Université de Mons
Faculté des Sciences
Institut d'Informatique

Copier/coller multi-plateformes

Directeur : Mr Olivier DELGRANGE

Projet réalisé par
Maëlick CLAES

Rapporteurs : Mr Bruno QUOITIN
Mr Sylvain DEGRANDSART

UMONS
Université de Mons

Année académique 2010-2011

Table des matières

Introduction **1**

1 Problème **2**
1.1 Objectifs 2
1.2 Contraintes 3

2 Analyse des solutions existantes **4**
2.1 Solutions lourdes 4
2.1.1 Échange d'e-mails 4
2.1.2 Utilisation d'un serveur FTP 5
2.1.3 Partage de fichiers 5
2.1.4 Remote Desktop Services 6
2.1.5 VNC 6
2.1.6 Citrix XenApp 6
2.1.7 Technologie NX 7
2.1.8 Comparaison 7
2.2 Solutions *a priori* légères 7
2.2.1 ClipboardMultiSharer 7
2.2.2 Clipboard Share 8
2.2.3 The Network Clipboard 8
2.2.4 Remote Clip 8
2.2.5 Comparaison 9

3 Solution proposée **12**
3.1 Principes d'une architecture P2P 12
3.2 Architecture logicielle 13
3.2.1 Client P2P 13
3.2.2 Client local 16
3.3 Protocoles 16
3.3.1 Protocole P2P 17
3.3.2 Protocole local 19

Conclusion **23**

Annexes **27**

A Codes sources utilisés **27**

B Principe KISS **29**

Introduction

Lorsque l'on travaille sur un ordinateur, il est souvent plus agréable de travailler en utilisant plusieurs écrans. Cela permet par exemple d'afficher des informations sur un écran tout en prenant note sur un autre. De plus lorsqu'un utilisateur dispose de deux ordinateurs, par exemple un PC fixe et un portable, il en arrive à travailler en utilisant plusieurs écrans.

Seulement une différence majeure existe entre le fait de travailler avec plusieurs écrans sur un ordinateur et le fait de travailler avec plusieurs ordinateurs ayant chacun leur écran. Dans le premier cas il est aisé de faire transiter de l'information d'un écran à un autre, ceux-ci étant des périphériques reliés au même ordinateur. Dans le second cas cela est impossible sans passer par un moyen de communication tel que l'e-mail ou un support externe tel qu'une clé USB. Or cela semble fastidieux s'il faut transmette peu d'information fréquemment.

Le but visé ici est donc de chercher comment résoudre ce problème grâce à un système de copier/coller entre plusieurs plateformes *i.e.* entre plusieurs machines connectées sur le même réseau. Pour cela un premier chapitre permettra de présenter le problème, d'énoncer les objectifs et les contraintes du projet. Ensuite un second chapitre permettra d'analyser les solutions existantes qui permettent de résoudre le problème. Ces solutions seront présentées, leurs avantages et inconvénients seront donnés et elles seront finalement comparées.

Un troisième chapitre permettra de définir une architecture réseau, ainsi qu'un protocole de communication, à utiliser pour implémenter un ensemble de logiciels permettant de résoudre le problème de copier/coller multi-plateformes. Enfin une conclusion permettra de rendre compte de l'avancement du travail réalisé, décrira comment seront planifiées la réalisation et l'implémentation du projet et quels outils seront utilisés.

Chapitre 1

Problème

1.1 Objectifs

Le premier objectif de ce projet est de donner à un utilisateur la possibilité de faire du copier/coller entre plusieurs ordinateurs allumés. Le logiciel créé doit être une solution, simple et légère, à installer sur chaque poste client. C'est-à-dire qu'il doit être facilement configurable, afin de pouvoir rapidement fournir le service voulu, et il doit avoir une charge minimale sur le système. Il doit permettre de faire facilement l'opération de copier/coller sans avoir à passer par un échange de mails ou de fichiers. Il doit aussi viser le monde Unix en premier lieu, mais doit être adaptable à d'autres systèmes d'exploitation.

Afin de fixer certains termes plusieurs définitions sont nécessaires.

Définition 1.1.1. *Un* presse-papier *est une zone mémoire dans laquelle une ou plusieurs données sont stockées. Celles-ci peuvent être de différents types,* e.g. *du texte, une image, un fichier.*

Remarque 1.1.1. *Bien qu'habituellement un presse-papier ne permet de stocker qu'une seule donnée à la fois, il existe un grand nombre de logiciels permettant de gérer un presse-papier contenant un ensemble de données permettant d'avoir ainsi un* historique *de ce qui a été copié dedans.* e.g. *GNU Emacs [Prors] permet de cycler sur cet historique en revenant au début de celui-ci lorsqu'il a été entièrement parcouru.*

Définition 1.1.2. Copier *est l'action d'écrire une donnée dans le presse-papier.*

Définition 1.1.3. Coller *est l'action de récupérer la donnée présente dans le presse-papier.*

Définition 1.1.4. Copier/coller *résume le concept permettant de copier quelque chose dans le presse-papier et le coller ensuite.*

Définition 1.1.5. *Copier/coller* multi-plateformes *décrit la possibilité de faire du copier/coller entre deux ordinateurs mis en réseau. L'abréviation* CCMP *sera parfois utilisée dans la suite de ce rapport* [1].

1.2 Contraintes

La priorité est de fournir un système léger. Celui-ci doit être facilement installable et configurable, discret et l'*overhead* sur le système doit être minimisé. Il ne doit donc pas reposer sur un autre système plus lourd comme le partage de fichiers. Dans un premier temps, il ne doit gérer que le texte mais il doit être extensible. Ceci afin de permettre facilement l'ajout du support pour d'autres formats de données comme les images. Une autre contrainte importante à prendre en considération est l'aspect réseau. Celui-ci introduit de potentiels problèmes de sécurité. Il faudra donc trouver un moyen de réduire ceux-ci *e.g.* en chiffrant la connexion réseau et en obligeant l'utilisateur à s'authentifier afin d'utiliser le logiciel. De même, le logiciel devant tourner sur plusieurs systèmes d'exploitations différents dont Linux, il est plus que préférable que le logiciel soit libre de droits et se base sur des protocoles et technologies ouverts.

1. Un protocole de chiffrement de la norme IEEE 802.11i est aussi abrégé CCMP, cependant celui-ci n'ayant aucun lien avec le sujet du projet, l'utilisation de cette abréviation ne créera pas d'ambiguïté.

Chapitre 2

Analyse des solutions existantes

Cette section détaille l'analyse des solutions existantes *i.e.* la présentation et comparaison de ces différentes solutions. Celles-ci sont divisées en deux catégories. La première reprend les solutions dites *lourdes*, *i.e.* qui reposent sur des logiciels permettant de faire beaucoup plus que du copier/coller *e.g.* les *bureaux virtuels*. La seconde reprend les solutions *potentiellement légères i.e.* qui sont des logiciels qui *a priori* conviendraient comme solution au problème.

Tout d'abord seront étudiés un ensemble de logiciels et protocoles donnés comme mots-clés par le directeur de projet au début de celui-ci. Ceux-ci sont principalement liés à des technologies lourdes et seront majoritairement ceux entrant dans la partie des solutions lourdes. Les autres solutions lourdes seront surtout des solutions plus conventionnelles comme l'échange d'e-mails ou le partage de fichiers.

