Marionnet : un logiciel graphique pour l'apprentissage et l'enseignement des réseaux locaux d'ordinateurs

Jean-Vincent Loddo
IUT de Villetaneuse, Laboratoire d'Informatique de Paris Nord (LIPN), Université Paris 13

***Abstract*— Les travaux pratiques de câblage, configuration et administration système et réseau (étude et test de services ou protocoles) nécessitent l'accès à des machines modifiables à loisir et à des composants spécialisés, tels que les répéteurs (hub) les commutateurs (switch), les routeurs (routers) et les éléments de câblage (câbles droits et croisés). D'une part, les structures d'enseignement technologiques sont confrontées au coût que représente un tel équipement si chaque étudiant doit en disposer et, d'autre part, le travail personnel des étudiants est impossible sans cet équipement. *Marionnet* permet de répondre à ces inconvénients en simulant d'une façon très fidèle des machines équipées d'un système *GNU/Linux* et en simulant les composants les plus couramment utilisés dans les réseaux locaux d'ordinateurs. Le logiciel est équipé d'une interface graphique permettant de définir, de lancer et de reconfigurer "à chaud" un réseau de façon très confortable et intuitive, et a été conçu pour un public à la fois d'étudiants et d'enseignants.**

I. Introduction

L'objectif pédagogique de Marionnet est de faciliter la pratique des réseaux d'ordinateurs, des protocoles de communication et des services ou applications réseau. Avec Marionnet il est possible de simuler le travail effectué dans une salle de travaux pratiques réseau à partir de la définition du réseau, c'est-à-dire de la phase de « câblage » des machines entre elles et avec les dispositifs de répétition (hub), de commutation (switch) et de routage ; ensuite, il est possible de simuler le travail de configuration du réseau, d'installation et réglage des services à activer ou de programmation des applications à exécuter sur l'ensemble des machines.

Marionnet est un logiciel libre qui tourne dans un environnement GNU/Linux en exploitant les fonctionnalités d'émulation[1] qui sont possibles dans ce système. Marionnet profite non seulement des capacités, de la robustesse et de l'efficacité de Linux, mais aussi du fait que Linux soit un logiciel libre et donc librement modifiable, donc adaptable à une cause particulière. L'émulation des machines est rendue techniquement possible sous Linux en exploitant une fonctionnalité particulière du noyau, le User Mode Linux (UML) (cf. [2], [3], [4]), qui permet de lancer un ou plusieurs autres noyaux par dessus le noyau principal (appelé noyau « hôte ») comme s'il s'agissait d'applications ordinaires. En effet, les noyaux ainsi lancés (appelés noyaux « uml ») tournent en espace utilisateur comme toute autre application. De plus, ils peuvent être associés à des fichiers particuliers dont le contenu est une image de disque dur. Autrement dit, le disque dur est en quelque sorte « enveloppé » dans un fichier tout à fait ordinaire du point de vue de la machine hôte. Ainsi, le lancement d'un noyau uml sur un fichier contenant l'image d'un disque dur simule exactement le démarrage réel d'un vrai système Linux sur un véritable ordinateur. Si on ajoute enfin la simulation des répéteurs (hub), des commutateurs (switch) et d'autres composants réseaux (routeurs, câbles, ...) par le biais de processus facilement programmables, on obtient une possibilité extraordinaire : celle de simuler fidèlement des réseaux locaux tels que ceux dont disposent les étudiants dans une salle de TP réseau.

Dans cet article nous décrirons les objectifs atteints dans ce projet d'un point de vue utilisateur, c'est-à-dire du point de vue de l'étudiant mais aussi de l'enseignant, avec quelques brèves explications sur l'implémentation des fonctionnalités réalisés. En particulier, nous verrons pourquoi et comment le noyau Linux a été adapté à des fins pédagogiques (cf. section IX-A).

II. Utilisation de base

Marionnet est un logiciel interactif avec une interface graphique basée sur le toolkit GTK (cf. [5]). Il offre à l'utilisateur une vision habituelle orientée *projet* : pour utiliser le logiciel, l'utilisateur peut soit commencer un nouveau projet, soit ouvrir un projet créé auparavant. Dans les deux cas, une fenêtre classique de création ou sélection de fichier s'ouvre, permettant à l'utilisateur de choisir le nom de l'archive à créer ou à ouvrir. Une fois créé ou ouvert un projet, l'utilisateur peut utiliser l'ensemble des fonctionnalités offertes par le logiciel : la palette de définition des composants réseau devient visible ainsi que la représentation graphique du réseau (onglet *Image*) et les onglets correspondants aux fonctionnalités avancées (*Énoncé*, *Interfaces*, *Anomalies*, *Disques*) deviennent accessibles. Dans le cas d'un nouveau projet, l'interface se présente comme dans la Figure 1.

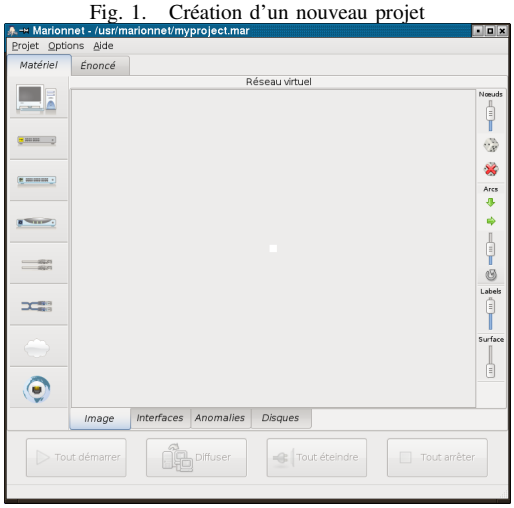
A. La palette des composants

Les composants du réseau virtuel sont introduits, modifiés et contrôlés à travers la palette des matériaux (ou palette des composants) qui s'affiche sur le côté gauchede la fenêtre principale de Marionnet, à l'intérieur de l'onglet *Matériel*.

Marionnet est un logiciel libre sous licence GPL (cf. [1]). Le développement du logiciel est financé comme projet *e-learning* par l'Université Paris 13. Une vidéo de présentation du logiciel est disponible aux formats ogg, avi et mpeg à l'adresse : `http://www-lipn.univ-paris13.fr/~loddo/marionnet/`

[1] Le terme *émulation* sera préféré dans le reste de cet article à *simulation* car le mot anglais *emulation* est employé par les spécialistes pour indiquer la technique de para-virtualisation utilisée, alors que le mot anglais *simulation* désigne des techniques différentes.

Fig. 1. Création d'un nouveau projet



L'utilisateur ne dessine pas le réseau, au sens propre du terme, il déclare simplement les éléments le constituant et règle leurs paramètres par des boîtes de dialogue. Pour des réglages particuliers, il utilisera les onglets donnant accès aux fonctionnalités avancées (*Interfaces*, *Anomalies*, *Disques*). Marionnet s'occupe automatiquement de dessiner le *plan* du réseau et de le mettre à jour à chaque changement du réseau ou d'état (éteint, exécution, suspendu) d'un de ses composants. L'image du réseau est à tout moment affichée au centre de la fenêtre dans le sous-onglet *Image* de l'onglet *Matériel*. L'ajout et la modification d'un composant se fait donc exclusivement par la palette, qui offre deux genres d'actions :

1) des actions de *définition* du réseau virtuel, telles que *ajouter*, *modifier* et *éliminer*
2) des actions de *contrôle* du réseau virtuel, telles que *démarrer*, *suspendre*, *réveiller*, *éteindre* et *arrêter*

et cela sur huit types de composant réseau :

1) la *machine* (ordinateur)
2) le *répéteur* (hub),
3) le *commutateur* (switch),
4) le *routeur* (router),
5) le *câble droi*t (direct cable),
6) le *câble croisé* (crossover cable),
7) le *nuage Ethernet*,
8) la *prise externe* (ou prise Internet).

