Utilisation de WebDAV dans ESUP-Portail

Thomas Bellembois
Thomas.Bellembois@univ-rennes1.fr

Raymond Bourges
Raymond.Bourges@univ-rennes1.fr

Yohan Colmant
Yohan.Colmant@univ-valenciennes.fr

Résumé

Ce document présente l'utilisation du protocole WebDAV au sein du projet ESUP-Portail. Une première partie présente le projet, les raisons du choix de ce protocole, le contexte de mise en œuvre et les deux composants principaux du stockage ESUP-Portail : canal stockage et serveur WebDAV. La seconde partie décrit le protocole WebDAV qui offre une gestion de ressources distante et intègre les notions de métadonnées, verrous et droits d'accès. La partie principale de ce document se focalise sur l'aspect pratique et technique du stockage en détaillant les développements effectués. Côté serveur WebDAV, l'accent est mis sur la couche d'authentification multiple, la gestion de groupes et de quotas. Le canal stockage, quant à lui, offre une interface conviviale permettant la gestion simple de ressources sur le serveur (dépôt, suppression, déplacement…) mais offre aussi des fonctionnalités de partage et de contrôle d'accès. Une conclusion expose les perspectives du projet ESUP-Portail ainsi que les améliorations à apporter.

Mots clefs

WebDAV, ACP, ESUP-Portail, Portail, uPortal, Slide, Shibboleth, inJAC, métadonnée

1 Introduction

1.1 ESUP-Portail

ESUP-Portail [1] est un consortium d'universités Françaises dont le but est de mettre à disposition en Open Source un Espace Numérique de Travail d'accès intégré aux services pour les étudiants et le personnel de l'enseignement supérieur.

La solution ESUP-Portail est à ce jour choisie par plus de 50 établissements (universités, IUFM et grandes écoles). Elle est actuellement en production sur de nombreux sites, utilisée par des dizaines de milliers d'utilisateurs.

Tout ou partie des composants qui composent la solution ESUP-Portail sont utilisés ou en cours d'évaluation au-delà du périmètre initialement prévu de l'enseignement supérieur français. Le portail est par exemple testé par des rectorats et des composants comme le helpdesk [2] sont mis en œuvre en Suisse ou au Japon.

1.2 Démarche

Parmi les problématiques abordées par ESUP-Portail (Portail, SSO, accès aux applications métier, etc.), il en était une importante concernant le stockage.

Nous reviendrons dans cet article sur les raisons du choix WebDAV et les spécifications du service. Nous nous baserons pour cela sur des articles précédemment publiés [3].

Nous aborderons aussi la partie cliente qui est très importante car elle offre des fonctionnalités intéressantes de délégation de l'administration du service. Elle permet d'accéder à de nombreux serveurs de stockage présents au sein d'un établissement (Cf. Figure 1).

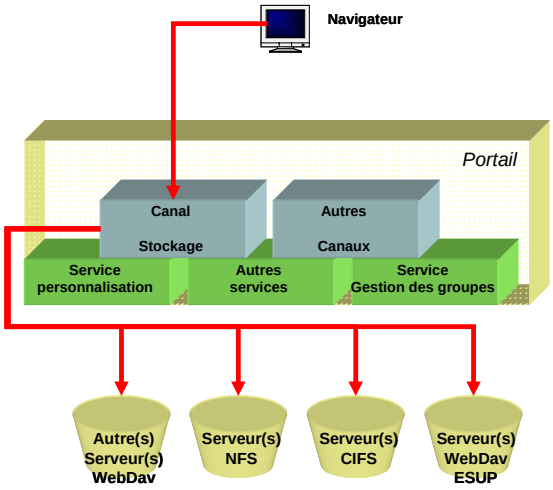

Figure 1 – *Le canal stockage accède à tous les serveurs*

D'une façon générale, cet article décrit les réalisations effectuées et donne des informations sur les protocoles et les standards utilisés.

Nous verrons enfin comment ce service de stockage ESUP-Portail est mis à contribution pour offrir une solution de CMS.

1.3 Pourquoi WebDAV

WebDAV [4] a été retenu. Basé sur le protocole HTTP, il est naturellement bien adapté aux utilisateurs de l'Internet. C'est une norme largement mise en application. Il existe de nombreux clients et des bibliothèques sont disponibles dans de nombreux langages de programmation. Néanmoins, certaines extensions du protocole WebDav, présentées dans cet article, ne sont mises en œuvre que dans le projet ESUP-Portail.

Il existe d'autres protocoles de partage de systèmes de fichiers sur le réseau comme NFS ou CIFS. Ces derniers sont pris en charge dans le module d'accès au stockage proposé dans le portail ESUP-Portail mais n'ont pas été retenus pour la partie serveur.

Il nous fallait une solution transverse Unix/Windows, qui puisse facilement être accessible localement comme à distance, par un composant du portail comme directement depuis les postes de travail des utilisateurs. De plus, WebDAV est bien adapté à de très grandes populations.