2.1 Solutions lourdes

Les solutions étudiées dans cette section seront d'abord l'échange d'e-mails, l'utilisation d'un serveur *FTP* (File Transfer Protocol) et le partage de fichiers. Les autres solutions étudiées seront *Remote Desktop Service*, *VNC*, *Citrix XenApp* et la technologie *NX*. La comparaison de ces solutions est résumée dans la table 2.1.

2.1.1 Échange d'e-mails

Une première solution qui vient rapidement à l'esprit est l'utilisation du courrier électronique. Il est en effet facile d'échanger du texte ou un fichier quelconque entre deux machines en utilisation celui-ci. Par exemple il suffit simplement d'envoyer à sa propre adresse le texte à copier/coller ou bien de joindre le fichier que l'on désire transférer. Cette solution à l'avantage d'être utilisable sur n'importe quelle machine (et n'importe quel système d'exploitation) sans l'installation d'un logiciel particulier autre qu'un client mail.

Cependant ceci reste très fastidieux. Outre l'utilisation d'un outil qui n'est pas prévu pour faire du copier/coller, ceci nécessite l'installation d'un serveur mail sur le réseau local, ou bien un accès à Internet. Cet accès à Internet pose d'ailleurs un problème important à l'heure actuelle où les connexions fournies aux particuliers ne proposent pas une vitesse d'envoi très élevée. Un réseau intranet câblé en *Fast Ethernet* permettra généralement un débit symétrique d'au moins 100 Mbits/s, alors qu'une connexion ADSL standard dépasse rarement les quelques Mbits/s en upload. Les différences de latence entre l'internet et l'intranet risquent aussi d'être fort importantes.

Ceci permet de montrer que, non seulement il n'est pas nécessaire d'utiliser Internet pour échanger des données entre deux machines qui sont normalement situées dans la même pièce, mais que cela peut en plus apporter une perte de performances si l'on dispose d'une connexion à faible débit [1]. L'utilisation d'Internet sera donc écartée par la suite pour ces raisons et seule l'utilisation sur un réseau local sera considérée.

2.1.2 Utilisation d'un serveur FTP

Un moyen de faire du copier/coller entre plusieurs plateformes est l'utilisation d'un serveur FTP. Pour cela il faut qu'un serveur soit installé sur le réseau local. De même chaque client devra disposer d'un client FTP. Sur Unix il est aussi possible de se passer d'un serveur FTP et d'utiliser un serveur *SSH* (Secure Shell Client), le protocole *SFTP* (SSH File Transfer Protocol) permettant d'utiliser un serveur SSH à la manière d'un serveur FTP sécurisé.

La solution du serveur FTP a donc comme contrainte l'installation du logiciel serveur. Celle-ci peut être en partie résolue sous Unix grâce à SSH. En effet il peut être assez fréquent d'avoir un serveur SSH installé en local pour l'utilisateur Unix utilisant plusieurs machines simultanément. Cependant le fait de créer un fichier pour copier/coller du texte engendre une lourdeur non désirée. Cette dernière remarque s'applique aussi pour le partage de fichiers.

2.1.3 Partage de fichiers

Le partage de fichier permet d'accéder à un répertoire se trouvant sur une machine A à partir d'une machine B se situant sur le même réseau. Celui-ci peut être mis en oeuvre de plusieurs manières. Sur Windows il est implémenté nativement via le protocole SMB. Sous Unix le protocole *NFS* (Network File System) est sans doute le plus répandu. Une alternative peut aussi être l'utilisation de SSH et *SSHFs* (SSH Filesystem qui permet de monter un dossier sur une machine distante comme si c'était un périphérique et qui repose sur SFTP). Le partage de fichiers de Windows n'étant pas compatible sous Unix et *vice versa*, l'utilisation de

1. Ce qui est actuellement le cas en Belgique ainsi que dans la plus grande partie du monde où la fibre optique et les connexions symétriques sont peu répandues

Samba [Samrs] sera obligatoire s'il est nécessaire de supporter ces deux mondes. Cette solution présente aussi les mêmes désavantages que l'utilisation de FTP.

2.1.4 Remote Desktop Services

Remote Desktop Services, autrefois *Terminal Services* est un composant de Microsoft Windows permettant l'utilisation d'un ordinateur à distance tournant sous Windows[Wik10e]. Ce système est basé sur un modèle client-serveur. Le serveur est appelé Terminal Server et est inclus dans Windows. Il est à noter que seule la version serveur de Windows permet une configuration avancée du programme serveur. Le protocole utilisé est appelé *RDP* (Remote Dekstop Protocol) et peut être transporté dans un tunnel *TLS* (Transport Layer Security, anciennement *SSL*, Secure Sockets Layer) afin d'améliorer la sécurité du protocole. L'utilisation de ce protocole est possible sous les systèmes d'exploitation basés sur Unix grâce à l'implémentation libre *rdekstop* [rders] du client. Le protocole et le serveur supportent le partage du presse-papier, cependant il est évident que cette solution est inadéquate pour être utilisée comme copier/coller multi-plateformes.

2.1.5 VNC

VNC (Virtual Network Computing) [Wik10f] est un système logiciel permettant d'utiliser un ordinateur à distance. Il a comme avantage sur le protocole de Microsoft d'être libre de droits et d'être indépendant du système d'exploitation. Bien que non sécurisé par défaut, il existe différents moyens de le sécuriser *e.g.* via une connexion SSH ou *VPN* (réseau privé virtuel). Cette solution souffre des mêmes problèmes que Remote Desktop Services, c'est-à-dire qu'il permet de faire bien plus que du copier/coller et allourdi le système.

2.1.6 Citrix XenApp

Citrix XenApp [Wik10a] est un ensemble de produits permettant de virtualiser des applications sur différentes machines. Il distribue des services tournant généralement sur un ou plusieurs serveurs à des clients dits légers, *i.e.* qui n'ont pas besoin d'avoir une grande quantité de ressources matérielles disponibles pour exécuter les applications, vu que celles-ci sont exécutées sur un serveur central. Contrairement à VNC qui ne distribue que ce qui est affiché, XenApp fonctionne de manière semblable à *X11* [2]. Cependant ceci ne l'empêche pas de souffrir de problèmes déjà évoqués, tels un overhead important lorsque l'on veut faire du copier/coller et l'utilisation de license propriétaire.

2. X Window, X11 ou X est le système graphique standard de Unix fonctionnant sous forme de serveur et où chaque application graphique est un client

2.1.7 Technologie NX

NX [Wik10b] est un protocole d'accès distant à X11 reposant sur un modèle client-serveur et utilisant SSH pour la sécurité. L'implémentation de base *NoMachine NX* est propriétaire mais une implémentation libre *FreeNX* [Frers] existe. Tout comme les solutions précédentes, NX permet bien plus que le copier/coller et souffre donc des mêmes problèmes.

2.1.8 Comparaison

En résumé, toutes ces solutions lourdes présentées reposent sur un modèle client-serveur et sont conçues, soit pour fournir un service qui n'est pas prévu pour être utilisé afin de réaliser du copier/coller, soit pour être utilisées comme bureau virtuel distant. Certaines sont plus simples que d'autres à mettre en oeuvre ; d'autres sont plus sécurisées, tandis que d'autres sont propriétaires ou visent un système d'exploitation particulier. Mais elles sont toutes inadaptées pour effectuer un copier/coller multi-plateformes simple. Leurs caractéristiques sont résumées dans la table 2.1.

