Nomadisme et compression X11

Eric DÉCORNOD
ULP MULTIMÉDIA
Université Louis PASTEUR de STRASBOURG
eric.decornod@ulpmm.u-strasbg.fr

Hervé JAUME
ULP MULTIMÉDIA
Université Louis PASTEUR de STRASBOURG
herve.jaume@ulpmm.u-strasbg.fr

Résumé

Depuis longtemps des solutions commerciales permettent d'ouvrir des sessions d'applications graphiques à distance sur des connexions de moyen débit. Les applications du monde UNIX reposant sur le protocole X11 peuvent désormais rivaliser avec ces dernières grâce à la compression NX (développée sous licence GPL par la société NoMachine).

Ceci ouvre une perspective nouvelle pour les utilisateurs nomades et les E.N.T.[1] : avoir un accès universel à son environnement applicatif.

C'est dans ce contexte que nous avons développé à l'ULP MULTIMÉDIA une solution de bureau virtuel appelée « Univ-RX ».

Mots clefs

NX, X11, Compression, Java, ENT, CAS.

1 Introduction

Le projet *UNIV-R* porté par l'ULP Multimédia vise à offrir une brique applicative de type *bureau virtuel* (applications bureautiques & pédagogiques, espace de stockage, espace de travail collaboratif) pour les E.N.T. fondés sur le système SSO[2] CAS[3] (comme E.P.P.U.N. [1] par exemple). L'équipe *UNIV-RX* (*UNIV-R LINUX*) y apporte une réponse fondée sur J2EE, la technologie NX et les applications du monde *LINUX*.

Dans ce cadre, nous verrons comment nous avons intégré la technologie NX pour donner un accès web nomade à l'ENT Univ-RX [2].

1.1 La brique Univ-RX

Les E.N.T. sont souvent vus comme des portails web authentifiés composés de différentes *briques* adaptées aux différents usages de leur public : de la gestion pour le personnel administratif habilité, des ressources pédagogiques pour les enseignants et les étudiants ou encore de la messagerie pour tous. Toutes ces briques sont la plupart du temps des *applications web*.

Univ-RX est une *brique* mais son objectif est non pas de se substituer aux outils existants mais de fournir un environnement de travail avec les vrais outils de bureautique et pédagogique traditionnels comme le sont la suite bureautique OPENOFFICE.ORG ou le logiciel de calcul formel MAPLE™.

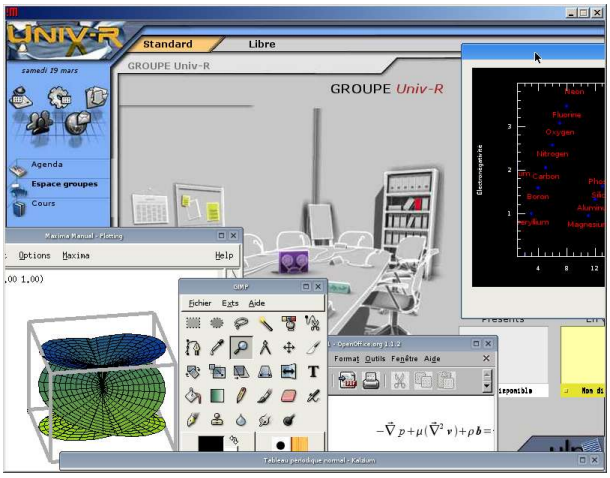

Figure 1 : session applicative à distance

À l'intérieur d'Univ-RX, l'accès aux applications se fait par le biais d'un navigateur FIREFOX qui se substitue au bureau (mode kiosque) en le remplaçant par une plateforme web de travail collaboratif (voir Figure 1).

1.2 L'infrastructure Univ-RX

Les sessions applicatives sont hébergées sur une ferme de serveurs DEBIAN/GNU-LINUX. Les connexions sont réparties sur les serveurs par deux redirecteurs LVS en haute-disponibilité avec reprise des connexions actives en cas de défaillance (voir Figure 2).