1.4 Serveur WebDAV

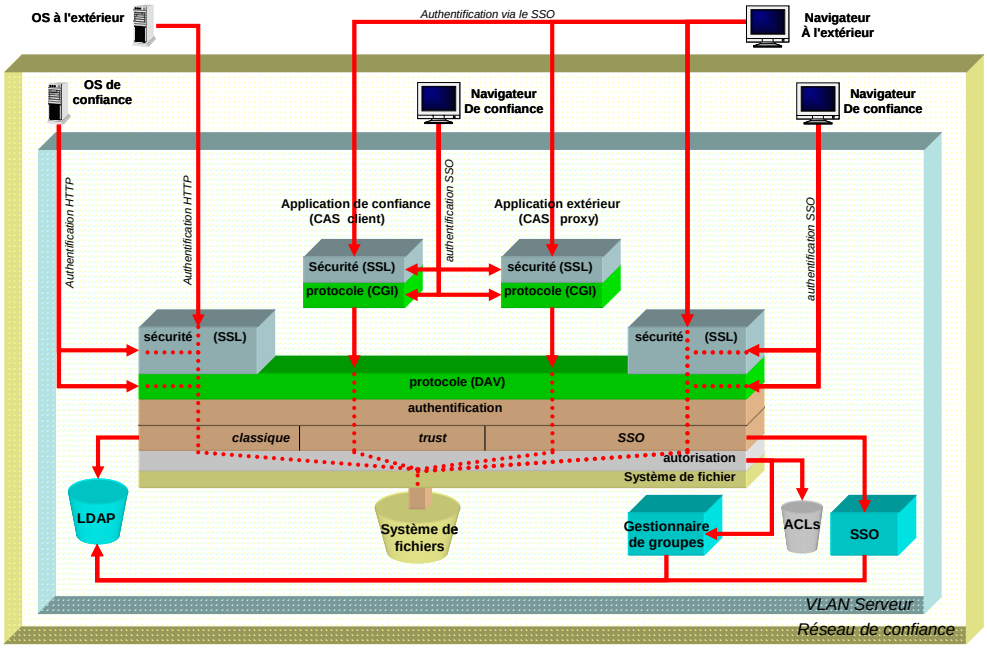

Figure 2 - *Architecture de l'espace de stockage [5]*

Plusieurs choses sont intéressantes dans ce schéma (Cf. Figure 2).

On peut accéder au service :

- Directement depuis un **système d'exploitation**
- Depuis un **navigateur web** et ce de deux façons différentes. Soit directement (le serveur WebDAV répondant comme un serveur HTTP classique), soit via une application spécialisée comme par exemple le canal Stockage ESUP-Portail. Pour ces deux accès il est possible d'utiliser le mécanisme de SSO.

Les accès peuvent se faire en HTTP ou en HTTPS. C'est notamment très utile pour les accès faits depuis le **système d'exploitation** puisque le SSO ne peut pas être utilisé et que l'on se connecte avec un mot de passe classique qu'il convient de protéger.

La couche d'identification est multiforme. Il est possible d'utiliser une identification classique, basée sur LDAP, avec nom d'utilisateur et mot de passe, une identification SSO ou une identification *trusted* où la confiance est déportée dans l'application cliente (généralement compatible avec le SSO).

Le contrôle d'accès utilise les ACL conformément au protocole ACP [6]. Le serveur référence des groupes externes (ceux du portail en l'occurrence).

Afin d'arriver au résultat souhaité, nous ne sommes pas parti de rien mais avons adapté le produit *Open Source* Slide [7] de Apache dans sa version 2.1. Slide est le premier serveur WebDAV à supporter le protocole ACP qui permet de positionner des contrôles d'accès fins sur des ressources WebDAV.

1.5 Canal Stockage pour uPortal

Le canal Stockage (Cf. Figure 3) est le moyen privilégié pour accéder aux espaces disques mis à disposition des utilisateurs.

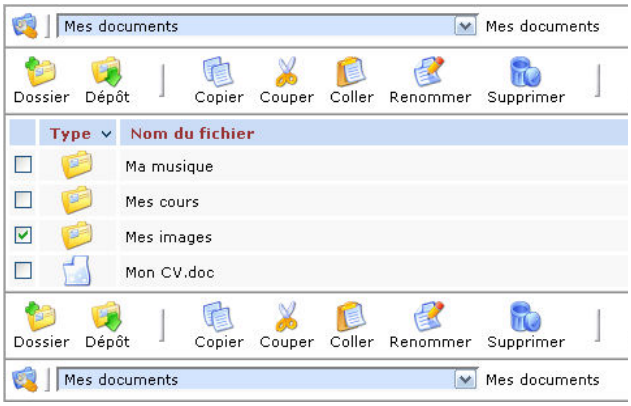

Figure 3 - *Interface utilisateur du canal stockage*

Le canal est intégré dans le portail ESUP-Portail. Il permet d'accéder à des espaces personnels ou partagés. Il travaille aussi bien avec le serveur WebDAV ESUP-Portail, tout autre serveur WebDAV ou des serveurs CIFS ou NFS.

Ceci permet d'offrir un point d'accès unique à tous les espaces de fichiers de l'utilisateur qu'il soit à l'intérieur ou à l'extérieur de l'université. A noter qu'il est possible de prendre un document ou un répertoire d'un serveur pour le déposer sur un autre serveur de technologie différente. L'opération est transparente pour l'utilisateur.

Sur un serveur WebDAV ESUP-Portail il est possible d'administrer directement depuis le canal les droits d'accès à des espaces partagés. Il est aussi possible de déléguer ces droits d'administration. Le service d'espace disque partagé est suffisamment flexible pour s'adapter à la taille d'un établissement.

2 Protocole et spécifications

2.1 WebDAV

WebDAV[1] est une extension du protocole HTTP, défini par plusieurs groupes de travail de l'IETF[2]. Normalisé depuis 1999, WebDAV permet de faciliter la gestion de fichiers à distance. Il permet de récupérer, déposer et publier des ressources (fichiers et/ou dossiers) rapidement et simplement. L'objectif principal de WebDAV étant de rendre possible à la fois la lecture et l'écriture de fichiers au travers du web.

2.1.1 Le contrôle des ressources

Dans WebDAV, une ressource correspond à tout objet pouvant être identifié par une URI. Une ressource peut alors être un fichier ou une collection.

Une **collection** est une ressource composée d'un ensemble d'URI, celles des membres qui la composent. Elle peut être considérée comme l'**équivalent d'un répertoire** dans un système de fichiers classique.

Le contrôle distant de ces ressources (par un client WebDAV) se fait via de nouvelles méthodes HTTP utilisant XML comme format de requête et de réponse.

Les méthodes sont `MKCOL`, `GET`, `POST`, `DELETE`, `PUT`, `COPY`, `MOVE`.