2.2 Solutions *a priori* légères

Les solutions présentées dans cette section sont pour la majorité des solutions trouvées en faisant des recherches sur Google sur base de mots clés français et anglais. Celles-ci ont permis de trouver des logiciels qui conviendraient au premier abord en fournissant un moyen simple de faire du copier/coller en réseau. Les logiciels présentés seront *ClipboardMultiSharer*, *Clipboard Share*, *The Network Clipboard* et *Remote Clip*. La comparaison de ces solutions est résumée dans la table 2.2.

2.2.1 ClipboardMultiSharer

ClipboardMultiSharer [Clirsc] est un logiciel écrit en *Java* et en *C#* qui permet de faire du copier/coller entre plusieurs ordinateurs. Celui-ci supporte le copier/coller de texte et d'image et repose sur l'utilisation d'un fichier partagé en réseau. Ceci en fait donc en réalité une solution lourde vu qu'il requiert l'utilisation du partage de fichiers. De plus, bien que le programme soit écrit en Java, la portabilité du programme dépend du type de partage de fichiers utilisé. Il sera donc sans doute nécessaire d'utiliser Samba s'il est nécessaire de travailler entre Unix et Windows.

2.2.2 Clipboard Share

Clipboard Share [Shars] est un autre logiciel qui permet de faire du CCMP. La connexion est chiffrée et il est possible de recevoir du contenu en provenance d'un envoyeur de confiance. Il est écrit en C#, requiert *Microsoft .NET 3.5* ainsi que *PNRP* (*Peer Name Resolution Protocol*), un protocole *P2P* (Peer-to-Peer, pair à pair en français, le principe de ce type de réseaux est décrit à la page 12) propriétaire de Microsoft, ce qui signifie que le logiciel ne tourne que sur Windows XP SP2 ou plus récent [Wik10c]. Bien qu'il propose des fonctionnalités intéressantes et réponde au critère de logiciel léger, il ne convient pas car il ne vise que Windows et repose sur des technologies propriétaires.

2.2.3 The Network Clipboard

The Network Clipboard [Clirsb] est lui écrit en *C++* et fonctionne aussi bien sous Windows que sous Linux. Le protocole mis en place permet d'utiliser l'application en P2P grâce au *broadcast IP*. Cependant il possède plusieurs défauts qui l'empêchent d'être un bon candidat. Premièrement le contenu n'est pas chiffré sur le réseau et aucun moyen d'authentification ne semble être mis en oeuvre pour sécuriser le système. Ensuite il ne semble pas certain que le programme tourne sur tous les Unix, *e.g.* il n'est fait mention nulle part d'une compatibilité assurée avec *MacOS*. Enfin, il faut tout de même noter que The Network Clipboard utilise la version 3 de *Qt*[3] et qui, avant la version 4, n'était libre que sous Linux [Wik10d]. Le programme repose donc sur une librairie propriétaire sous les autres systèmes d'exploitation.

2.2.4 Remote Clip

Remote Clip [Clirsa] est à la base un outil pour synchroniser du contenu entre Windows et un *PDA Palm*. Celui-ci existe aussi en version Java et lui permet ainsi d'être portable. Il repose sur une architecture P2P dont le fonctionnement est expliqué dans un article de Robert C. Miller et Brad A. Myers [MM99]. Les connexions sont également chiffrées via TLS à partir de la version 1.4 de Java et l'ajout d'un pair dans un groupe de partage du presse-papier nécessite l'accord explicite de la machine gérant celui-ci. Il permet de gérer aussi bien le texte que les fichiers et est distribué sous licence libre. Remote Clip semble donc être le logiciel répondant à toutes les exigences requises mais contrairement aux solutions citées plus haut, le projet ne semble plus actif. En effet la dernière version du logiciel est datée de juillet 2002. Cela signifie que de potentielles failles de sécurité ne seront pas corrigées et que le logiciel ne sera pas amélioré.

3. framework C++ utilisé entre autre dans le projet *KDE*

2.2.5 Comparaison

En résumé toutes les solutions présentées ont chacune des qualités et des défauts. ClipboardMultiSharer repose sur du partage de fichiers (et donc un modèle client-serveur). Clipboard Share repose sur des technologies fermées et ne tourne que sous Windows. The Network Clipboard n'est pas sécurisé. Seul Remote Clips répond réellement aux exigences du projet même si celui-ci est quelque peu vieillissant. Pour cette raison c'est les concepts de ce dernier qui seront principalement utilisés pour développer le projet. Les caractéristiques de l'ensemble des logiciels sont résumées dans la table 2.2.

Solution	Service rendu	Architecture	Sécurité	Indépendance de la plateforme	Ouverture de la solution
E-mails	E-mails	Client-serveur	Dépend du protocole	Oui	Protocoles ouverts
FTP	Transfert de fichiers	Client-serveur	SSH grâce à SFTP	Oui sauf SFTP	Protocoles ouverts
Partage de fichiers	Partage de fichiers	Client-serveur	SSH sous Unix	Oui grâce à Samba	Libre sous Unix, fermé sous Windows
RDS	Bureau distant	Client-serveur	Tunnel TLS possible	Windows mais clients Unix existants	Protocole propriétaire
VNC	Bureau distant	Client-serveur	Possibilité d'utiliser SSH ou un VPN	Oui	Logiciel libre
XenApp	Bureau distant	Client-serveur	Possibilité d'utiliser HTTPS	MS Windows Server, HP-UX, Solaris, AIX	Logiciel propriétaire
NX	Bureau distant	Client-serveur	Oui	Vise Unix	Implémentation libre FreeNX

TABLE 2.1 – Comparaison des solutions lourdes

Solution	Service rendu	Architecture	Sécurité	Indépendance de la plateforme	Ouverture de la solution
ClipboardMultiSharer	CCMP	Client-serveur	Partage de fichiers	Java + partage de fichiers	Logiciel libre
Clipboard Share	CCMP	P2P	Connexion cryptée + envoyeur de confiance	Windows (.NET 3.5 + PNRP)	Logiciel libre mais technologies MS
The Network Clipboard	CCMP	P2P	aucune	Linux + Windows	Logiciel libre
Remote Clip	CCMP	P2P	TLS + validation de connexion d'un pair	Java	Logiciel libre
Solution	Service rendu	Architecture	Sécurité	Indépendance de la plateforme	Ouverture de la solution

TABLE 2.2 – Comparaison des solutions légères

Chapitre 3

Solution proposée

Dans le chapitre précédent, les différentes solutions existantes pour faire du CCMP ont été analysées. La solution jugée la plus efficace est Remote Clip, celle-ci proposant une architecture P2P adéquate et un niveau de sécurité correct. Pour ces raisons, les idées mises en avant par Remote Clip seront réutilisées.

3.1 Principes d'une architecture P2P

Avant de décrire l'architecture qui sera utilisée, il est préférable de rappeler quels sont les principes de base d'une architecture P2P. Habituellement, une architecture client-serveur est utilisée lorsqu'il faut fournir un service à un ensemble de machines connectées en réseau, c'est-à-dire que chaque client va se connecter à un (ou parfois plusieurs) serveur central qui s'occupera de fournir le service désiré. Par opposition à ce mode de fonctionnement centralisé, un réseau organisé en pair-à-pair permet de se passer de serveur central, chaque client jouant à la fois le rôle de client et de serveur. La figure 3.1 illustre la différence de topologie réseau existante entre ces deux types d'architectures réseau. De manière plus précise les systèmes P2P sont définis dans [ATS04] comme étant :

> des systèmes distribués constitués de noeuds interconnectés, capables de s'auto-organiser dans des topologies de réseaux avec comme but le partage de ressources, telles que le contenu, les cycles CPU, le stockage, la bande passante tout en ayant la capacité de s'adapter aux erreurs et de s'accommoder de populations de noeuds transitoires ; tout en maintenant une connectivité et des performances acceptables sans requérir l'intermédiaire ou le support d'un serveur ou d'une autorité centralisée globalement.