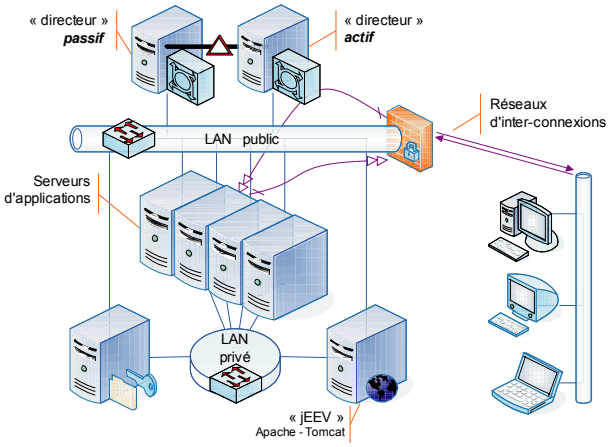

Figure 2 : Schéma d'infrastructure d'Univ-RX

[1] Espaces Numériques de Travail

[2] Single Sign On

[3] Central Authentication Service

La plateforme de travail collaboratif (jEEV) et ses bases de données ainsi que les espaces disque utilisateurs et l'annuaire sont déportés sur des serveurs spécialisés.

Les connexions à la ferme UNIV-R LINUX se font au sein de sessions SSH transportant les sessions graphique des utilisateurs.

2 Le Protocole NX

Le protocole de communication (X11) entre applications graphiques et les serveurs d'affichage des clients est si gourmand que même sur des liaisons 100 Mb l'impact sur le réseau est pénalisant. La compression NX [3] permet d'alléger cette charge réseau jusqu'à permettre :

- une utilisation bureautique sur une liaison ADSL-128k,
- un grand nombre de connexions simultanées sur les réseaux des universités.

Cette méthode de compression — dérivée de DXPC [4] — est développée sous licence GPL par la société italienne NoMachine [5] qui commercialise le logiciel serveur et distribue (sous licence fermée) les clients pour LINUX, WINDOWS et MAC OSX (compression = GPL, serveur = payant, client = gratuit sont trois éléments distincts).

FreeNX [6] est un équivalent GPL au serveur de NoMachine écrit en SHELL SCRIPT ; et KNX sera un client libre pour KDE.

2.1 Principe de la compression NX.

Lorsqu'une application graphique fait appel à des fonctions graphiques (dessiner un polygone, un caractère, une image), la librairie graphique (Xlib) transmet cet appel de fonction au serveur graphique (Xserver) via un socket (unix ou réseau). La communication entre Xlib et Xserver est le protocole X11 [7].

NX s'intercale au milieu de cette communication en plaçant deux agents (voir Figure 3 et Figure 8) pour réaliser un certain nombre d'économies et de modifications sur les échanges. Pour cela chacun des agents tient à jour le cache de son pair.

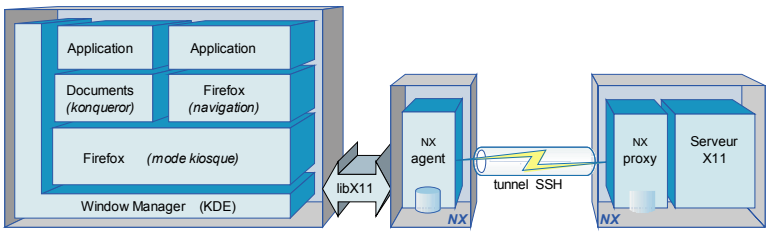

Figure 3 : Schéma de fonctionnement

Plusieurs techniques sont utilisées simultanément :

- *cache de messages* (MessageStore),
- *encodage différentiel* (Differential Encoding),
- *encodage des identifiants avec algorithme MTF* (Cache Based Value Encoding),
- *compression d'images et découpage des trames* (Compression & Streaming),
- *cache d'images sur le disque*,
- *contrôle de la bande passante*.