Nous allons passer en revue l'ensemble des composants disponibles et discuter les possibilités de paramétrage offertes par Marionnet.

B. Le composant machine

Le composant machine représente un ordinateur sur lequel est installé un système d'exploitation de la famille GNU/Linux. Comme dans une situation réelle, une machine peut être *éteinte* ou bien en *exécution*. Comme pour tous les composants, à l'exception des câbles, il a été prévu un état supplémentaire appelé *suspendu*. Il s'agit d'un état où le composant ne réagit pas aux sollicitations de l'extérieur comme s'il était (très) occupé ou temporairement inaccessible. Ce type de fonctionnalité permet de construire des exercices particulièrement intéressants et originaux, notamment pour étudier les algorithmes de routage dynamique. Des icônes différentes sont utilisées pour représenter la machine dans ses états possibles : éteint, en exécution, suspendu :

Paramétrage d'une machine virtuelle: Lorsque un utilisateur ajoute ou modifie une machine virtuelle, la fenêtre de dialogue de Figure 2 permet de définir le *nom* de la machine virtuelle et de régler plusieurs paramètres matériels et logiciels. Dans la section *Hardware*, l'utilisateur peut régler des paramètres liés au support matériel de l'émulation :

- la *mémoire*

  c'est-à-dire la quantité de mémoire vive (RAM) que le système hôte doit réserver pour cette machine virtuelle (48Mo par défaut, ce qui permet sur la machine virtuelle de lancer confortablement des applications gourmandes comme *Firefox* ou *Ethereal*/*Wireshark*)

- le nombre de *cartes Ethernet* (1 par défaut)

Dans la section *Software* l'utilisateur peut régler des paramètres liés au support logiciel de la machine virtuelle :

- la *distribution* GNU/Linux (*Debian*, *Mandriva*, *Gentoo*,..), à choisir parmi les disponibles dans l'installation courante de Marionnet et de ses paquets supplémentaires sur le système hôte;

- la *variante*

  c'est-à-dire une mise à jour (un *patch*) de la distribution concernée ; le patch est un simple fichier, tout comme la distribution auquel il s'applique : la technologie COW (Copy On Write) supportée par Linux permet, en effet, de réunir un disque "enveloppé" dans un fichier (*backend*) et son patch (*frontend*) dans une unique entité utilisable comme disque virtuel;

- le *noyau*

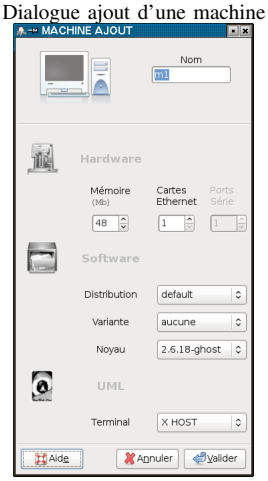
  c'est-à-dire la version du noyau Linux, à choisir parmi les disponibles dans l'installation courante de Marionnet et de ses paquets supplémentaires sur le système hôte;

Enfin, dans la section *UML* l'utilisateur peut régler des paramètres liés à la manière de prendre les commandes de la machine virtuelle depuis la machine hôte :

- le *terminal*

  les choix possibles sont ici "X HOST" et "X NEST"; le premier choix permet de lancer des applications graphiques à partir d'un terminal textuel où l'utilisateur pourra prendre les commandes de la machine virtuelle

Fig. 2. Dialogue ajout d'une machine virtuelle



(avec le login "root" et mot de passe "root"); le choix "X NEST" permet en revanche d'avoir un véritable serveur graphique réservé à la machine virtuelle, avec des gestionnaires de fenêtres et de bureaux indépendants de ceux de l'hôte.

C. *Le composant* répéteur

Un *répéteur* ou *concentrateur* (*hub* en anglais) est un dispositif électronique très simple permettant de réaliser un réseau informatique local de type Ethernet. Il permet la connexion de plusieurs appareils sur une même ligne de communication en régénérant le signal, d'où le nom de répéteur, et en répercutant ainsi les données émises par l'un vers les autres. Tous les noeuds rattachés partagent le même domaine de diffusion ainsi que le même domaine de collision : un seul noeud transmet à la fois et, lorsque une collision se produit, les émetteurs impliqués retransmettent alors leurs trames après avoir attendu un temps aléatoire.

Même si ce type d'appareil est à juste titre considéré inactuel, voire obsolète, par rapport à un commutateur (cf. section II-D), sa caractéristique principale, celle de répéter les trames écoutées depuis un port sur tous les autres ports, présente des avantages d'un point de vue pédagogique. Avec un tel appareil, le trafic réseau entre un émetteur et un récepteur peut être à tout moment intercepté et analysé par un tiers connecté au répéteur. Marionnet permet l'émulation de ce type d'appareil essentiellement en raison de cet intérêt pédagogique et en dépit de l'effort supplémentaire que son implémentation a requis. En effet, le code source en langage C du logiciel libre VDE (cf. [6], [7]), permettant l'émulation d'un commutateur (cf. section II-D), a du être étudié et modifié pour obtenir un émulation fidèle du comportement du répéteur. Les icônes représentants dans Marionnet un répéteur dans ses différents états possibles (éteint, exécution, suspendu) sont respectivement :

Paramétrage d'un répéteur virtuel: À la demande de l'utilisateur, une fenêtre de dialogue très simple permet d'ajouter ou de modifier le composant répéteur virtuel. Avec le *nom* de l'appareil, les paramètres demandés sont ici :

- une éventuelle *étiquette*

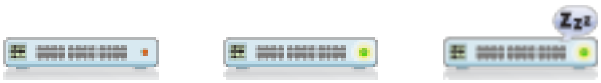
  c'est-à-dire une chaîne de caractères qui est utilisée pour décorer l'icône du répéteur dans l'image du réseau; ceci permet d'avoir en un clin d'oeil des informations (par exemple 192.168.1.0/24) sur le réseau Ethernet réalisé ou qui devra être réalisé par ce composant; cette information est utilisée exclusivement à des fins graphiques et n'est aucunement prise en compte à des fins de configuration

- le *nombre de ports*

  ce nombre (par défaut 4) ne doit pas être inutilement grand car le nombre de processus nécessaires à l'émulation de ce composant lui sera proportionnel

D. *Le composant* commutateur

Un *commutateur* réseau (*switch* en anglais) est un équipement permettant de relier plusieurs segments d'un réseau informatique. Comme le répéteur, il prend le plus souvent la forme d'un boîtier disposant de plusieurs ports (typiquement entre 4 et 100), mais à l'opposé du répéteur, il dirige les données uniquement vers la machine destinataire, ce qui le rend plus efficace en vitesse de transmission et bien plus discret lorsque une protection des données est requise. Le commutateur dans ses trois états possibles (éteint, exécution, suspendu) est représenté dans Marionnet par les icônes suivantes :



Paramétrage d'un commutateur virtuel: La fenêtre de dialogue permettant d'ajouter ou modifier un commutateur ne présente aucune différence notable avec celle du répéteur (cf. section II-C) : sont demandés ici un *nom*, une éventuelle *étiquette* et le *nombre de ports* (par défaut 4). La simulation d'un commutateur a été réalisée en modifiant les sources du projet VDE (http://vde.sourceforge.net/), essentiellement pour ajouter la simulation des anomalies qui seront décrites dans la section .