Nous pouvons souligner que, dans WebDAV, la notion de ressource est prépondérante. En effet, en plus des fichiers et des collections, les utilisateurs et les groupes d'utilisateurs sont également des ressources. Chaque utilisateur ou groupe correspond à une ressource, identifiée par une URI et comportant elle-même des métadonnées (Cf. 2.1.2). Un exemple d'URI pour le groupe `local.12` est `/slide/roles/local.0/local.12` où `local.12` et un sous-groupe de `local.0`.

2.1.2 Les métadonnées

WebDAV permet la représentation de l'information relative aux ressources via des propriétés (métadonnées), exprimées à l'aide d'éléments XML. Utilisant les *namespaces* XML, leur nom est par conséquent unique.

Il existe différents types de métadonnées :

Métadonnée vivante (*Live property*)

> Propriété calculée par le serveur. Ex : `getcontentlength`.

Métadonnée morte (*Dead property*)

> Propriété pouvant être fixée librement par le client.

La récupération d'une métadonnée sur une ressource se fait via la méthode `PROPFIND`.

La modification d'une métadonnée se fait via la méthode `PROPPATCH`.

En ce qui concerne la modification, il existe deux types de métadonnées :

Métadonnée protégée

> Une métadonnée protégée ne peut pas être modifiée par une requête `PROPPATCH` mais est modifiable par une autre méthode. Exemple : la métadonnée `owner` qui représente le propriétaire d'une ressource. Celle-ci est placée par le serveur lors d'une requête `PUT` ou `COPY`.

Métadonnée non protégée

> Une métadonnée de ce type peut à la fois être modifiée par le serveur et par tout client via une requête `PROPPATCH`. Exemple : la métadonnée `displayname` qui stocke le nom de la ressource qui est affiché à l'utilisateur.

[1] Web-based Distributed Authoring and Versioning

[2] Internet Engineering Task Force

2.1.3 Le verrouillage

Il est possible de sérialiser les accès aux ressources. En utilisant un verrou, WebDAV garantit qu'une ressource en cours de modification ne peut pas être modifiée par un autre client.

Les méthodes permettant ce contrôle sont les suivantes :

LOCK

Méthode qui verrouille la ressource. Elle crée le verrou et l'applique sur la ressource. Notons que le verrou est placé sur la ressource elle-même, non sur l'URI qui mène à la ressource. Ceci évite les incohérences lorsque des ressources peuvent être accessibles via différentes URI, dans le cas de redirections par exemple.

UNLOCK

Méthode qui déverrouille la ressource précédemment verrouillée.

2.2 ACP

ACP [8] (Access Control Protocol) est une extension du protocole WebDAV fournissant un mécanisme de contrôle d'accès interopérable entre serveurs WebDAV.

Ce mécanisme contrôle les accès aux ressources. A chaque ressource est attachée une liste d'autorisations pour un ensemble d'utilisateurs ou groupes.

2.2.1 Termes utilisés

Afin de mieux appréhender cette section, voyons les principaux termes qui y sont utilisés :

principal

Un *principal* peut être distinctement un acteur humain ou informatique qui tente d'accéder à une ressource sur le réseau. Dans les autorisations, le *principal* peut être à la fois un utilisateur ou un groupe d'utilisateurs. Parmi les groupes, il peut également y avoir une hiérarchie de sous-groupes.

Nous pouvons noter qu'il existe quelques *principals* spéciaux :

unauthenticated

Désigne toutes les personnes non authentifiées qui tentent d'accéder à une ressource.

authenticated

Désigne toute personne authentifiée qui tente d'accéder à une ressource.

all

Agrégation de tous les *principals* (utilisateurs, groupes, *unauthenticated*, etc.).

self

L'utilisateur courant est reconnu à ce niveau uniquement si la ressource est un *principal* et que celui-ci est l'utilisateur courant. Une autre possibilité est que ce *principal* soit un groupe et que l'utilisateur courant fasse partie de ce groupe.

groupe

Un groupe est un *principal* qui représente un ensemble d'autres *principals*.

privilège

Un privilège permet de contrôler l'accès à un ensemble particulier d'opérations HTTP sur une ressource. Les actions possibles pour ces privilèges sont détaillées dans la partie 2.2.2.

ACL (access control list - liste de contrôle d'accès)

Une ACL est une liste d'ACE qui définit le contrôle d'accès sur une ressource. Les ACL sont des métadonnées particulières, vivantes et non protégées.

ACE (access control element – élément de contrôle d'accès)

Un ACE permet d'autoriser (grant) ou de refuser (deny) certains privilèges sur une ressource pour un principal donné.

ACE hérité

Un ACE hérité est un ACE qui est dynamiquement partagé avec une autre ressource référencée par l'URI contenue dans l'élément `href`. Il ne peut être modifié que sur la ressource référencée.

2.2.2 Les privilèges

A chaque privilège correspond une action particulière. En effet, il est possible de donner différents droits au travers de différents ACE. De cette manière, nous pouvons donner une autorisation en lecture sur une ressource pour un usager donné, mais lui interdire l'accès en écriture.

Les différentes actions sont `read`, `write`, `write-properties`, `write-content`, `unlock`, `read-acl`, `read-current-user-privilege-set`, `write-acl`, `bind`, `unbind`, `all`.

Notons que certains privilèges sont des agrégations d'autres privilèges. Il n'existe pas une liste exhaustive des agrégations, mais la RFC 3744 spécifie les règles à respecter par les différentes implémentations des serveurs WebDAV.

2.3 Quotas

Le protocole WebDAV permet de définir des limitations de quotas. La RFC est encore au statut de *draft* [9] et définit les métadonnées et comportements des serveurs supportant les quotas.

Cette gestion de quotas repose sur deux métadonnées *vivantes protégées* (Cf. 2.1) nommées `DAV:quota-available-bytes` et `DAV:quota-used-bytes` définies sur des collections (répertoires). La définition de ces deux métadonnées repose étroitement sur la définition des quotas dans la spécification NFS [10].