Les difficultés potentielles à mettre en avant dans ce genre de systèmes sont la gestion des va-et-vient de clients et la tolérance aux erreurs sans l'aide de serveur central, et tout en minimisant la charge sur le réseau, due à cette gestion. Il faut cependant noter que l'importance de la tolérance aux erreurs est tout de

même assez minime dans le contexte du copier/coller. Les données stockées sont en général destinées à être utilisées à court terme et non pas à être stockées de manière persistante. De même le nombre de pairs est normalement peu élevé, et donc le nombre de va-et-vient n'est pas aussi important que sur des systèmes à grande échelle.

Une remarque supplémentaire à faire est qu'une différence est souvent faite entre réseaux P2P structurés et non structurés [ATS04 ; Lua+05]. Les premiers ont une topologie contrôlée de manière précise permettant de placer chaque donnée à un endroit précis et d'effectuer des recherches efficaces. Le réseau ici étant censé être de taille relativement petite et le contenu changeant très rapidement, il n'est pas nécessaire de structurer le réseau, ceci risquant de créer un overhead important et non nécessaire en complexité. En effet les réseaux non structurés utilisent souvent le mécanisme de flooding plus simple à comprendre et à mettre en oeuvre qu'une table de hashage distribuée.

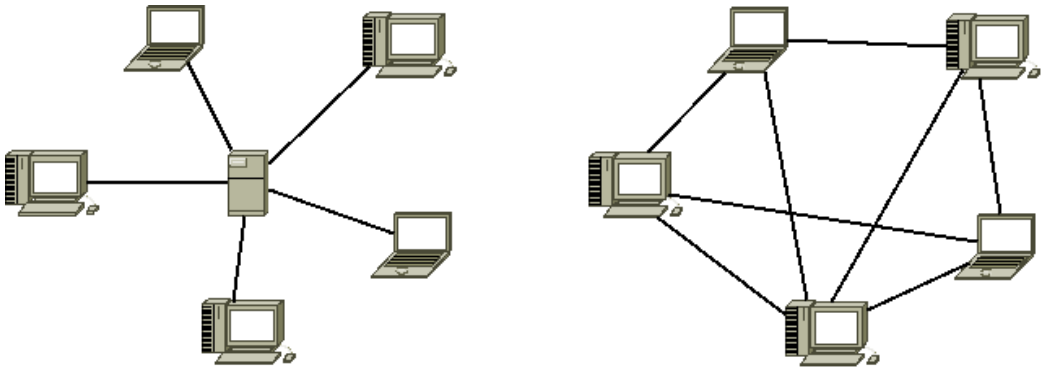

FIGURE 3.1 – Exemple de différence entre une topologie client-serveur et P2P

3.2 Architecture logicielle

Afin de suivre le principe *KISS* (Keep It Simple Stupid, *cf.* Annexe B) et la philosophie Unix, le projet sera divisé en plusieurs programmes, chacun d'entre eux s'occupant d'une tâche particulière. Une autre raison justifiant une telle conception est le fait qu'elle introduit la possibilité de développer le projet de manière incrémentale, en se concentrant d'abord sur l'aspect réseau et ensuite sur l'implémentation propre à un environnement particulier.

3.2.1 Client P2P

Premièrement un logiciel s'occupant uniquement de la gestion du presse-papier sur le réseau est nécessaire. Celui-ci a comme rôle de se connecter aux autres pairs et de s'organiser avec ceux-ci quand cela est nécessaire. Sa seule fonctionnalité est en fait de gérer la partie réseau du système, toute interaction

locale (*i.e.* avec l'utilisateur) se fait par l'intermédiaire d'autres logiciels qui seront définis par la suite.

Il faut définir de manière plus précise comment l'ensemble des clients P2P vont s'organiser afin de s'informer pour savoir quel client *détient* le presse-papier, comment un pair entre dans le réseau et comment détecter qu'un pair a quitté celui-ci (volontairement ou pas). Des exemples de fonctionnement de cette architecture et du protocole qui seront mis en oeuvre sont montrés dans les figures 3.2 et 3.3.

Gestion du presse-papier en P2P

Il y a principalement deux possibilités pour gérer le presse-papier sur le réseau. Lorsqu'un copier est effectué en local, le client P2P doit prévenir les autres pairs qu'il détient le presse-papier. La première option est d'envoyer en même temps la donnée copiée à tous les pairs. La deuxième est de n'envoyer cette donnée que lorsque qu'un coller est effectué chez un pair, celui-ci peut alors demander la donnée au pair détenant le presse-papier.

La première manière de faire a les avantages et inconvénients suivants :

- Avantages :
  - Possibilité de garder une copie du presse-papier sur chaque pair.
  - Tolérance aux erreurs dans le cas où le pair détenant le presse-papier aurait quitté le réseau.
  - Absence de communications réseaux en cas de multiples coller.
- Inconvénients :
  - Consommation de bande passante accrue dans le cas où des copiers sont effectués fréquemment car il faudra envoyer la même donnée à chacun des pairs qui ne feront peut être pas forcément de coller. De plus la consommation de bande passante dépend du nombre de pair présent sur le réseau.

La seconde a les avantages et inconvénients suivants :

- Avantages :
  - Il n'est pas nécessaire de notifier l'ensemble des pairs à chaque copier : si un pair effectue plusieurs copies d'affilée, il ne faudrait notifier le réseau qu'une seule fois et sans envoyer la moindre donnée.
  - En général le coller ne sera effectué que par un seul pair, il n'a donc qu'à demander lui même au pair détenant le presse-papier de la lui envoyer.
- Inconvénients :
  - Impossibilité de récupérer le presse-papier d'un pair ayant quitté le réseau.
  - Si plusieurs collers ont lieu les uns à la suite des autres, il faudra demander plusieurs fois la même donnée au pair détenant le presse-

papier.

Il est donc nécessaire de faire un choix entre une de ces options et c'est la seconde qui est choisie. C'est ainsi que fonctionne non seulement Remote Clip, mais aussi X Window[Nye92] à la différence que c'est une architecture client-serveur qui est utilisée. En effet, X Window est un serveur qui considère chaque fenêtre comme un client. Lorsque l'utilisateur copie un élément de cette fenêtre, elle notifie le serveur X qu'elle détient le presse-papier. Ensuite lorsque l'utilisateur veut coller le contenu du presse-papier, la fenêtre dans laquelle est effectuée le coller doit demander au serveur X quel client détient le presse-papier et ensuite s'adresser à celui-ci pour l'obtenir. L'application devant viser en priorité le monde Unix, c'est ce principe qui est adapté à une architecture sans serveur. Il faut aussi noter que sous X, il n'est pas possible de savoir si le presse-papier a changé : si deux copies sont effectuées au sein de la même application, il n'y a qu'une seule notification qui est faite auprès des autres fenêtres, il faudrait donc constamment vérifier que celui-ci a changé ou pas afin de le synchroniser sur le réseau.