E. *Le composant* routeur

Un *routeur* est un dispositif de communication des réseaux informatiques qui permet d'acheminer les paquet IP, en provenance d'un réseau local, sur un second réseau local en direction de leur destination finale. La fonction principale du routeur est de déterminer le prochain noeud du réseau, auquel le paquet doit être envoyé, afin que ce dernier atteigne sa destination finale le plus rapidement possible. Ce mécanisme nommé *routage* intervient à la couche 3 (couche réseau) du modèle OSI. Dans un réseau de type Ethernet, le routeur utilise les informations au niveau IP, à la différence d'un commutateur qui ne regarde que les adresses au niveau MAC (couche 2 du modèle ISO). Pour réaliser sa fonction, un routeur utilise une ou plusieurs *tables de routage*. Ces tables

Fig. 3. Dialogue ajout d'un câble droit

peuvent être configurées "en dur" sur le routeur, on parle alors de routage *statique*, mais elles peuvent aussi être mises à jour automatiquement et dynamiquement, on parle alors de routage *dynamique*.

L'émulation de ce composant dans Marionnet est réalisée avec le logiciel *quagga* (cf. [8], [9]), dérivé du projet *zebra* (cf. http://www.zebra.org/), permettant à la fois le routage statique et le routage dynamique (ce dernier selon différents algorithmes ou protocoles tels que RIP, OSF, BGP, ISIS). Ainsi, dans Marionnet un routeur n'est autre qu'une machine virtuelle pré-configurée proposant les services réalisés en réalité par *quagga*. Chaque interface du routeur peut être configurée depuis l'onglet *Interfaces* de Marionnet. Une fois démarré, le routeur répondra au protocole *telnet* (avec le mot de passe *zebra*) sur toutes les interfaces configurées et sur les ports tcp suivants :

Protocole	Port tcp
zebra	2601
RIPd	2602
RIPngd	2603
OSPFd	2604
BGPd	2605
OSPF6d	2606
ISISd	2608

Les icônes représentant le routeur dans ses trois états possibles (éteint, exécution, suspendu) sont :

Paramétrage d'un routeur virtuel: La configuration des interfaces du routeur se faisant par l'onglet *Interfaces*, la fenêtre de dialogue qui permet d'ajouter ou modifier un routeur ne se distingue pas de façon significative de celles des composants répéteur et commutateur. Comme pour ces derniers, l'utilisateur doit définir le *nom* du routeur virtuel, une éventuelle *étiquette* et le *nombre de ports* du routeur (par défaut 4).

F. Le composant câble droit

Un *câble* est un assemblage de fils électriques enveloppés dans une gaine. Dans un réseau informatique, les câbles permettent de relier physiquement des équipements divers assurant l'interconnexion des moyens physiques et des protocoles. Si l'infrastructure d'un réseau est constituée seulement de câbles, le réseau est dit *filaire*. Plusieurs types de câbles, donc de réseaux filaires existent (câble coaxial 10BASE5/10BASE2, paires torsadées 10BASE-T/100BASE-T/ 1000BASE-T, fibres optiques). En simulant les câbles comme les machines et les autres appareils, Marionnet permet donc de simuler un réseau informatique filaire.

Les actions de contrôle accessibles par la palette des composants sont, dans le cas des câbles, seulement *déconnecter* et *reconnecter*. Un câble ne s'allume ni s'éteint, mais peut être déconnecté temporairement de ses extrémités, puis reconnecté au besoin. Dans l'image du réseau, l'état *déconnecté* est représenté par une ligne pointillée.

Paramétrage d'un câble droit virtuel: L'icône du composant *câble droit* évoque les câbles à paires torsadées (prise RJ45), les plus généralisés aujourd'hui en réseau local, mais il s'agit d'un choix de représentation plutôt arbitraire, conditionné par la présence du composant *câble croisé* dans Marionnet (cf. section II-G). En réalité, Marionnet fait abstraction des différences et aucune assomption sur la vitesse de transmission des signaux électriques n'est faite. Cela revient à dire, d'un point de vue utilisateur, qu'aucun choix est prévu et aucun réglage n'est possible sur la nature et capacité du câble. Les seuls paramètres demandés à l'utilisateur dans le dialogue de la Figure 3 sont le *nom* du câble, une éventuelle *étiquette* et, surtout, les deux *extrémités*, c'est-à-dire les deux composants réseaux (machine, répéteur, commutateur,..) reliés par le câble, et les deux interfaces impliquées. Ces dernières seront à choisir parmi les disponibles, c'est-à-dire parmi les existantes non déjà occupées par d'autres câbles.

Le dialogue de définition d'un câble droit mérite une précision importante : il permet de définir des câbles droits même lorsqu'ils ne pourront pas fonctionner, par exemple, entre deux machines. La possibilité de définir un câblage incorrect n'est pas liée à des raisons techniques mais a une justification et un intérêt exclusivement d'ordre pédagogique.

G. Le composant câble croisé

Un *câble croisé* est un type de câble Ethernet à paires torsadées permettant de connecter directement des dispositifs qui seraient normalement reliés par un répéteur ou un commutateur. Par exemple, il est possible d'utiliser un câble croisé pour connecter directement deux machines par leurs interfaces Ethernet.

Paramétrage d'un câble croisé virtuel: Le dialogue permettant de définir ou modifier un câble croisé ne diffère pas de façon significative par rapport à celui des câbles droits. Sont demandés le *nom*, une éventuelle *étiquette* et les deux *extrémités* reliées (composant et interface, des deux côtés). Comme pour les câbles droits et avec la même justification d'ordre pédagogique, le dialogue permet de définir un câblage *incorrect*. Il est, en effet, tout à fait possible de

définir des câbles croisés même lorsqu'ils ne pourront pas fonctionner, comme, par exemple, entre une machine et un commutateur.

H. *Le composant* nuage Ethernet

Le composant *nuage* représente un réseau Ethernet (couche 2), c'est-à-dire un assemblage de répéteurs, commutateurs et câbles croisés, dont la structure interne est sans intérêt, mais qui provoque des retards ou d'autres anomalies observables lors d'un passage de trames entre ses deux extrémités. L'intérêt de ce type de composant se justifie principalement dans le cadre d'exercices de routage dynamique (cf. section II-E).

Paramétrage d'un nuage Ethernet: Le dialogue de définition d'un nuage n'est pas particulièrement intéressant : il permet juste de définir le *nom* et une éventuelle *étiquette*. Une fois le nuage défini, l'utilisateur pourra sélectionner l'onglet *Anomalies* de Marionnet pour régler les retards, les pertes de trames et les autres anomalies qu'il souhaitera provoquer artificiellement entre les deux extrémités du nuage (cf. section VI-B).

I. *Le composant* prise externe

Avec les composants décrits jusqu'à présent, un réseau virtuel serait un ensemble de machines, répéteurs, commutateurs, routeurs et nuages inter-connectés par une infrastructure de câbles Ethernet droits et croisés. Le réseau virtuel serait ainsi un système totalement *clos*, isolé du système hôte qui l'exécute et par conséquent isolé du reste du monde (réel ou virtuel). Ceci n'est en réalité qu'une apparence. À vrai dire, il existe une connexion particulière qui permet l'affichage des applications graphiques d'une machine virtuelle sur le serveur graphique (serveur X) de l'hôte. Mis à part cette liaison cachée à l'utilisateur et dont l'unique but est de donner à l'utilisateur une vision simplifiée de la machine virtuelle (ce sujet est développé dans la section IX-A), le réseau virtuel ne peut avoir de contact avec l'extérieur. Le composant *prise externe* ouvre une brèche dans cette cloison apparente. On se sert de ce composant comme, dans la réalité, on se sert d'une prise Ethernet murale (femelle RJ45) : en connectant n'importe quel composant du réseau virtuel à la prise (par un câble droit ou croisé selon le cas), le composant aura accès à l'extérieur, c'est-à-dire au même réseau (réel) sur lequel est branché le système hôte.

