nom d'un site (peut être le site de création de l'objet), parfois son type

permet de référencer les objets élémentaires **indépendamment de leur localisation**, (certains objets peuvent migrer)

Capacités

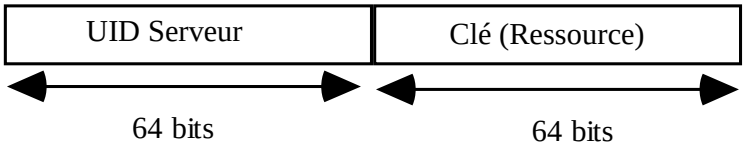
Noms intermédiaires d'une ressource/d'un objet d'un sous-système:

- connus hors du noyau

- gérés par un serveur

- n'ont de sens que dans le contexte du serveur associé à l'objet désigné par une capacité

- liés aux droits d'accès à l'objet désigné



Les Capacités sont composées de :

- nom interne du serveur gérant l'objet

- clef qui permet d'accéder à la ressource chez le serveur

- protection (droits d'accès)

concept principalement issu d'AMOEBA

Noms contextuels ou Identificateur Local

Nom qui n'a de sens que par rapport au contexte d'une entité système.

Equivalent au n°de fichier ouvert pour un processus dans Unix.

Exemple :

N° de Thread au sein d'un acteur CHORUS

Noms - API Chorus

uiBuild/uiCreate : création d'un nom unique pour un utilisateur

Pour uiBuild, l'utilisateur doit fournir un certain nombre d'indications: le type de l'UI (K_UIPORT, K_UIGROUP, K_UISITE), un numéro de site (indication nécessaire au protocole de localisation), le noyau forge alors l'identificateur unique en fonction de ces paramètres

uiCreate a la même fonction, mais le noyau prend des valeurs par défaut (en particulier il choisit K_UISITE)

uiDeclare : déclare un UI utilisateur auprès du noyau sur un site, nécessaire avec UI de type K_UISITE

uiForget : le noyau "oublie" toute information à propos d'un UI précédemment déclaré

Acteurs/Tâches/Processus (1)

Unité de Structuration d'architecture =

Acteur/Tâche/Processus

Espace d'allocation de ressources :

. Mémoire : Espace d'adressage (protégé)

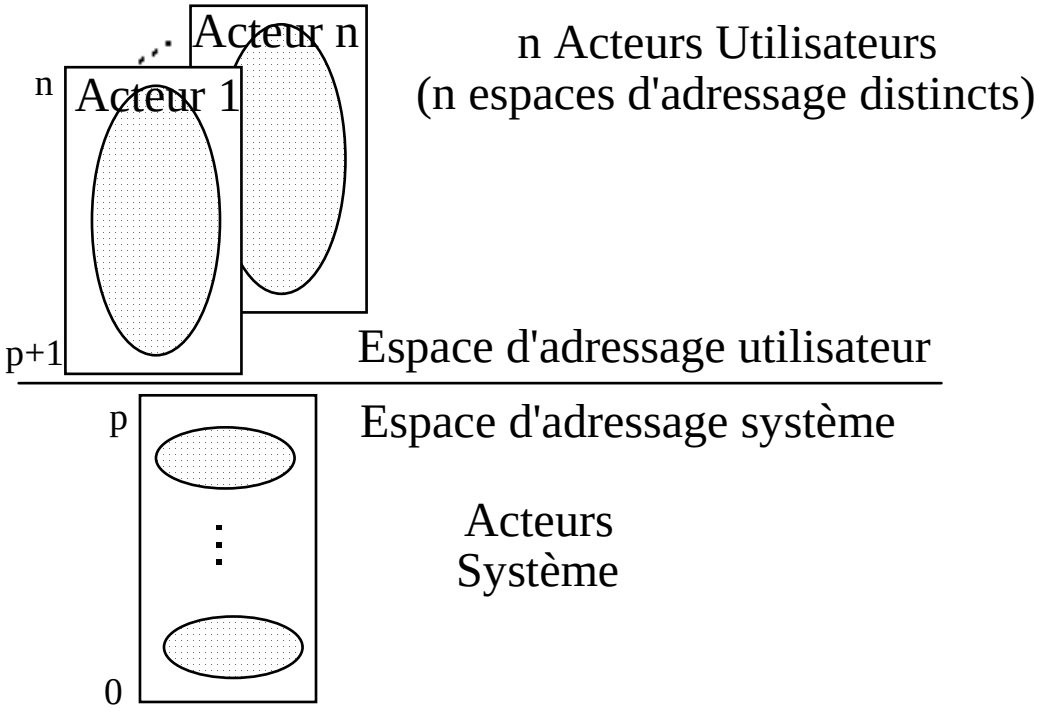
. Ressources système : ports, sémaphores, canaux d'E/S, ...

. Activités

On retrouve le modèle UNIX du processus mais affiné

Pour Chorus, un acteur représente une machine virtuelle.

Acteurs/Tâches/Processus (2)



Acteurs Chorus (1)

Création *actorCreate* **:** Comme lors d'un fork Unix, l'acteur créé (fils) peut hériter d'un certain nombre d'attributs de l'acteur créateur (père).

Acteur est créé avec un minimum de ressources:

- une porte

- pas d'activité

- possibilité d'hériter d'une partie de l'espace mémoire de l'acteur père (régions)

Un acteur ne migre pas d'un site à un autre.

Acteurs Chorus (2)

Deux types d'acteurs :

- Utilisateurs :

. Utilisateur sans aucun privilège

. Serveur Système accède à certaines primitives du système (équivalent au statut d'un "daemon" Unix)

- **Superviseurs :** leurs activités s'exécutent en mode système dans l'espace d'adresse du noyau

=> pb graves en cas d'erreur de programmation

Thread

Unité d'exécution =

Thread/Processus Léger/Activité

. Son exécution a un caractère **séquentiel**

. Contexte d'exécution :

- état
- compteur ordinal
- registres
- pile d'exécution
- descripteur

. Lié à un acteur et un seul (espace d'adressage)

. Partage l'espace d'adressage et les ressources d'un acteur avec d'autres threads

Threads

Ordonnancement

Thread = Unité d'ordonnancement indépendante pour le noyau

Parallèlisme

pseudo-parallèlisme

ou

parallèlisme réel sur multiprocesseur[2]

Que se passe-t-il de différent entre les deux situations suivantes?

. commutation de contexte entre deux threads d'un même acteur

. commutation de contexte entre deux threads d'acteurs différents

---

[2] **Symétrique** : les processeurs ne sont pas distingués, ils ont tous les mêmes capacités d'exécution,
Asymétrique : les processeurs sont distingués, certains ont des capacités différentes, un co-processeur arithmétique par exemple, et certaines threads doivent s'exécuter exclusivement sur ces processeurs

Threads Chorus

Thread :

. Liée à un et un seul acteur

. Une thread est créée par une thread éventuellement dans un acteur différent

. Sur un multiprocesseur, les threads d'un même acteur peuvent s'exécuter en même temps de façon concurrente

. Unité d'ordonnancement considérée par le noyau

. Priorité : la priorité d'une thread n'a de sens qu'au sein d'une classe d'ordonnancement

. Une partie du contexte du thread est fournie à la création :

- compteur ordinal

- le pointeur de pile (mais pile elle même non allouée)

- le mode d'exécution (système ou utilisateur)

Le noyau est préemptif => **temps réel**

Coopération entre Threads

Entre threads d'un même acteur

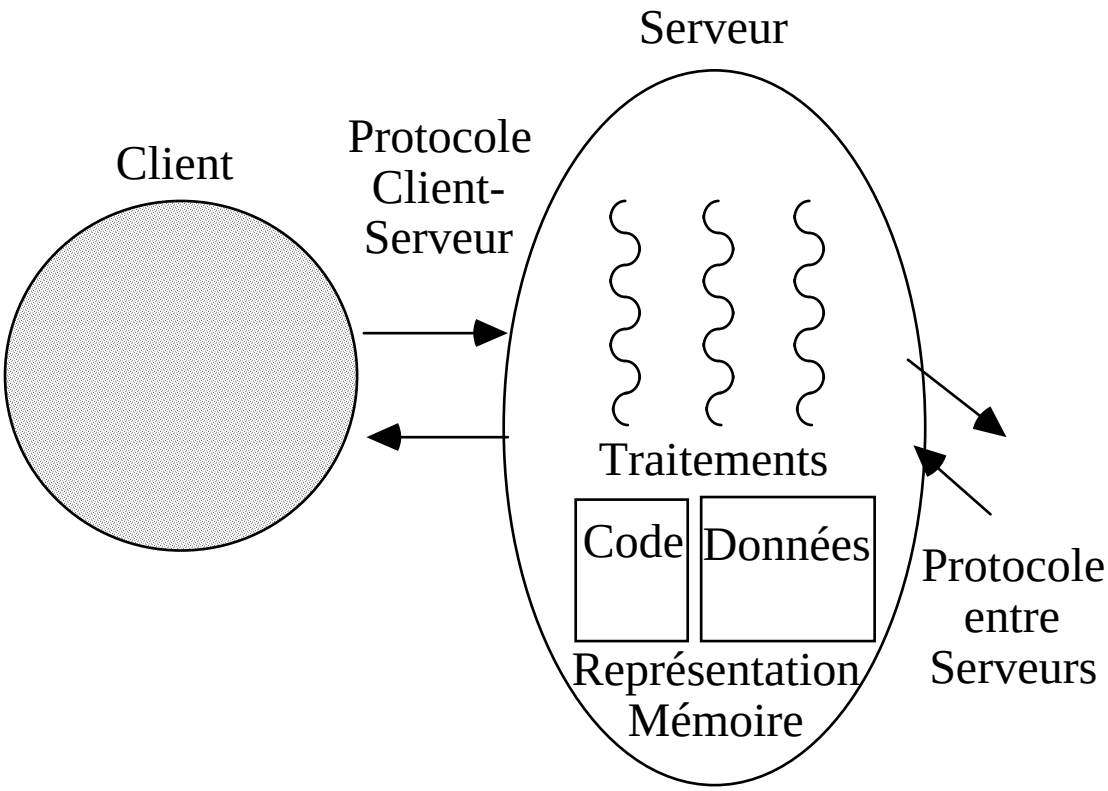
=> par partage de mémoire avec synchronisation

Entre acteurs

=> par messages, en utilisant des portes

Communication Client-Serveur en local

Threads => possibilité de traiter plusieurs requêtes simultanément chez un serveur



Messages

Unité d'échange entre acteurs = Message

Suite d'octets

Il est non typé dans CHORUS, Amoeba, WindowsNT

Typé dans Mach en fonction de la nature des données qu'il contient

(plusieurs données dans un message => plusieurs types[3] coexistent dans un message)

Dans CHORUS, un message est constitué d'une partie principale (body) de taille variable, et d'une partie secondaire de 80 octets (annex).

[3] Il ne faut pas entendre type au sens du typage des données dans les langages de programmation, d'ASN1 ou de XDR, mais plutôt nature des informations pour la gestion du système.

Portes

Unité d'adressage = Porte

Porte :

- adresse : Destination d'un message
- boite aux lettres : File de messages (avec un message courant dans CHORUS)

. Droits d'utilisation : [Emission[4]], Réception

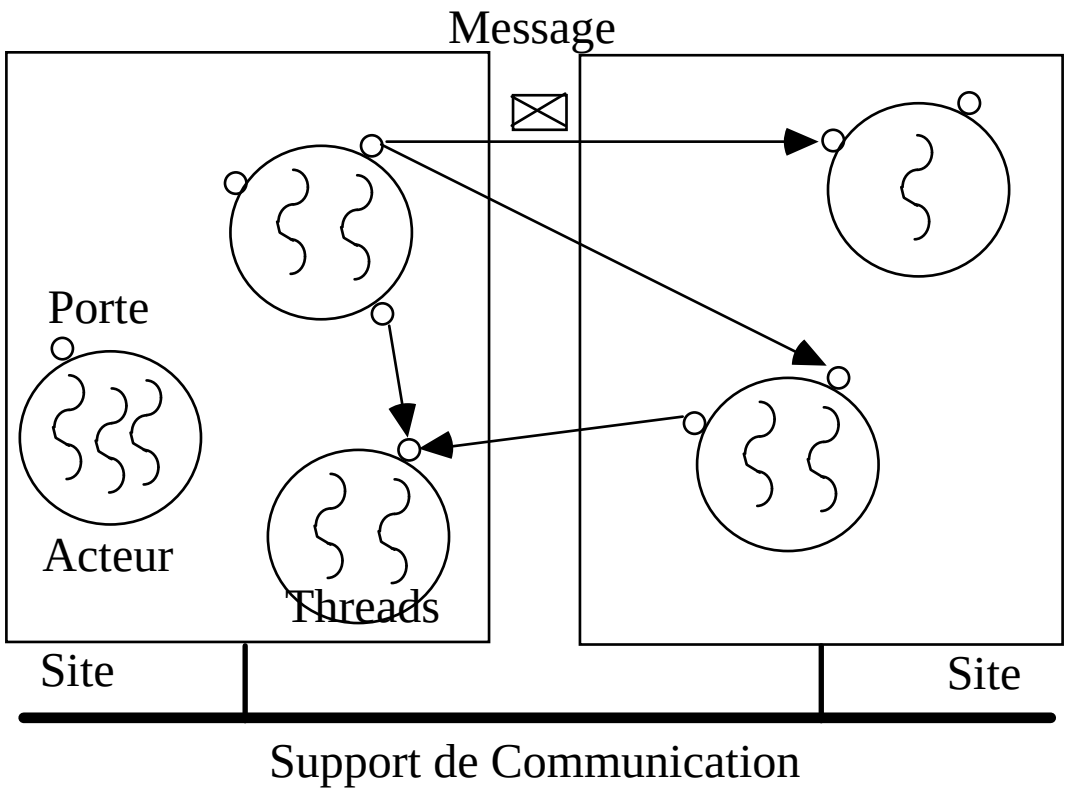
. Porte liée à un acteur : seules les activités de cet acteur possèdent le droit de réception

. Droit de réception transmissible (migration de porte vers un autre acteur)

. Droit d'émission transmissible et partageable

[4] Portes unidirectionnelles dans Mach.

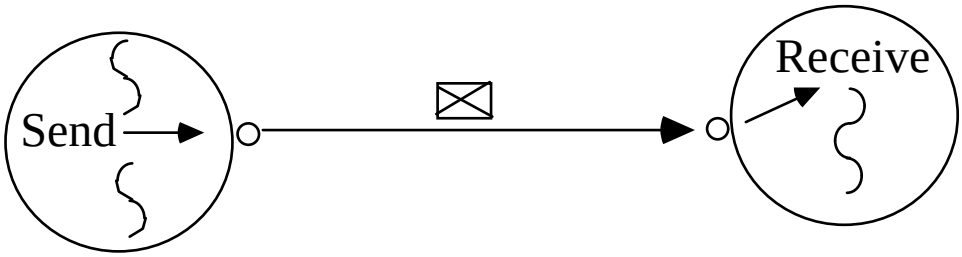
IPC - Communication entre acteurs



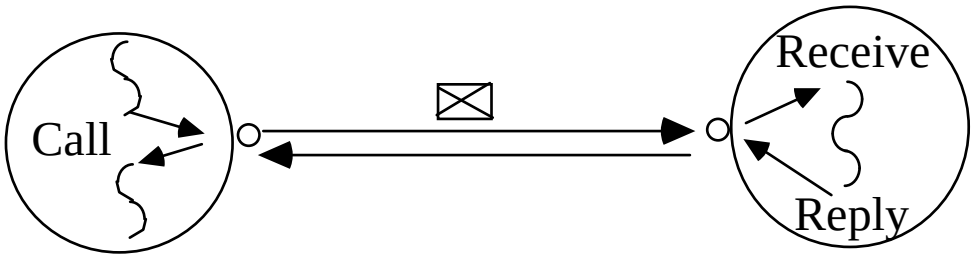
IPC = Inter-Process Communication

IPC

IPC non bloquant/asynchrone :



IPC synchrone :



IPC sans threads dans CHORUS

Programmation par **interruption logicielle** (mécanisme de"message handler") :

On peut associer un traitement à une porte. On connecte un service à une porte.

Dans ce cas, toute réception de message provoque l'exécution du traitement :

- l'ipc est local (LRPC[5]), l'appelant exécute le traitement du serveur chez lui

- l'ipc est distant vers un serveur (FRPC[6]), le noyau du site distant dispose d'un "pool de threads", une de ces threads est utilisée quand le serveur est sollicité pour traiter une demande.

Ce mode de traitement est restreint aux acteurs superviseurs et est incompatible avec le mode normal.

