Nom : Groupe ISC1

Laboratoire découverte OS et Réseaux 2022-2023

Marc SCHAEFER
Marc.Schaefer@he-arc.ch

http://www.he-arc.ch/ingenierie

31 octobre 2022

Neuchâtel, le 31 octobre 2022

Ces laboratoires ont pour but de permettre aux étudiants de 1ère année ISC de découvrir, en pratique, les concepts vus aux cours OS, Réseaux et Sécurité pratique, avec une application également du cours C (programmation socket).

Licence et droits d'auteurs

Marc SCHAEFER
Maître d'enseignement
Ing. inf. dipl. EPFL
http://www.he-arc.ch/ingenierie

Table des matières

0. Introduction

Contenu du chapitre

— objectifs du laboratoire *Découverte OS et Réseaux*
— votre implication
— évaluation

1

Objectifs du chapitre

— vous savez ce qui est attendu de vous
— vous connaissez les modalités d'évaluation
— vous savez que l'enseignant est disponible, mais pas le service informatique

Objectifs du laboratoire Découverte OS et Réseaux – 0.1

- mettre en pratique les éléments théoriques des cours du module OS+Réseaux, en particulier :
  - la mise en place de machines virtuelles GNU/Linux et Microsoft avec le logiciel de virtualisation VirtualBox
  - la gestion des fichiers et processus POSIX
2 - l'émulation (simulation) de réseaux avec Netkit
  - le développement système et réseau en C POSIX
  - appliquer des éléments de sécurité des réseaux
  - en option : la virtualisation légère (conteneurs Docker) et l'acquisition de données de gestion réseau (SNMP) et leur visualisation en Python ; ou le déploiement de services automatisés
- s'autonomiser (recherches documentaires, questions adéquates, travail régulier, etc)

Planification annuelle du cours-laboratoire La planification initiale du cours est disponible sur le Git [1]. Elle pourra changer en fonction de différents impondérables (synchronisation des autres unités d'enseignement du module, expérimentation du contenu).

Chaque étudiant vient au moins **une fois toutes les 2 semaines** : si votre emploi du temps le permet, il est possible de venir avec un autre groupe en cas d'empêchement – envoyer un petit mail à l'enseignant.

Errata et indications complémentaires S'il y a des erreurs dans ce polycopié (errata), vous trouverez également les corrections dans le Git, ainsi que des indications complémentaires, des références, etc.

Disponibilité de l'enseignant L'enseignant est disponible durant les cours, mais aussi par e-mail (*Marc.Schaefer@he-arc.ch*) et sur rendez-vous.

Attention : le service informatique ne doit *pas* être contacté pour des questions liées à ce cours.

1. `00_Org/deroule.md`

Votre implication – 0.2

— maintenance régulière des checklists
— présentation des checklists et objectifs atteints à l'enseignant (en général 2 semaines après le début d'un labo, vous avez un peu plus de temps pour le premier)
— recherches de solutions efficaces
— interaction avec vos collègues
3 — formulation de questions (en direct ou par e-mail) à l'enseignant

Evaluation – 0.3

— vous remplissez régulièrement les checklists pour vous situer
— l'enseignant passera régulièrement pour vérifier l'atteinte des objectifs

Votre implication Ce cours a pour but de vous aider à assimiler les notions des cours OS, Réseaux et Sécurité pratique, et à les mettre en pratique. Investissez suffisamment de temps pour que cela vous soit profitable. Echangez avec vos collègues. Utilisez les checklists pour vous situez. Cherchez des informations si nécessaire, posez des questions, devenez de plus en plus autonome.

Une idée : créez un fichier texte de *notes* dans lequel vous stockez vos trucs et astuces. Ainsi cela sera facilement recherchable dans un outil simple. Utilisez aussi la fonction de marque-page (bookmark) ou signets de votre navigateur Web.

Référencez ! Même si ce n'est pas indispensable pour ce cours, prenez l'habitude de référencer vos sources. Il y a plusieurs façons de faire, mais la meilleure consiste à référencer la source que vous avez lue (y compris Wikipedia), en indiquant de préférence la date de consultation. Vous pouvez aussi donner la référence directe à la version concernée, ou dans certains cas utiliser des services comme `https://archive.org` (archive du web à une date donnée) ou de raccourcissement d'URL.

Wikipedia est un outil formidable si on l'utilise de manière correcte : n'hésitez pas à le consulter, pour obtenir des informations de base sur divers sujets, pour *désambiguifier* des acronymes surchargés, et surtout pour obtenir des sources *primaires* : en effet, Wikipedia est une source secondaire : elle ne devrait en général pas comporter de travaux originaux, mais uniquement référencer des sources primaires. Ces sources primaires sont des livres ou des sites Internet archivés et vous permettent de remonter à la **source** de l'information et d'approfondir un sujet, voire de vérifier si Wikipedia a vraiment des informations correctes et à jour.