Avec la prise externe, tout réseau accessible au système hôte, dont Internet est un exemple, sera accessible depuis le réseau virtuel. Les implications pratiques sont multiples :

- donner un accès Internet aux machines virtuelles;
- installer des logiciels supplémentaires par le réseau (par exemple, par un simple *apt-get install* sur une Debian);
- permettre aux machines virtuelles d'être, elles aussi, clientes de tous les services exploitables par l'hôte, c'est-à-dire de tous les services offerts sur le réseau de l'hôte (DHCP, DNS, NFS, NTP,..);
- connecter des réseaux virtuels différents dans une salle de TP en réseau; cela permettrait des exercices en groupe d'étudiants (binôme, trinôme,..) difficile à réaliser dans la réalité : chaque réseau virtuel (composé de plusieurs éléments tels que machines, appareils et câbles) serait administré par *un* seul étudiant sur une machine hôte de la salle de TP. Pour un binôme, par exemple, on pourrait imaginer alors des exercices ludiques du type :
  - déterminer une faille de sécurité chez son partenaire (ou plutôt *adversaire*);
  - exploiter ou tester un service offert par son partenaire;
  - programmer, tester ou monitorer une application client-serveur;

Choix du réseau externe: Une précision s'impose quant à la politique adoptée par Marionnet pour sélectionner le réseau externe lorsque plusieurs réseaux sont accessibles depuis l'hôte. Le réseau est choisi avec la stratégie suivante :

1) si l'hôte a une route par défaut définie dans sa table de routage, le réseau accessible par la prise est celui de l'interface connectée à la passerelle;
2) sinon, le réseau est la réunion (en *bridge* Linux) des réseaux de toutes les interfaces réseau actives de l'hôte; autrement dit, si Marionnet ne peut déterminer un réseau "principal", il réunit tous les réseaux accessibles dans un bridge et donne accès au bridge à travers la prise; de cette manière, les machines virtuelles pourront accéder à n'importe quel réseau externe accessible par l'hôte.

Paramétrage d'une prise externe: Sont demandés à l'utilisateur un *nom* pour la prise et une éventuelle *étiquette*. Aucun autre paramétrage n'est possible, à l'exception d'éventuelles anomalies de la prise que l'utilisateur pourra configurer dans l'onglet *Anomalies*.

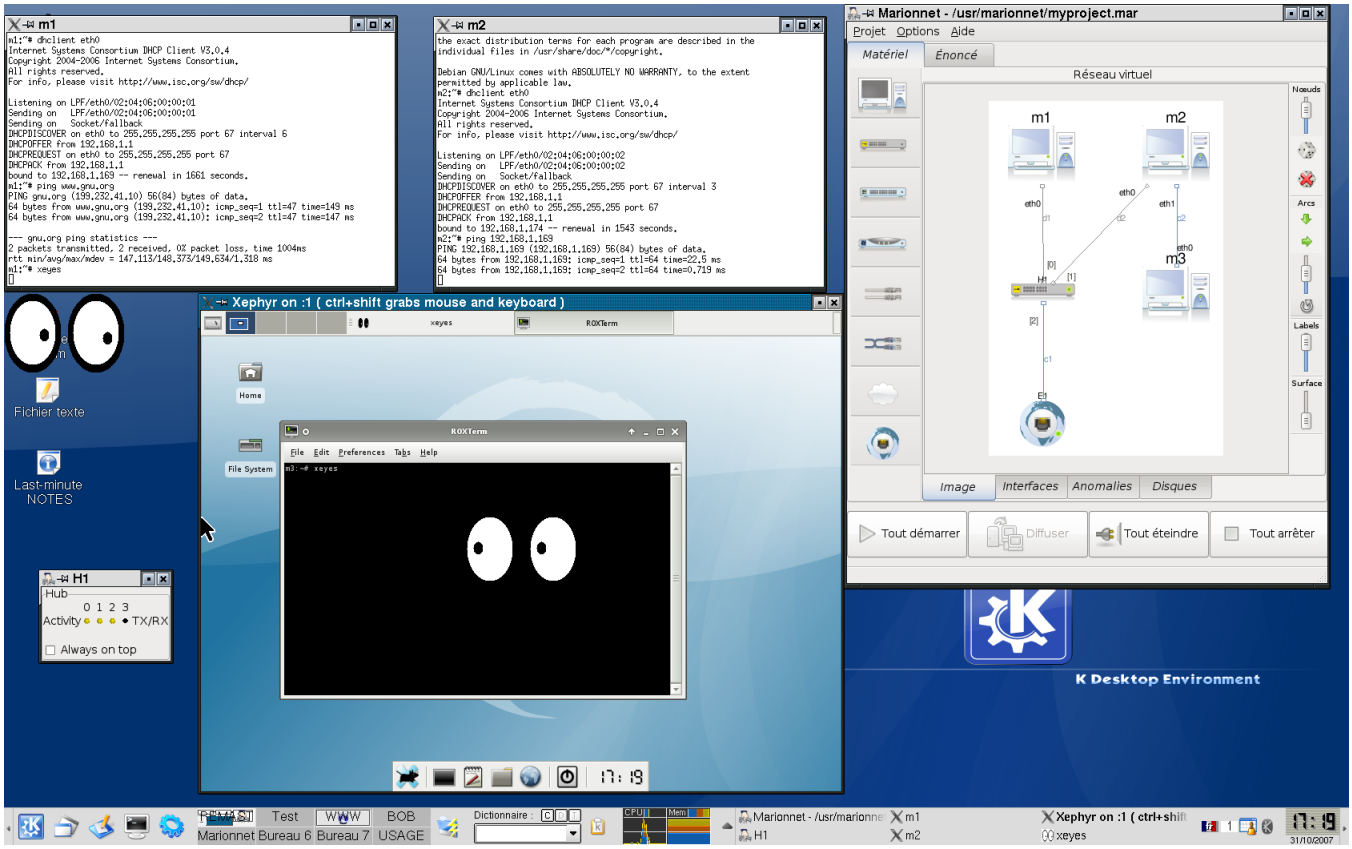
III. L'IMAGE DU RÉSEAU

Lorsqu'un projet est ouvert, l'image du réseau défini dans le projet est affichée à tout moment au centre de la fenêtre dans le sous-onglet *Image* de l'onglet *Matériel*. Cette image est régénérée automatiquement à chaque changement d'état du projet, c'est-à-dire :

1) lorsque un composant est ajouté ou modifié dans le projet
2) lorsque un composant change d'état (de éteint à exécution,..)

Cette fonctionnalité a été implémentée en utilisant le logiciel libre *graphviz* (cf. [10]). Il s'agit d'un outil permettant

Fig. 4. Exemple de réseau en exécution



de construire l'image d'un graphe (orienté ou non) dans plusieurs formats possibles tels que *postscript*, *jpeg*, et *png*, à partir d'une spécification formelle du graphe. Lorsque Marionnet doit générer une image, il s'occupe de traduire l'état courant du réseau dans le langage formel proposé par *graphviz*, puis de lancer la compilation de *graphviz* sur la spécification générée.

A. La palette d'ajustement de l'image

Plusieurs paramètres de compilation concernant la façon de dessiner les noeuds et les arcs sont réglables et l'utilisateur peut accéder à certains de ces réglages par la palette *à droite* de l'image. Cette palette est divisée en quatre sections permettant de régler la représentation des noeuds, des arcs (arêtes), des étiquettes (associées aux noeuds et aux arcs) et de la surface contenant l'image.

Section Noeuds*:* Dans cette section de la palette, trois réglages différents sont proposés à l'utilisateur :

1) une barre de type *échelle* permet d'ajuster la *taille* des icônes représentant les composants réseau;
2) un bouton figurant un *dé* permet de réagencer les noeuds de façon aléatoire;
3) un bouton figurant un *dé barré* permet de revenir à l'agencement initial des noeuds;

Section Arcs*:* Cette section offre à l'utilisateur quatre réglages possibles :

1) un bouton figurant une *flèche vers la droite* permet de développer le plan à l'horizontale, de la gauche vers la droite;
2) un bouton figurant une *flèche vers le bas* permet de développer le plan à la verticale, du haut vers le bas;
3) une barre de type *échelle* permet d'ajuster la *taille minimale* des arcs du graphe;
4) un bouton figurant une *flèche circulaire* permet d'inverser n'importe quel arc (câble) du graphe;

Section Labels*:* Dans cette section il est possible, par l'intermédiaire d'une barre de type échelle, d'ajuster la *distance* entre les étiquettes et l'extrémité des arcs qu'elles décorent.