[5] LRPC = Lightweight Remote Procedure Call
[6] FRPC = Full weight Remote Procedure Call

Diffusion sur groupe dans Chorus

Groupe de portes

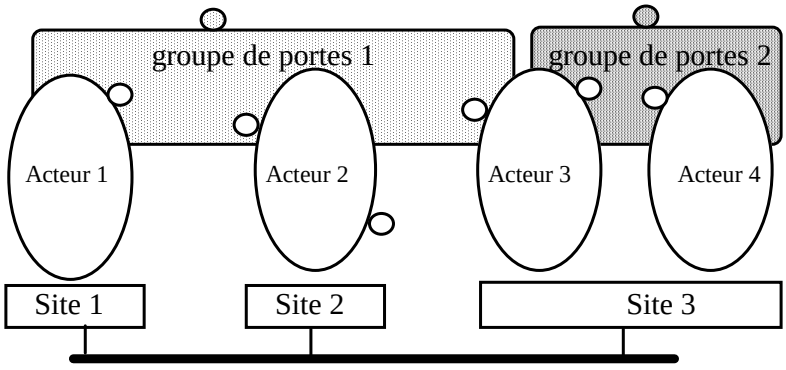
Un groupe de portes est désigné par une capacité

Deux modes d'adressage pour l'émission des messages :

- Diffusion : toutes les portes d'un groupe reçoivent le message

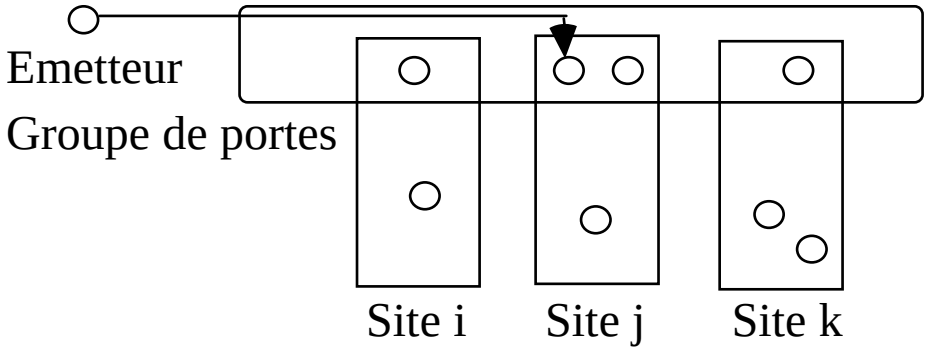
- Fonctionnel : une seule porte dans le groupe reçoit le message

L'adressage fonctionnel peut être "avec" ou "sans" sélection du destinataire. Cette sélection est effectuée par l'émetteur sous la forme d'un critère.

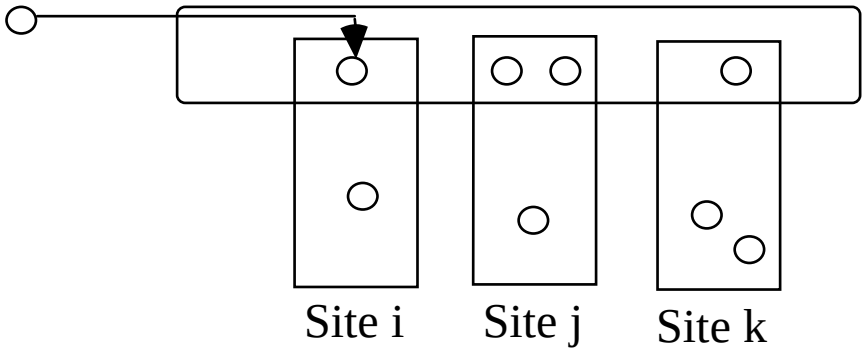


Adressage Fonctionnel dans Chorus

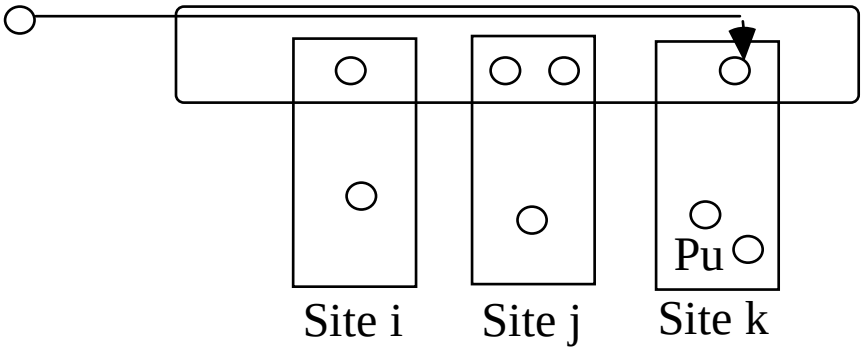
- Adressage Fonctionnel sans sélection : une dans le groupe (idem pour Mach)



- Adressage Fonctionnel avec sélection sur site i



- Adressage Fonctionnel avec sélection sur même site que porte Pu (Pu peut ne pas appartenir au groupe)



Protocoles

Communications locales

=> éviter les recopies (RPC Léger)

Communication à l'intérieur d'un domaine

- une première couche au dessus du réseau avec un protocole de type IP (mode datagramme)

- une seconde couche qui joue le rôle d'une couche transport ou session

Communications extérieures

=> serveurs spécialisés qui sont équipés de piles de communication plus complexes : TCP-IP, OSI ...

IPC Chorus : portes et groupes - API

portCreate : création d'une porte

portDelete : destruction d'une porte et de tous les messages qui y sont rattachés

portUi(&portUI, portLi) : retourne l'UI d'une porte à partir de son Li

portLi(&portUI) : donne le Li d'une porte à partir de son UI

grpAllocate : crée une capacité de groupe de portes (UI+Clé), le groupe peut-être nouveau K_DYNAMIC, statique de portée utilisateurs K_STATUSER, statique de portée système, prédéfini, K_STATSYS

grpPortInsert/grpRemove : insérer/retirer une porte dont on donne l'UI dans un groupe dont on spécifie la capacité

IPC Chorus : opérations - API

L'IPC attache la notion de message courant à l'utilisation d'une porte

ipcTarget (groupUi, mode) : construit une destination pour la communication sur groupe, ajoute le mode d'adressage à l'UI du groupe, le mode peut être : K_BROADMODE, K_FUNCMODE, K_FUNCUMODE, K_FUNXMODE

ipcCall : envoie de message en mode synchrone, le message est considéré comme faisant partie d'une transaction entre le demandeur et le serveur invoqué, on peut envoyé sur un groupe mais seulement en mode fonctionnel (spécification dans le champ de type *msgdest de type KnIpcdest : target : UI groupe; d'une coTarget : UI site)

ipcReceive : réception d'un message sur une porte, appel qui bloque le demandeur, la taille du message est fournie au retour, le message reçu est le nouveau message courant, l'ancien est perdu

ipcGetData : extrait le corps du message courant

ipcSave : sauve le message courant dans un tampon du noyau, l'utilisateur pourra y accéder de nouveau en fournissant le msgid retourné par l'appel

ipcRestore : remet un message comme message courant dans la file des messages d'une porte

ipcReply : réponse à un message

ipcSend : envoie de message suivant politique best-effort, diffusion sur groupe possible

Handlers logiciels- API Chorus

Pour les acteurs superviseurs uniquement, au lieu d'utiliser des messages pour aller plus vite :

svMsgHandler : attachement d'un gérant de message à une porte en réception

svExcHandler : attachement d'un gérant d'exception

svItConnect : attachement d'un gérant d'interruption

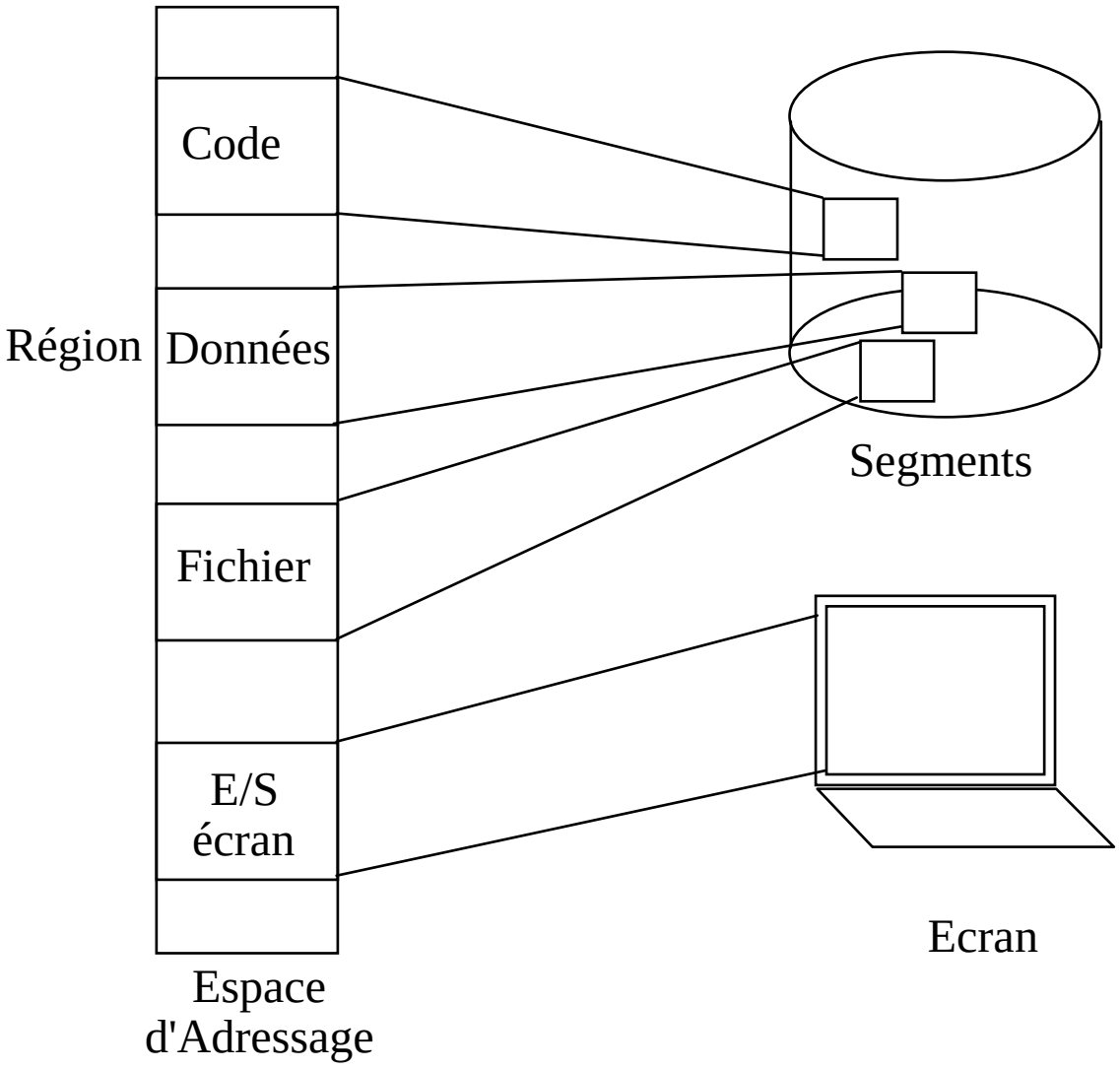
svCallConnect : attachement d'un gérant de trappe qui associe une table à un déroutement provoqué par le matériel, lors de la génération d'une trappe (cas d'un appel système), l'utilisateur dépose un n° dans un registre qui sert d'index dans la table

svTimeOut : attachement d'un gérant de fin de délai de garde

Objets mémoire

Unité de représentation des données = **Segment**

Unité d'accès aux données en mémoire d'exécution = **Région**



Segments et Régions

Segment :

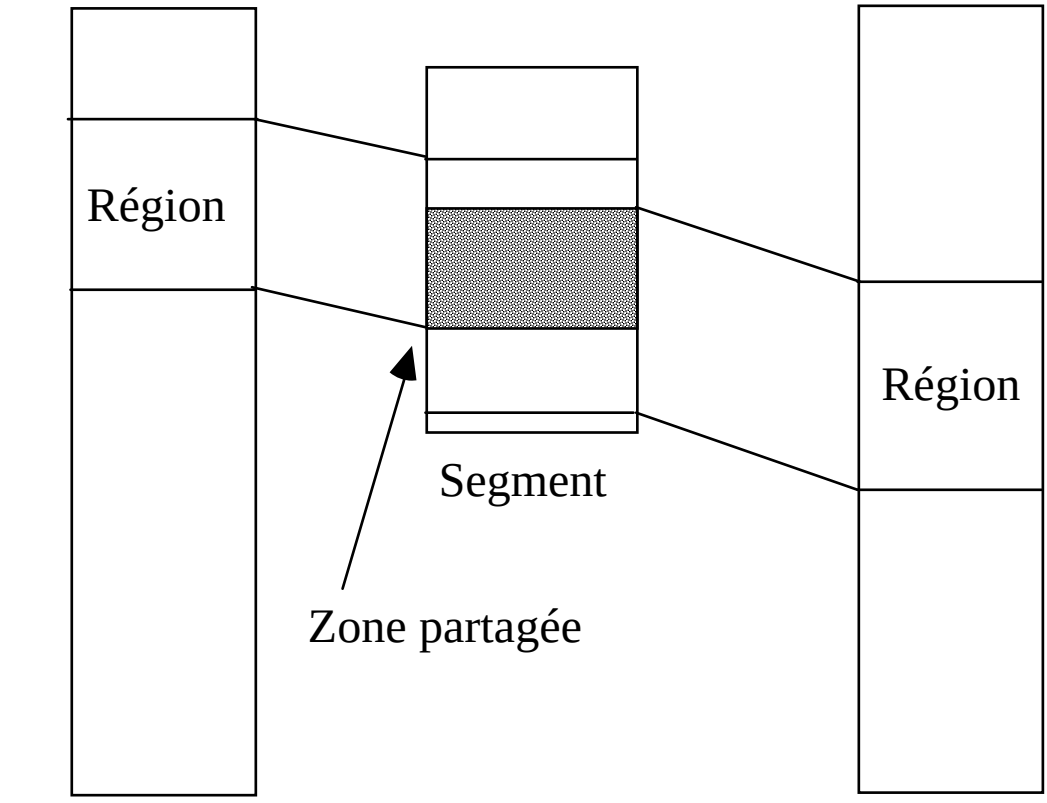
- Données permanentes (fichier) ou temporaires (espace swap)
- **Géré par serveur** externe au noyau (mappeur)
- **Identifié par une capacité**

Région :

- L'espace d'adressage d'un acteur est divisé en plusieurs régions
- Partie visible en mémoire d'un segment : fenêtre sur un segment qui correspond à sa projection partielle dans l'espace d'adressage d'un acteur

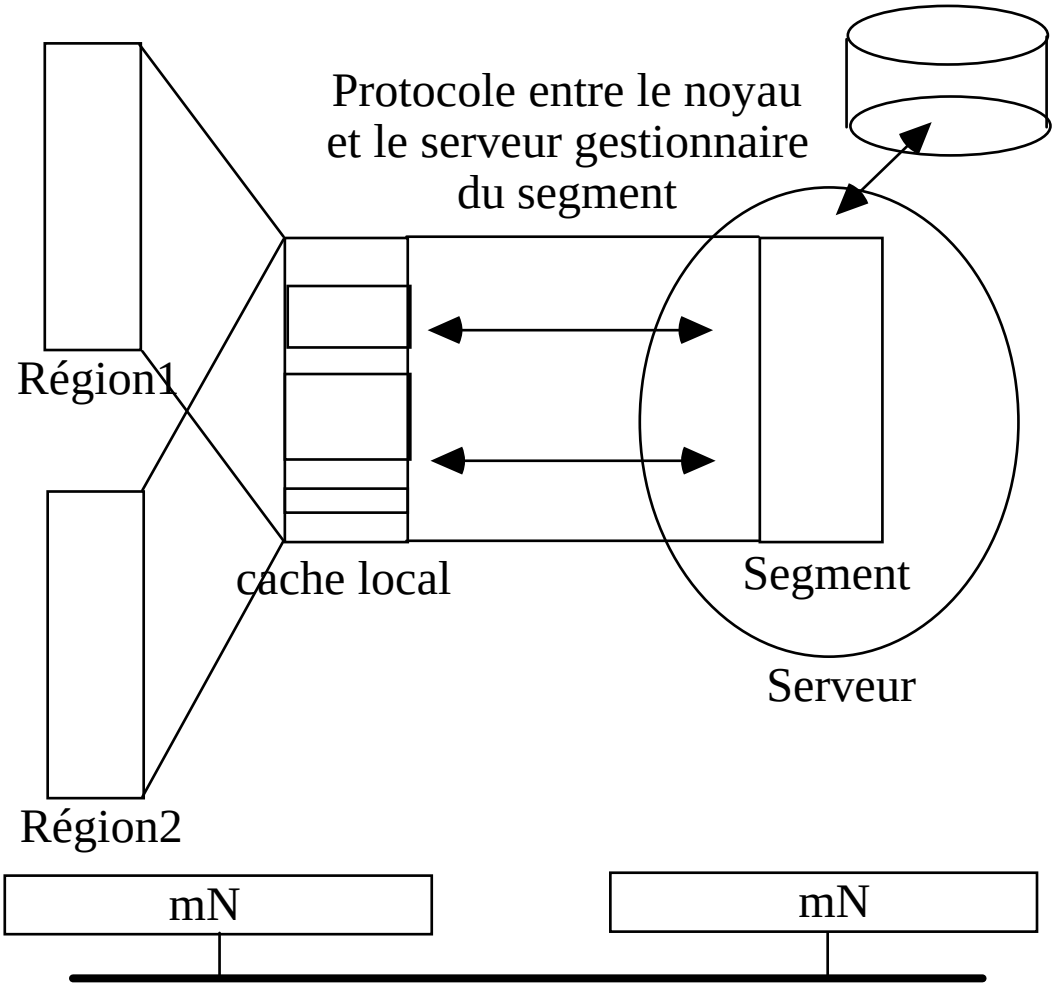
Partage

Partage d'un segment entre acteurs par région



Représentation d'un segment en mémoire physique : Cache local d'un segment

Le noyau peut gérer un cache des pages physiques associées à un segment.