1. Virtualisation classique en pratique

Objectifs – 1.1

4

— installer Oracle VirtualBox (voir slide 1.2 en page 10 et le Git)
— créer et configurer une machine virtuelle Debian GNU/Linux buster (Debian 10 amd64 – voir dès la page 11)
— procéder à l'installation de base de cette machine virtuelle
— tester le fonctionnement
— améliorer l'intégration à l'hôte avec les *guest additions*
— accéder à un répertoire partagé avec le *host* et à un partage de la HE-Arc
— répondre aux questions

allez à votre rythme : n'oubliez pas maintenir la checklist en fin de ce labo (page 30)

Vocabulaire

architecture se dit de l'organisation d'une plateforme d'exécution : organisation de la mémoire, jeu d'instruction du processeur, ... : concrètement par exemple i386, i486, i686 (pour les processeurs classiques Intel et AMD 32 bits), armv6 (processeur ARM du RASPBERRY PI), amd64 (processeurs 64 bits évolués de l'architecture x86 – Intel qui a suivi le précurseur AMD l'appelle EMT64, Linux l'appelle x86_64), ...

la virtualisation a pour but de faire coexister plusieurs systèmes d'exploitation identiques ou différents, côte à côte, en limitant de manière plus ou moins étendue l'accès au matériel par une couche d'abstraction logicielle : le(s) processeur(s), la RAM, le stockage, le réseau peuvent notamment être simulés.

système hôte ou *host* en anglais, héberge une ou plusieurs machines virtuelles grâce à un **hyperviseur** qui les gère et les isole : l'hôte peut se limiter à une couche très fine : p.ex. VMWare ESX comporte un mini-système hôte Linux

machine virtuelle ou VM, *guest*, VPS, VE, conteneur ... : le système virtualisé.

isolation il existe plusieurs type de virtualisation qui se différencient principalement par le niveau d'isolation, le degré d'émulation du matériel, la coopération éventuelle du système hébergé et la performance résultante : les *containers* en sont un exemple particulièrement efficace.

I/O ou E/S (input-output, entrées-sorties) : un élément particulier dont le niveau d'isolation et de virtualisation peut varier.

Installation de VirtualBox sur votre laptop – 1.2

- téléchargez l'installeur VirtualBox depuis l'URL indiquée ci-dessous
- exécutez pour installer
- familiarisez-vous avec le lancement (qui est automatique à l'installation) et avec l'agencement de l'interface graphique
- en général, il sera possible de créer des machines virtuelles *64 bits* depuis VirtualBox : si cela n'est pas disponible, assurez-vous que la virtualisation VT est activée dans votre BIOS :
  - procédure pour aller dans le BIOS – faites appel à l'enseignant
    - chaque fabriquant propose une fonction au démarrage (p.ex. DEL, F10, etc)
    - alternative, depuis Microsoft Windows 10 : Réglages/Mise à jour, puis Sécurité/Récupération, puis Démarrage avancé, puis Options avancées, et Configuration du BIOS ou approchant
  - la configuration VT se trouvera sous sous CPU ou sous Sécurité / Virtualisation

Téléchargement de VirtualBox Téléchargez l'installateur de VirtualBox depuis `https://gitlab-etuing.he-arc.ch/isc/2022-23/niveau-1/decouverte-os-et-reseau/-/tree/main/00_Org/binaries` et profitez également de télécharger l'image ISO Debian – que l'on installera plus tard en machine virtuelle – qui s'y trouve aussi.

Support multiplateforme Nous supposerons que vous avez Microsoft Windows sur votre laptop (le *host*). VirtualBox étant multiplateforme (et en grande partie *logiciel libre* selon la licence GPL), si vous n'avez pas Microsoft installé sur votre laptop, il est également possible de procéder en installant sur GNU/Linux (de préférence par les packages, p.ex. `virtualbox-ose`) ou Mac OS X par `http://www.virtualbox.org/`).

Spécificités d'installation de VirtualBox Il faut que l'utilisateur connecté puisse avoir les droits d'administration (ce qui est normalement le cas sous Microsoft Windows 7 – et ultérieur – pour les premiers comptes configurés dans le système).

Il faut simplement accepter les configurations d'installation par défaut.

L'installation dure environ 5 minutes et ne nécessite pas de redémarrage. Elle installe le logiciel et des composants systèmes comme des pilotes pour la virtualisation du réseau.

Création d'une machine virtuelle – 1.3

- utilisez l'icône Nouvelle (ou Machine → Nouvelle)
- paramètres du Wizard de création de machine virtuelle :
  - nom : debian-buster-amd64-simple
  - type : Linux, Debian 64 bit (préconfiguration automatique)
  - mémoire : 1 GB
  - espace disque virtuel : 12 GB (type VDI, dynamiquement alloué)
- une fois le Wizard terminé, activez le bouton Configuration et faites les observations demandées en page 12
- assurez-vous que la fonction *Presse-papier bidirectionnel* soit activée
- téléchargez le CD d'installation net-install depuis le même emplacement que celui de VBox (voir slide 1.2) et **associez-le** comme CD de démarrage (pas Live)
- démarrez la machine virtuelle et visualisez sa console : choisissez le démarrage sur CD ; le BIOS virtuel se lance puis le CD d'installation démarre sur un menu

Modes réseau d'une machine virtuelle Les modes possibles sont :

NAT le *host* s'occupe des détails de l'accès au réseau : notre machine virtuelle (le *guest*) aura une adresse IP privée (non routable sur Internet) derrière un routeur/NAT (comme chez vous à la maison en ADSL p.ex.).