Section Surface*:* Une dernière barre de type échelle permet, dans cette section, d'ajuster la *taille du canevas* contenant l'image entière.

IV. Exécution et contrôle des composants

L'interface de Marionnet propose deux manières de contrôler l'exécution des composants : une forme de contrôle *local*, orienté composant, et une forme de contrôle *global*, orienté réseau.

A. Contrôle local vs global

La palette des composants permet le contrôle local : l'utilisateur sélectionne avant tout le bouton correspondant

au type de composant (p.e. la machine), ensuite il sélectionne l'action (p.e. démarrer), et il sélectionne enfin le nom de la machine dans un menu déroulant. Ce menu est calculé à chaque instant, c'est-à-dire de façon dynamique, de façon à proposer seulement les noms des composants susceptibles d'être l'objet de l'action sélectionnée par l'utilisateur. Par exemple, si l'utilisateur entend éteindre une machine, seulement les noms des machines actuellement en exécution seront proposées. Ainsi, s'il entend réveiller un commutateur, seulement les commutateurs suspendus seront proposés dans le menu.

Vice-versa, une forme de contrôle global est possible par l'intermédiaire des gros boutons sur la barre d'outils en bas de la fenêtre de Marionnet. Un bouton *Tout démarrer* permet de démarrer tout ce qui n'est pas déjà en exécution; un bouton *Tout éteindre* permet d'éteindre (enlever le courant) tout ce qui ne l'est pas déjà, et un bouton *Tout arrêter* permet d'arrêter proprement (shutdown) tous les composants encore en exécution.

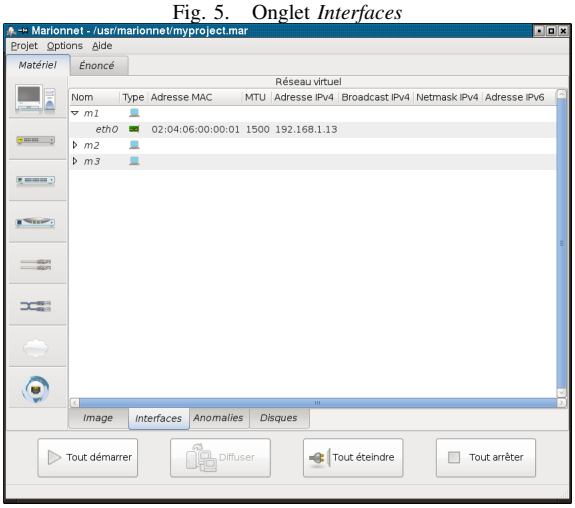
B. Éteindre vs arrêter

Dans Marionnet le mot *éteindre* veut signifier *éteindre brusquement*, comme si on enlevait le courant au composant, ce qui se traduit le plus souvent, dans l'implémentation, par l'envoi du signal SIGKILL à l'ensemble des processus qui en réalisent l'émulation. À l'opposé, le mot *arrêter* veut signifier *éteindre gentiment* ou *proprement*. Il s'agit d'une pratique qui est désormais la norme dans la réalité : tout utilisateur a l'habitude d'éteindre proprement son ordinateur, car les systèmes d'exploitation modernes proposent des écritures disque non synchrones et risquent donc, lors d'une coupure du courant, de rester dans un état inconsistant. Dans Marionnet, l'action d'éteindre brusquement une machine ou un routeur peut avoir les mêmes conséquences fâcheuses que dans la réalité, en termes de pertes de données ou d'inconsistances de l'état du disque.

V. Exemple de projet en exécution

Dans l'exemple de la figure 4, trois machines m1, m2 et m3 ont été définies, les deux premières avec un terminal textuel (option "X HOST"), la troisième avec son propre serveur graphique et son propre gestionnaire de fenêtres et de bureau (option "X NEST"). Deux câbles droits relient m1 et m2 à un répéteur nommé H1, un câble croisé relie directement m2 et m3. Une prise externe a été également définie, branchée par un câble croisé au répéteur (un câble droit ne fonctionnerait pas!), puis démarrée. Elle permet aux machines m1 et m2 d'être configurées par un simple appel du programme *dhclient* (auquel répond le même serveur DHCP qui a configuré automatiquement l'hôte lors de son démarrage). Ensuite, m1 exécute avec succès un *ping* vers le site internet www.gnu.org. Puis, tandis que m2 exécute, elle aussi avec succès, un *ping* vers m1, ce dernier lance une instance du programme *xeyes* s'affichant sur l'écran graphique de l'hôte. De son coté, m3 lance aussi une instance de *xeyes* qui s'affiche (et ne peut sortir) de l'espace de 800x600 pixels réservé à son environnement graphique.

Fig. 5. Onglet *Interfaces*



En bas de l'écran, sur le côté gauche, est visible une fenêtre représentant le répéteur. On y aperçoit 4 leds, un pour chacun de ses 4 ports, mais dont seulement 3 sont allumés, c'est-à-dire utilisés (par m1, m2 et la prise). À chaque passage de trames, tous les leds clignotent, le comportement de cet appareil étant bien celui de reporter les données écoutées depuis un port sur tous les ports.

L'exemple que nous venons de décrire pour un répéteur est valide pour l'exécution des commutateurs et des routeurs à quelques différences près. En effet, une fenêtre du même type apparaît et permet ainsi d'en monitorer l'exécution. Bien sûr, à la différence du répéteur, seulement les leds correspondants aux ports impliqués par une transmission clignoteront au passage des données.

VI. Utilisation avancée

Les fonctionnalités de base de Marionnet permettent la définition et l'émulation d'un réseau informatique. La reconfiguration du réseau peut avoir lieu "à chaud", c'est-à-dire pendant l'exécution de tous ou un sous-ensemble de composants. Marionnet propose aussi des fonctionnalités "avancées", dont le but est de :

- faciliter la *pré-configuration* IPv4 et IPv6 des interfaces réseau (pour machines et routeurs);
- simuler les *anomalies*, c'est-à-dire simuler des interfaces réseau ou des câbles *défectueux*;
- garder une trace (historique) des images de disque avant tout démarrage d'une machine ou d'un routeur;
- offrir un module gestionnaire d'énoncés et/ou de documentation associés au projet;
- offrir un mode de fonctionnement facilitant le travail de correction et de notation de l'enseignant.

A. Pré-configuration des interfaces

Par l'onglet *Interfaces* (cf. figure 5) l'utilisateur peut configurer les interfaces réseau des machines et routeurs en

Fig. 6. Onglet *Anomalies*

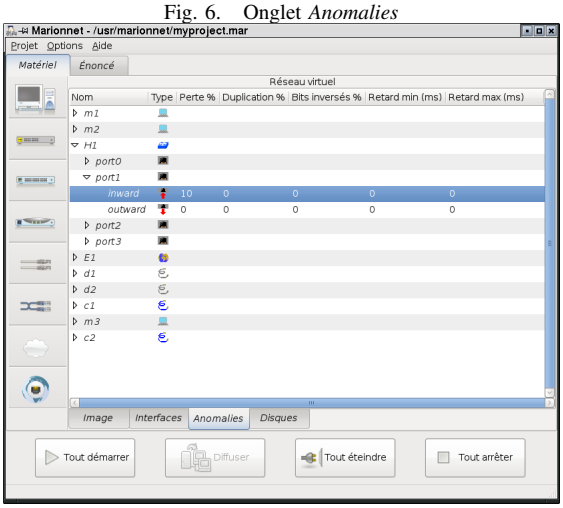
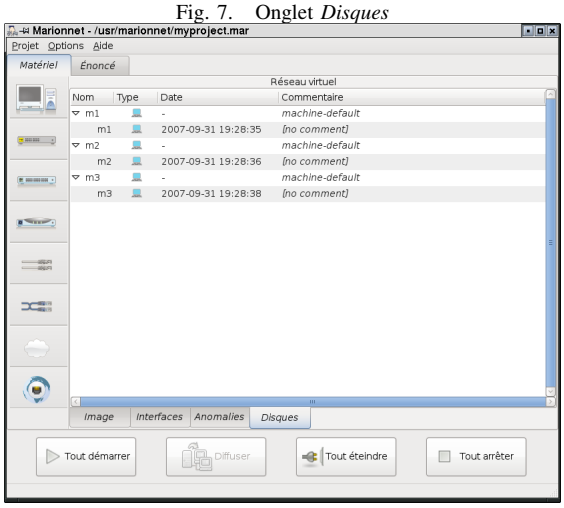