-> Pas plus d'un cache local par segment et par site

-> Le noyau assure la cohérence d'un segment partagé par plusieurs acteurs sur le même site mais ne gère pas sa cohérence quand il est partagé entre plusieurs sites

Mémoire Chorus : Régions et Segments - API

rgnInit : allocation et initialisation d'une région à partir d'un segment, rgnInitFromActor initialise à partir de l'espace d'adressage d'un autre acteur

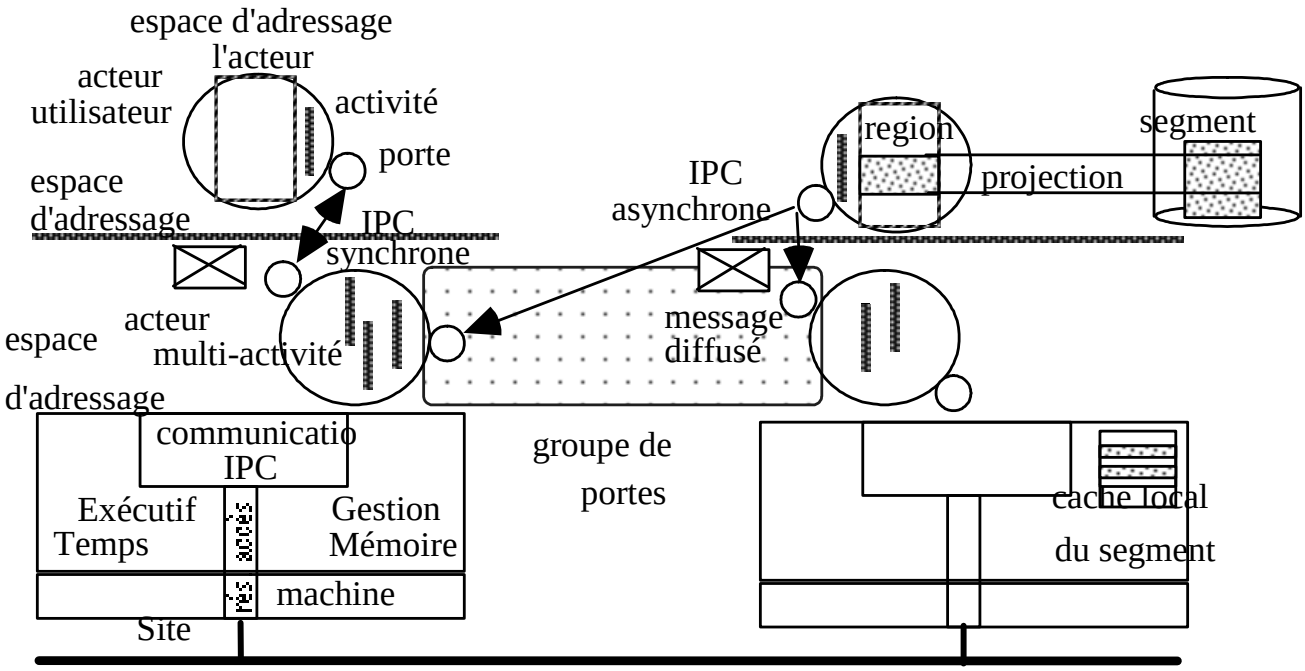
rgnAllocate/rgnFree : allocation/libération d'une région

adressage virtuel seulement:

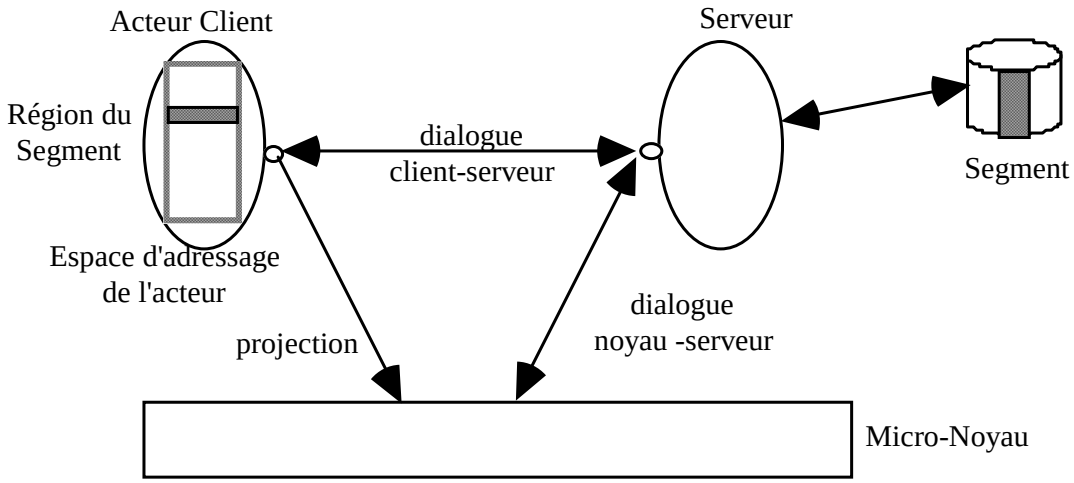
rgnMap : allocation et initialisation d'une région à partir d'un segment, équivalent au mmap d'Unix, les modifications apportées sur la région sont répercutées sur le segment

sgRead/sgWrite : lire/écrire le contenu d'un segment et le charger dans l'espace d'adressage d'un acteur

Vue d'ensemble



Accès aux données : vue globale



accès aux données entre acteur client, noyau et serveur

Gestion des données en mémoire physique

Un segment dans le micro-noyau :

un cache local : portion de mémoire physique gérée par le noyau d'un site, qui contient les données d'un segment. Sur un site Chorus, un segment a son propre cache local et un cache local ne peut contenir que des données relatives à un seul segment. A chaque cache local sont associées deux capacités et un historique :

. la capacité du cache local construite par le noyau lors de la création du cache local,

. la capacité du segment associé au cache local qui est utilisée lors du transfert de données du noyau vers/depuis le serveur du segment,

. un historique mémorisant l'ensemble des entrées et des modifications apportées sur le segment dans le cache.

Un segment peut être accédé sur plusieurs sites. La cohérence des caches locaux d'un même segment situés sur des sites différents n'est pas gérée par les différents noyaux et nécessite l'utilisation d'un service de gestion de cohérence externe.

Mémoire Chorus : cache local et MPR - API

Le noyau dispose de quatre types de requêtes à destination d'un serveur :

· *MpPullIn,* le noyau demande au serveur de fournir les données et les droits d'accès d'un segment, pour remplir le cache local correspondant. La capacité du cache local est transmise au serveur. Une demande de droits d'accès sur ces données est également associée au *MpPullIn*. Le serveur répond en fournissant les données demandées et en indiquant les droits d'accès effectivement obtenus.

Lors du premier *MpPullIn,* le serveur récupère la capacité du cache local du segment qu'il n'avait pas moyen de connaître avant. Cette capacité permettra au serveur d'accéder au cache local.

· *MpGetAccess,* le noyau demande au serveur des droits d'accès sur des données. Le mappeur répond en indiquant les droits d'accès effectivement obtenus. Le *MpGetAccess* est un *MpPullIn* sans demande de données. Il est utile lors de changement de droits d'accès, par exemple, lorsque le noyau possède la page en lecture seulement, et qu'il en a besoin en écriture.

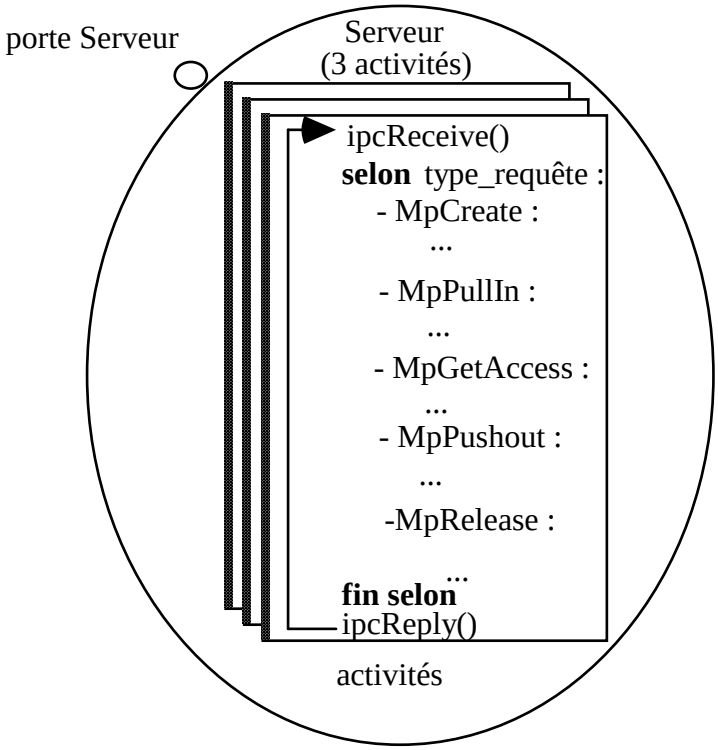
· *MpPushOut,* le noyau demande au serveur de recopier sur le segment géré les données correspondantes contenues dans le cache local.

· *MpRelease,* le noyau avertit le serveur que le cache local du segment a été détruit. Message optionnel, le serveur décide de recevoir cet avertissement ou non. La destruction réelle du cache local est effectuée :

- lorsqu'il n'existe plus sur le site de régions clientes projetées sur le segment associé,

- lorsque le taux d'occupation de la mémoire physique descend en dessous d'un certain quota, la valeur de ce quota étant paramétrable

Architecture d'un serveur de segment



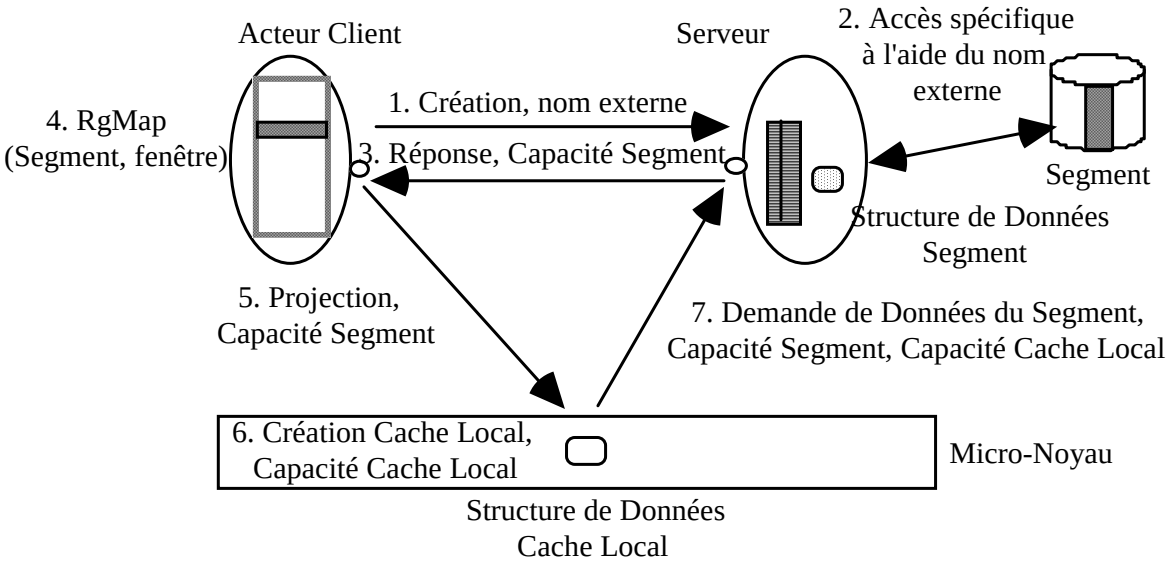
Modèle de traitement d'une activité d'un serveur de segments

Gestion du cache local par le noyau

-*lcFlush* : permet à l'appelant de contrôler le contenu du cache local. L'utilisateur spécifie la portion du segment qui doit être recopiée du cache local vers le segment. Cette primitive a pour effet de provoquer une éjection de pages hors du noyau. Optionnellement, l’accès à la portion du segment dans le cache local peut être modifiée. Il peut éventuellement y avoir une invalidation de la portion de segment spécifiée.

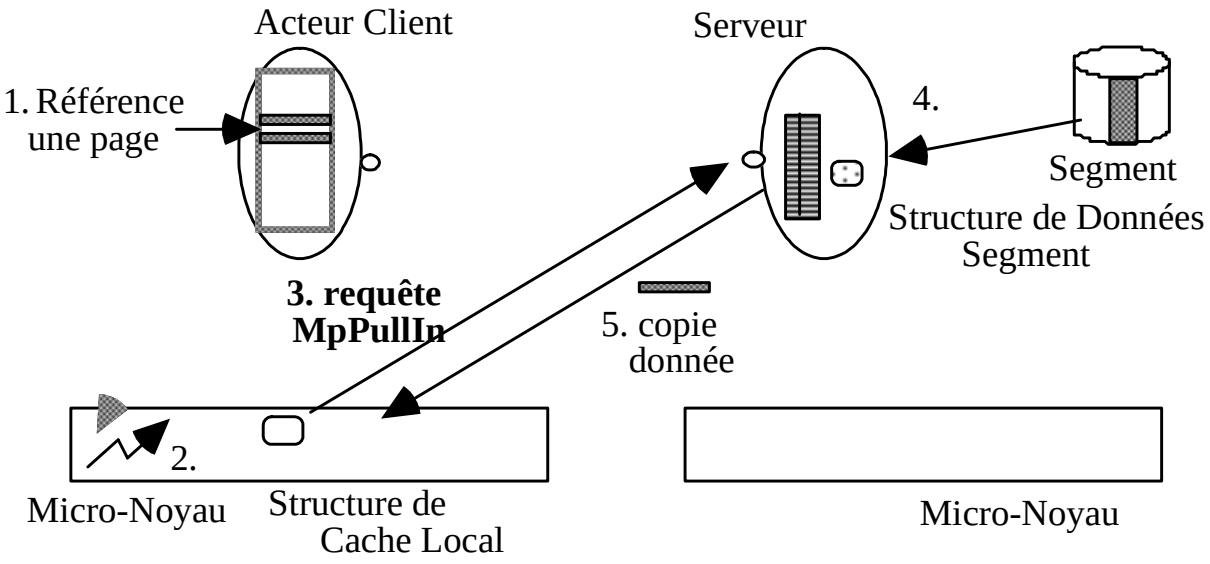
-*lcSetRights* : cette primitive agit de la même façon que le lcFlush, mais ne provoque aucune éjection de pages du cache local vers le segment. Elle est essentiellement utilisée pour les invalidations de données.

Capacités associées à un Segment

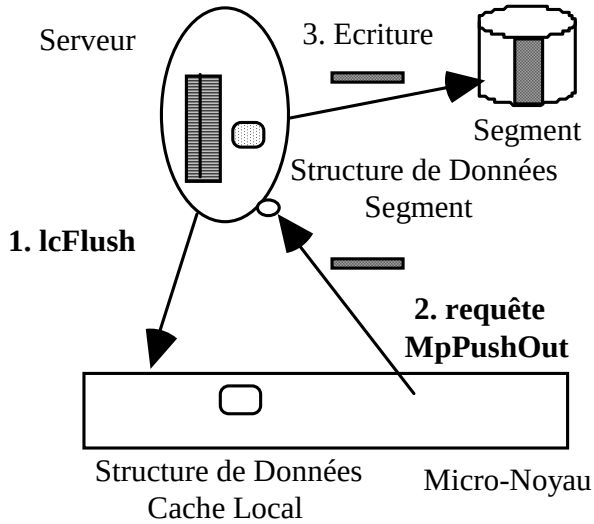


La gestion de segments est assurée par un dialogue entre le noyau et les différents serveurs associés aux segments. Ces serveurs peuvent se coordonner ou non, être répartis ou non.

Fonctionnement global



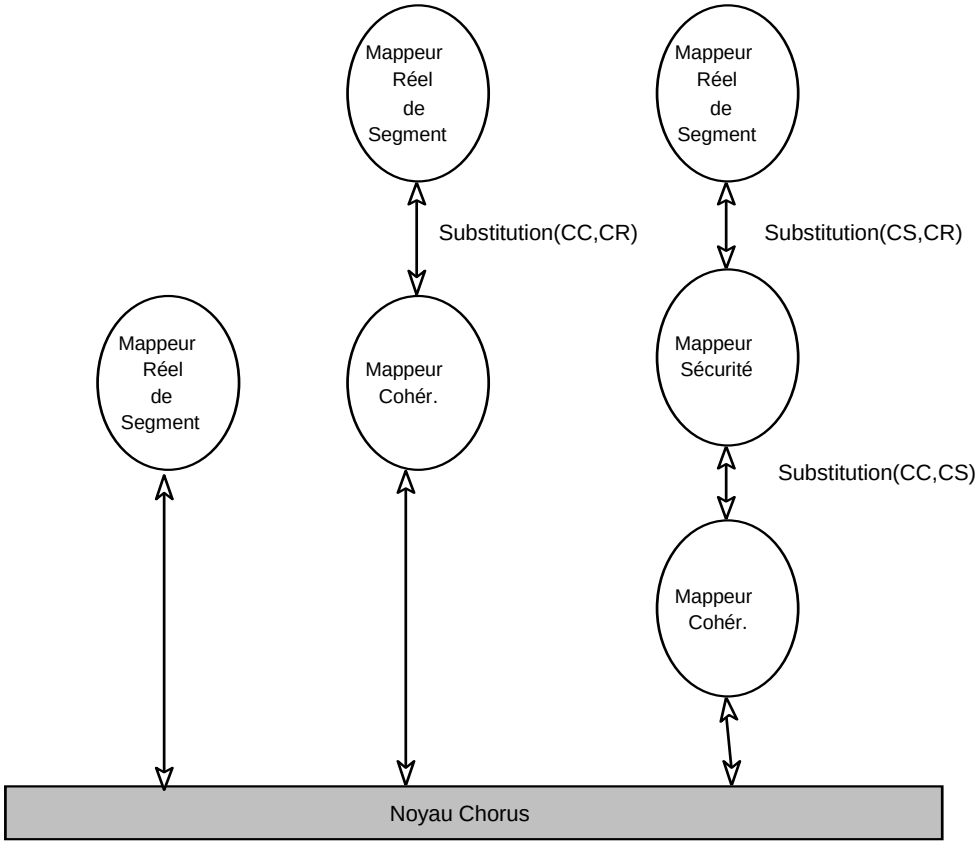
Défaut de Page



La réponse du MpPushOut ne figure pas dans le dessin pour ne pas l'allourdir.