Le principe du mode NAT est qu'il isole en partie notre machine virtuelle du réseau réel par routage et réécriture d'adresse (NAT, *Network Address Translation*) : notre machine virtuelle n'est pas accessible [1] depuis le réseau par exemple, mais elle peut accéder tout Internet (couche 3 OSI).

L'avantage est que la configuration est possible quelque soit la façon dont vous accédez réellement le réseau externe (WiFi, filaire).

bridge ou pont, switch : comme si le *guest* était directement connecté à l'interface réseau réelle qui est mentionnée et obtient donc p.ex. une adresse IP dynamiquement dans le même sous-réseau ; cela offre la plus grande flexibilité (p.ex. pour offrir des services virtualisés au sous-réseau, ou pour installer via réseau avec PXE) ; l'isolation est moindre (l'adresse MAC de la carte réseau virtuelle doit être unique dans le réseau ainsi bridgé), et c'est plus difficile à faire en WiFi à cause de l'authentification (couche 2 OSI)

interne uniquement pour relier en réseau des *guests* entre eux

privé hôte idem interne, mais avec un accès possible par le *host* seul.

générique divers protocoles permettant p.ex. la liaison entre *guests* situés sur des *hosts* distants.

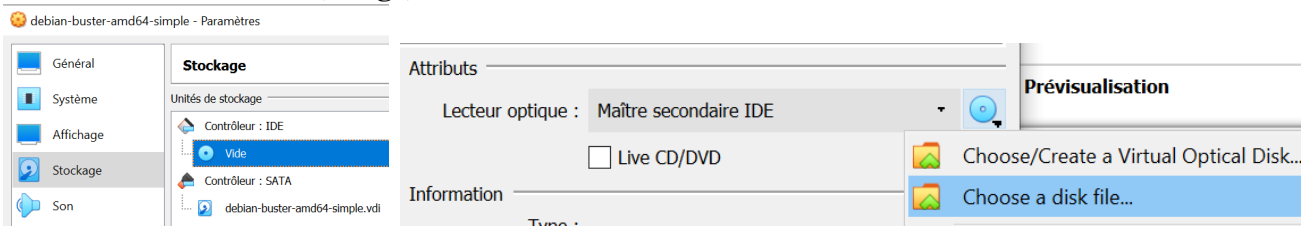
1. il est possible de configurer des redirections de port dans ce mode, toutefois.

Importation de machines virtuelles On peut importer une machine virtuelle, via Fichier → Importer. Le format OVF est interopérable entre VirtualBox et VMWare. On peut bien sûr exporter une machine virtuelle.

Types de fichiers de disque-dur virtuel Les différents types permettent une compatibilité avec d'autres logiciels de virtualisation (on peut aussi exporter si nécessaire après coup), ou, proposer des fonctions avancées : redimensionnement automatique (avec un *sparse file* – fichier à trou – sous POSIX) ou des métadonnées sous Microsoft) ; ou le partage économique d'un seul fichier de stockage de base par plusieurs machines virtuelles en utilisant des fichiers annexes de type *copy-on-write*, avec le format QCOW.

Ces disques-durs virtuels seront ici alloués comme fichiers dynamiques sur le host, mais pourraient également être fournis au sein d'un stockage virtualisé : par exemple un SAN (*Storage Area Network*) offrant un disque brut par réseau (*Network Block Device*) via les protocoles iSCSI ou DRBD (*Distributed Replicated Block Device*).

Associer le fichier ISO (image) au lecteur CD virtuel



Live CD/DVD Un Live CD/DVD est un CD/DVD (ou une clé USB) qui permet d'utiliser un système d'exploitation sans l'installer, ce qui est pratique pour les tests ou la sécurité. Toutefois, ici, nous allons installer l'OS, donc n'activez pas cette option.

Observations

1. notez combien de CPU virtuels sont activés : . . .
2. comment la 1ère carte réseau virtuelle est configurée, notamment son mode [2] : . . . et si elle est activée : □
3. de quel nom de bus [3] et type est le contrôleur de stockage du disque simulé : . . .
4. constatez, avant l'association ou après l'installation, que le CD/DVD-ROM simulé est *vide* (sous contrôleur IDE) : □.

2. voir page 11
3. un bus est un média d'interconnexion entre plusieurs périphériques : par exemple le bus PCI ou PCI-Express, ou par extension SATA, USB – c'est aussi une topologie (cf cours réseau) mais ceci n'est pas le sujet ici

Installation de base de Debian buster – 1.4

— l'installation se fait dans l'espace confiné (isolé) de la machine virtuelle
— l'installation (voir ci-dessous) se fait dans ce cas en mode texte
— vous pouvez attribuer le focus en cliquant dans la console de la VM
— vous pouvez mettre en/sortir du plein écran avec CTRLdroite–f (touche Host–f) ; si nécessaire utilisez le mode mise à l'échelle
7 — si pendant l'installation vous vous ennuyez à certains moments, lisez toutes les pages de ce document et répondez aux premières questions en page 27 (en vous aidant bien sûr des ressources indiquées en page 160 et d'Internet)
— dès que les fichiers sont téléchargés, vous pouvez suspendre la VM pour reprendre plus tard si nécessaire
— après l'installation, la machine redémarrera en mode graphique

Eléments importants de l'installation Navigation : touches curseurs gauche/droite ou haut/bas ; touche ENTER pour confirmer ; touche TAB pour changer de secteur.