Fig. 7. Onglet *Disques*



définissant au choix un ou plusieurs paramètres parmi les suivants :

- l'adresse MAC
- le MTU
- l'adresse IPv4
- l'adresse de diffusion IPv4
- le masque réseau IPv4
- l'adresse IPv6

B. Anomalies

L'infrastructure de communication peut présenter des comportements artificiellement anormaux. Ce type de simulation ouvre la voie à des exercices du type "trouver la panne". La panne ou anomalie peut se manifester sous plusieurs formes :

- pertes de trames
- duplication de trames
- bits inversés (flip) n'importe où dans la trame
- retard de transmission (minimum et maximum, selon une loi gaussienne)

Les défauts sont programmables par l'onglet *Anomalies* (cf. figure 6) avec la granularité de la ligne électrique, autrement dit, avec la granularité de la *direction* (entrée ou sortie) de toute prise RJ45 femelle (ports et cartes Ethernet) ou RJ45 mâle (câbles). Les anomalies peuvent être définies "à chaud", c'est-à-dire pendant que le composant concerné est en exécution (le comportement en sera immédiatement affecté).

C. Historique des disques

Pour toute machine ou routeur, une historique complète des états disques, sauvegardés avant tout démarrage, est accessible depuis l'onglet *Disques* (cf. figure 7). Garder une trace des images disques permet de redémarrer la machine ou le routeur non seulement dans son état courant (ce qui est le comportement par défaut), mais aussi dans un état précédent (dans l'onglet *Disques* on utilisera le bouton droit de la souris pour cela). La relation de filiation entre images disque engendre, en général, une véritable arborescence (car deux images peuvent être soeurs, c'est-à-dire dérivées, de la même image mère).

Étant donnée une machine, la racine de son arborescence d'états disque est l'état initial (distribution, variante) choisi au moment de l'ajout de ce composant dans le réseau virtuel. Toute image disque enregistrée dans l'onglet *Disques* peut être donc considérée comme une variante, c'est-à-dire une sorte de "patch", de la distribution GNU/Linux se trouvant à la racine de son arborescence. Toute image disque peut être alors *exportée* (bouton droit de la souris) comme variante et être réutilisée dans d'autres projets.

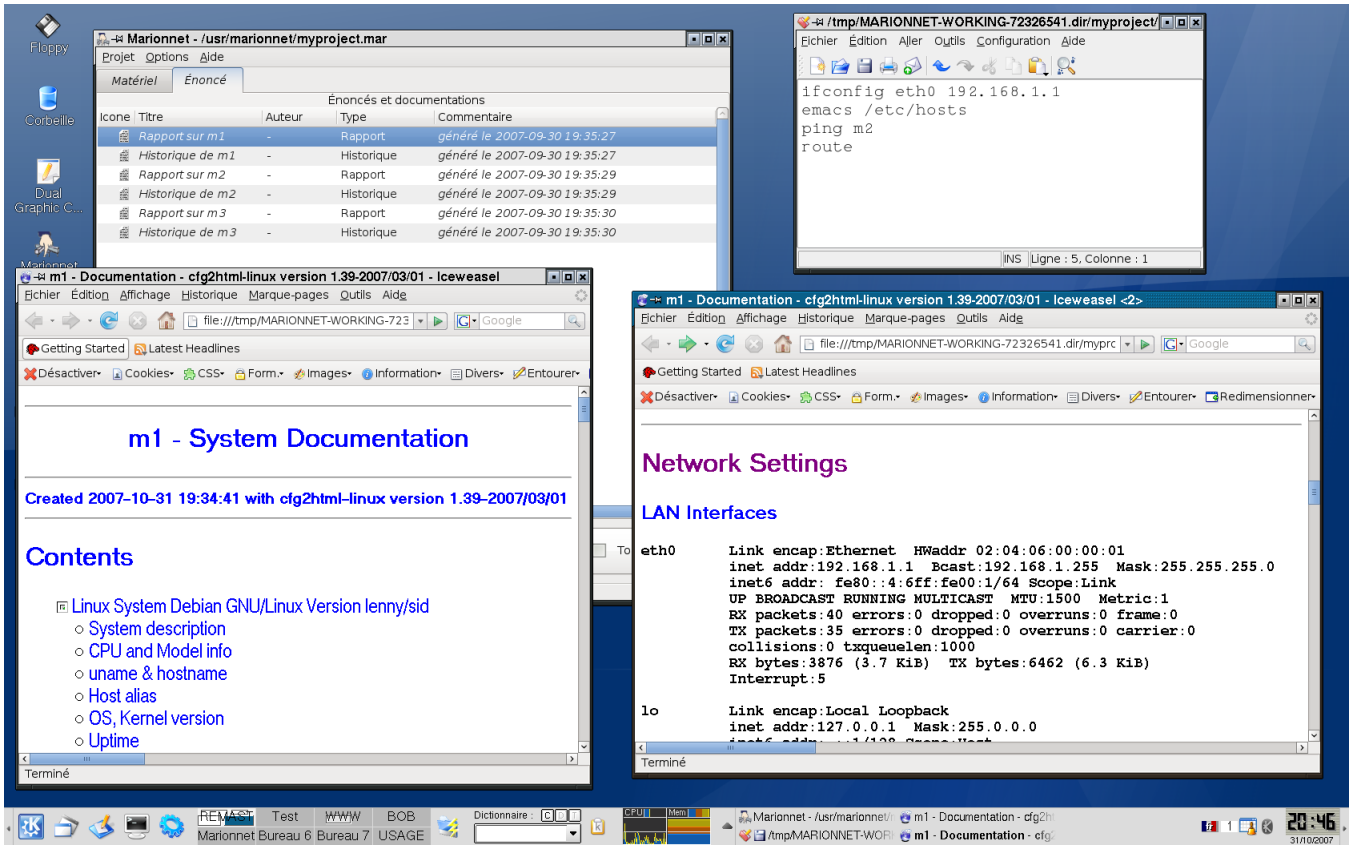
VII. Énoncés et rapports d'exécution

Par l'onglet *Énoncé*, Marionnet offre d'autres fonctionnalités avancées, dont l'intérêt est strictement pédagogique. Dans cet onglet, n'importe quel fichier au format *postscript*, *pdf*, *dvi*, *html* ou *texte* peut être ajouté au projet. Les fichiers importés pourrons être ensuite affichés et servir ainsi d'énoncé pour les étudiants. L'enseignant pourra accessoirement introduire tout complément d'information, comme par exemple un corrigé, des notes de cours, des scripts utiles, des RFC ou des pages web.