Certains messages échangés sont conservés dans le *cache de messages*, ainsi pour des nouveaux messages très proches, seules les petites différences avec la référence du cache doivent être transférées pour reconstruire le message original (chaque type de message a son mode de calcul propre). NoMachine annonce que NX arrive à coder 60 à 80 % des messages par ce biais.

Pour chaque message ne rentrant pas dans le cas précédent, chaque champ est encodé séparément, en tirant au maximum parti des informations déjà connues (d'où le nom d'*encodage différentiel*) ; par exemple pour un appel PolySegment, chaque coordonnée est transmise relativement à la précédente de façon à économiser un maximum de bits (en général moins de 8 bits dans ce cas contre 16 bits ferme pour le protocole X11). NoMachine annonce un ratio de 5:1 à 8:1 pour cette méthode.

Chaque objet graphique (bouton, zone de texte, icône) est en général porté par une fenêtre conteneur représentée par un identifiant sur 32 bits ; de même la majorité des identifiants du protocole X11 sont représentés sur 32 bits. *L'algorithme MTF* [8] (pour Move To Front, déplacer vers l'avant) est utilisé pour encoder ces identifiants sur très peu de bits (le plus souvent un seul).

Figure 4 : exemples de compression PNG (faible) et JPEG (forte) avec la barre d'outils FIREFOX (des artefacts de compression apparaissent avec JPEG)

Les transferts d'*images* (X_PutImage) sont *compressés* à l'aide des algorithmes PNG ou JPEG (voir Figure 4) en plus des algorithmes de cache de message et d'encodage différentiel. Le niveau de compression est ajusté en fonction du type de la ligne (modem, ADSL, LAN). La méthode JPEG rapporte un ratio typique de 20:1 ramenant une image de 256 Kb à 12 Kb, mais l'impact sur la bande passante reste fort, ainsi pour ne pas pénaliser les autres applications, les gros messages sont *découpés* en petits morceaux et intercalés dans le flux des messages — pénalisant les applications intensivement graphiques au profit des autres —. De surcroît, en collaboration avec le cache de messages, les images sont conservées dans un *cache sur le disque* afin d'être réutilisées.

NX utilise un système de *contrôle de la bande passante*, suspendant une application lorsqu'elle remplit le quota de bande passante, pour assurer des temps de réponse satisfaisants et un usage équitable de la fenêtre TCP.

2.2 Problèmes posés par le modèle client/serveur NoMachine.

Ce modèle pose trois problèmes essentiels :

- sécurité,
- intégration aux systèmes SSO (single sign on),
- et installation.

Sécurité

La connexion du client au serveur débute par une session SSH vers le serveur sous l'identité générique « nx », identifié par un couple de clefs publique/privée. Une redirection de ports dynamique[4] est établie pour porter les différents canaux de compression (graphique, son et autres ressources locales). Une fois cette session utilisateur nx établie, la session de l'utilisateur est ouverte[5] utilisant son mot de passe (transmis dans le tunnel SSH par le client).

Or la *clef privée* est contenue dans le client et donc distribuée à grande échelle, notamment sur le site internet de NoMachine. Ainsi sans aucun compte utilisateur, n'importe qui a la possibilité d'utiliser le serveur comme relais pour ses connections TCP.

```
spammer@sp.am> ssh -i client_dsa.key \
          -L 2525:mailhost.univ-truc.fr:25 \
          -N  nx@nxserver.univ-truc.fr  &
spammer@sp.am> ./sendspam 127.0.0.1:2525
```

Figure 5 : envoi de spam par serveur NoMachine/FreeNX

L'équipe FreeNX [6] tente de contourner le problème en imposant quelques limitations[6] à SSH sur le compte utilisateur « nx » (depuis le 5 mars 2005 d'après le changelog). Ils suggèrent de même de changer le couple de clefs, mais cette option ne convient pas à un large public (on ne peut pas référencer un millier de clefs pour l'utilisateur nx sans perturber significativement l'ouverture de session). Mais toutes ces restrictions laissent un script SHELL sous l'identité nx potentiellement vulnérable (voir [9]).