Rejoindre le réseau

Outre un moyen d'authentification, il est nécessaire de définir comment un pair peut rejoindre le réseau afin de partager son presse-papier. Pour cela il doit connaître l'adresse IP et l'identifiant d'au moins un pair. Une fois authentifié, le *parent* contacté lui envoie la liste des autres pairs faisant partie du réseau en précisant lequel d'entre eux détient le presse-papier. Une fois cette liste reçue, le *fils* informe l'ensemble des autres pairs de son arrivée sur le réseau tout en s'authentifiant auprès de ces pairs. Une fois ceci fait, le pair est considéré comme faisant partie du réseau et garde chaque connexion ouverte. Ceci crée une topologie où chaque pair possède une connexion ouverte avec tous les autres noeuds, ce qui signifie que pour un réseau de n pairs, chaque pair doit maintenir $n - 1$ connexions TCP sécurisées. Ceci pourrait devenir une charge importante à supporter dans un grand réseau, mais en général celui-ci est restreint à quelques pairs.

Quitter le réseau

Lorsqu'un pair quitte le réseau, il doit notifier l'ensemble des autres pairs de son départ. Une fois ceci fait, il peut fermer chaque connexion ouverte avec chacun des pairs. En revanche il se peut qu'un pair quitte le réseau sans pouvoir notifier les pairs de son départ, *e.g.* à cause d'une déconnexion du lien physique. Pour cette raison, lorsqu'un pair n'est plus joignable, celui-ci est considéré comme sorti du réseau et la connexion avec celui-ci peut être coupée.

3.2.2 Client local

Le client P2P est le *front-end* avec le réseau, il est chargé de communiquer avec les pairs du réseau afin de les découvrir et savoir lequel détient le presse-papier. Le front-end avec l'utilisateur est en revanche le client local. Celui-ci tourne sur le même ordinateur que le client P2P, se charge de notifier le client P2P lorsque l'utilisateur copie quelque chose et le notifie lorsqu'il désire accéder au presse-papier. Le client P2P notifie le client local lorsqu'un pair du réseau détient le presse-papier c'est-à-dire lorsque ce pair effectue un copir. Un exemple d'interaction entre clients locaux et clients P2P utilisant le protocole qui sera mis en oeuvre est montré sur la figure 3.4.

Cependant, contrairement au client P2P, il peut y avoir plusieurs clients locaux tournant sur le même ordinateur. Chacun d'entre eux étant destiné à un environnement précis. Dans ce projet, seront développés deux types de clients différents :

- Un client tournant en tâche de fond (comme *daemon*) et synchronisant le contenu du presse-papier de X Window.
- Un client gérant le copier/coller dans un terminal, composé de trois logiciels :
  - Un daemon gérant le presse-papier, cette fonctionnalité n'étant en général pas implémentée dans un shell, seuls les émulateurs de terminaux peuvent communiquer avec celui de X Window.
  - Une commande permettant de copier une donnée à partir de l'entrée standard du shell.
  - Une commande permettant de coller une donnée sur la sortie standard du shell.

Il est à noter que dans le cas du premier client, le rôle des deux commandes est en fait joué par les clients X Window *i.e.* les fenêtres ouvertes et réalisant le copier/coller.

3.3 Protocoles

Deux protocoles sont à définir afin de faire fonctionner l'application correctement. Le premier est celui utilisé sur le réseau par les clients P2P afin de communiquer entre eux. Le second est utilisé entre le client P2P et les clients locaux afin de se notifier mutuellement de changements dans le presse papier. De même ils doivent tous les deux prendre en compte l'aspect sécurité, en proposant un moyen d'identification. Ces deux protocoles seront appelés respectivement protocole P2P et protocole local.

Les protocoles sont caractérisés par un ensemble de types de messages. Ceux-ci sont décrits en utilisant la syntaxe suivant :

```
MSG <SP> <VAR1> <SP> <VAR2> ... <SP> <VARN> <LF>
```

MSG définit le type du message. Celui-ci contient N variables <VAR1>, <VAR2>, . . ., <VARN>. Chacune est séparée par un espace (<SP>) et le message se termine par un passage à ligne (<LF>). Chaque variable est ensuite décrite et le type de messages pouvant être reçu comme réponse est décrit.

3.3.1 Protocole P2P

AUTH

Afin de s'authentifier lors de l'ouverture d'une connexion TCP/TLS, un identifiant est choisi pour le pair et un mot de passe est défini dans la configuration du client. Lorsqu'un pair veut contacter pour la première fois un pair, il doit s'authentifier auprès de celui-ci. Pour cela il envoie un message d'authentification :

```
AUTH <SP> <NAME> <SP> <PSWD> <LF>
```

Variable NAME : identifiant de l'hôte à joindre.

Variable PSWD : mot de passe.

Réponse : si le mot de passe est correct *i.e.* est le même utilisé par les deux pairs, le pair contacté peut accepter la connexion et répondre par un message de type OK, sinon il répond par un message de type KO avec un code d'erreur 1.

OK

Ce type de message permet de confirmer la plupart des opérations :

```
OK <LF>
```

KO

Ce type de message permet de signaler un refus ou une erreur :

```
KO <SP> <ERRNO> <LF>
```

Variable ERRNO : code d'erreur pouvant être :

- **0** erreur non définie/générique.
- **1** échec d'authentification.
- **2** pair non valide.

JOIN

Un message de type JOIN permet de rejoindre le réseau :

```
JOIN <SP> <TYPE> <SP> <NAME1> <SP> <NAME2> <LF>
```

Variable NAME1 : identifiant du pair.

Variable NAME2 : identifiant du pair contacté.

Variable TYPE : 0 signifie que le pair n'a pas encore obtenu la liste des pairs

1 signifie que le pair a déjà la liste des pairs

Réponse : si l'identifiant du pair contacté n'est pas le bon, la réponse est un message KO avec un code d'erreur 2, si celui-ci est bon la réponse est un message de type LIST si le pair a besoin de la liste des pairs, sinon un message de type OK.

LIST

Un message de type LIST annonce une liste de messages de type PEER :

```
LIST <SP> <N> <LF>
```

Variable N : nombre de messages PEER qui suivent.

PEER

Un message de type PEER décrit ce qui identifie un pair :

```
PEER <SP> <NAME> <SP> <IP> <SP> <PORT> <LF>
```

Variable NAME : identifiant du pair permettant d'effectuer un JOIN.

Variable IP : adresse IP permettant de joindre le pair.

Variable PORT : port sur lequel le pair écoute.

LEAVE

Un message de type LEAVE doit être envoyé pour quitter le réseau et ne nécessite aucune réponse :

```
LEAVE <LF>
```

COPY

Un message de type COPY est envoyé lorsque le pair détient le presse-papier :

```
COPY <LF>
```

PASTE

Un message de type PASTE est envoyé lorsqu'un pair doit obtenir le presse-papier :

`PASTE <LF>`

Réponse : Un message de type DATA. Si le pair ne possède pas le presse-papier un message de type PEER doit être envoyé avec les informations du pair détenant le presse-papier.

DATA

Un message de type DATA permet d'envoyer le contenu du presse-papier :

`DATA <SP> <TYPE> <SP> <LENGTH> <SP> <CONTENT> <LF>`

Variable TYPE : cette variable permet de préciser le type de donnée et ainsi d'étendre facilement le protocole afin de supporter d'autres types de données. Dans ce cas il faudra préciser comment ce type de données est encodé.

0 type non défini/inconnu.

1 texte.