A. La modalité examen

Lorsque Marionnet est lancé dans une modalité spéciale appelée *modalité examen* (marionnet –exam) au moment d'un arrêt *propre* (cf. section IV-B) d'une machine ou d'un routeur, un rapport très détaillé est automatiquement généré. Il s'agit d'une sorte d'état des lieux sur la configuration du composant juste avant son arrêt. Dans le cas d'une machine virtuelle, au rapport s'ajoute un deuxième fichier contenant l'historique des commandes lancées par l'étudiant dans les shells *bash* activées au cours de l'émulation de la machine virtuelle (fichier *bash_history*). Le rapport (au format *html*) et l'éventuelle historique (au format *texte*) seront rendus accessibles, donc analysable par l'enseignant, à partir de l'onglet *Énoncé*.

Fig. 8. Inspection d'une copie rendue



B. *Déroulement d'un contrôle*

Le fonctionnement préconisé dans le cas d'un contrôle peut se résumer par les étapes suivantes :

1) l'enseignant transmet un projet contenant un énoncé aux étudiants; le projet peut contenir une éventuelle définition (un câblage) et une éventuelle pré-configuration du réseau virtuel;
2) l'étudiant lance Marionnet en modalité examen et travaille sur sa copie pour la durée du contrôle;
3) une fois le contrôle terminé, l'étudiant éteint proprement tous les composants encore en exécution (par exemple par le bouton *Tout arrêter*);
4) l'étudiant rend sa copie, c'est-à-dire le projet avec les modifications qu'il y a apporté; le projet contient à présent l'historique des disques avant tout démarrage (onglet *Disques*) ainsi que les rapports d'état avant tout arrêt propre, pour tout composant du type monitoré en modalité examen (machines et routeurs);
5) l'enseignant récupère la copie de l'étudiant, l'ouvre avec Marionnet, inspecte les rapports d'exécution et les historiques et peut alors noter la copie.

La figure 8 montre l'utilisation des rapports d'exécution et de l'historique des commandes. L'enseignant y découvre une correcte configuration de l'interface eth0 de la machine m1 et une correcte définition du nom m2 dans le fichier /etc/hosts.

VIII. Gestion de projets

L'entrée Projet du menu principal de l'application permet de réaliser les tâches courantes et habituelles liées à la gestion de projets. Les sous-entrées de ce menu sont :

- *Nouveau* : création d'un nouveau projet
- *Ouvrir* : ouverture d'un projet existant
- *Enregistrer* : sauvegarde du projet courant
- *Enregistrer sous* : sauvegarde du projet courant sous un autre nom
- *Copier sous* : création d'une copie du projet courant
- *Fermer* : fermeture du projet courant

Un projet consiste en une hiérarchie de fichiers et répertoires stockés dans une seule archive de type *tar* (cf. http://www.gnu.org/software/tar). Dans cette archive sont enregistrées les informations suivantes :

- les composants réseau utilisés (machines, répéteurs,..), chacun avec le paramétrage défini par l'utilisateur;
- le paramétrage concernant l'image (dessin) du réseau;
- les éventuelles configurations IP (numéro, masque, MTU, ..) des interfaces réseau;
- les éventuelles anomalies à simuler;

- l'historique des états disque de chaque machine;
- les énoncés, les rapports d'exécution et les autres éventuels fichiers (documentation, RFC, ..) au format *pdf*, *postscript*, *dvi*, *html* ou *texte*.

A. Un format d'échange pour les enseignants ?

Du point de vue de l'enseignant, un projet Marionnet est aussi une façon de transmettre aux étudiants et aux collègues enseignants un exercice défini dans tous ses aspects et devant s'exécuter sur une seule plateforme, c'est-à-dire Marionnet. Cette plateforme, qui exécute les composants de la même manière sur tout ordinateur hôte, peut s'entendre comme un moyen de *standardiser* ses exercices de pratique réseau : la "salle machines" sera la même partout, les appareils seront les mêmes, les câbles aussi.

1) Compatibilité vers le haut et vers le bas: Marionnet propose ainsi son modèle de projet comme format d'échange pour exercices de pratique réseau. Pour renforcer ce potentiel, le logiciel a été programmé dès la première version publique (octobre 2007) de façon à être :

- *compatible vers le bas*, c'est-à-dire compatible avec les versions précédentes du logiciel

  cela signifie, en pratique, qu'une version récente de Marionnet sera capable de lire un projet produit par une ancienne version du logiciel;
- *compatible vers le haut*, c'est-à-dire compatible avec les versions futures du logiciel

  cela signifie, en pratique, qu'une ancienne version de Marionnet sera capable de lire (en partie, pour ce qu'elle pourra comprendre) un projet produit par une version plus récente du logiciel.

IX. Détails de l'implémentation

Marionnet a été initialement conçu par l'auteur de cet article, Jean-Vincent Loddo. Il a été ensuite développé par deux personnes, l'auteur et Luca Saiu, avec *OCaml* (cf. [11], [12], [13], [14]), un langage de programmation fonctionnel avec typage statique et appel par valeur. Nous considérons que ce choix a réduit de façon considérable le temps de développement (ce sujet est traité dans [15]). L'application est concurrente (gestion de thread) avec un certain degré de tolérance aux pannes et aux comportement inattendus des fils d'exécution. La fonctionnalité qui caractérise le plus Marionnet par rapport à d'autres logiciels d'émulation (cf. [16], [17], [18], [19], [20], [21]) est sa capacité de reconfigurer le réseau virtuel "à chaud". Tout composant, configuration, anomalies peut être défini *dynamiquement*, c'est-à-dire pendant que d'autres composants réseau s'exécutent. Cela est rendu possible par une implémentation sophistiquée de l'émulation : un composant "logique" correspond toujours à un ensemble de composant "physiques", c'est-à-dire un ensemble de processus système. La figure 9 montre le détail des processus (rectangles blancs) nécessaires à l'émulation un réseau virtuel simple, constitué de deux machines (chacune avec deux interfaces), un commutateur et deux câbles droits.

A. Ghostification

Les machines virtuelles ont besoin de communiquer avec le serveur graphique de l'hôte pour afficher les fenêtres de leurs applications. Il s'agit toutefois d'une liaison complètement cachée à l'utilisateur, dont l'objectif est de donner à l'utilisateur une vision simplifiée de la machine virtuelle : l'utilisateur peut lancer des applications graphiques (*firefox*, *ethereal*, *kate*, ..) sans se préoccuper du comment et du pourquoi elles sont en mesure de s'afficher sur l'écran. Le réseau virtuel paraît ainsi isolé du système hôte. Cachée derrière cet apparence, il existe une interface réseau particulière dans chaque machine virtuelle, l'interface eth42, que le noyau Linux, opportunément modifié, fait fonctionner parfaitement sans toutefois en montrer la présence au niveau des applications. Des outils tels que *ifconfig*, *route*, *netstat* n'ont aucune possibilité de détecter l'interface eth42 et l'utilisateur profite de l'affichage graphique sans être perturbé par une interface réseau supplémentaire et par le trafic réseau qui la traverse.