Intégration aux systèmes SSO

Une intégration transparente dans un portail web d'E.N.T. doit respecter trois critères principaux :

- faire partie du portail, c.-à-d. être accessible depuis un navigateur internet,
- respecter le SSO,
- être accessible depuis un maximum d'endroits (aussi bien sous LINUX, sous WINDOWS, que sous MACINTOSH ; aussi bien depuis chez soi que d'un cyber-centre).

Nous avons contacté la société NoMachine afin de mettre en place un partenariat pour développer un accès internet (de type appliquette) intégrable dans des portails ; mais notre offre ne les intéressait pas, ils estimaient le temps de développement à 3 ans.

De plus aucun des clients actuels (NoMachine, KNX) n'implémente de système SSO ; ils ne peuvent donc pas s'intégrer pleinement aux portails applicatifs et E.N.T. des universités (notamment dans le cas du système SSO CAS [10] choisi par EPPUN).

Les plate-formes cibles que nous visons sont principalement les clients sous Windows et sous Linux (pas encore les version 64 bits) ; l'intégration des MACINTOSH OSX est à l'étude.

Installation des clients

Les clients — notamment le client NoMachine pour MAC OSX — sont parfois très difficiles à installer et requièrent des droits très élevés sur la machine lors de l'installation. C'est un gros problème pour permettre des accès de type « cyber-centre ».

Or les accès de ce type représentent l'atout majeur des performances de la compression NX (en dehors des salles de ressources informatiques et des chambres universitaires CROUS où des clients spécifiques « maison[7] » remplissent cette tâche).

3 Une appliquette comme mode d'accès

3.1 Appliquette signée et exécution de code

Pour répondre à la fois à la contrainte de pouvoir s'exécuter sur une large variété de machines et à celle d'être accessible depuis un navigateur, nous avons développé une « *appliquette JAVA* » qui forme le cœur du dispositif d'accès à Univ-RX [11].

Une appliquette *signée* est affranchie des contraintes du « *bac à sable*[8] » ; dans une configuration standard de JAVA elle peut donc :

- ouvrir des connexions réseau,
- accéder au système de fichiers (même en écriture),
- exécuter des applications natives (non-java).

3.2 Modèle client/serveur

Pour contourner les problèmes et/ou réticences vis-à-vis du modèle client/serveur NoMachine évoqués précédemment, on se passe totalement du serveur NX (NoMachine ou FreeNX).

Il suffit d'exécuter à distance et localement les deux agents NX (nxproxy côté client, nxagent côté serveur) pour obtenir un « tunnel de compression ».

Ceci peut être réalisé simplement au sein d'une session SSH authentifiée de façon traditionnelle : plus besoin du compte générique nx ni de clef privée[9].

Pour cela nous utilisons la bibliothèque « JSCH » [12] qui est un client SSH entièrement JAVA et distribuée sous licence BSD. On a la possibilité de gérer nous même l'authentification, les transferts de ports et la session utilisateur à distance depuis le client.

---

[4] le client SSH se comporte alors comme un mandataire SOCKS

[5] soit par le biais d'un `ssh` sur localhost, soit par une commande `su`

[6] `no-port-forwarding, no-X11-forwarding, no-agent-forwarding, command="$PATH_BIN/nxserver"`

[7] Un système linux complet fondé sur KNOPPIX et allégé pour les salles de ressources, et un client NoMachine packagé et intégré à des terminaux clients légers

[8] Une appliquette JAVA s'exécute dans un environnement restreint appelé « bac à sable » pour prévenir des actions malveillantes des appliquettes

[9] appelée « semi-privée » dans le jargon FreeNX

3.3 Principe de fonctionnement

L'appliquette réalise les opérations suivantes :

- identification du système,
- téléchargement sur le système (si nécessaire) des applications natives (non-java) indispensables à l'établissement du tunnel de compression,
- démarrage de ces dernières,
  notamment un serveur X11 sur les systèmes qui en sont dépourvus (comme WINDOWS et MAC OSX).