Vidage d'un cache par *lcFlush*

Empilage dynamique de Sémantiques



La mise en oeuvre d'une pile de mappeurs s'appuie sur **le mécanisme de substitution de capacités**. La substitution de capacité se fait au niveau de chaque mappeur, qui échange, pour un segment donné, une capacité reçue avec sa propre capacité.

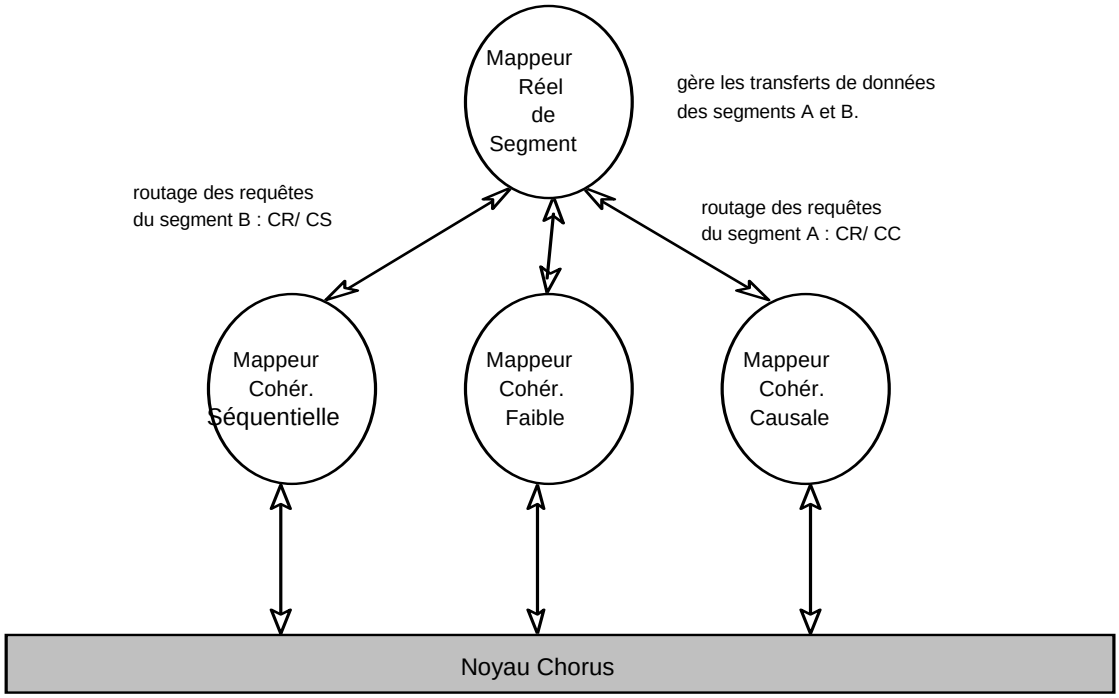
Avantages de la substitution

- Cette substitution permet de désigner de facon unique le segment pour les traitements du mappeur du niveau considéré. Par exemple, le mappeur de cohérence utilise une capacité cohérente pour assurer la cohérence d'un segment donné, alors que ce même segment est désigné par une capacité réelle pour la gestion des entrées-sorties par le mappeur réel.

- Elle permet également d'adresser les mappeurs de niveau directement adjacent, car la capacité comporte l'adresse de la porte de communication du mappeur destinataire de la requête.

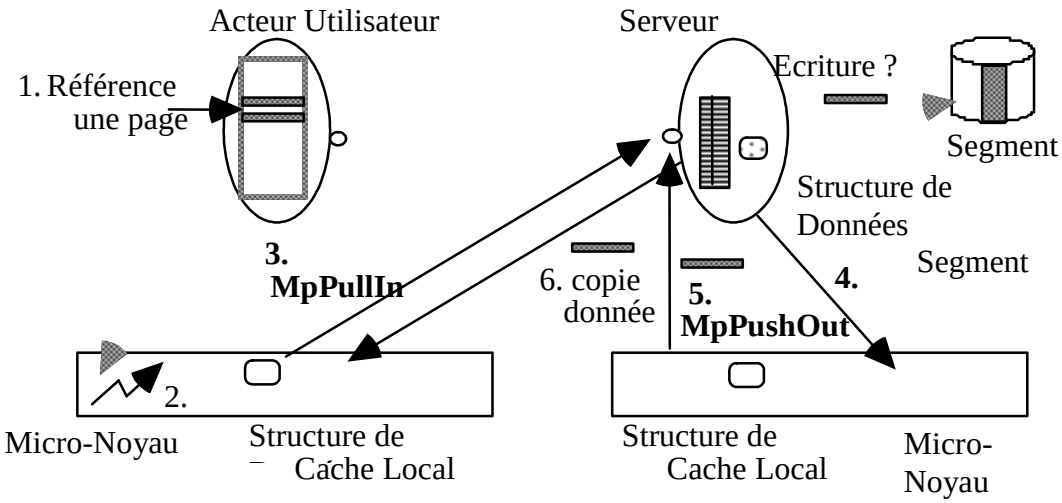
- Enfin, elle garantit que le circuit emprunté par les requêtes du noyau ou du client pour traverser les différentes couches de mappeurs restera constant pour un segment donné. Cette dernière propriété devient importante lorsque le système propose différentes piles de mappeurs, pour différents types de gestion de segments. Dans ce dernier cas, plusieurs routages sont potentiellement possibles et il est indispensable d'emprunter toujours le même pour un segment donné.

Circuit des requêtes dans un système multi-cohérent.



CC : Capacité Cohérente
CS : Capacité Séquentielle
CR : Capacité Réelle

Exemple serveur de cohérence



Défaut de page avec *lcFlush*

Exemple de résolution de noms par serveur de noms

Architecture RFID et EPC Global

RFID

Un tag RFID (Radio Frequence Identification) :

Tous les tags n'ont pas de la mémoire et un processeur.

Plusieurs types de tags :

- Tags passifs : le tag passif reçoit son énergie du lecteur, quand il est alimenté il peut avoir sa propre intelligence, restitué les informations qu'il contient en mémoriser de nouvelles
- Tags actifs : le tag actif est alimenté par un énergie propre (batterie le plus souvent) il est autonome, il peut être relié à des capteurs
- Tags semi-actifs le tag est alimenté par une énergie propre mais un lecteur doit fournir l'énergie pour communiquer

Pour accéder à l'information qu'ils contiennent, il faut un lecteur qui peut aussi écrire.

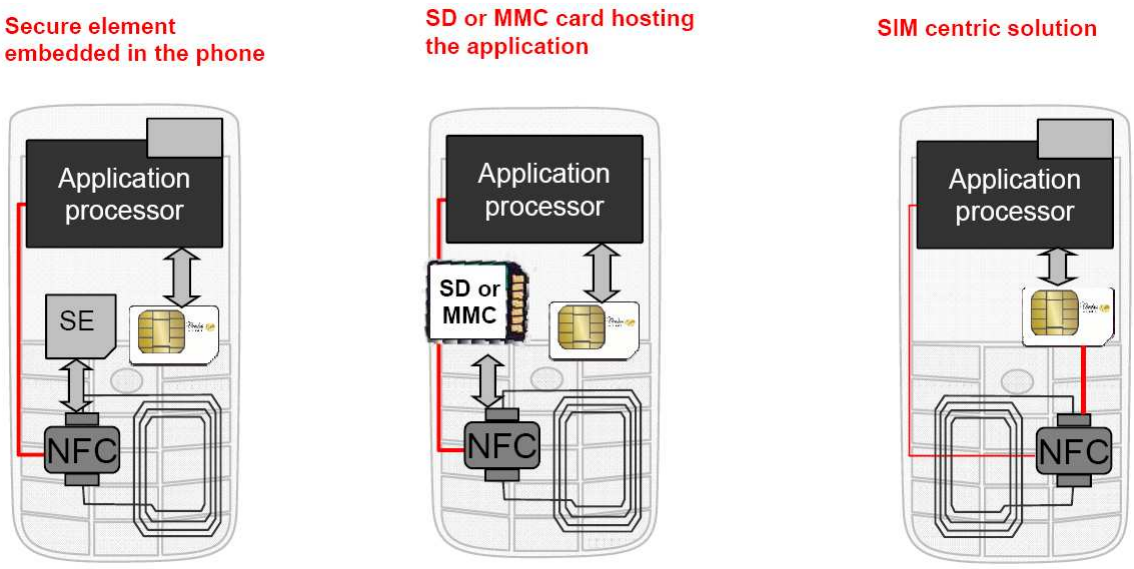
Accès au RFID et Système d'Information associé

Mode d'accès le plus fréquent :

Un lecteur lit le contenu du tag, récupère l'identificateur qu'il contient, et accède à un serveur pour récupérer l'information associée au tag.

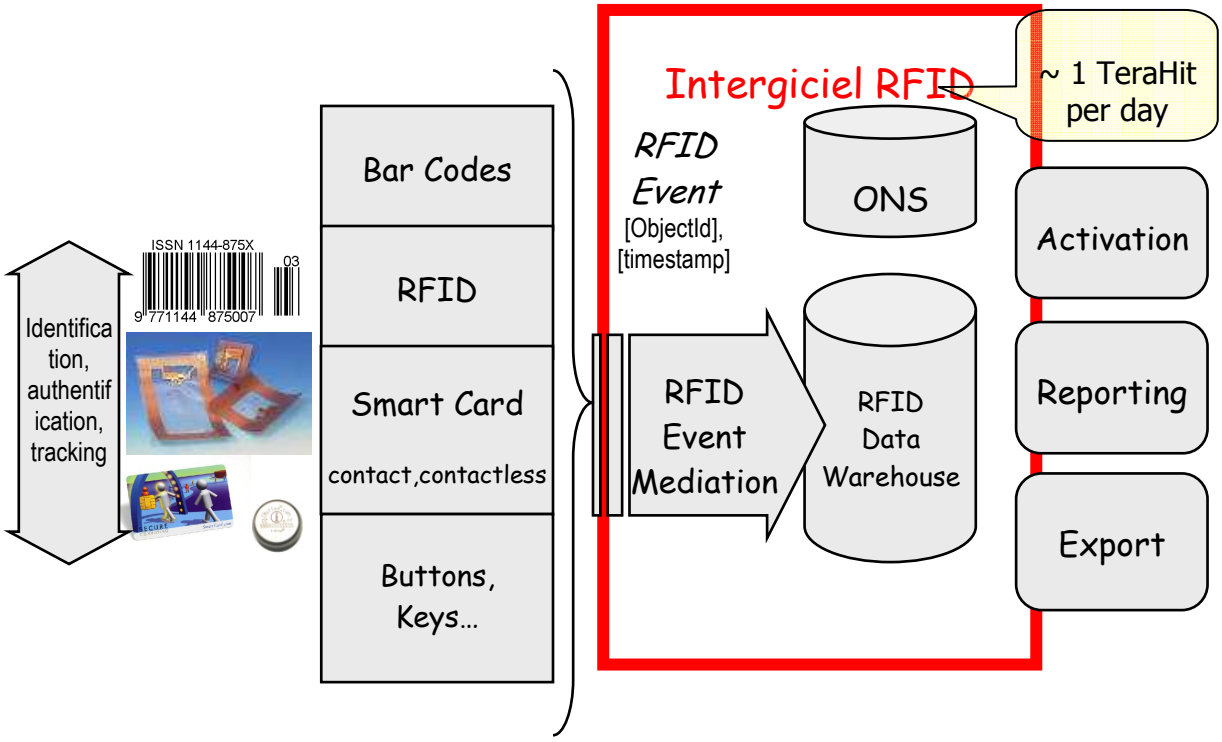
L'architecture EPCGlobal a été proposée pour normaliser l'accès à des tags RFID dans le cadre de la grande consommation.

Architectures sécurisées d'un téléphone NFC

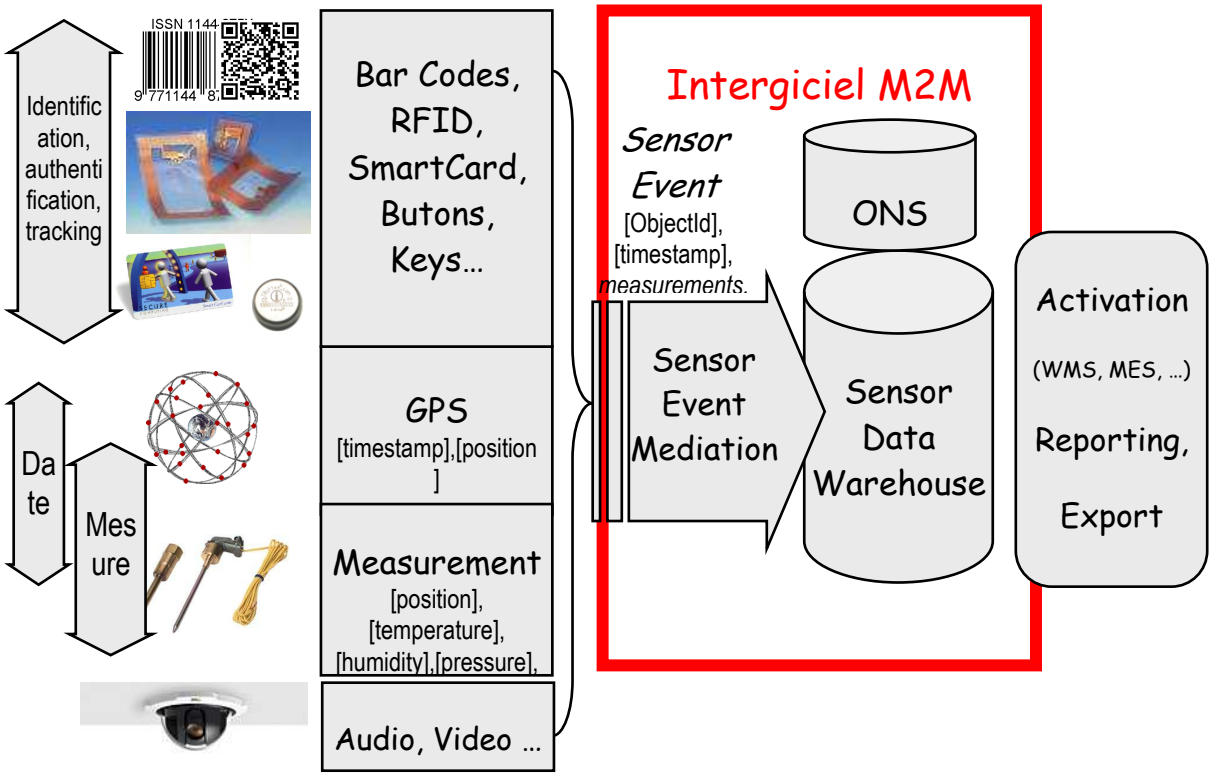


Sur le Nokia 6131NFC, prédécesseur du 6212, on peut accéder au tag via une lecture NFC (Near Field Communication) en Java via l'API de la JSR 257.

Besoins communs aux applications RFID



Evolution des besoins vers le M2M : RFID + Capteurs environnementaux



EPC Global[7] (1)

- Le projet EPCGlobal démarre en 2003 succède à l'AutoID Center (un projet de recherché du MIT débuté en 1999 et supporté par l'industrie).
- EPCglobal est un groupement d'intérêt industriel entre European Article Number (EAN) International et le Uniform Code Council (UCC). Il gère la compatibilité avec d'autres systèmes d'identification plus anciens ou concurrents.
  - La famille UPC/UCC/EAN de codes barres est connue comme Global Trade Item Number (GTIN), a worldwide system of supply chain identification.
  - Le code barre de base de l' Universal Product Code (UPC), est étendu pour inclure le pays, la classe et le produit
  - et il intégre de nouveaux formats

[7] D'après le support de François Fornaciari, équipe ADELE du LIG

EPC Global (2)

- Chaque objet de la chaîne d'approvisionnement est muni d'un tag RFID, identifié par un code unique : **E**lectronic **P**roduct **C**ode (**EPC**)

- Le réseau EPC permet aux membres EPCGlobal de consulter les données à partir d'Internet (aspect clef).

Objectif : Interopérabilité en boucle ouverte (chaîne d'approvisionnement)

- L'architecture réseau EPC permet d'interconnecter un ensemble de composants matériels et logiciels et d'offrir des services aux différents composants au travers d'interfaces.

Par abus on associe parfois Internet des Choses et EPC global.

Electronic Product Code - principes

Tous les EPC contiennent un entête qui identifie la méthode d'encodage utilisée. Cette méthode détermine la longueur, le type, et la structure de l'EPC. Un EPC contient souvent un n° de série qui permet d'identifier de façon unique un objet.