Variable LENGTH : longueur de la donnée.

Variable CONTENT : contenu de la donnée dont la longueur doit vérifier la variable LENGTH.

3.3.2 Protocole local

Le protocole local utilise un sous-ensemble des messages du protocole P2P, celui-ci ayant seulement besoin de notifications de copier/coller et de s'authentifier. Les messages utilisés seront ceux de type AUTH, OK, KO, COPY, PASTE et DATA. Les messages de type AUTH, OK et KO sont toujours utilisés pour l'authentification.

La sémantique des messages de type COPY change légèrement : le client local envoie un tel message lorsque l'utilisateur copie une donnée dans le presse-papier et le client P2P envoie un COPY au client local lorsque c'est un autre pair (ou un autre client local) qui détient le presse-papier.

Les messages de type PASTE sont quant à eux envoyés lorsque le client local désire faire un coller et qu'il ne détient pas le presse-papier ou bien lorsqu'un autre pair désire obtenir le contenu du presse-papier détenu par le client local. Les messages de type DATA ont toujours la même sémantique : servir de réponse à un message de type PASTE.

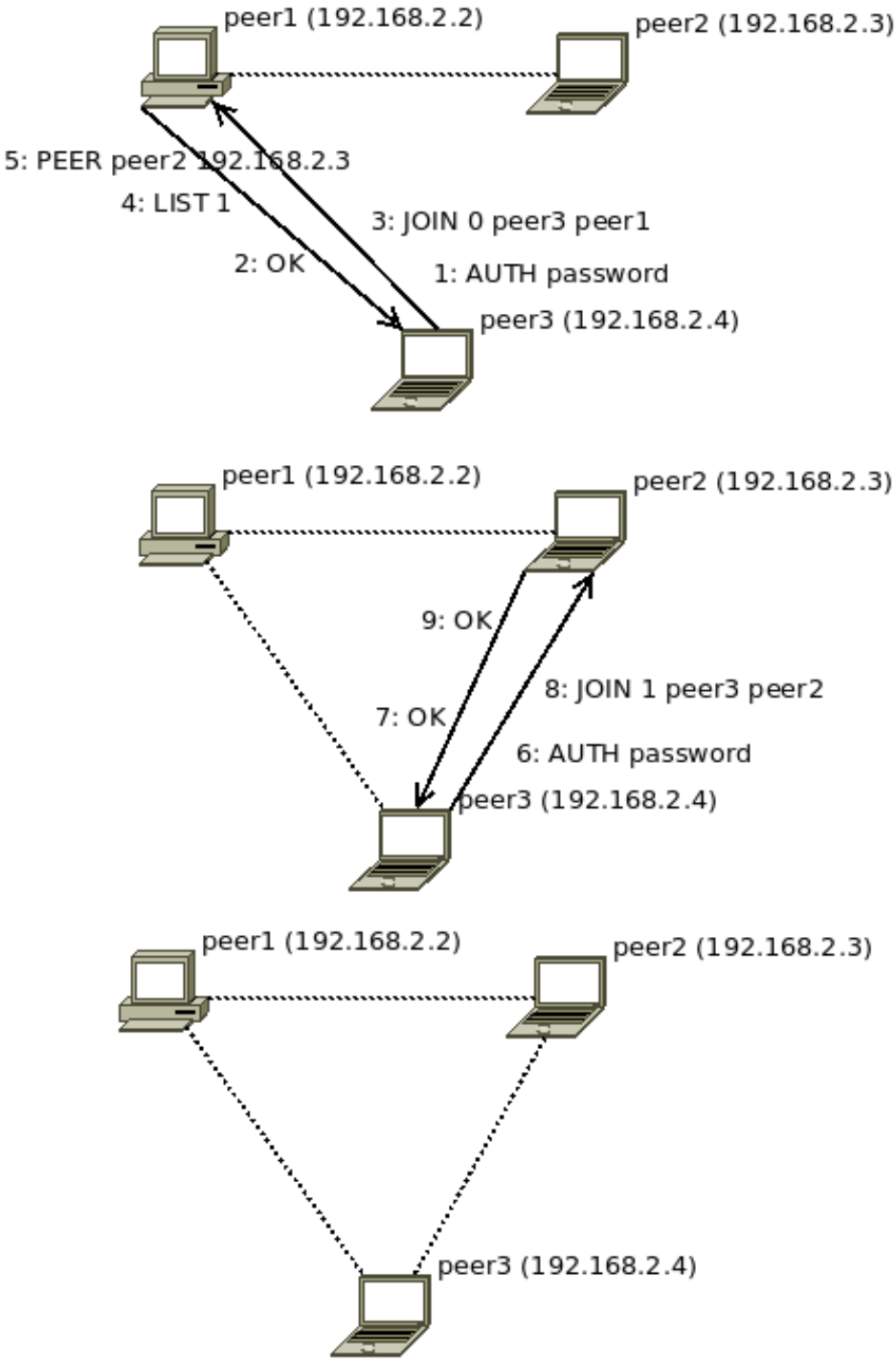

FIGURE 3.2 – Le pair 3 qui rejoint un réseau. Les lignes en pointillés représentent les pairs qui se connaissent mutuellement *i.e.* qui ont une connexion TCP/TLS ouverte, et les flèches les communications réseaux entre deux pairs. Les messages envoyés sont numérotés afin de suivre l'ordre dans lequel ils sont envoyés.