1. activez la fonction **Install** avec curseur bas, puis touche ENTER
2. **localisation** : choisissez avec les touches curseurs bas et haut la langue qui vous convient (French) et confirmez avec ENTER
3. indiquez que vous êtes en Suisse
4. confirmez le choix du **clavier suisse romand** [1]
5. le **réseau** devrait se configurer automatiquement
6. choisissez un nom de machine à votre convenance (de préférence en minuscules, sans espaces ni caractères spéciaux) ou laissez le nom par défaut (aucune importance pour Linux)
7. le domaine peut être laissé vide
8. configurez le **mot de passe de root** [2] (mot de passe, confirmation), (notez-le [3]) : ...

1. ou le clavier que vous avez réellement sur votre machine !
2. le super-utilisateur – le compte `root` peut aussi être désactivé si l'on emploie `sudo` comme nous le verrons plus tard
3. en général une mauvaise pratique !

9. créez un utilisateur non privilégié [4] (nom complet, login [5], mot de passe, confirmation), notez son login et son mot de passe : ...
10. vous pouvez annuler la Récupération de l'heure depuis un serveur de temps si ça coince (ENTER)
11. choisissez le **partitionnement** Assisté disque entier (par défaut [6], ENTER)
12. confirmez le périphérique virtuel reconnu (ENTER) – il correspond au fichier de 12 GB maximum que vous avez créé sur le *host*
13. tout (les fichiers) dans une seule partition (par défaut, ENTER)
14. consultez le partitionnement retenu : il y a le système de fichiers racine / (où seront les fichiers systèmes et ceux des utilisateurs) et une partition de mémoire virtuelle (*swap*, mémoire d'échange) ; confirmez avec ENTER (Terminer)
15. vous devez encore une fois confirmer (touche curseur gauche, puis ENTER)
16. la création du système de fichiers [7] et l'installation du système de base s'effectuent
17. il n'y a pas de CD ou DVD supplémentaire à scanner, nous installerons les **paquets logiciels** (packages) par réseau
18. les paquets doivent être installés depuis un miroir, choisissez Suisse → `debian.ethz.ch`, puis ENTER
19. configurez le **proxy** du laboratoire (pour la performance, car c'est aussi un cache qui évitera de transférer plusieurs fois les paquets logiciels pour chaque étudiant) `http://proxy.teleinf.labinfo.eiaj.ch:3128/` – uniquement si vous faites l'installation à l'Ecole ou en VPN !
20. les listes de paquets se chargent
21. les logiciels déjà installés sont mis à jour et configurés automatiquement ; d'autres paquets logiciels sont installés
22. acceptez ou non l'activation de *popularity-contest* – en fonction de votre sensibilité à la confidentialité de vos choix de paquets logiciels
23. ajoutez **avec précaution** [8] le groupe de logiciels **MATE** [9], laissez les autres groupes présélectionnés tels quels
24. les paquets logiciels sélectionnés sont téléchargés et installés automatiquement (environ 1200)

4. certaines distributions Linux autorisent le premier utilisateur créé à devenir `root` avec `sudo`, Debian ne fait rien de tout cela par défaut
5. de préférence en minuscules, sans accent, espace ni caractères spéciaux ; un login de 8 caractères maximum est recommandé
6. les autres choix permettent la configuration d'un gestionnaire de volumes logiques (LVM, pour redimensionner ou ajouter de la capacité facilement), voire même de chiffrer les données, ou une configuration manuelle, par exemple pour faire coexister plusieurs OS, pas nécessaire ici
7. se fait ici automatiquement, sinon via la commande `mkfs.ext4`, fonction parfois imprécisément appelée *formatage*
8. déplacement avec touches curseurs, sélection ou déselection avec espace, confirmation finale avec ENTER
9. si vous ratez cette étape, refaites-la en revenant en arrière un peu plus tard dans l'installation ; et au premier login choisissez MATE plutôt que GNOME – en alternative `tasksel` pourra y être lancé à nouveau

25. confirmez l'installation du programme de démarrage GRUB (bootloader) sur le premier disque brut (virtuel) `/dev/sda` (utilisez les touches curseur puis ENTER)
26. confirmez le redémarrage de votre machine virtuelle juste installée
27. la machine redémarre en mode graphique (désassociation automatique du CD de démarrage par VirtualBox – vous pouvez aussi le supprimer dans la configuration, à choix)

Disque brut, partitions, volumes logiques et RAID Un périphérique type bloc brut (*raw block device*) est un arrangement linéaire de blocs présenté par un disque. Sous Linux, `/dev/sda` est le premier disque brut. On peut l'utiliser directement, mais généralement on le partitionne en sous-unités dédiées chacune à une application particulière. Par exemple, le partitionnement automatique que nous avons choisi produit la table de partitions suivantes :

```
/dev/sda1  partition primaire          fs racine / ext4
/dev/sda2  partition primaire/étendue  contient des p. logiques
/dev/sda5  partition logique 1         mémoire d'échange (swap)
```

Le format d'une table de partitions peut être historique ou GPT (étendu, pour les très grands disques modernes).