Le modèle objet de JAVA est tout à fait adapté à l'identification : un objet *générique* (NXarch) factorise les opérations communes aux différentes architectures et systèmes clients, tandis qu'une classe fille *spécialise* les caractéristiques propres à chaque système (voir Figure 6 : classes NXlinux, NXwin32 et NXnone en cas d'erreur).

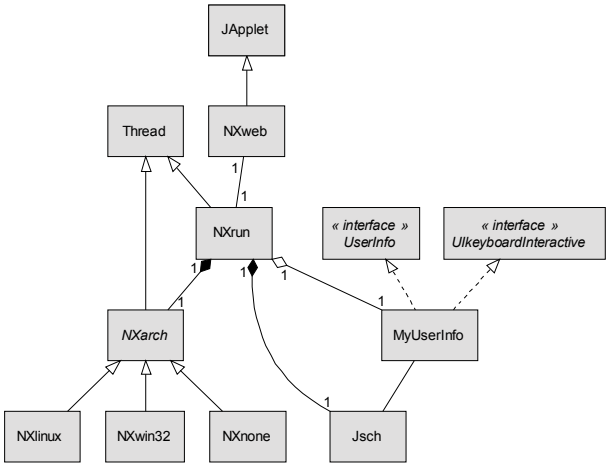

Figure 6 : Partie du diagramme de classes UML

Par exemple le paquetage de binaires spécifiques qu'il faut télécharger, extraire et exécuter est une de ces caractéristiques. Dans le cas des plateformes dépourvues de serveur graphique X11, une version est incluse dans le paquetage (le serveur XFREE de CYGWIN pour WINDOWS par exemple comme dans la Figure 8).

L'ensemble des opérations réalisées par l'appliquette sont spécialement étudiées afin de requérir un minimum de droits sur la machine cliente.

3.4 Intégration au système SSO CAS

Le système SSO CAS [10]

Il s'agit d'un système où l'utilisateur entre son login et son mot de passe une seule fois *via* un formulaire web auprès du serveur CAS. Ce dernier maintient la session de l'utilisateur par un identifiant *opaque*[10] et *rejouable*[11] sous la forme d'un cookie dans le navigateur (TGC).

De cette manière les applications qui nécessitent une authentification font passer le navigateur par le serveur CAS qui, grâce au TGC, lui fournit un ticket *opaque* (ST), *non-rejouable*[12] et associé à l'application (identifiée par son URL).

[10] on ne peux pas en tirer d'informations sur l'utilisateur, sa session ou son mot de passe

[11] réutilisable à volonté pour toute la durée de la session (jusqu'au *logout*)

[12] à usage unique

Le navigateur transmet en retour ce ticket à l'application. Celle-ci doit finalement le valider auprès du serveur CAS pour obtenir le login de l'utilisateur.

Ces mécanismes limitent l'usurpation d'identité.

Dans le cas des portails de services, un système particulier (CAS-2, Figure 10) permet à une application dite proxy d'obtenir un ticket opaque rejouable (PGT) qui comme le cookie TGC permet de délivrer des tickets non-rejouables associés à des applications tierces (PT). L'objectif est de propager l'identification vers les applications tierces de façon transparente à l'utilisateur (plus besoin pour lui de repasser par le serveur CAS).

Ticket	*Nom*	*rejouable ?*	*Délivré sur présentation de*	*Stockage*
TGC	Ticket Granting Cookie	oui	login + mot de passe	dans le navigateur de l'utilisateur sous la forme d'un cookie
ST	Service Ticket	non	TGC	dans l'URL
PT	Proxy Ticket	non	PGT	dans l'URL
PGT	Proxy Granting Ticket	oui	ST ou PT et URL	dans l'application proxy.