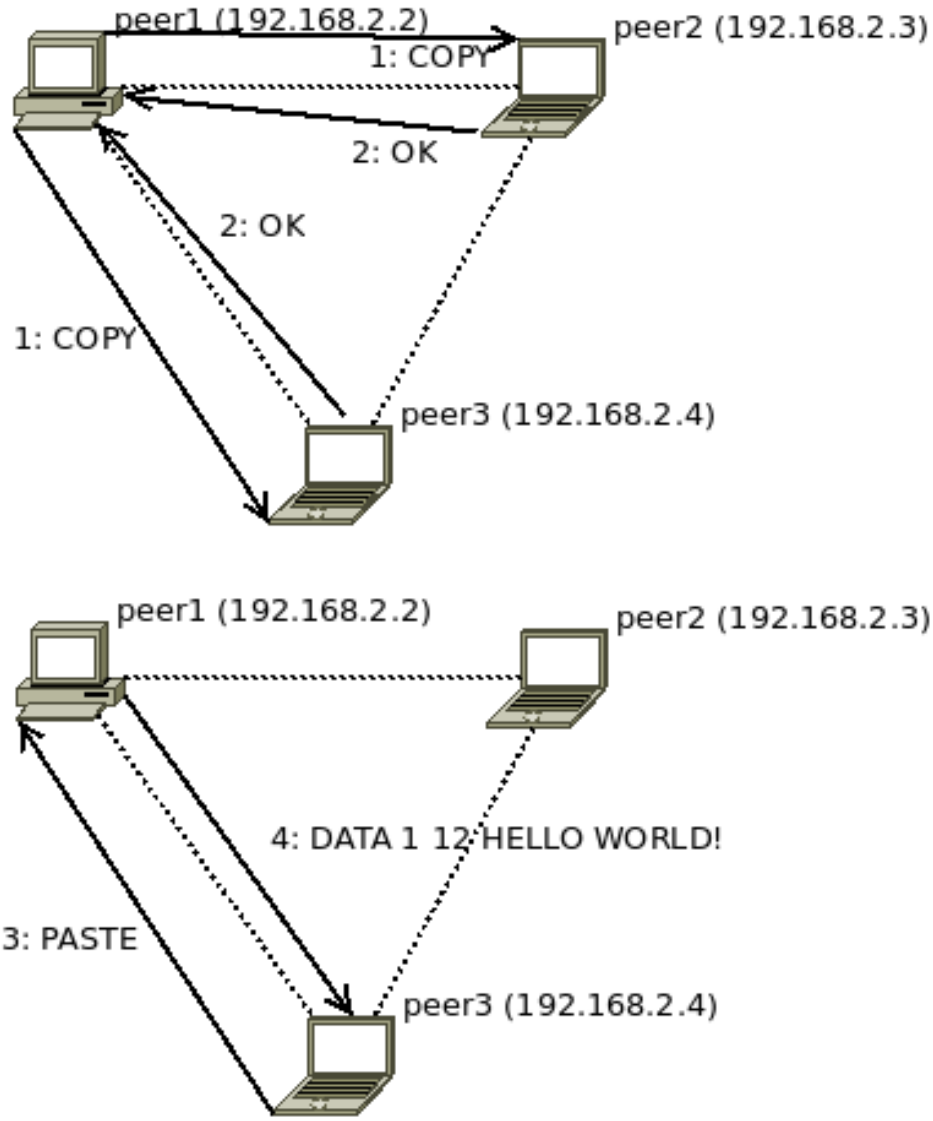

FIGURE 3.3 – Exemple de copier/coller : le pair 1 copie le texte HELLO WORLD ! et l'envoie au pair 3 qui fait un coller. Les lignes en pointillés représentent les pairs qui se connaissent mutuellement *i.e.* qui ont une connexion TCP/TLS ouverte, et les flèches les communications réseaux entre deux pairs. Les messages envoyés sont numérotés afin de suivre l'ordre dans lequel ils sont envoyés.

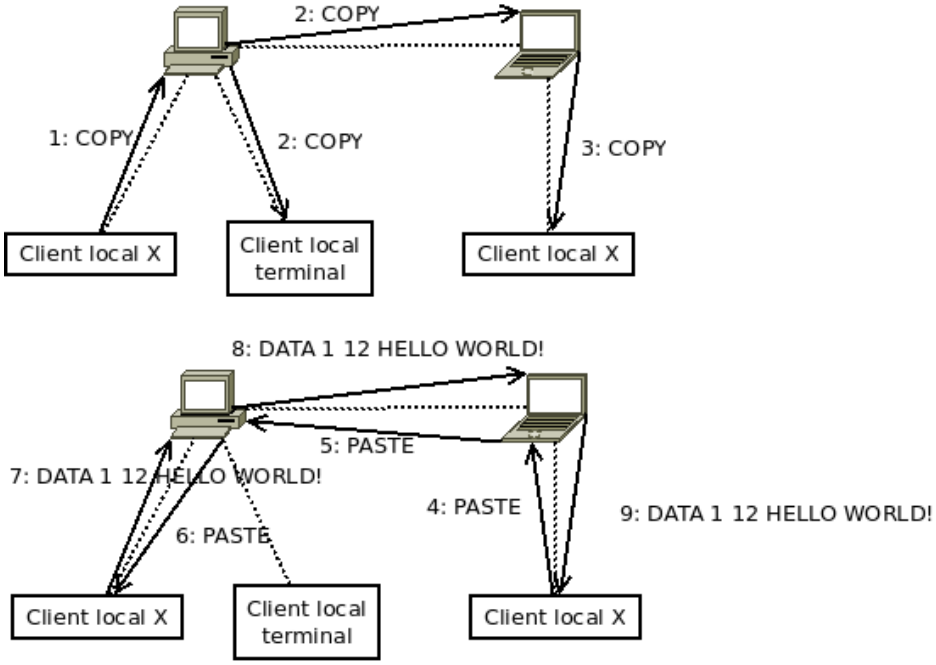

FIGURE 3.4 – Exemple de copier/coller avec clients locaux. Les lignes en pointillés représentent les pairs qui se connaissent mutuellement *i.e.* qui ont une connexion TCP/TLS ouverte, et les flèches les communications réseaux entre deux pairs. Les messages envoyés sont numérotés afin de suivre l'ordre dans lequel ils sont envoyés.

Conclusion

Finalement, après avoir examiné le problème, les solutions existantes pour le résoudre se sont avérées pour la plupart inefficaces. Certaines ont pourtant de bonnes idées qui ont été reprises afin d'imaginer une architecture et une ébauche de protocole permettant à des pairs de synchroniser leur presse-papier sur un réseau. Grâce à cela, il est désormais possible d'implémenter ce programme afin de fournir une solution efficace.

Pour cela le langage C++ sera utilisé, celui-ci permettant d'avoir un programme compilé optimisé pour un système précis, il permet de mieux satisfaire la contrainte de solutions légères qu'avec un langage interprété par une machine virtuelle. De même le C++ offre l'avantage de donner l'accès directement à la Xlib et de communiquer facilement avec le serveur X Window.

La librairie Qt, célèbre pour être utilisée dans l'environnement de bureau KDE, sera aussi utilisée. Celle-ci permet cependant de faire bien plus que des programmes fenêtrés. Elle contient entre autre un ensemble de classes permettant d'établir des connexions TCP/TLS de manière indépendante du système d'exploitation. Par manque d'expérience avec cette librairie, il n'est cependant pas possible de dire ce qu'elle pourrait apporter en plus, celle-ci contenant un grand nombre d'outils différents.

Enfin les prochains mois seront consacrés, tout d'abord à l'étude de la librairie Qt (et en particulier de sa partie concernant les sockets). Ensuite une fois ceci fait, il sera possible d'implémenter l'ensemble des logiciels à fournir de manière incrémentale. Le client P2P sera d'abord développé afin d'offrir un protocole de base fonctionnel. Une fois celui-ci implémenté et testé, il sera alors possible de passer à l'implémentation de trois types de clients locaux.

Un premier permettant une utilisation simple sur l'entrée et la sortie standard d'un terminal Unix. Le second devra synchroniser le presse-papier de X Window. Le dernier fournira une interface graphique basique pour les systèmes ne supportant pas X, il permettra de copier/coller du texte dans un champ texte et de le synchroniser sur le réseau. Cependant si une solution est trouvée afin de fournir un client permettant de synchroniser le presse-papier indépendemment du système d'exploitation, les deux clients n'en formeront qu'un seul.

Bibliographie

[ATS04] Stephanos ANDROUTSELLIS-THEOTOKIS et Diomidis SPINELLIS. “A Survey of Peer-to-Peer Content Distribution Technologies”. Dans : *ACM Computing Surveys* 36.4 (déc. 2004), p. 335–371. ISSN : 0360-0300. DOI : doi:10.1145/1041680.1041681. URL : http://www.spinellis.gr/pubs/jrnl/2004-ACMCS-p2p/html/AS04.html.

[Bro95] Frederick P. BROOKS. *The Mythical Man-Month : Essays on Software Engineering*. 20th Anniversary. Addison-Wesley, 1995. ISBN : 0201835959.

[Clirsa] Remote CLIP. *Remote Clip : copy-and-paste between Palms, Windows, and Unix*. (consulté le 18 décembre 2010). Version 3.1. URL : http://www.cs.cmu.edu/~rcm/RemoteClip/.

[Clirsb] The Network CLIPBOARD. *The Network Clipboard*. (consulté le 18 décembre 2010). Version 0.60. URL : http://netclipboard.sourceforge.net.