Dès le moment qu'on a un périphérique bloc brut, on peut créer [10] un système de fichier Linux puis le monter [11]. Ou s'il s'agit d'un swap, de l'initialiser [12].

Il aurait aussi été possible de partitionner différemment : que des partitions primaires, ou alors une seule partition pour LVM, avec le découpage en volume logique dans LVM ; ou même une partition RAID dans laquelle LVM est mis en place, puis un système de fichiers à l'intérieur.

Voici un exemple plus complexe (sortie de la commande `lsblk`) :

```
NAME                MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                   8:0    0   1.8T  0 disk
`-sda1                8:2    0   1.8T  0 part
  `-md0               9:1    0   1.8T  0 raid1
    |-vg1-root      253:0    0  18.6G  0 lvm   /
    |-vg1-swap      253:8    0     4G  0 lvm   [SWAP]
sdb                   8:16   0   1.8T  0 disk
`-sdb1                8:18   0   1.8T  0 part
  `-md0               9:1    0   1.8T  0 raid1
    |-vg1-root      253:0    0  18.6G  0 lvm   /
    |-vg1-swap      253:8    0     4G  0 lvm   [SWAP]
```

Deux partitions de deux disques bruts (`/dev/sd[ab]1`) sont intégrées dans un array RAID1 (miroir, fiabilité), lui même formant un groupe de volumes dynamiques (vg1), lui-même découpé en plusieurs unités logiques (`/dev/vg1/root`, `/dev/vg1/swap`) servant respectivement à la racine du système de fichiers / et à du swap. L'avantage est que LVM permet du redimensionnement dynamique en-ligne.

Sans l'abstraction offerte par LVM, qui évite de dépendre des noms actuels des périphériques bruts, il est aussi possible de référencer les partitions contenant un système de fichiers ou du swap par identificateurs uniques (UUID).

10. `mkfs.ext4 /dev/sda1` par exemple ci-dessus a été fait par l'installeur
11. `mount /dev/sda1 /mnt` – voir `/etc/fstab` pour ce qu'a fait l'installeur, probablement un montage par identificateur unique de la partition
12. avec `mkswap /dev/sda5` – déjà fait par l'installeur

Environnements de bureau GNU/Linux En plus du gestionnaire de login (*display and login manager*, comme `gdm3` ou `lightdm`), X11 [13] propose de nombreux environnements de bureau [14], en fonction de critères comme l'expérience utilisateur (UX), la performance, la stabilité, le niveau de configurabilité.

Parmi les choix possibles [15], citons :

GNOME 3 expérience utilisateur très moderne (proche des tablettes), utilisation de l'*accélération matérielle*, nécessite un matériel performant, évolue vite, relativement peu de configurations à part les éléments essentiels ; utilisation typique : débutant, tablette ; basé sur un toolkit C

MATE expérience utilisateur classique GNOME, ne nécessite pas de matériel très performant, grande stabilité de l'expérience utilisateur, relativement peu de configuration à part les éléments essentiels : utilisateur classique débutant – cela sera **notre choix** !

KDE expérience utilisateur classique basée Qt, un toolkit C++, qui sera étudié en $2e$ année

OpenBox pour l'expert qui désire tout configurer et optimiser, voir `https://wiki.debian.org/Openbox`

13. The X Window System, version 11, Release 6
14. on distingue entre gestionnaire de fenêtre (*window-manager*, qui s'occupe de la décoration des fenêtres et des fonctions de base), et bureau complet (*desktop manager*, comprenant un gestionnaire de fenêtre)
15. installables simultanément avec choix au login

Se logguer et installer des packages – 1.5

— logguez-vous en mode graphique avec l'utilisateur non privilégié créé lors de l'installation (ne travaillez pas sous `root`)
— lancez une ligne de commande (shell `bash`) dans une fenêtre de terminal MATE (tapez ALT–F2, puis `mate-terminal` ; alternative : menu Applications → Outils systèmes → Terminal MATE)
— votre utilisateur n'est pas privilégié : listez vos droits avec la commande `id`
8
— devenez `root` avec la commande de changement d'utilisateur (*switch user*) `su -` et remarquez les deux différences sur l'invite (*prompt*) du shell, puis comparez la sortie de la commande `id` avec précédemment.
— revenez au shell normal avec `exit` (ou CTRL–d pour fin de fichier), remarquez le prompt normal à nouveau
— installez avec `apt-get install` le package `xterm` (un terminal plus classique) : testez d'abord l'installation sous votre utilisateur et expliquez pourquoi cela ne fonctionne pas, puis faites-la sous `root`, enfin faites ALT–F2 `xterm` pour tester ce terminal

Mode texte Vous pouvez aussi vous logguer en mode texte (consoles textes accessibles avec CTRL–ALT–Fx, où x est le numéro de la console ; retour au mode graphique avec ALT–F7). Attention, si le *host* est Linux, cela risque de basculer les consoles du *host* à la place du *guest* : utiliser CTRLdroite-Fx.