`DAV:quota-available-bytes` indique l'espace disponible sur la collection en octets. Si celle-ci s'approche de zéro, le client peut s'attendre à se voir refuser toute allocation de ressource supplémentaire. `DAV:quota-used-bytes` indique l'espace actuellement utilisé par la collection et ses sous collections (ie. par l'ensemble des fichiers et sous répertoires). Ces deux métadonnées vont de paire et ne devraient normalement pas apparaître l'une sans l'autre. Les métadonnées associées aux ressources peuvent être comptabilisées ou non.

Le dépôt d'une ressource entraînant un dépassement de l'espace disponible doit être refusé par le renvoi au client d'une réponse HTTP portant le code erreur 507 (*insufficient storage)*. Par contre, le remplacement d'une ressource existante doit être possible (si la nouvelle ressource est de taille égale ou inférieure).

Le deux métadonnées de quotas peuvent être récupérées par le client via la méthode WebDAV `propfind <DAV:propname>` mais ne doivent pas être retournées lors d'une requête `propfind <DAV:allprop>`. Si une ressource n'a pas de quotas ou un quota infini, le serveur peut choisir de ne pas retourner la métadonnée `DAV:quota-available-bytes` mais la spécification recommande de retourner une valeur appropriée comme par exemple la quantité d'espace physique disponible.

3 Mise en œuvre

3.1 Gestion des authentifications

Le serveur Slide utilise par défaut un mécanisme d'authentification spécifique à Tomcat et basé sur une base d'utilisateurs internes. Il était intéressant de disposer d'un système d'authentification multiple et indépendant du conteneur d'applications. L'utilisation de filtres J2EE a permis de répondre à ce besoin (Cf. Figure 4).

Un filtre CAS [11] développé par JA-SIG [12] gère l'authentification SSO. Deux filtres LDAP et TRUSTED on été développés en supplément. Le filtre TRUSTED permet l'authentification depuis les serveurs de confiance par simple vérification d'un mot de passe partagé [13].

Un filtre de routage permet de sélectionner un des 3 filtres ci-dessus en fonction de critères paramétrables :

- L'adresse IP du client
- L'agent client (navigateur Web, client JAVA…)
- Le nom d'hôte destinataire
- Les paramètres de la requête
- Le type de requête : HTTP ou HTTPS

Ce filtre apporte une grande flexibilité. Il permet ainsi dans le cadre d'ESUP-Portail d'utiliser une authentification CAS quand le client est un navigateur Web, LDAP quand celui-ci est un dossier web et TRUSTED quand l'utilisateur se connecte par le portail faisant parti d'un réseau sécurité (VLAN des serveurs).

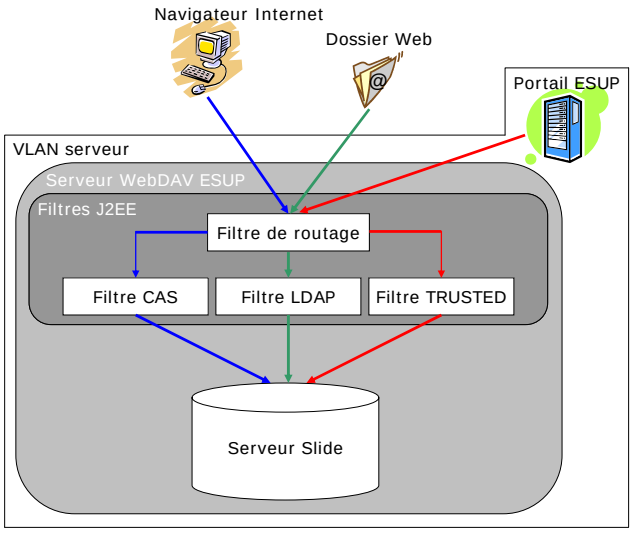

Figure 4 – *Authentification multiple ESUP-Portail*

3.2 Gestion des groupes

3.2.1 Définition des groupes dans Slide

Le serveur Slide intègre la notion de groupes d'utilisateurs. Ces groupes (qui sont des ressources WebDAV - Cf. 2.1) sont visibles sous l'URL `/roles` – cette URL pouvant être changée (ex : `/roles/UFR Math`, `/roles/Informatique`, …). Lister les ressources de l'URL `/roles` permet de connaître les groupes définis sur le serveur.

Une métadonnée nommée `group-member-set`, attachée à chaque groupe, définit les membres du groupe (sous groupes ou utilisateurs).

Le groupe `UFR Math` contient le groupe `licence Math` et l'utilisateur `john`.

Les groupes (et informations associées) peuvent être définis statiquement dans un des fichiers de configuration du serveur, ou récupérés dynamiquement depuis un annuaire LDAP.

3.2.2 *Les stores Slide*

Le serveur Slide offre la notion de *store* permettant de définir la façon dont sont stockées les ressources et métadonnées sur le serveur (Cf. Figure 5). Ainsi il permet d'associer un pattern d'URL à un mode de stockage. L'administrateur peut décider par exemple que toutes les ressources sous l'URL `/files` seront stockées sur un système de fichiers classique, les ressources sous l'URL `/roles` (donc les groupes) dans une base de données SQL et les ressources sous `/users` (donc les utilisateurs) récupérées depuis un annuaire LDAP.

Slide offre des interfaces permettant de développer des *stores* personnalisés.

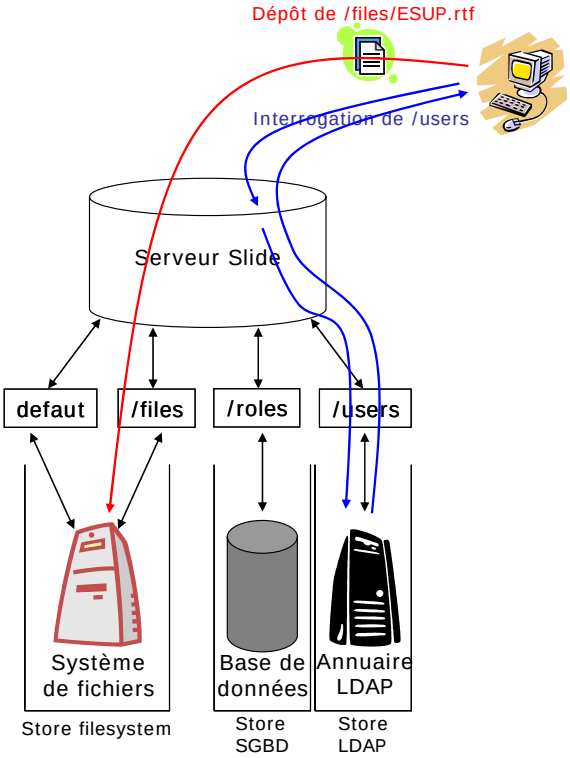

Figure 5 – *Les stores Slide*

3.2.3 *Intégration dans le portail ESUP-Portail*

Le portail possède sa propre base de groupes dynamiques très riche, facilement administrable et configurable.