Figure 7 : Les différents tickets opaques dans CAS

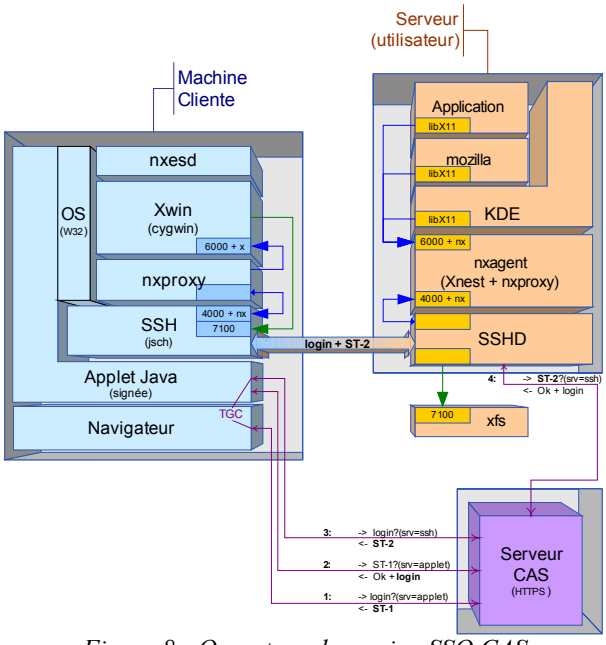

Figure 8 : Ouverture de session SSO CAS

« CAS-ification »

SSH a besoin d'un nom d'utilisateur (*login*) **et** d'un mot de passe pour ouvrir une session (un ticket se substitue au mot de passe dans ce cas). Or le portail (ou le serveur CAS suivant le cas) donne un ticket (ST ou PT suivant les cas) valide *une seule et unique fois*. Il nous faut donc un mécanisme pour construire le nom de l'utilisateur et envoyer un ticket encore valide.

La validation d'un ticket donne le nom d'utilisateur mais invalide le ticket qui sera perdu puisqu'il n'est *pas* rejouable.

Or l'appliquette (grâce à JAVA) dispose d'un certain nombre de paramètres du navigateur et notamment le cookie de session CAS (TGC). Ceci autorise JAVA à contacter CAS pour obtenir un nouveau ticket de service à son compte (voir Figure 9).

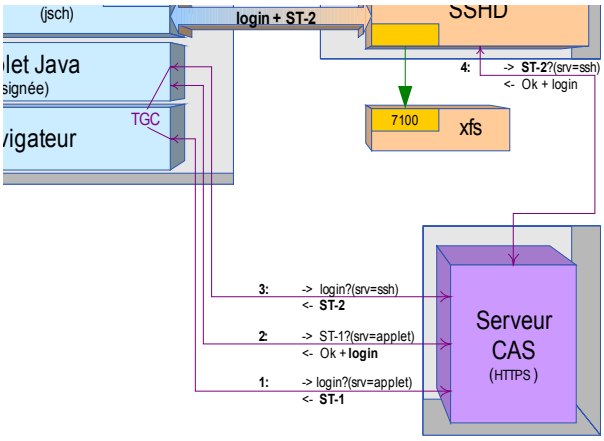

Figure 9 : Agrandissement de la Figure 8

Problème du « proxy N-tiers »

Pour être pleinement SSO, il devrait être possible de pouvoir ouvrir des services du portail de l'E.N.T. comme un web-mail par exemple.

Le mécanisme de CAS pour ce genre de situations est appelé « proxy N-tiers » [13].

Lors de la validation du ticket de service reçu, le serveur CAS répond avec un PGT_{IOU}, suite à quoi le serveur, de son propre chef, contacte le service en lui donnant la correspondance PGT_{IOU}↔PGT. Il est à la charge du service de maintenir sa table d'association pour retrouver le PGT à partir du PGT_{IOU}. Muni du PGT pour l'utilisateur, le service peut désormais obtenir des tickets (PT) pour ses services tiers (voir Figure 10).