Super-utilisateur Avec une installation comme nous l'avons faite, le super-utilisateur (ou administrateur) `root` (UID 0) peut aussi se connecter en mode texte ou graphique : cela n'est pas recommandé (sécurité) : le mieux est de se logguer sous un utilisateur normal puis d'exécuter des commandes sous le super-utilisateur via la commande `su -` (l'argument – étant nécessaire pour que les chemins (`PATH`) soient corrects), ou, comment nous allons le configurer ensuite, avec `sudo` : dans ce cas, le mot de passe `root` n'est plus nécessaire et pourrait être bloqué avec `passwd -l`.

En ligne de commande (shell bash), le fait qu'on travaille sous root est dénoté par le caractère `#` [1] (à la place de l'usuel `$` de l'invite de commande). De plus, l'utilisateur root est parfois spécifiquement affiché, en plus.

Installation de packages Debian GNU/Linux buster propose plusieurs interfaces d'installation de paquets logiciels (*packages*), que cela soit en ligne de commande (shell) ou en mode graphique. La liste des sources d'installation se trouve dans le fichier `/etc/apt/sources.list` : essayez de l'afficher avec `cat` : les lignes désactivées sont commentées avec un `#`.

1. on retrouve aussi cette convention en ligne de commande de matériel réseau CISCO

9

— redevenez `root` et ajoutez le privilège (groupe) `sudo` à votre utilisateur : `addgroup demo sudo`[a], si `demo` est votre utilisateur, puis quittez le shell `root`
— délogguez-vous de l'interface graphique (Système → Fermer la session → Déconnexion) et relogguez-vous pour activer le groupe précédemment ajouté
— sous votre utilisateur, relancez un terminal puis vérifiez que l'installation de packages peut également être faite via `sudo apt-get install bc`[b], tapez votre *propre* mot de passe.
— calculez 2^{128} à l'aide de la commande `bc` (^ est une touche morte, tapez un espace après ; terminez la commande avec `quit` ou CTRL–d)
— comparez la commande `id` avec la commande `sudo id`
— faut-il à nouveau entrer votre mot de passe si vous exécutez plusieurs `sudo` en séquence rapide ?
— lancez un shell sous `root` avec `sudo -s`, vérifiez que le prompt vous indique que vous êtes `root`, puis terminez avec `exit` ou CTRL–d.

a. si cela ne marche pas, lisez le paragraphe Super-utilisateur à la page précédente
b. ce package est en fait déjà auto-installé : vérifiez qu'il n'y a pas de message d'erreur

Mises à jour Pour pouvoir installer (ou mettre à jour) des packages, il faut mettre à jour la liste des packages avec `apt-get update`.

Il y a une différence entre mettre à jour la liste des packages (`update`) et mettre à jour, ensuite, les paquets logiciels installés (`-u dist-upgrade`). Debian peut faire les mises à jour automatiquement, ou, par défaut, vous pouvez les faire vous même.

NB : il existe aussi la commande `apt`, qui est plus moderne mais dont la sortie texte est plus difficilement interprétable dans des scripts.

Edition de la ligne de commande du shell bash Les touches curseur droite et gauche, backspace et delete peuvent être utilisées. Il existe également des raccourcis de positionnement [1] (p.ex. CTRL–a début de ligne, CTRL–e fin de ligne, CTRL–k couper, CTRL–y coller, ou les touches de positionnement du pavé numérique comme Home et End).

La souris ne sert en général qu'au copier coller : vous ne pouvez pas utiliser CTRL–c, CTRL–x et CTRL–v car ils servent au shell : utilisez le menu Editer avec `mate-terminal`. On peut aussi utiliser l'ancien système de sélection X11 : on sélectionne *et* copie simultanément avec le bouton de gauche, et on colle avec le bouton du milieu ou les deux boutons en même temps.

1. compatible GNU readline

Navigation dans l'historique L'historique [2] des commandes se visite avec les touches curseur haut et bas : on peut aussi rechercher dans l'historique inverse avec CTRL–r puis un début du texte de la commande déjà exécutée ; on passe au précédent avec CTRL–r, on exécute avec ENTER, on annule avec CTRL–c.

Signaux et fonctions spéciales du terminal En plus, dans un terminal, CTRL–d signifie fin de fichier (essayez avec un `cat`, qui, sans argument, va copier le clavier à l'écran) et CTRL–c envoie un signal de terminaison au processus.

Avant-plan et arrière-plan On peut aussi suspendre une commande avec CTRL–z, puis la remettre en avant-plan avec `fg` (foreground) ou `%`, ou en arrière-plan avec `bg` (background). On peut lancer une commande en arrière-plan déjà au lancement en y ajoutant `&`. Un programme en avant-plan est relié au clavier (il "bloque" le clavier). Nous verrons les détails au chapitre 4.