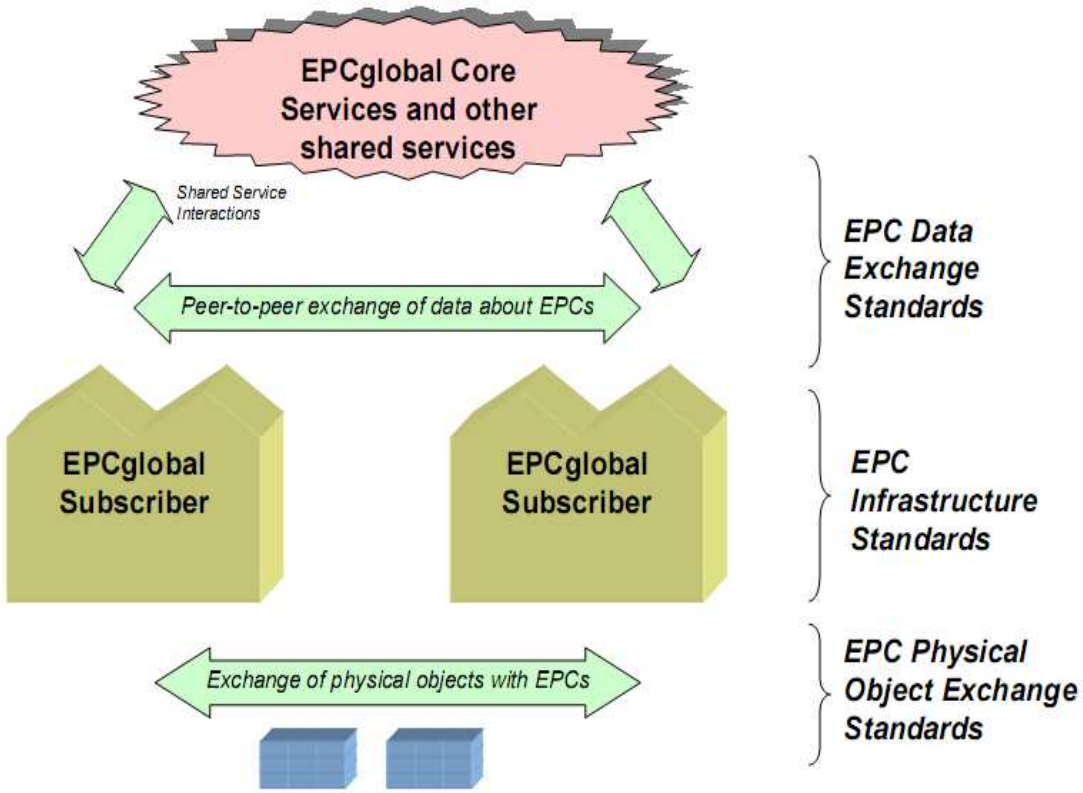
EPC Version 1.3 propose les codages suivants :

- General Identifier (GID) GID-96
- Une version sérialisée du GS1 (General Specification 1) Global Trade Item Number (GTIN), soit <manufacturer, product, version, et en plus un n° de série> pour identifier de façon unique un article. SGTIN-96 SGTIN-198
- GS1 Serial Shipping Container Code (SSCC) SSCC-96
- GS1 Global Location Number (GLN), SGLN-96 SGLN-195
- GS1 Global Returnable Asset Identifier (GRAI) GRAI-96 GRAI-170
- GS1 Global Individual Asset Identifier (GIAI) GIAI-96 GIAI-202 and
- DOD Construct DoD-96

La version 1.4 ajoute aussi:

- Global Service Relation Number (GSRN) GSRN-96
- Global Document Type Identifier (GDTI) GDTI-96

Architecture Globale (1)



EPC Physical Object Exchange

- Les membres du consortium EPCglobal échangent des objets physiques.
- Chaque objet est muni d'un identifiant unique (EPC) qui permet de suivre l'objet à travers sa propre entreprise ou dans une entreprise partenaire
- Lors d'échanges d'objets entre membres EPCglobal, tous les acteurs doivent pouvoir lire et interpréter l'EPC associé à l'objet.

Architecture Globale (2)

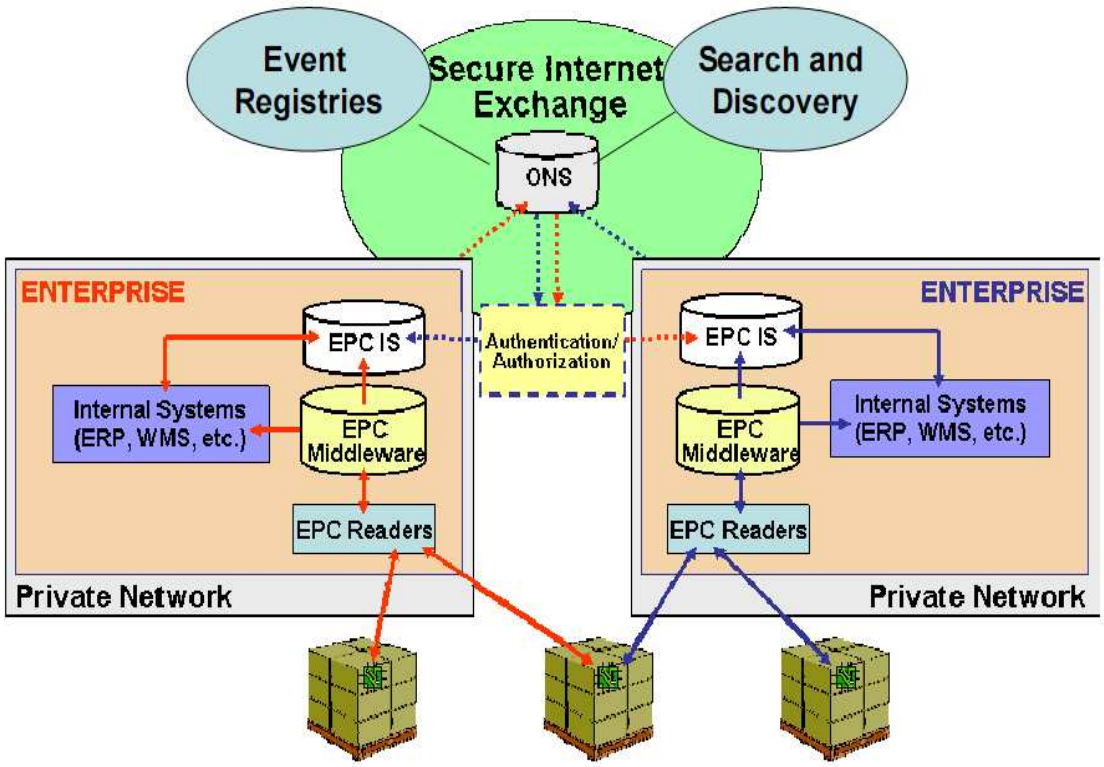
EPC Data Exchange

- Les membres EPCglobal échangent des données sur des objets physiques de manière à suivre leurs mouvements externes.
- Une interconnexion directe **peer-to-peer** et **sécurisée (authentification et autorisation)** est utilisée pour les échanges.
- Il existe d'autres services d'échange en plus du service lié à la sécurité.

EPC Infrastructure

- Opérations nécessaires à l'échange de données
  - ◆Générer les EPC des objets.
  - ◆Suivre les mouvements des objets.
  - ◆Stocker les données dans une base de données interne
- Des standards (interfaces) sont proposés pour mettre en place l'architecture. C'est la base du système interne.
- Définition de standards de gestion et de configuration de matériel et de logiciel EPC.

Architecture Détaillée (1)



Tag RFID

- Le tag RFID de l'objet contient l'EPC.
- L'identifiant EPC détermine un identificateur unique (possibilité d'accéder à des données concernant l'objet, stockées ailleurs).
- Le tag RFID contient des caractéristiques additionnelles (lock, kill, access control, password, ...). L'utilisateur spécifie les valeurs des caractéristiques dont il veut se servir.

Architecture Détaillée (2)

Les Classes RFID

Catégorie	Caractéristiques
0	Lecture (enregistrement de l'information de la puce par le fournisseur)
1	Lecture, écriture unique (enregistrement sur la chaîne de fabrication de l'objet).
2	Lecture, écriture
3	Lecture, écriture plus une source d'énergie sur la puce (tag actif)
4	Lecture, écriture, source d'énergie sur la puce et communication active des étiquettes entre elles
5	Lecture, écriture, source d'énergie sur la puce et communication avec les étiquettes passives.

Le Lecteur EPC

- Il lit l'EPC présent sur le tag RFID et l'envoie vers le middleware EPC.
- Peut également écrire sur le tag dans le cas où l'information est descendante.
- Possibilité d'effectuer un premier filtrage à la demande du middleware EPC.

Architecture Détaillée (3)

Le middleware EPC (le LIG est impliqué dans le projet de middleware open source ASPIRE)

- Il reçoit les données des lecteurs.
- Filtrage et agrégation des données (réduction du volume d'informations).
- Le middleware EPC ajoute une information métier aux données reçues. Les données pourront être par la suite utilisées par des applications spécifiques (contexte d'entreprise).
- Liaison du middleware EPC avec des systèmes internes :
  - ERP (PGI), WMS (Warehouse Management System), ...

EPC Information System (EPC IS)

- L'EPC IS est le système à la base de l'échange des données.
- Possibilité de filtrage des données.
- Les partenaires externes accèdent aux données d'un EPC spécifique via l'EPC IS qui détient l'objet.

EPCglobal Core services (1)

- Object Name Service (ONS confié à Verisign) : Fait le lien entre l'EPC et l'EPC IS de manière à récupérer des données spécifiques (souvent logistiques) sur l'objet muni d'un tag EPC.

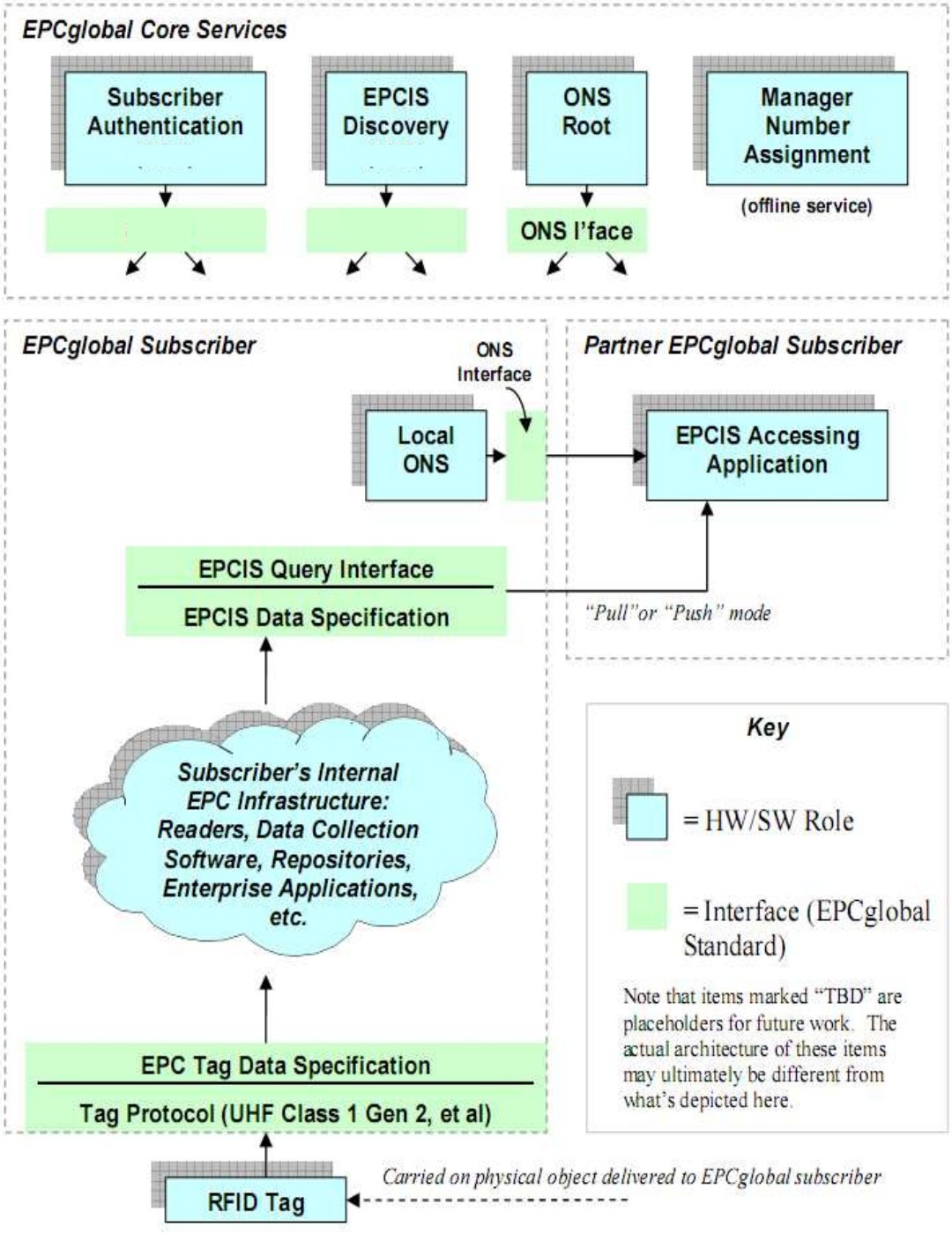
- Security Service : présent à tous les niveaux de l'architecture
  - Un utilisateur doit être authentifié et autorisé pour consulter un EPC IS.
  - Codes access control, kill et lock présents sur le tag RFID.

- Event registries
  - Nécessaire lorsqu'un objet physique muni d'un EPC a été transformé par différents partenaires (commerciaux) et que chacun de ces partenaires possède des données sur celui-ci dans son EPC IS.
  - Mise en place d'un registre qui pour chaque chaîne d'approvisionnement, tiendra à jour tous les EPC IS concernés.

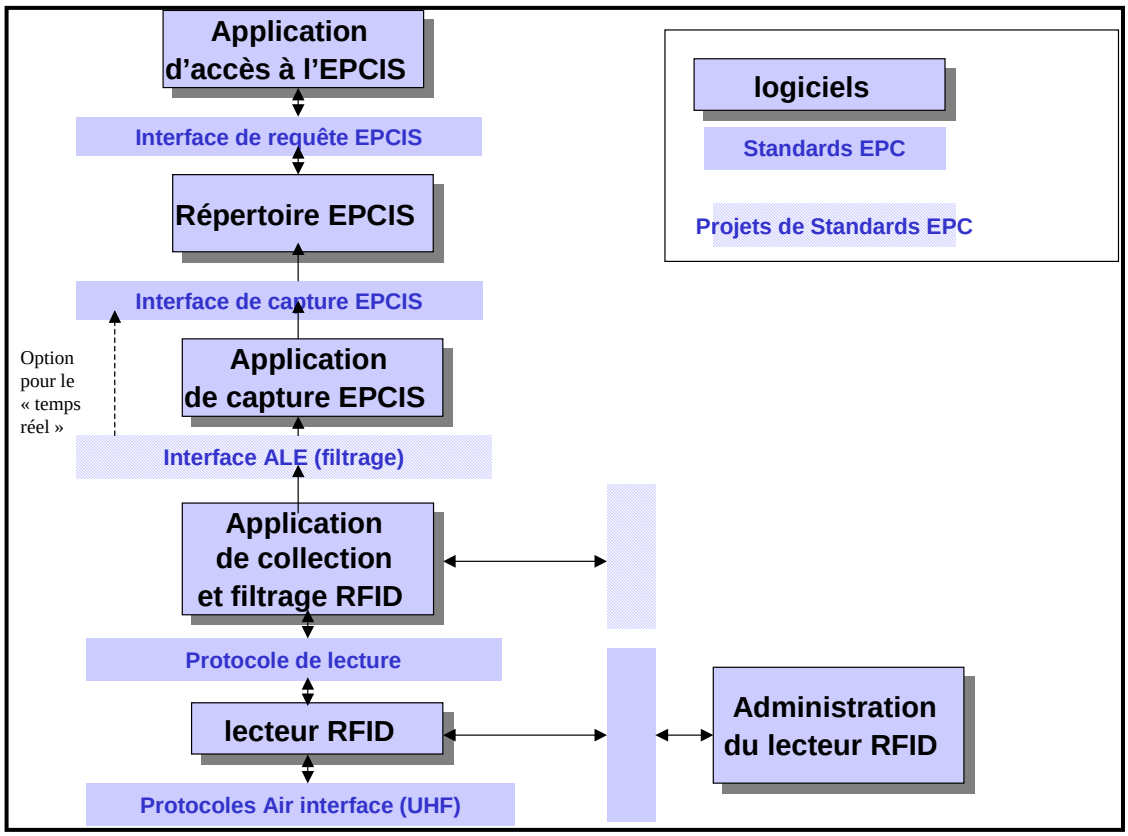
- EPC IS Discovery Services : pour localiser tous les EPC IS qui contiennent des informations sur un objet recherché : cela inclut tous les EPC IS autres que le EPC Manager.

- Subscriber Authentication : l'authentification d'EPC Subscribers afin de gérer le conrôle d'accès à l'EPC IS.