Mais dans le cas où le service proxy est une *ferme de serveurs* la requête HTTPS de CAS (d'association PGT_{IOU}↔PGT) a peu de chances de tomber sur le serveur de la ferme où se trouve l'utilisateur.

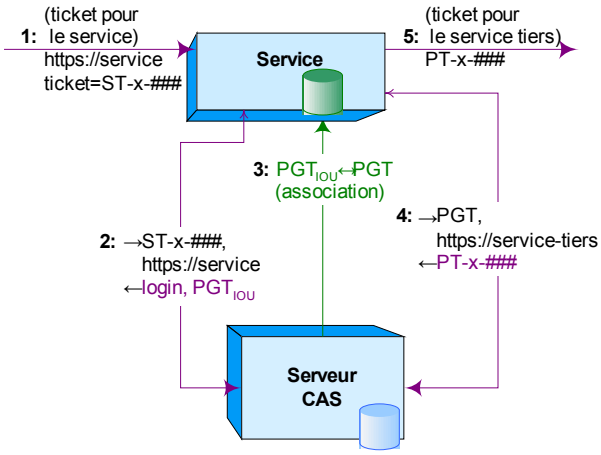

Figure 10 : Principe de service proxy (CAS-2)

L'alternative est :

- maintenir la table d'associations sur un serveur dédié, consulté par les serveurs d'applications pour obtenir le précieux PGT qui permet d'obtenir des tickets (PT),
- ou bien contourner le mécanisme en s'inspirant de la technique précédente pour obtenir le nom d'utilisateur en communiquant avec l'appliquette pour demander les tickets (ST) que l'on souhaite comme si c'était le navigateur du client qui les demandait.

CAS-ification des serveurs

Côté serveur d'applications, au niveau du démon ssh, la solution semble être toute trouvée avec les différentes implémentations de CAS au niveau de PAM [14].

Or ces derniers n'authentifient que les tickets ignorant les authentifications traditionnelles (login + mot de passe). Dans ce cas, le module PAM suivant (LDAP par exemple) peut s'en charger mais en perdant les fonctions de SSO.

Nous avons donc écrit notre propre module PAM CAS qui se comporte dans ce cas comme un navigateur en soumettant les identifiants de connection au login CAS, obtenant en réponse le TGC.

Ainsi nous avons la possibilité à l'intérieur de la session utilisateur d'obtenir un ticket pour ouvrir le portail de l'ENT.

Ce cas de figure se présente pour des connections établies depuis des postes informatiques situés dans des salles de ressources informatiques sous notre gestion : la session est établie directement sans passer par le portail (celui-ci est ouvert au sein de la session authentifiée).

3.5 Ressources locales

Une fonctionnalité très importante, dans un accès nomade distant, est de pouvoir accéder aux ressources de la machine cliente depuis les applications distantes.

Son

Les clients NoMachine utilisent les fonctionnalités réseau des démons de son ARTS ou ESD suivant le système client. Ce comportement est simplement reproduit dans l'appliquette (par le biais de transferts de ports au sein de la session SSH).

Échanges de fichiers locaux ↔ distants

Après avoir patché JSCH, un serveur FTP JAVA a pu être écrit pour permettre l'échange de documents tout en faisant un minimum de présuppositions sur les droits et le système du client. Tout cela restant sécurisé car au sein de la session encryptée SSH.

Impression

Le support de l'impression locale est encore à l'étude.

4 Évolutions, perspectives

L'appliquette de connection n'est pas restreinte à la brique Univ-RX (ferme de serveur, interface KDE-kiosk avec firefox remplaçant le bureau, SSO CAS), elle conserve son intérêt en dehors de ce contexte comme mode de connexion nomade.