Nous voulions que le serveur WebDAV ESUP-Portail soit complètement intégré au portail concernant sa gestion des groupes dans un souci de cohérence et de facilité de gestion. Il était inconcevable de créer une base locale copie conforme de la base du portail (Cf. Figure 6).

Nous avons donc développé un *store* permettant de récupérer dynamiquement les groupes du portail.

D'une part un *web service* a été développé pour le portail afin d'exposer toutes ses informations de groupes. Ainsi ce *web service* permet de connaître l'arborescence des groupes du portail avec les sous-groupes et membres. D'autre part, un *store* a été développé pour interroger ce *web service*. Toutes les requêtes internes ou externes au serveur vers une URL sous `/roles` entraîne donc une interrogation du *web service*. Un cache permet cependant de limiter ces accès.

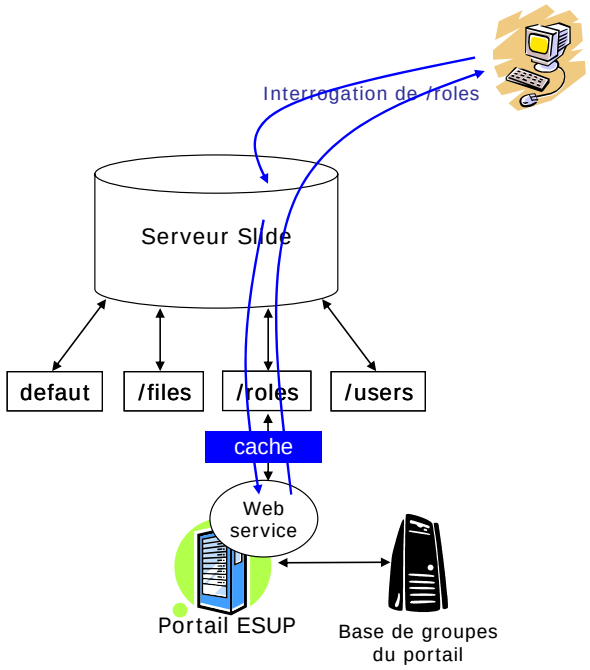

Figure 6 –*Store de gestion de groupes ESUP-Portail*

La base de groupes du serveur WebDAV ESUP est donc exactement la même que celle du portail (Cf. Figure 7). On le remarque en explorant à la fois l'URL `/roles` du serveur avec un dossier web et l'interface de gestion des groupes du portail.

Figure 7 – *Vue des groupes du portail et du serveur WebDAV*

3.3 Positionnement des ACL

3.3.1 *Outil de positionnement*

Le canal stockage, intégré au portail ESUP-Portail, nous permet de positionner des droits d'accès aux répertoires pour différents utilisateurs et groupes d'utilisateurs (Cf. Figure 8). Cette fonctionnalité a été qualifiée de partage.

Voici un aperçu de l'interface de gestion des droits :

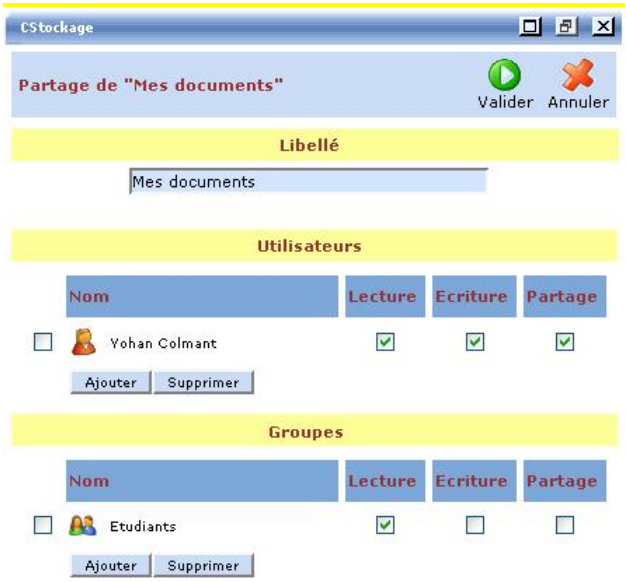

Figure 8 – *Interface de gestion des droits*

3.3.2 API de tri

Le positionnement des ACL sur le serveur Slide nécessite un tri fin et précis des ACE. En effet, l'interprétation des droits d'accès aux ressources se fait dans l'ordre de lecture de celles-ci. Le premier ACE qui correspond à l'utilisateur courant est appliqué dans le cas où il en existe plusieurs.

Afin de justifier le développement d'une bibliothèque de positionnement des ACL, prenons l'exemple suivant :

Supposons que `tbellemb` et `ycolmant` fassent tous deux partie du groupe `local.12` :

Autorisation	Permission	Principal
grant	*read*	`tbellemb` (utilisateur)
deny	*read*	`local.12` (groupe)
grant	*read*	`ycolmant` (utilisateur)

Dans cet exemple, si nous gardons les ACE placés dans cet ordre, `tbellemb` aura le droit de lire la ressource (car placé avant `local.12`), tandis que `ycolmant` ne pourra pas la lire (étant donné que le *deny* pour `local.12` est placé avant le *grant* pour `ycolmant`).