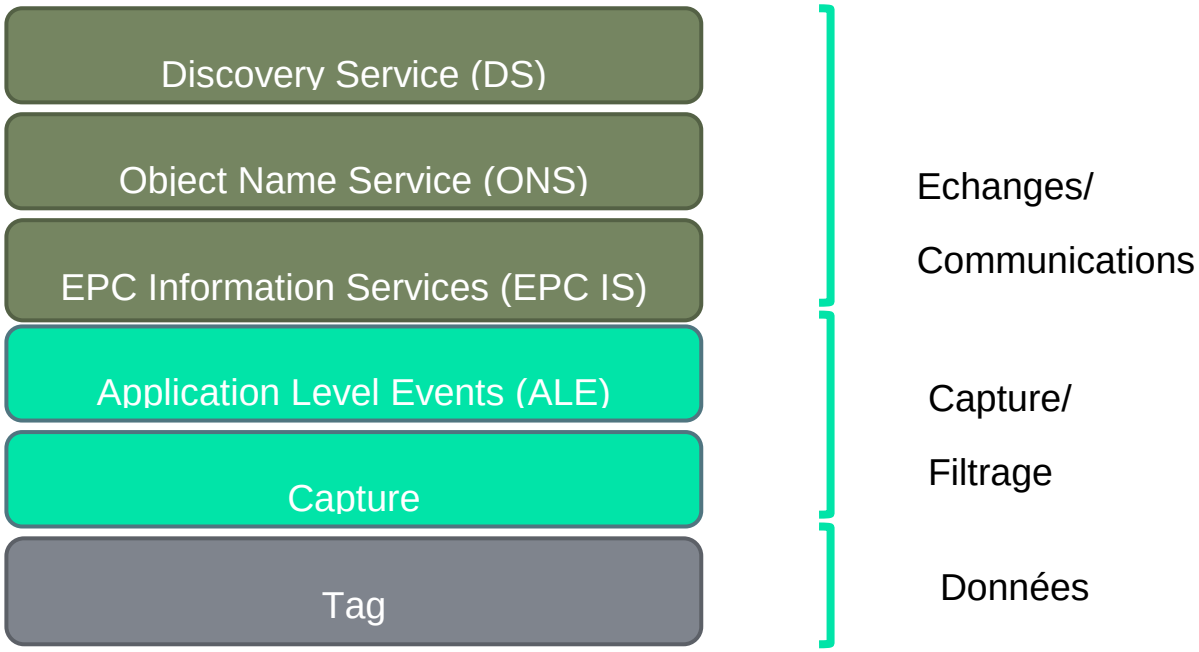
EPCglobal Core services (2)



EPCglobal Core services (3)

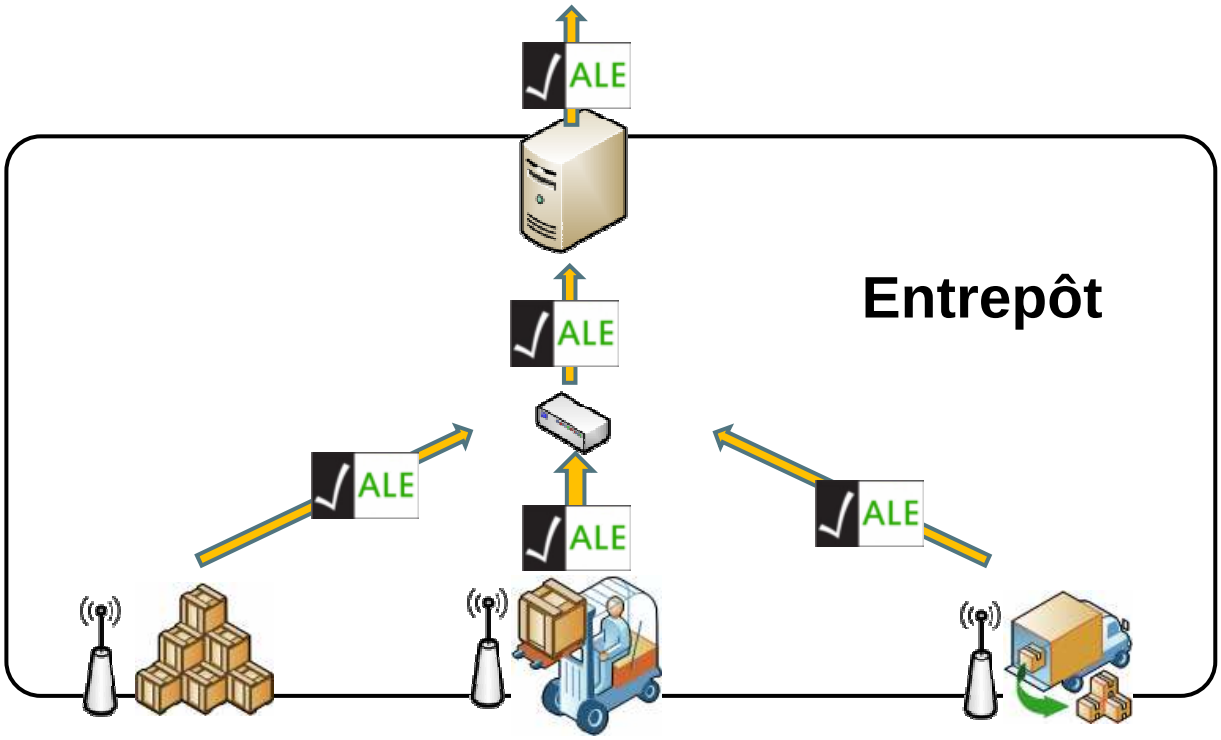


Spécifications EPC Global



ALE :

- Objectifs
  - Réduire volume des données entre lecteurs et applications
  - Isoler les applications des spécificités des lecteurs
  - Partager les données entre plusieurs applications
  - Utiliser des événements haut niveau pour les applications
- Agrégation & Filtrage des données générées par les lecteurs RFID pour les transformer en événements, utilisés par les applications métier.
- Rapport ALE (Format XML)
  - Contient les informations associées aux tags RFID détectés durant un cycle de lecture.
  - et éventuellement d'autres données dans l'élément <extension> (current sensors, sensor histories, tag data, réponses d'un usager d'un téléphone NFC, …)



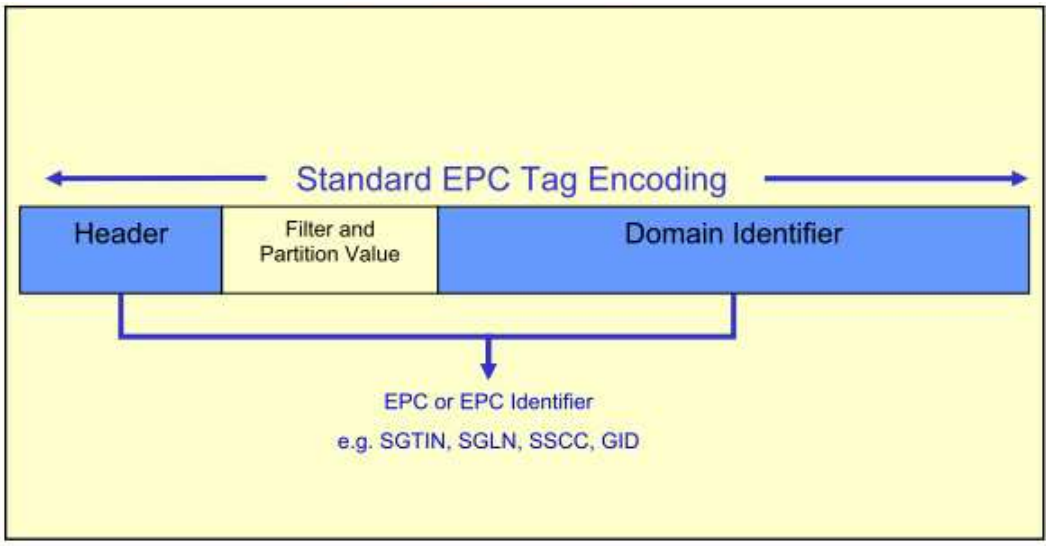
- Sur quel lieu lire ?
  - Sur la porte n°3 de l'entrepôt
- Accumuler les données combien de temps ?
  - Temps absolu
  - Utilisation de triggers
- Comment filtrer les données ?
  - Seulement les tags des palettes, seulement les tags des produits...
- Comment grouper les produits ?
  - Par entreprise, par produit, par tag, ...
- Quels tags m'intéressent ?
  - Tous, les nouveaux, les disparus
- Ai-je besoin de tous les tags ou juste le nombre de tags lus ?

Electronic Product Code – principes

Structure d'un TAG RFID :

CRC	EPC	Mot de passe

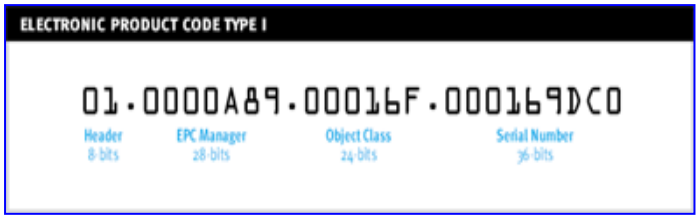
Tous les EPC contiennent un entête qui identifie la méthode d'encodage utilisée. Cette méthode détermine la longueur, le type, et la structure de l'EPC. Un EPC contient souvent un n° de série qui permet d'identifier de façon unique un objet.

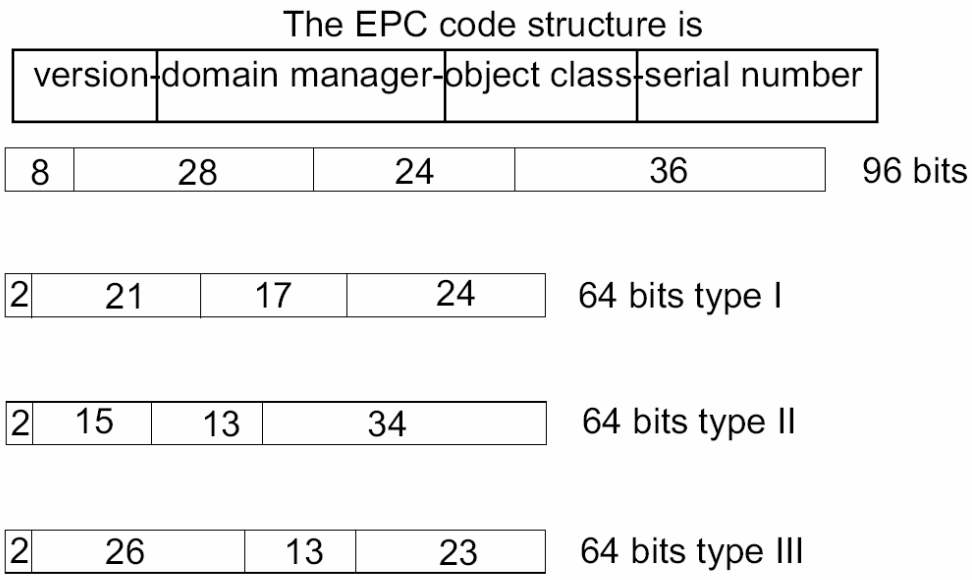


EPCglobal Tag Data Standards Version 1.4, Ratified on June 11, 2008

Header: identifie le n° de version d'EPC: **Type I, Type II, Type III, Type IV.**

Electronic Product Code - structure





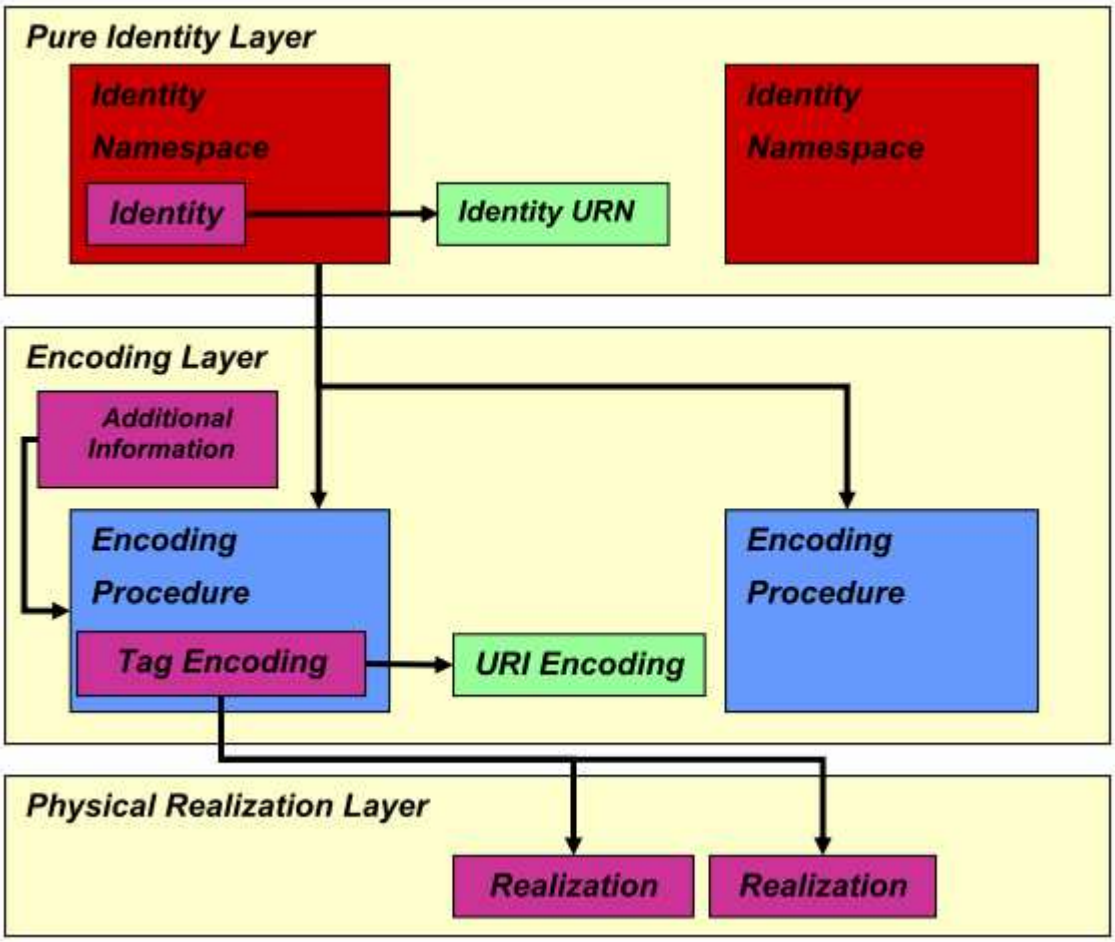
EPC Manager: le fabricant du produit auquel l'EPC est attaché

Object Class: type du produit, souvent le SKU (Stock Keeping Unit)

Serial Number: n° de série

Electronic Product Code – couches de noms

L'identification est structurée en 3 niveaux dans EPCglobal :



Defined Identity Namespaces, Encodings, and Realizations.

On retrouve les 3 couches de noms : nom interne, nom externe, et nom intermédiaire.

Electronic Product Code – types de noms

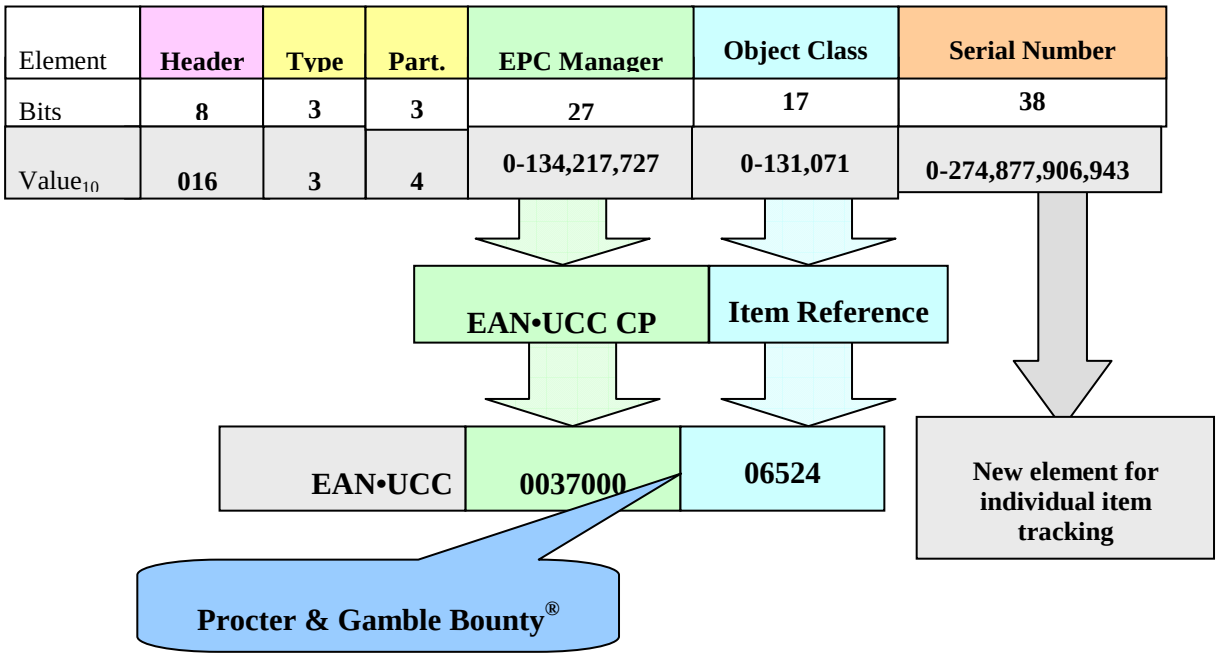
- **Pure identity :** nom abstrait ou numéro pour identifier un objet, elle ne varie pas quelle que soit la technologie utilisée pour attacher l'objet et son identité (on pourrait passer d'un tag RFID à un code barre suite à une transformation de l'objet)

- **Encoding Identity :** nom intermédiaire qui combine "Pure Identity" et syntaxe de l'objet final

- **Realization Identity :** c'est la façon dont l'identité encodée est écrite sur l'objet (code barre, dans la mémoire d'un tag, ou autre technologie)

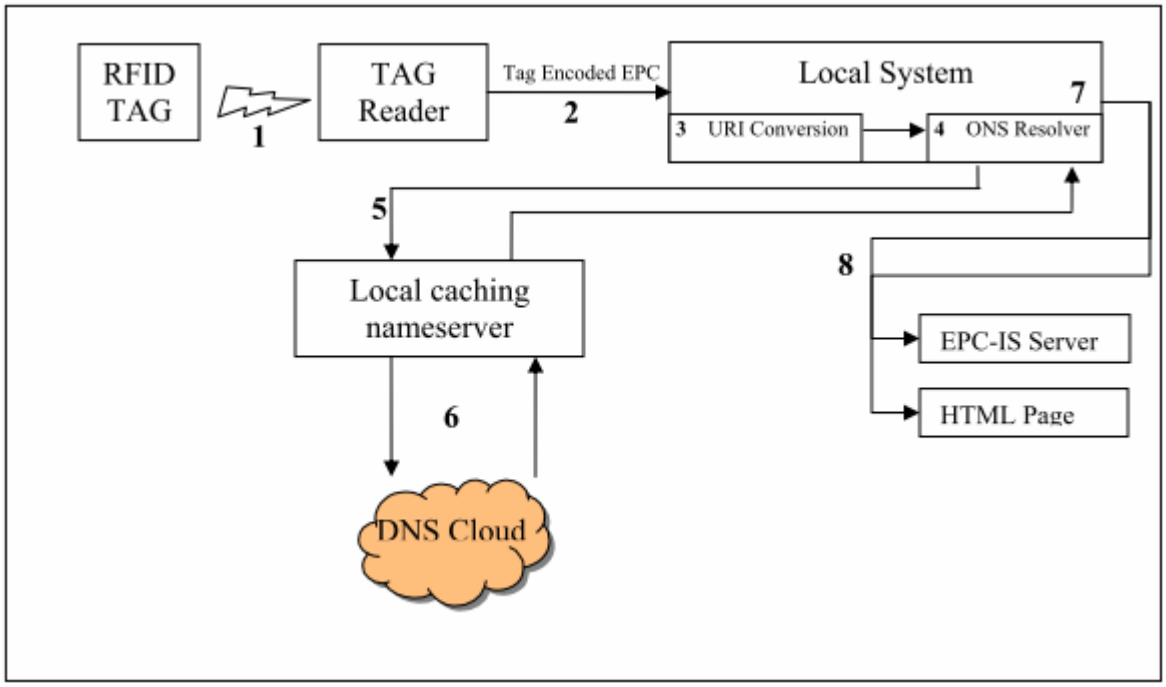
Electronic Product Code – Exemple

EPC-96: dans un EPC



ONS - Object Name Service

L'ONS s'appuie sur le DNS pour trouver un serveur EPCIS.