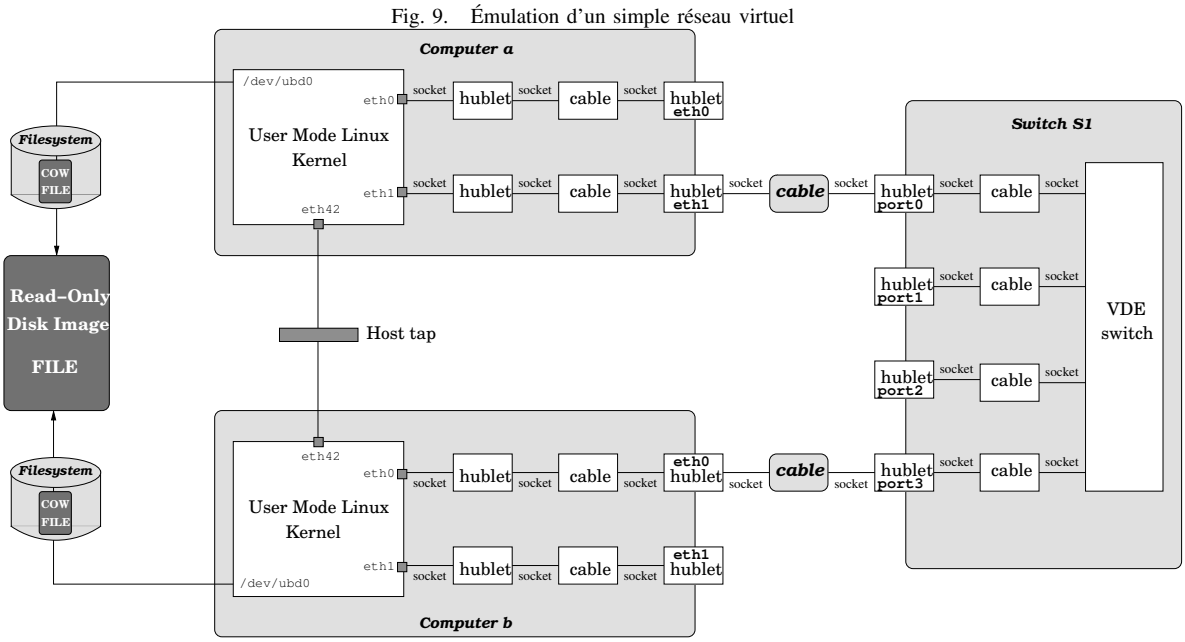
Ce comportement à été réalisé en ajoutant au noyau Linux un appel système spécial, que nous avons nommé *ghostify*, qui permet de cacher une interface réseau quelconque en la laissant toutefois totalement opérationnelle. Un appel à *ghostify* est réalisé à chaque démarrage d'une machine virtuelle de façon à rendre invisible l'interface eth42.

X. Conclusions

Nous avons parcouru dans cet article les caractéristiques principales de Marionnet, un logiciel qui permet de définir, configurer et exécuter un réseau virtuel d'ordinateurs et d'appareils tels que des répéteurs, commutateurs et routeurs connectés par une infrastructure de câbles virtuels. Un simple ordinateur, équipé du système d'exploitation GNU/Linux est suffisant pour l'émulation du réseau entier. Marionnet propose des fonctionnalités de base par une interface utilisateur graphique réfléchie pour être intuitive et accessible à un public sans compétences particulières. Il propose aussi des fonctionnalités avancées pour un public plus avisé et des fonctionnalités dont le but principal est pédagogique. Les enseignants peuvent utiliser Marionnet pour préparer leurs énoncés et pour les transmettre aux étudiants et à leurs collègues enseignants.

Marionnet est un moyen de faciliter l'apprentissage à distance et le travail personnel dans la pratique des réseaux informatiques. L'étudiant peut, en effet, travailler à distance : tout exercice sur un réseau à base de machines Unix peut avoir lieu (à quelques rares exceptions près) sur une seule machine, l'hôte qui héberge le réseau virtuel, chez soi, sur son poste fixe ou son ordinateur portable. L'étudiant peut télécharger le projet sur un site regroupant des énoncés, réaliser l'exercice et même rendre sa copie. La fonctionnalité Copy On Write (COW) de Linux permet, en effet, de découper une image de disque en deux parties complémentaires : une partie principale et « lourde » (en Méga ou Giga octets) accessible en lecture seule (read-only) contenant le gros de l'image, et une partie auxiliaire « légère » accessible en lecture-écriture (read-write)

Fig. 9. Émulation d'un simple réseau virtuel



et contenant juste les modifications apportées à la partie lourde. Ainsi, l'étudiant pourra rendre à l'enseignant un projet Marionnet contenant seulement les parties légères des disques du réseau virtuel, et représentant précisément le travail effectué sur l'ensemble du réseau. Grâce à la taille limitée des parties légères (typiquement quelques Méga octets) il pourra aisément renvoyer sa copie par internet (par courrier électronique à l'enseignant ou en téléchargement sur un site ad-hoc). De son côté, l'enseignant aura moyen d'inspecter l'état des disques durs correspondant au travail rendu, et cela grâce à des fonctionnalités du logiciel spécifiquement conçues pour les enseignants.

XI. Remerciements

Je remercie Luca Saiu pour son excellent travail de modélisation et développement. L'Université Paris 13, l'IUT de Villetaneuse et le Laboratoire d'Informatique de Paris Nord (LIPN) pour leur support.

References

[1] Free Software Foundation, "GNU General Public License." URL: http://www.gnu.org/copyleft/gpl.html, 2007.
[2] J. Dike, *User Mode Linux*. Prentice-Hall, 2006.
[3] J. Dike, "User Mode Linux Community Site." URL: http://usermodelinux.org.
[4] J. Dike, "User Mode Linux Kernel Home Page." URL: http://user-mode-linux.sourceforge.net.
[5] O. Taylor *et al.*, "Gtk+ - GNU toolkit for X windows development." URL: http://www.gtk.org.
[6] R. Davoli, "VDE: Virtual Distributed Ethernet," in *TRIDENTCOM*, pp. 213–220, IEEE Computer Society, 2005.
[7] R. Davoli, "Teaching Operating Systems Administration with User Mode Linux," in *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, (New York, NY, USA), pp. 112–116, ACM Press, 2004.
[8] K. Ishiguro *et al.*, "Quagga Home Page." URL: http://www.quagga.net.
[9] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, (Berkeley, CA, USA), pp. 2–2, USENIX Association, 2005.
[10] AT&T Labs, "Graphviz - open source graph drawing software." URL: http://www.research.att.com/sw/tools/graphviz/.
[11] P. Narbel, *Programmation fonctionnelle, générique et objet. Une introduction avec le langage OCaml*. Vuibert Informatique, 2005.
[12] X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon), *The Objective Caml system, Documentation and user's manual*, release 3.10 ed., 2007. URL: http://caml.inria.fr/pub/docs/manual-ocaml/.
[13] G. Cousineau and M. Mauny, *The Functional Approach to Programming*. Cambridge University Press, 1998.
[14] E. Chailloux, P. Manoury, and B. Pagano, *Developing Applications with Objective Caml.* 2000. Developement d'applications avec Objective Caml, O'Reilly, France.
[15] J.-V. Loddo and L. Saiu, "Status Report: Marionnet – "How to Implement a Virtual Network Laboratory in Six Months and be Happy"," in *ACM SIGPLAN Workshop on ML*, ACM Press, 2007.
[16] F. Galán and D. Fernández, "Virtual Network User Mode Linux." URL: http://jungla.dit.upm.es/vnuml/.
[17] F. Galán and C. T. Deccio, "VNUML Language Reference." URL: http://jungla.dit.upm.es/vnuml/doc/1.6/reference/index.html.
[18] M. Rimondini, "Emulation of Computer Networks with Netkit," tech. rep., 2007. Università degli Studi di Roma Tre. URL: http://dipartimento.dia.uniroma3.it/ FILE: ricerca/rapporti/rt/2007-113.pdf.
[19] M. Blanc, "The vmumlgui Project Home Page." URL: http://pagesperso.erasme.org/michel/vnumlgui/.
[20] S. C. Nemesio, P. de las Heras Quirós, E. M. C. Barbero, and J. A. C. Gonzàlez, "Early experiences with NetGUI laboratories,"
[21] A. Krap, "Setting up a virtual network laboratory with User-Mode Linux," tech. rep., 2004. Masters programme on System and Network Administration, University of Amsterdam. URL: http://www.os3.nl/arjen/snb/asp/asp-report.pdf.
[22] W. R. Stevens, *Unix Network Programming*. Prentice Hall, 1990.
[23] W. R. Stevens, *Advanced Programming in the UNIX Environment*. Addison-Wesley, 1992.
[24] Free Software Foundation, "GNU Home Page." URL: http://www.gnu.org.
[25] L. Torvalds, "Linux Home Page." URL: http://www.linux.org.