L'API de tri que nous avons développée permet donc de trier ces ACE dans un ordre partant des *principals* les plus fins jusqu'au *principals* les plus larges, et ensuite des permissions les plus fines aux permissions les plus larges.

Nous avons, de plus, choisi de privilégier les *deny* avant les *grant*. C'est-à-dire que pour deux ACE portant sur la même ressource, avec le même privilège, nous faisons passer le *deny* avant le *grant*.

Voyons un exemple de tri de ces ACE :

Autorisation	Permission	Principal
grant	*read-acl*	`ycolmant` (utilisateur)
deny	*read*	`ycolmant` (utilisateur)
grant	*read*	`ycolmant` (utilisateur)
deny	*read*	`local.12` (groupe)

3.3.3 Algorithme de partage

Les spécificités du serveur Slide utilisé ici rendent délicate la mise en place de la fonctionnalité de partage. En effet, supposons que nous voulons modifier les ACL sur `/files/homedir/y/yc/ycolmant/partage` afin de donner à `tbellemb` le droit d'écrire (*write*). Dans ce cas, il ne suffit pas de modifier les droits sur la seule collection `partage`, mais il est nécessaire que `tbellemb` ait le droit de lire (*read*) sur toute l'arborescence. Autrement dit, pour que `tbellemb` puisse écrire sur `partage`, il faut qu'il ait le droit de lire sur `files`, `homedir`, `y`, `yc`, `ycolmant` et `partage`.

Cette contrainte fait que l'utilisateur, via le canal stockage, doit positionner les droits nécessaires sur toutes les collections précédemment citées.

3.4 Utilisation des Métadonnées

Le serveur Slide sépare distinctement le contenu des ressources et les métadonnées qui leurs sont associées. Les métadonnées peuvent être stockées dans des fichiers XML, une base de données, un annuaire LDAP, etc. grâce au mécanisme des *stores* (Cf. 3.2.2) de Slide. Dans le cas d'un stockage sous forme de fichiers XML, les ressources sont contenues de façon arborescente dans un dossier `content` et les métadonnées avec la même arborescence dans le dossier `metadata`.

Voici un extrait du fichier de métadonnées du fichier `Informatique.doc` :

```
<properties>
     <property name="owner" namespace="DAV:"
value="ycolmant" protected="true"/>
     <property name="getlastmodified"
namespace="DAV:" value="Thu, 02 Jun 2005 12:35:17
GMT" protected="true"/>
     <property name="titre" namespace="INJAC:"
value="L'informatique tout public"
protected="false"/>
</properties>
```

Nous pouvons y distinguer les métadonnées définies par le protocole avec le namespace `DAV:`, et celles définies par l'utilisateur. Dans cet exemple, ce sont les métadonnées contenant le namespace `INJAC:`. Ces métadonnées sont utilisées dans le cadre du CMS inJAC développé par ESUP-Portail.

L'interface paramétrable de saisie de ces métadonnées dans le canal stockage est présentée Figure 9.

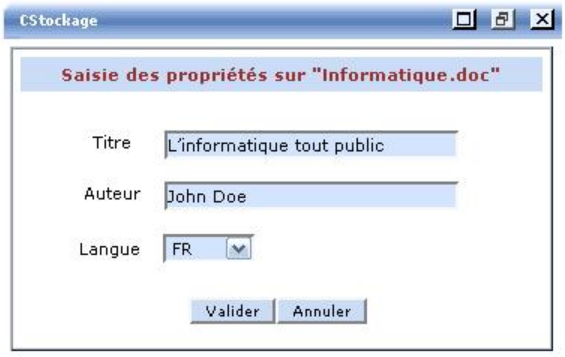

Figure 9 – *Interface de saisie des métadonnées*

3.5 Quotas

3.5.1 Principe

Le serveur Slide ne supporte pas la gestion de quotas dans sa version 2.1. Il fallait développer cette fonctionnalité en restant conforme au *draft* en cours. Nous avons cependant étendu les spécifications du *draft* pour répondre à nos besoins.

Nous voulions notamment pouvoir casser l'héritage de quotas.

Prenons un exemple avec la hiérarchie présentée Figure 10.

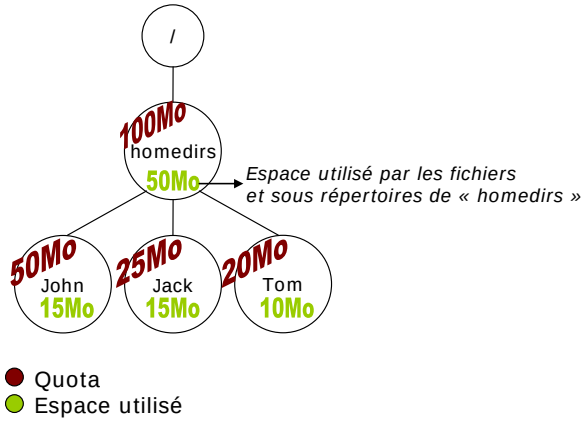

Figure 10 – *Exemple de positionnement de quotas*

L'espace restant sous le répertoire `homedirs` est de 100-(50+25+20)=5Mo. Imaginons que nous voulions créer un répertoire pour l'utilisateur privilégié `Boss` avec un quota de 200Mo, la seule solution pour être certain que `boss` disposera bien de 200Mo même si les autres répertoires du même niveau sont complètement pleins est d'augmenter le quota de `homedirs` jusqu'à 200+(50+25+20)=295Mo. Ce mécanisme est donc peu flexible.

Une solution beaucoup plus simple consiste à casser l'héritage à un nœud donné de l'arborescence (sur le répertoire `Boss` dans notre exemple) en redéfinissant une racine (appelée racine virtuelle).

De ce fait :

– La racine virtuelle est complètement indépendante de ses nœuds pères concernant les quotas, c'est-à-dire qu'elle peut avoir un quota supérieur à ceux-ci.

– La place utilisée par cette racine virtuelle (par ses fichiers et sous répertoires) n'est pas comptabilisée dans l'espace utilisé par ses nœuds pères.