De plus, l'appliquette peut facilement fonctionner de façon autonome sans navigateur en tant qu'application java. L'intérêt réside alors dans le fait qu'il n'est pas nécessaire d'avoir des droits très élevés sur le serveur non plus :

disposer d'une version des binaires côté serveur suffit (un accès SSH sans privilèges particuliers est suffisant pour y parvenir).

On pourrait même envisager (de la même manière que l'appliquette télécharge les binaires qui lui conviennent) que l'appliquette télécharge vers l'amont (*upload*) sur le serveur les binaires dont elle aurait besoin pour son fonctionnement lorsque ceux-ci n'y sont pas ; tout comme l'utilisateur pourrait le faire lui-même « à la main » avec `scp` par exemple.

Une des évolution futures est de pouvoir ouvrir des documents depuis une application web de travail collaboratif. L'application prenant en charge chaque type de document sera ouverte depuis le serveur d'applications comme si tout se passait localement (serveur X11 en mode fenêtré dit *rootless*).

4.1 Licence, distribution

L'appliquette [11] est développée sous licence GPL et les sources sont disponibles à l'adresse http://univ-rx.u-strasbg.fr/applet-univrx/

L'installation se fait par le biais de la publication des divers archives `.jar`[13] sur un serveur web. Une page (statique ou dynamique) construit une balise `<applet>` avec les paramètres qui conviennent. À noter que ces paramètres peuvent être forcés ou initialisés dans le `.jar`.

4.2 Conclusion

Grâce à la compression NX et aux développements que nous avons réalisés à l'ULP MULTIMÉDIA, il est désormais possible de se connecter depuis un navigateur internet sur des serveurs d'applications et pas uniquement en console ou avec des latences pénalisantes.

[13] Java ARchive

Bibliographie

[1] ULP Multimédia, *Espaces Pédagogiques Pour les Universités Numériques*, 2005, http://eppun.u-strasbg.fr.

[2] Hervé Jaume, Univ-RX : Pédagogie dans les ENT. Dans *Actes du congrès JRES2005*, Marseille, Décembre 2005.

[3] Gian Filippo Pinzari, *NX X Protocol Compression*, D-309/3-NXP-DOC, NoMachine, Septembre 2003.

[4] Brian Pane, *Differential X Protocol Compressor*, 1995, http://www.vigor.nu/dxpc/.

[5] NoMachine, *Building the network computing on the power of X*, 2002, http://www.nomachine.com.

[6] Fabian Franz, Kurt Pfeifle, *FreeNX - Free Software (GPL) Implementation of the NX Server*, Juin 2004, http://freenx.berlios.de/.

[7] Robert W. Scheifler, *X Window System Protocol*, X Version 11 Release 6, Massachusetts Institute of Technology, 1994.

[8] Wikipédia, *L'algorithme MTF*, 2005, http://fr.wikipedia.org/wiki/Move-To-Front.

[9] CERTA, *Sécurité des applications Web et vulnérabilité de type "injection de données"*, CERTA-2004-INF-001, SGDN, janvier 2005.

[10] Pascal Aubry, Julien Marchal, Vincent Mathieu, Single Sign-On open-source avec CAS (Central Authentication Service). Dans *Actes du congrès JRES2003*, chapitre 3.2.2, Lille, Novembre 2003.

[11] Eric Décornod, *Univ-RX : Applet*, ULP Multimédia, 2005, http://univ-rx.u-strasbg.fr/applet-univrx/.

[12] Atsuhiko Yamanaka, *JSch -- Java Secure Channel*, JCraft, http://www.jcraft.com/jsch/.

[13] David Spencer, *Proxy CAS Walkthrough*, CAS mailing list, JA-SIG, http://jasigch.princeton.edu:9000/display/CAS/Proxy+CAS+Walkthrough.

[14] Andrew Petro, *CAS > Clients > PAM Module*, JA-SIG, mars 2005, http://jasigch.princeton.edu:9000/display/CAS/PAM+Module.