[Clirsc] CLIPBOARDMULTISHARER. *ClipboardMultiSharer*. (consulté le 18 décembre 2010). Version 1.1.2. URL : http://clipboardmshare.steweche.co.uk.

[Frers] FREENX. *FreeNX - the free NX*. (consulté le 18 décembre 2010). Version 0.9 (client) et 0.7.3 (serveur). URL : http://www.freenx.berlios.de.

[Han10] Filip HANIK. *The Kiss Principle*. (en ligne ; consulté le 19 décembre 2010). 2010. URL : http://people.apache.org/~fhanik/kiss.html.

[Lua+05] Eng Keong LUA et al. “A Survey and Comparison of Peer-to-Peer Overlay Network Schemes”. Dans : *IEEE Communications Surveys and Tutorials* 7 (2005), p. 72–93.

[MM99] R. C. MILLER et B. A. MYERS. “Synchronizing Clipboards of Multiple Computers”. Dans : *Proceedings UIST’99 : ACM SIGGRAPH Symposium on User Interface Software and Technology*. 1999, p. 65–66.

[Nye92] A. NYE. *Xlib programming manual : for version 11 of the X Window System*. Definitive guides to the X Window System. O'Reilly & Associates, 1992. ISBN : 9781565920026. URL : http://books.google.com/books?id=d8tByjvMmIwC.

[Prors] GNU PROJECT. *GNU Emacs*. (consulté le 19 décembre 2010). Version 23.2. URL : http://www.gnu.org/software/emacs/.

[Ray01] Eric S. RAYMOND. *The cathedral and the bazaar : musings on Linux and open source by an accidental revolutionary*. Revised. O'Reilly & Associates, Inc., 2001, p. xiv + 241. ISBN : 0-596-00131-2.

[rders] RDESKTOP. *rdesktop : A Remote Desktop Protocol client*. (consulté le 18 décembre 2010). Version 1.6.0. URL : http://www.rdesktop.org.

[Samrs] SAMBA. *Samba - opening windows to a wider world*. (consulté le 18 décembre 2010). Version 3.5.6. URL : http://www.samba.org.

[Shars] Clipboard SHARE. *Clipboard Share*. (consulté le 18 décembre 2010). Version 1.0.1. URL : http://clipboardshare.codeplex.com.

[Wik05] WIKIPEDIA. *Keep it Simple, Stupid — Wikipedia, The Free Encyclopedia*. (en ligne ; consulté le 19 décembre 2010). 2005. URL : http://en.wikipedia.org/w/index.php?title=Keep_it_Simple,_Stupid&oldid=17481579.

[Wik10a] WIKIPEDIA. *Citrix XenApp — Wikipedia, The Free Encyclopedia*. (en ligne ; consulté le 18 décembre 2010). 2010. URL : http://en.wikipedia.org/w/index.php?title=Citrix_XenApp&oldid=402691976.

[Wik10b] WIKIPEDIA. *NX technology — Wikipedia, The Free Encyclopedia*. (en ligne ; consulté le 18 décembre 2010). 2010. URL : http://en.wikipedia.org/w/index.php?title=NX_technology&oldid=398679460.

[Wik10c] WIKIPEDIA. *Peer Name Resolution Protocol — Wikipedia, The Free Encyclopedia*. (en ligne ; consulté le 18 décembre 2010). 2010. URL : http://en.wikipedia.org/w/index.php?title=Peer_Name_Resolution_Protocol&oldid=366357724.

[Wik10d] WIKIPEDIA. *Qt (framework) — Wikipedia, The Free Encyclopedia*. (en ligne ; consulté le 18 décembre 2010). 2010. URL : http://en.wikipedia.org/w/index.php?title=Qt_(framework)&oldid=402326618.

[Wik10e] WIKIPEDIA. *Remote Desktop Services — Wikipedia, The Free Encyclopedia*. (en ligne ; consulté le 18 décembre 2010). 2010. URL : `http://en.wikipedia.org/w/index.php?title=Remote_Desktop_Services&oldid=402329590`.

[Wik10f] WIKIPEDIA. *Virtual Network Computing — Wikipedia, The Free Encyclopedia*. (en ligne ; consulté le 18 décembre 2010). 2010. URL : `http://en.wikipedia.org/w/index.php?title=Virtual_Network_Computing&oldid=402696467`.

Annexe A

Codes sources utilisés

Afin de comprendre comment fonctionnent certains logiciels étudiés dans l'étude de faisabilité, certains codes sources disponibles en ligne ont été utilisés. Ceux-ci étant disponibles sous licences libres (GPL et MIT), ils sont non seulement disponibles en ligne mais peuvent aussi être réutilisés librement.

Bien que ces codes ne seront pas réutilisés pour des raisons évidentes (utilisation d'un environnement de développement différent), certains principes ont été compris grâce à ceux-ci. De même les principes de base utilisé dans le projet s'inspirent largement de ces logiciels, c'est pour cette raison qu'une annexe leur est consacrée.

ClipboardMultiSharer

Le code source Java de l'application sous licence GPL [1] a été parcouru afin de comprendre comment fonctionnait le logiciel, entre autre afin de voir comment il était possible d'accéder au presse-papier en Java. La révision consultée était la 16 datant du 10 juillet 2010.

Clipboard Share

Bien que disponible sous licence MIT [2], n'ayant pas de connaissances en C# et en MS .NET, le code de ce logiciel n'a été ni utilisé dans un but précis, ni lu.

1. `http://sourceforge.net/projects/clipboardmshare/develop`
2. `http://clipboardshare.codeplex.com/SourceControl/list/changesets`

The Network Clipboard

Le code source sous licence GPL [3] a été consulté dans le but d'examiner le protocole utilisé. La révision consultée était la 88.

Remote Clip

Le code source sous licence GPL est disponible avec la dernière version du logiciel [4]. Celui-ci a été consulté dans le but d'étudier le protocole réseau utilisé.

3. `http://sourceforge.net/projects/netclipboard/develop`
4. `http://www.cs.cmu.edu/~rcm/RemoteClip/RemoteClip-3.1.zip`

Annexe B

Principe KISS

Le principe *KISS* (Keep It Simple, Stupid !) [Wik05] est un principe prônant la simplicité, tout particulièrement dans le monde du design. Il est utilisé dans le développement logiciel dans le but d'exprimer le fait que la conception doit être simple et éviter toute complexité inutile.

Celui-ci est à la base même de la philosophie Unix, en effet celle-ci prône l'utilisation de petits utilitaires simples ayant chacun une fonctionnalité bien précise. Le projet sera donc développé de manière à fournir une implémentation suivant cette philosophie. Plusieurs auteurs [Bro95 ; Ray01] prônent cette philosophie sans pour autant y faire référence explicitement. *e.g.* Brooks explique par exemple que le design est d'une importance capitale car il permet de comprendre plus facilement le code source, Raymond dit qu'il faut concevoir les logiciels de manière ce qu'ils suivent la philosophie d'Unix. Il dit aussi que la perfection en conception n'est pas atteinte lorsque l'on n'a plus rien à ajouter mais plutôt lorsque l'on n'a plus rien à retirer.

Filip Hanik, un ingénieur logiciel membre de l'*Apache Foundation* parle aussi du principe KISS et donne des conseils pour appliquer ces principes en Java (mais ceux-ci sont applicables dans tous les langages de programmation) [Han10].