Dans notre exemple nous pouvons donc avoir la configuration présentée Figure 11.

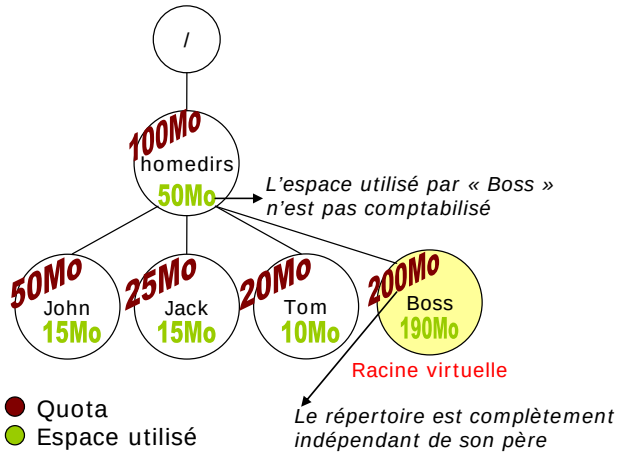

Figure 11 – *Rôle de la racine virtuelle*

Une nouvelle métadonnée optionnelle nommée `ESUP:virtual-root` prenant la valeur `true` ou `false` à été définie pour spécifier qu'un nœud est racine virtuelle ou non.

3.5.2 Développement

Slide implémente des *event listener* Java. Un listener appelé `detailedWebdavListener` écoute des évènements lancés par le serveur Slide à chaque requête WebDAV, plus exactement avant et après chaque requête. Ainsi lors d'une requête `put`, un événement `put` est lancé avant la requête et `putAfter` après la requête. Ce mécanisme est donc idéal pour implémenter les quotas car il est nécessaire de vérifier qu'une requête est possible avant qu'elle ne soit effectuée (le dépôt de document va-t-il engendrer un dépassement de quota ?) et des métadonnées (`DAV:quota-used-bytes` et `DAV:quota-available-bytes`) doivent être mise à jour après celle-ci.

Un `QuotaListener` a été développé pour faire ce travail (Cf. Figure 12). Il vérifie avant toute requête de stockage (dépôt, déplacement, copie, remplacement de fichier) que l'action n'entraînera pas de dépassement de quota pour le répertoire courant mais aussi pour tous les pères jusqu'à la racine (ou première racine virtuelle) et si c'est le cas génère une réponse HTTP avec le code erreur

507 (*insufficient storage)*. Après une requête de stockage réussie il met à jour les métadonnées `DAV:quota-used-bytes` et `DAV:quota-available-bytes`.

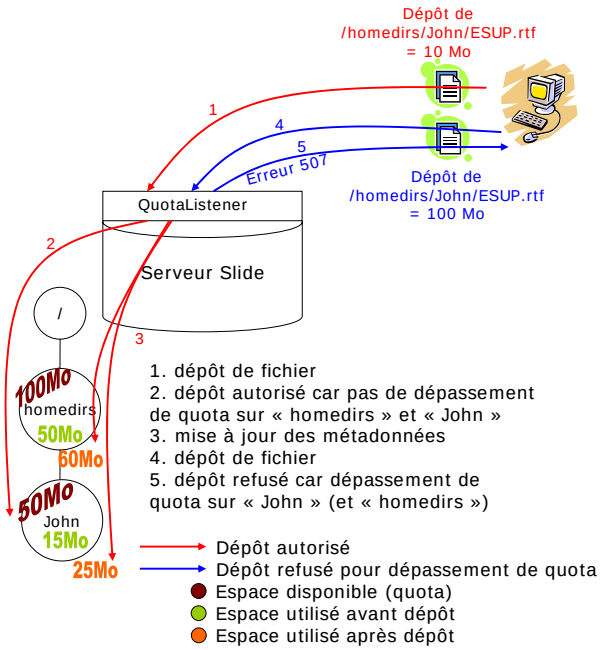

Figure 12 – *Listener de gestion de quotas*

4 Contexte de mise en application

4.1 inJAC

4.1.1 Présentation

ESUP-Portail propose un outil de référencement de ressources : inJAC [14]. Le but de cet outil est de pouvoir stocker des documents, d'y adjoindre différents types de métadonnées et de spécifier des droits d'accès sur ces documents. Une fois les documents stockés ils peuvent être restitués à l'utilisateur via un moteur de rendu. Ce moteur transforme les documents stockés en XML et XHTML afin de leurs appliquer une charte graphique lors d'une sortie web (XHTML) ou papier (PDF). Les documents stockés dans d'autres formats (Word, Open Office ou autres) sont restitués à l'identique. L'outil dispose aussi d'un moteur de recherche pour retrouver les documents localement et d'une compatibilité OAI [15] pour retrouver les documents dans le cadre d'un réseau d'outils de référencement.

4.1.2 Lien avec le serveur WebDAV

inJAC s'appuie sur le serveur WebDAV ESUP-Portail pour ses besoins de stockage des documents et des métadonnées ainsi que pour le contrôle d'accès aux documents (Cf. Figure 13).

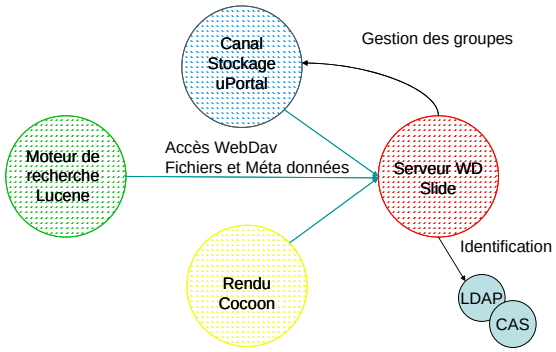

Figure 13 – *Interaction entre les services*

En effet, WebDAV permet nativement de stocker les documents et les métadonnées sur ces documents. ACP apporte les ACL nécessaires au contrôle d'accès. Le moteur de rendu, utilisant Cocoon [16], ne stocke aucun document et s'appuie sur WebDAV pour retrouver les documents. Il bénéficie de fait de la compatibilité avec les groupes du portail car il fait des requêtes WebDAV identifiées et, en fonction des ACL positionnées sur les ressources, n'affiche que les contenus accessibles par l'utilisateur connecté.