Découvrez votre nouveau système Debian en ligne de commande Par exemple :

- listez les systèmes de fichier : `df -h` : vous devriez trouver le système de fichier racine (/) sur lequel le système a été installé, ainsi que des systèmes de fichiers virtuels du système
- déterminez la version du kernel et l'architecture : `uname -a` : en 64 bits, cela devrait être x86_64
- déterminez l'usage actuel de la mémoire vive et d'échange (swap) : `free`
- déterminez la configuration réseau : `ip addr show; ip route show` : nous en reparlerons plus tard dans le cours
- listez les bus PCI et USB : `lspci; lsusb` : sont-des périphériques réels ou virtuels ici ?

HE-Arc et proxy pour cette VM (informatif, ne pas faire !) Nous avons configuré le proxy du laboratoire de téléinformatique pour augmenter la performance (il dispose d'un cache), juste pour l'installation de packages, dans le sous-système APT (*A Packaging Tool*).

La machine virtuelle ainsi configurée ne peut *pas* être utilisée en-dehors de la HE-Arc ou VPN [3] pour installer des logiciels. Pour déconfigurer le proxy en cas de nécessité, il suffit de commenter le proxy (avec #) dans `/etc/apt/apt.conf` ou un des fichiers de `/etc/apt/apt.conf.d/`.

2. affichable avec la commande `history`, stocké en fin de session shell dans le fichier `.bash_history`
3. la VM ayant été configurée en mode NAT, cela fonctionne

Guest additions et paravirtualisation – 1.6

— il est possible de modifier légèrement l'OS virtualisé pour qu'il soit *conscient* de cette virtualisation et utilise des interfaces logicielles plus performantes, via l'installation de *guest additions* (si elles existent pour l'OS considéré !)
— vos buts seront (voir ci-après) :

1. installer les *guest additions*
2. vérifier que l'on peut facilement redimensionner la fenêtre de la VM et qu'elle s'adapte automatiquement (hors mise à l'échelle !)
3. vérifier que le copier-coller *host* vers votre *guest* et vice-versa fonctionne
4. configurer un partage de données *host* / *guest persistant* et tester l'échange de données
5. reconfigurer l'interface réseau VirtualBox pour qu'elle utilise le pilote efficace `virtio-net` et vérifier que cela fonctionne dans le *guest*

Virtualisation de base Sans logiciel spécifique installé dans la machine virtuelle (*guest*), l'hyperviseur VirtualBox va simplement présenter au *guest* du matériel factice, souvent d'ailleurs de vieux périphériques pour une compatibilité maximale.

Ces périphériques vont être accédés par le *guest* de manière très inefficace :
— émulation *registre par registre* de vieilles cartes réseaux, SATA (stockage) ou VGA : performance insuffisante
— partage de données entre l'hôte et la machine virtuelle par partage de fichier classique, complexe à mettre en place
— impossibilité d'accéder directement au matériel réel de la machine
— impossibilité de contrôler en temps réel la mémoire allouée à la machine virtuelle (il faut la redémarrer pour changer ses paramètres)

Guest additions et paravirtualisation Il est possible pour remédier à ces inconvénients d'installer du logiciel (des pilotes) dans la machine virtuelle – développé spécialement pour l'OS *guest*, par exemple :

— pilote VirtIO pour les I/O disques ou réseau
— pilote d'accès au système de fichiers de l'hôte
— *balloon driver* pour la gestion de la mémoire du *guest* par l'hôte
— accès direct à certains matériels de l'hôte (carte réseau ou graphique dédiée par exemple)

Le type de virtualisation change légèrement (*paravirtualisation*), la machine virtuelle devenant consciente qu'elle parle avec un hyperviseur.

Installation des Guests additions Certains OS vont détecter qu'ils sont virtualisés et vont donc installer automatiquement les bons pilotes et outils de paravirtualisation. Ici, l'installation doit être faite manuellement, d'une des manières suivantes :

— en insérant un CD virtuel depuis VirtualBox ou comme package Debian et en installant les outils qui s'y trouvent (et en mettant à jour manuellement à chaque fois que c'est nécessaire)
— ou, si existant, via le système de paquet Debian, ce qui garantit la maintenabilité et la sécurité
— ou via des packages gérés par des tiers (avec des risques de maintenabilité et de sécurité)

Pour des raisons de maintenance à long terme non garantie par Oracle VirtualBox, Debian buster n'intègre pas les packages VirtualBox directement, dont les *guest additions*, ni n'assure leur support sécurité spécifiquement.

Nous allons utiliser le CD des Guest additions, disponible comme package Debian : toutefois, il faut garder à l'esprit qu'il faudra refaire dès le point 5 de l'installation ci-après, manuellement, à chaque mise à jour du package.

De plus, ce package Debian ne se trouve pas dans la section générale `main` mais dans la section `non-free` (packages non libres au sens de Debian) : il faut activer cette section, et il faudrait vérifier la licence individuelle du package dans `/usr/share/doc/NOM-PACKAGE/` : un tel package n'est pas forcément redistribuable sans restrictions !