ONS query

ONS - exemple

1. Une suite de bits représentant un EPC est lue depuis un tag RFID 64-bit.

Exemple: <10 000 00000000000000 0000000000000011000 00000000000000000110010000>

2. Le lecteur de tags envoie cette suite vers un serveur local.

3. Le serveur local convertit cette suite en identité pure au format URI.

Exemple: urn:epc:id:sgtin:0614141.000024.400

4. Le serveur local présente cette URI à la résolution par le serveur ONS local.

5. Ce serveur convertit l'URI en nom de domaine et génère une requête DNS pour un enregistrement (DNS record) de type NAPTR (Naming Authority PoinTeR) pour ce domaine. Le numéro de série a été enlevé, il ne sera utilisé que lorsque le service EPCIS correspondant sera trouvé.

Exemple: 000024.0614141.sgtin.id.onsepc.com

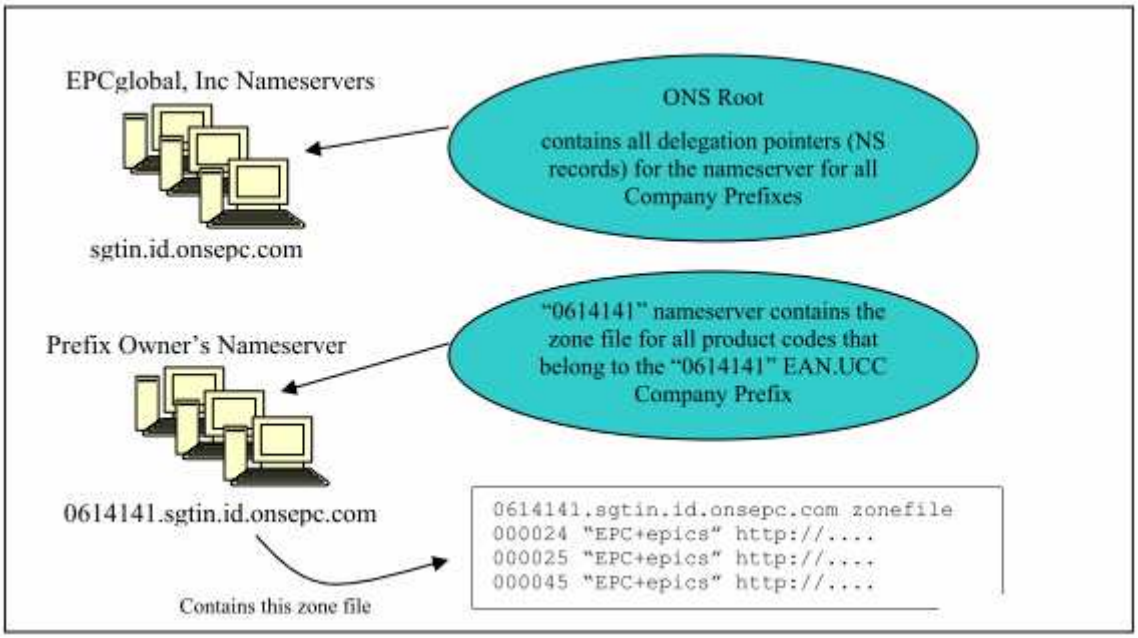
6. L'infrastructure DNS renvoie une série de réponses qui contiennent des URLs qui référencent un ou plusieurs services, par exemple des serveurs EPCIS.

7. Le serveur ONS local extrait l'URL de l'enregistrement DNS et le présente au serveur local.

Exemple: http://epc-is.example.com/epc-wsdl.xml

8. Le serveur local contacte le serveur EPC-IS indiqué dans l'URL pour l'EPC initial

Hierarchie de serveurs de noms



Bilan de la Désignation

Dans un systèmes distribué, les noms doivent être uniques dans le temps et dans l'espace, ils doivent de plus être indépendants de la localisation de l'objet.

On a vu deux mécaniques pour résoudre les noms :

- la résolution de noms par annuaire
- la substitution de noms

La résolution de noms est de loin la plus populaire et la plus simple à mettre en œuvre car elle permet de tenir compte des univers hétérogènes.

La substitution de noms est la plus flexible mais suppose l'adoption par tous les acteurs de la gestion de noms de la même structure de noms.

La protection dans les Systèmes Répartis

(partie figurant à titre d'information pour ceux qui n'auraient pas suivi l'ue RSX112, sécurité réseaux)

Problème général de la protection

Ce qu'on attend du système:

- **secret**: pouvoir conserver des données secrètes
- **confidentialité**: données fournies dans un but ne sont utilisées que dans ce but
- **authenticité**: données indiquées comme venant de X viennent bien de lui
- **intégrité**: données mémorisées par le systèmes ne peuvent pas être altérées

Ce que peut tenter un intrus:

- **inspection**: lecture des fichiers, espionnage des paquets
- **fuite**: obtenir d'un complice des données confidentielles
- **déduction**: reconstruire un information sensible depuis un ensemble d'informations
- **déguisement**: se faire passer pour un utilisateur autorisé.

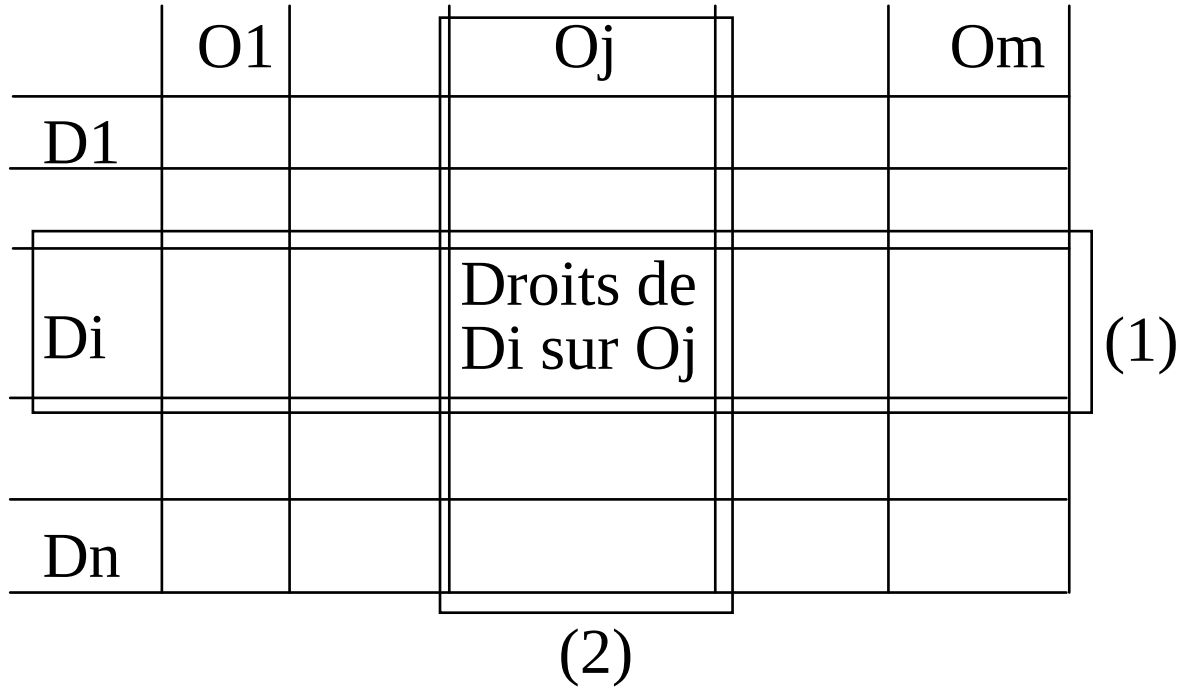
Domaine et matrice d'accès

domaine = ensemble d'objets accessibles, ainsi que les opérations (droits) autorisées sur ces objets

un processus s'exécute dans un domaine, mais il peut en changer.

Les changements de domaine sont contrôlés

matrice de droits :



(1) liste de droits, ou encore liste de capacité

(2) liste de contrôle d'accès

Où ranger les droits?

Une liste de contrôle d'accès, pour chaque objet

- près de l'objet, donc dans le serveur qui le gère (solution retenue pour AFS)
- problème: authentification du demandeur, et du domaine dans lequel il s'exécute

Une liste de capacité, pour chaque domaine

- près du processus, donc chez le client
- problème: falsification de la capacité par le client, ou par le réseau

Exemple d'Amoeba

Les portes

une porte est un couple de deux nombres assez grands, 64 bits, (P, G), reliés par une fonction non inversible F, connue de tous, de telle sorte que P = F(G)

G est privé, et n'est connu que du serveur qui reçoit les messages sur cette porte "get (G)"

P est connu de ceux qui peuvent envoyer des messages sur cette porte "put(P)"

la connaissance d'une porte est synonyme du droit d'envoyer des messages sur la porte

La protection suppose qu'il est impossible de contourner la transformation P = F(G):

- un intrus ne peut s'approprier un message envoyé sur la porte, car il ne connaît pas G
- le serveur répond par le biais d'une porte du client; un intrus ne peut s'approprier la réponse

Les capacités

les objets sont protégés par des capacités

Une capacité a 4 champs:

64 bits	32 bits	32 bits	64 bits
porte	objet	droits	contrôle

la porte est celle du serveur qui gère l'objet

le champs objet n'a de sens que pour le serveur; il identifie l'objet concerné

le champs droit contient 1 bit par opération

le champs contrôle permet au serveur de vérifier la validité de la capacité:

- à la création de l'objet, le serveur tire un nombre aléatoire N, qu'il conserve dans ses tables
- le champs contrôle = F(N xor droits), où F est une fonction non inversible.

Note: la révocation (pour tous) est simple: il suffit de tirer un nouveau nombre aléatoire N!

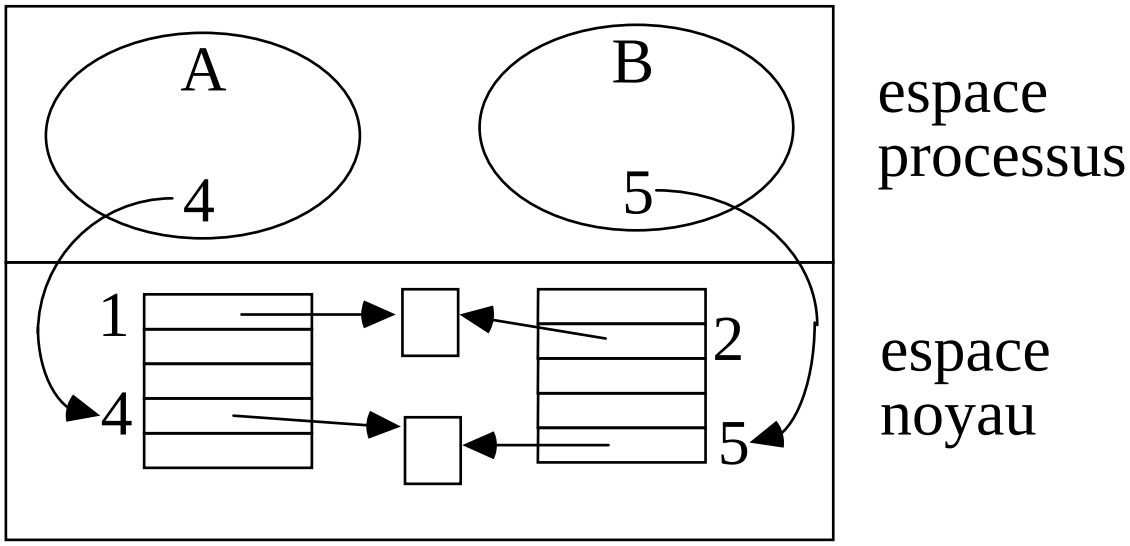
Exemple de Mach

les portes sont protégées par des capacités

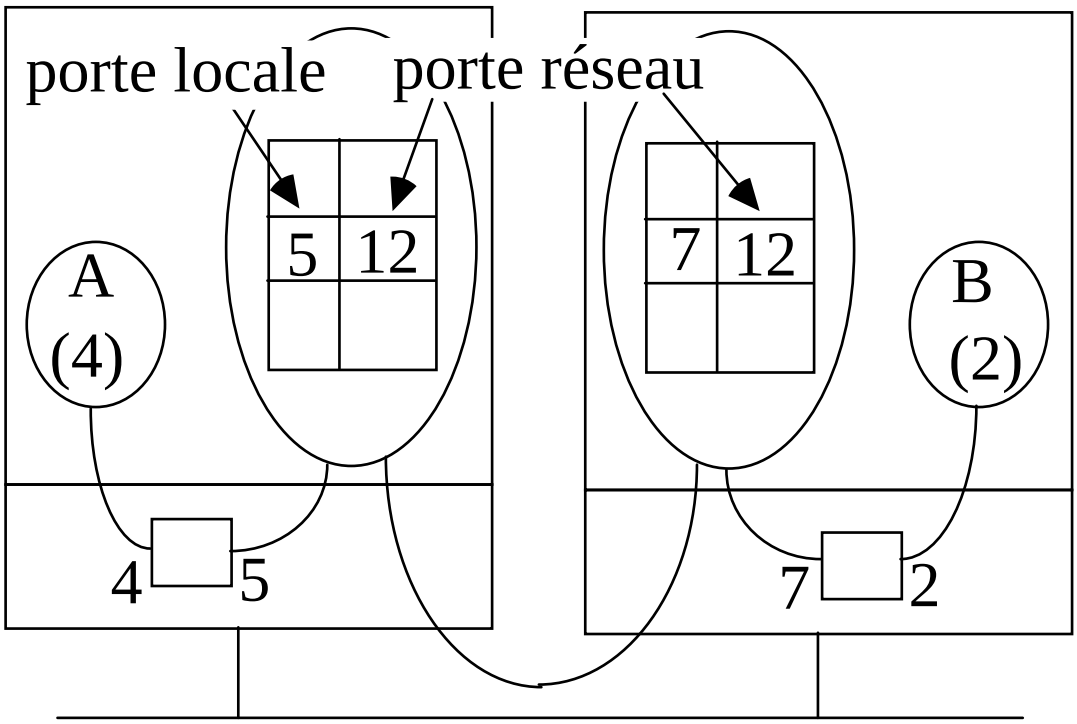
les capacités ne sont pas accessibles aux processus, qui les désignent par un nom local

droits: recevoir, envoyer, envoyer-une-fois

sur un site:



Entre sites:



On passe par l'intermédiaire du serveur de message réseau, qui assure la correspondance

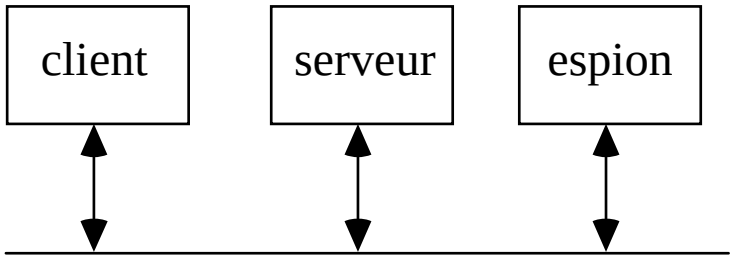
porte mandataire<->porte réseau

le transfert d’une capacité entre processus fonctionne ainsi:

- formatage du message pour indiquer qu’un emplacement du message est un nom local de capacité
- émission du message; le système remplace le nom local par la capacité...
- réception du message; le système range la capacité dans l’espace protégé, lui attribue un nom local et la remplace dans le message par ce nom

les messages sur le réseau peuvent être cryptés si nécessaire

Notion de sécurité



L’espion peut se faire passer pour le client auprès du serveur,

L’espion peut se faire passer pour le serveur auprès du client,

L’espion peut observer l’échange client-serveur

Nécessité d’authentification mutuelle et de conservation du secret

Cryptage par clé privée

La même clé est utilisée pour le cryptage et le décryptage

Message chiffré = cryptage(Message, Clé)

Message = décryptage(Message chiffré, Clé)

On note parfois $\{M\}_K$ le message M crypté par la clé K

Exemple DES (Data Encryption Standard)

utilisation de clés de 56 bits pour coder des blocs de 64 bits

fonction de cryptage/décryptage simple, efficace, peut être câblée,

pratiquement impossible de trouver la clé

Cryptage par clé publique clé privée

cryptage et décryptage utilisent des clés différentes

Message chiffré = cryptage (Message, Clé1)

Message=décryptage(Message chiffré,Clé2)

Clé-1 privée => qqun qui connaît Clé-2 trouve M

Clé-2 publique => seul celui qui a crypté connaît Clé-1

On notera $\{M\}_K$ le message M crypté par la clé K.

On note souvent K, la clé privée et K^{-1} la clé publique.

Exemple RSA (Rivest, Shamir, Adleman)

- basé sur la difficulté de décomposer les grands nombres (150 à 300 chiffres) en facteurs premiers
- fonction de cryptage/décryptage longues et coûteuses
- connaissant l'une des clés, il est pratiquement impossible de trouver l'autre
- les deux clés fonctionnent dans n'importe quel ordre

Hypothèses générales

1 le cryptage est parfait: décryptage impossible sans connaissance de la clé

2 il est possible de savoir si un message a été décrypté avec la bonne clé
Ce n'est pas le cas si le message est une suite de bits non prévisible.