Néanmoins, le protocole WebDAV n'est pas suffisant. Autant il est bien adapté à la consultation, il manque de richesse dans la phase de mise à jour du contenu. Il n'est, par exemple, pas possible de donner un droit d'écriture sur une métadonnée et pas sur une autre. Prenons l'exemple de deux métadonnées (niveau de validation et titre du document). Dans la logique métier inJAC le titre du document doit pouvoir être mise à jour par l'auteur du document alors qu'il ne faut pas qu'il puisse modifier la métadonnée niveau de validation pour, par exemple, spécifier que le document est publiable. Ceci relève de la responsabilité de l'éditeur et pas de celle de l'auteur.

De ce fait, toutes les opérations de mise à jour du contenu passent par une couche applicative contrôlant la cohérence des données. Elle utilise le protocole WebDAV pour dialoguer avec le serveur mais c'est la seule à pouvoir le faire avec des droits importants sur des espaces de type inJAC. Cette couche applicative prend aujourd'hui la forme du canal stockage qui, en plus de ses capacités à accéder à des espaces WebDAV, CFS et NFS, sait aussi travailler sur des espaces inJAC. Il est programmé d'en faire un *web service* afin de rendre accessible cette logique métier depuis d'autres applications (Ex : extraction de documents CDM [17] depuis une base de données vers un espace inJAC).

5 Conclusion

5.1 Perspective - Shibboleth

A l'heure où nous écrivons cet article le serveur WebDAV ESUP-Portail n'est pas encore compatible avec Shibboleth [18] mais nous espérons que ce sera le cas au

moment de JRES 2005. Ceci permettra de donner accès à des utilisateurs de plusieurs établissements à des ressources partagées.

Le CRU [19] propose une fédération d'identités et d'attributs pour les établissements de l'enseignement supérieur français [20]. Cette fédération utilise l'outil Shibboleth développé dans le cadre d'internet2.

Le but d'une fédération d'identité est de rendre accessible un service à des utilisateurs qui n'appartiennent pas obligatoirement à l'établissement qui offre le service en question.

Il y a une répartition des responsabilités. Les autorisations d'accès au service sont décidées par l'établissement hébergeur (*Service Provider*) alors que la gestion de l'identification de chaque utilisateur et la fourniture au service d'attributs le caractérisant est confiée à l'établissement d'appartenance de l'utilisateur (*Identity Provider*).

Le mécanisme de fédération d'identité étend la notion de SSO et permet d'utiliser des services d'autres établissements sans avoir à s'identifier à nouveau. Néanmoins, il ne remet pas en cause le choix de SSO des établissements et des essais avec CAS ont déjà été effectués.

Dans le cadre de la mise en œuvre de Shibboleth nous utiliserons le mécanisme de *stores* vu au point 3.2.

Dans un fichier de configuration XML sera définie une arborescence de groupes. Chaque groupe aura pour définition une expression logique sur des attributs de l'utilisateur (ex : Niveau=L1 et établissement=A ou B ou C). Cette arborescence de groupes sera visible au même niveau que les groupes du portail (sous `/roles`). Il sera alors possible de spécifier une ACL sur une ressource avec un des ces groupes.

Au moment où un utilisateur tentera d'accéder à cette ressource, ses attributs, remontés par le mécanisme Shibboleth, seront évalués afin de déduire sont appartenance ou non au groupe. Ceci afin de savoir si l'ACL s'applique à l'utilisateur ou non.

5.2 Critiques

Le système présenté dans cet article est relativement complexe. Il met en œuvre de nombreux serveurs qui dialoguent avec différents protocoles.

La gestion des quotas nécessite une version particulière de Slide qui n'est pas encore largement diffusée.

Les composants d'administration évoluent en fonction de retours des utilisateurs pour offrir plus de cohérence et de simplicité.

En bref, la solution est encore jeune et la mise en production récente dans différentes universités devrait permettre de la stabiliser.

Bibliographie

[1] http://www.esup-portail.org/

[2] http://esup-helpdesk.sourceforge.net/

[3] Pascal Aubry et Pierre Gambarotto, ESUP-Portail: a pure WebDAV-based Network Attached Storage. Dans actes de la conférence EUNIS2004, Bled, Slovénie, Juin 2004.

[4] http://www.webdav.org/

[5] Pascal Aubry - http://www.esup-portail.org/consortium/espace/Stockage_2F/architecture/storage_11_12_03.ppt

[6] http://www.webdav.org/acl/

[7] http://jakarta.apache.org/slide/

[8] http://www.webdav.org/specs/rfc3744.html

[9] http://www.ietf.org/internet-drafts/draft-ietf-webdav-quota-07.txt

[10] http://www.ietf.org/rfc/rfc3530.txt

[11] Vincent Mathieu, Pascal Aubry et Julien Marchal, Single Sign-On open-source avec CAS (Central Authentication Service). Dans Actes de JRES2003, Lille, France, décembre 2003.

[12] http://jasigch.princeton.edu:9000/display/CAS

[13] http://www.esup-portail.org/consortium/espace/Stockage_2F/serveur/serveur_V3/authenticationFilters.html

[14] http://www.esup-portail.org/consortium/espace/CMS_3F/index.html

[15] http://www.openarchives.org/

[16] http://cocoon.apache.org/

[17] http://cdm.nou.no/ et http://acces.inrp.fr/cdm/

[18] http://shibboleth.internet2.edu/

[19] Olivier Salaun, Florent Guilleux et Pacal Aubry, Gérer la propagation d'identités et d'attributs pour le web. Dans actes de JRES2005, Marseille, France, décembre 2005, à paraître.

[20] http://federation.cru.fr/