3 $\{M\}_K = \{M'\}_{K'}$ => M = M' et K = K'

4 si un message est décrypté avec la bonne clé, alors quelqu'un l'a crypté avec la clé ad-hoc

Remarques :

Si est K_A la clé privée connue de A, et K_A^{-1} la clé publique

1 $\{M\}_{K_A}$ vient de A, qui connaît M, et est compréhensible par tous (signature),

2 $\{M\}_{K_A^{-1}}$ est crypté par n'importe qui, mais n'est compréhensible que par A,

3 $\left\{\{M\}_{K_A}\right\}_{K_B^{-1}}$ vient de A, qui connaît M, et n'est compris que par B.

4 $\left\{\{M\}_{K_B^{-1}}\right\}_{K_A}$ vient de A et n'est compris que par B. Rien ne dit que A connaît M: le cryptage de M a pu être fait par n'importe qui!

Principe général: ne jamais crypter quelque chose que l'on ne connaît pas.

Serveur d'authentification

Il est impossible que tout couple de communicants potentiels se partage statiquement une clé secrète, car il y en a trop!

L'utilisation systématique de clés publiques est trop coûteuse (cryptage est 1000 à 5000 fois plus long).

Solution: si deux processus veulent partager une clé secrète, ils font appel à un serveur:

1 Ils ont tous les deux confiance en lui.

2 Le serveur garantit à chacun l'identité de l'autre, ce qui implique qu'il est sûr de l'identité de chacun!

3 Le serveur choisit la clé secrète, et garantit qu'ils sont les seuls à la connaître.

Le protocole entre processus et serveur se fait initialement par le biais de clés publiques:

1 je suis certain que le serveur a fourni la clé, car elle est cryptée par une clé secrète qu'il est seul à connaître, et dont je suis seul à connaître la clé de décryptage

2 je suis certain que cette clé a été fournie récemment car on utilise un mécanisme d'estampilles pour garantir la fraîcheur

Protocole d'authentification à clé privé avec serveur de Needham et Schroeder :

1. A -> S : N_A,A,B (N_A change à chaque session)

2. S -> A : $\{N_A,B,K_{AB},\mathbf{\{K_{AB},A\}_{K_B}}\}_{K_A}$

3. A -> B : $\mathbf{\{K_{AB},A\}_{K_B}}$

4. B -> A : $\{N_B\}_{K_{AB}}$ (N_B change à chaque session)

5. A -> B : $\{ N_B - 1 \}_{K_{AB}}$

Attention, on ne cherche qu'à authentifier les différentes entités, et rien d'autre, on cherche que A est bien A et B est bien B !!!

Protocole d'authentification à clé publique avec serveur de Needham et Schroeder :

Seul le serveur a connaissance des clefs publiques de A et B (K_A^{-1} et K_B^{-1}), les sites n'ont que la clef publique du serveur :

1. A -> S : A,B
2. S -> A : $\{B,K_B^{-1}\}K_S$
3. A -> B : $\{N_A,A\}K_B^{-1}$
4. B -> S : B,A
5. S -> B : $\{A,K_A^{-1}\}K_S$
6. B -> A : $\{ N_A , N_B \} K_A^{-1}$
7. A -> B : $\{N_B \}K_B^{-1}$

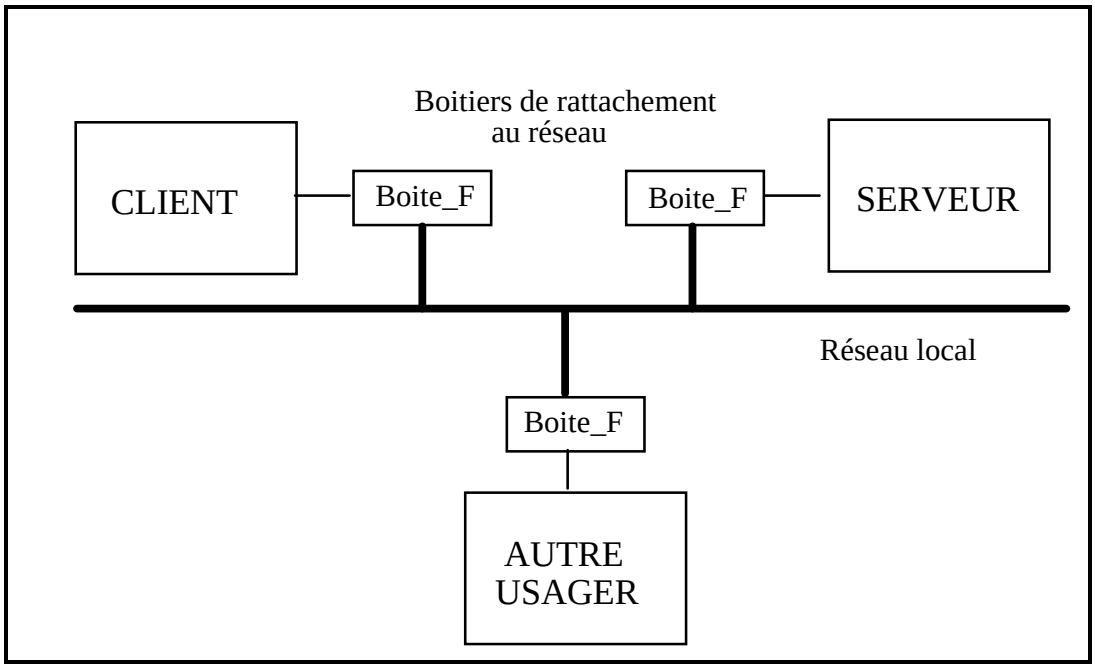
Le protocole n'est pas suffisant :

- l'étape 3 peut être un replay, il faut N_A change à chaque session

- on peut se faire passer pour A dès qu'on s'empare de sa clef publique, le protocole suppose qu'aucun attaquant ne s'est procuré une clef publique, en particulier celle du serveur

Exercice: Etude du système d'authentification AMOEBA (exercice et correction G. Florin)

Ce problème s'intéresse à l'authentification des entités communicantes dans une relation client-serveur du système AMOEBA. On ne considère pas les autres problèmes de confidentialité, d'intégrité, ... Ce système réparti suppose que tous les utilisateurs du réseau de communication d'entreprise ne peuvent communiquer qu'au moyen d'un boitier matériel dédié aux fonctions de sécurité appelé dans la suite boite-F ("F-box").



Dans ce système les adresses de ports ou point d'accès de service de communication (adresses de communication) sont définies sur un grand nombre de bits (ici 48 bits) et tirés aléatoirement dans cet espace.

1 Ce simple choix est déjà une technique de protection contre certaines erreurs ou attaques. Lesquelles?

Pourquoi? (répondez en quelques phrases)

Correction : Le fait d'utiliser des adresses tirées aléatoirement dans un grand espace de numérotation fait perdre la correspondance entre l'adresse et le rôle d'un service. Il est donc plus difficile de déterminer une adresse pour un service donné dont on voudrait usurper l'accès.

D'une part lorsqu'on est un utilisateur bien intentionné mais qui fait des erreurs de programmation et génère des adresses fausses ("par hasard") on peut difficilement (avec une probabilité très faible) tomber sur une adresse correcte et mettre en danger le bon fonctionnement du destinataire.

D'autre part si l'on est un utilisateur mal intentionné le grand nombre d'essais et d'erreurs qu'il faut effectuer dans des attaques "force brute" interdit pratiquement cette forme de piratage.

Il est bien certain que ce choix et l'ensemble de remarques précédentes ne conduit pas à un niveau de sécurité très élevé mais la technique est d'un coût très faible.

On utilise pour les fonctions de sécurité une technique de cryptage au moyen d'une fonction univoque F (fonction à sens unique, "one way function"). Par exemple dans la suite une adresse de port de communication aura en fait deux versions, une version en clair G (connue seulement de l'usager et de la boite_F) et une version cryptée P = F(G) (version qui circule sur le réseau).

2 Rappelez la définition d'une fonction univoque

Correction : Une fonction à sens unique F est une fonction telle que son calcul est facile mais qu'il est très difficile voire impossible de déterminer la fonction inverse F^{-1} lorsque l'on connait F. C'est l'un des éléments essentiels d'un système de mot de passe. On peut utiliser par exemple l'algorithme du DES fonctionnant avec pour clé la valeur X sur laquelle s'applique la fonction. On peut déterminer un couple clé publique et clé privée dans l'algorithme RSA et détruire l'une des clés.

Les boites F ont pour rôle principal de calculer la fonction F selon le mode de fonctionnement suivant. Soit Gs une adresse choisie aléatoirement par un serveur S. Le serveur garde Gs secret et communique Ps=F(Gs) aux clients à qui il souhaite offrir son service. Un client posséde également une adresse Gc (avec Pc=F(Gc)). Il présente à sa boite_F locale pour l'émission une requête de la forme:

put (Ps, Gc, Kc, Dappel)

- Ps est l'adresse de port cryptée du destinataire

- Gc est l'adresse de port de l'émetteur (pour la réponse)

- Kc est un mot de passe (une clé secrète) authentifiant du client.

- Dappel est l'ensemble des données constituant la requête.

La boite_F du client effectue en fait une primitive notée put_F (pour la distinguer de celle du client)

put _F(Ps, Pc, F(Kc), Dappel)

Elle émet sur le réseau un message comportant une adresse client cryptée, une adresse serveur cryptée et la clé du client cryptée.

Le serveur demande la réception des messages à sa boite_F locale en lui présentant une requête:

get (Gs, X, Y, Dappel)

La boite_F du serveur calcule la clé cryptée du serveur Ps=F(Gs) et n'accepte que les messages qui comportent l'adresse cryptée Ps. Le serveur peut alors vérifier la requête du client et récupérer l'adresse X=Pc du client.

3 Comment le serveur peut-il vérifier l'identité du client?

Correction : En fait le serveur dispose de deux moyens de vérification: les adresses de ports et la clé (mot de passe) qui sont relativement redondants pour la fonction d'authentification souhaitée. Ces deux informations circulent cryptées. Le serveur peut donc disposer d'un fichier de mots de passe (ou de nom de ports) cryptés comme dans tout système d'authentification à l'ouverture d'une session en temps partagé. Comme les mots de passe cryptés ont été fabriqués par une boite_F en qui on peut avoir confiance il suffit de faire une comparaison entre la valeur présentée et la valeur stockée.

Le serveur répond ensuite au client par: put(X=Pc,Gs,Ks,Dreponse)

La boite_F du serveur retransmet en fait sur le réseau: put(Pc,Ps,F(Ks),Dreponse)

4 Expliquez pourquoi ce mécanisme interdit à un autre usager que le serveur d'usurper l'identité du serveur S. De la même façon le client est le seul à pouvoir recevoir la réponse. Pourquoi?

Correction : Pour arriver à se faire passer pour le serveur du point de vue de sa boite_F un utilisateur du système n'a pour seule solution que de connaitre Gs qui normalement est une information connue uniquement du vrai serveur S. La seule information qui circule sur le réseau est F(Gs). Si un intrus présente cette valeur à sa boite_F d'interface celle ci utilisera F(F(Gs)) qui n'est d'aucune utilité. La fonction F étant univoque il n'est pas possible de déduire Gs de F(Gs).

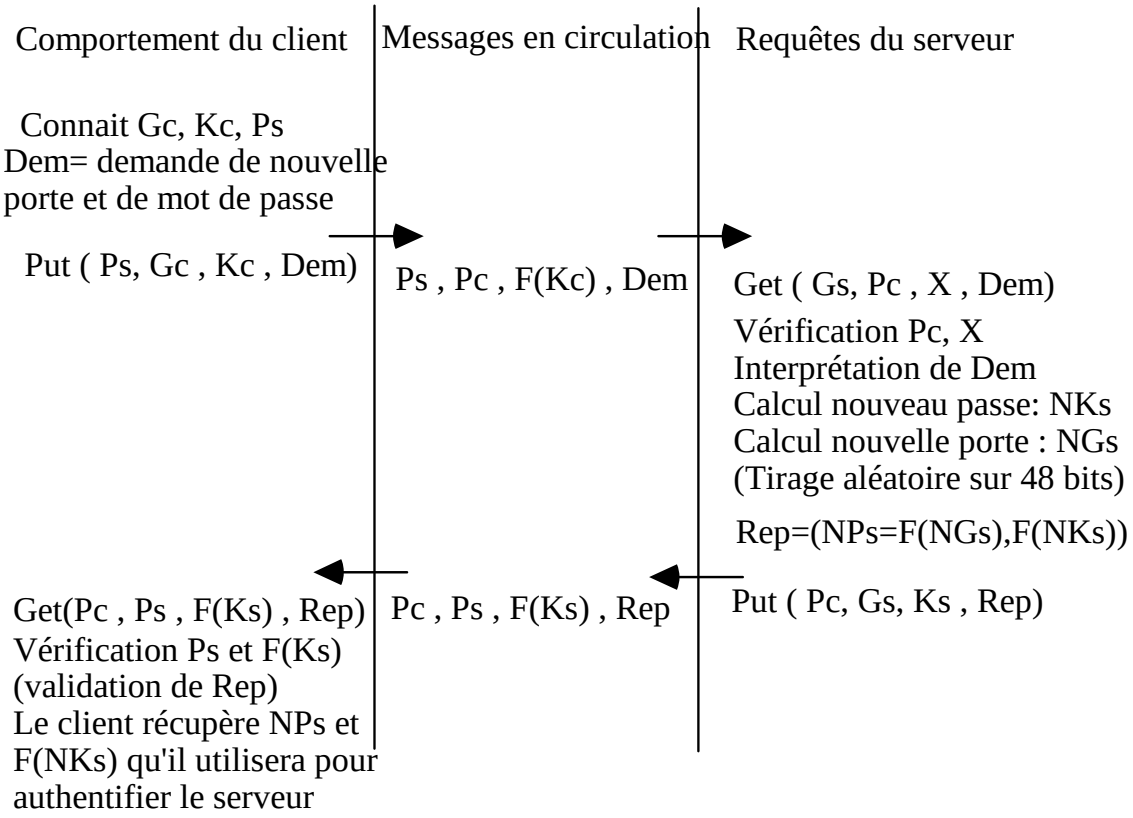
On souhaite maintenant augmenter la sécurité du système en introduisant une obligation de changement périodique des informations de sécurité employées. On s'intéresse donc à la procédure de changement de l'adresse de port de communication et du mot de passe du serveur.

5 Définissez l'échange permettant le changement sécurisé de l'adresse du serveur et de mot de passe du serveur (le client demande et récupère une nouvelle adresse de serveur cryptée NPs et un nouveau mot de passe du serveur).

Correction : Un aspect important du changement d'adresse et de mot de passe est que celui-ci doit se faire sous les anciennes protections. Le client doit pouvoir authentifier le

serveur pour savoir s'il ne s'agit pas d'un serveur déguisé qui répond. Le changement doit prendre effet des la fin d'un échange de changement ce qui signifie qu'il ne peut y avoir de panne (perte de message) pendant cette phase sous peine de ne plus pouvoir communiquer.

Le serveur doit calculer une nouvelle clé et une nouvelle adresse de port. Il doit les communiquer au client par message mais uniquement sous leur forme cryptée car le client ne doit pas pouvoir y accéder en clair. La boite_F ne peut faire ce travail. Elle ne sait le faire que sur les zones adresses et clés du message. On ne peut avoir mis dans ces zones directement les nouvelles adresses et clés car on ne pourrait pas vérifier l'origine de cette modification.



6 On suppose qu'un intrus C' a pu s'approprier illégalement un accès au serveur S (par un moyen non

décrit, apprentissage des adresses et clé d'accès au serveur, usurpation d'identité d'un client C accrédité). Montrez comment le changement d'adresse et de mot de passe du serveur permet de détecter cette situation.

Correction : La situation d'usurpation décrite implique qu'un client accrédité était en fonctionnement correct (connexion correctement ouverte, procédure de changement de secret déjà opérante) et que suite à une usurpation deux clients ont simultanément accès à un serveur S (le vrai et le faux). La procédure avec changement implique que périodiquement les adresses de ports et les mots de passe sont changés. Les échanges sont point à point et lors d'un changement un seul site est demandeur du changement et est avisé de ce changement (le vrai ou le faux peu importe). Ensuite l'un des deux continue sur les anciennes valeurs d'accès et le serveur a la possibilité de détecter une anomalie pour faire cesser tous les transferts immédiatement.

7 La sécurité de cette solution repose essentiellement sur l'usage impératif et unique des boite_F par tous les sites. Si un intrus arrive à se brancher directement sur le réseau que doit-il faire pour utiliser illégalement un serveur? (2,5 points)

Correction : Un intrus peut en écoutant le réseau obtenir la totalité des informations qui circulent dans des messages corrects. En particulier les adresses cryptées des clients et des serveurs les clés cryptées des serveurs.

Il peut donc fabriquer sans difficultés des messages qui sont corrects (la technique est en fait pratiquement celle de

reémission"replay") sur la partie contrôle des messages).

Le problème posé par les adresses (sur un grand espace d'adressage déjà souligné) est qu'il faut que l'intrus découvre parmi tout le traffic des messages qui ont une structure correspondante à celle du serveur qu'il veut atteindre. Il lui faut donc une connaissance du protocole du serveur pour lui permettre de reconnaitre en filtrant le traffic des messages correspondant au serveur qu'il veut atteindre.

Par la suite lorsqu'il a trouvé le bon serveur il doit également s'adapter en permanence aux changements d'adresse.

Notons enfin qu'il ne peut pas faire émettre ses messages par une boite_F dans la mesure ou il en aurait une aussi car il ne connaît pas les clés secrètes des usagers.