Opérations à effectuer :

1. ajouter le mot `non-free` après `main` dans toutes les lignes du fichier des sources d'installation :
   `sudo sed --in-place "s/main$/main non-free/" /etc/apt/sources.list`
   (`sed` permet de modifier des flux, et l'option `--in-place` de modifier le(s) fichier(s) indiqués en argument(s) de ligne de commande ; ici la modification est une substitution (s) ; grâce à la spécification de fin de ligne (`$`), la commande n'a pas d'effet si elle a déjà été lancée (idem-potente)
2. faire la mise à jour de la liste des paquets, et vérifier qu'il n'y a aucune erreur :
   `sudo apt-get update`
3. mettre à jour les paquets :
   `sudo apt-get -u dist-upgrade`
   redémarrer si un nouveau kernel a été installé
4. installer les outils et fichiers includes du langage C nécessaires à la compilation des pilotes (*drivers*) VirtualBox :
   `sudo apt-get install build-essential dkms linux-headers-$(uname -r)`
   (ici `uname -r` retourne la version du kernel, l'expansion sous la forme `$( ... )` permet de remplacer la sortie dans l'argument)
5. installer le package contenant les additions invité du CD ISO :
   `sudo apt-get install virtualbox-guest-additions-iso`
6. monter ce fichier de CD ISO comme fichier virtuel, exécuter l'installation, et démonter :
   (c'est l'occasion d'essayer la touche TAB pour compléter les chemins)
   ```
   sudo mount -o loop /usr/share/virtualbox/VBoxGuestAdditions.iso \[1]
                      /mnt
   sudo bash /mnt/VBoxLinuxAdditions.run
   ```

1. le backslash, qui annule la signification spéciale du caractère qui le suit, ici le saut de ligne, n'est là que pour indiquer que tout doit être écrit sur la même ligne, sans backslash !

(l'erreur sur l'insertion du module vboxsf est normale, les pilotes seront compilés au prochain démarrage)
`sudo umount /mnt`
supprimer les packages téléchargés : `sudo apt-get clean`[2]

7. désactiver éventuellement le mode écran mise à l'échelle (avec deux fois Host-f), sauf si écran 4k
8. redémarrer la machine virtuelle (à cause de la modification du kernel), via le menu Système → Eteindre → Redémarrer.

Copier-coller Par défaut, X11 (l'interface graphique UNIX utilisée par Linux) supporte un copier-coller simplifié que vous pouvez tester à l'intérieur de la machine virtuelle buster dans un `xterm` : sélectionner et copier avec le bouton de gauche, copier avec le bouton du milieu (ou les deux boutons en même temps).

Les applications modernes supportent également les fameux CTRL–c et CTRL–v ou les options ad-hoc du menu Editer : vous pouvez tester entre le host (Notepad p.ex.) et le guest (`pluma &` ou une console `mate-terminal`[3]).

Notez que le copier/coller X11 ne fonctionne qu'entre hôte (copier sous Microsoft Windows avec CTRL–c) et VM (coller sous Linux avec le bouton du milieu). Pour le faire aussi dans l'autre sens (VM vers hôte), il faut utiliser une des applications modernes ci-dessus et le CTRL–c.

Partage avec le host Ajouter un partage dans la configuration VirtualBox de votre VM (p.ex. quand la VM est activée, dans Périphériques → Réglages des dossiers partagés). La première case est le répertoire sur le *host* (p.ex. un répertoire à créer dans votre Bureau Microsoft) et la deuxième le nom du partage tel que vu par le *guest* (ci-après `PARTAGE`). Cochez les coches `Montage automatique` et `Configuration permanente`.

Recommandation : ne pas utiliser un emplacement de répertoire partagé sur votre Microsoft OneDrive : contrôlez dans le navigateur de fichier Microsoft le chemin d'accès, et en cas de doute, mettez le répertoire de partage par exemple dans `C:\Users\votre.utilisateur\Partage`.

Au prochain démarrage de la VM, le partage sera automatiquement monté sous `/media/sf_PARTAGE`[4], mais il ne sera accessible qu'aux membres du groupe `vboxsf` (*Virtual Box Shared Folder*), ce qui n'est pas forcément très pratique[5]. Vous verrez probablement ce nouveau partage – inaccessible – sur le bureau. Vous pouvez visualiser avec la commande `df` les différents systèmes de fichiers montés sur votre machine.

Arrêtez la VM proprement. Désactivez l'option `Montage automatique`. Démarrez la VM. La commande `df` devrait montrer que le montage automatique n'a plus eu lieu. Puis montez manuellement le répertoire partagé à un endroit spécifique de l'arborescence UNIX (p.ex. dans un sous-répertoire de votre propre répertoire), comme ceci :

2. on peut comparer la place libre sur / avant et après avec la commande `df`
3. CTRL–c y est inutilisable car c'est la commande d'interruption d'une commande du shell : consulter le menu Edition
4. si vous avez configuré l'option d'automontage : visualiser l'effet avec la commande `df`
5. on pourrait aussi ajouter votre utilisateur dans ce groupe avec `addgroup`, mais cela n'est pas ce que l'on va faire ici

1. décider où vous voulez monter ce répertoire partagé dans le *guest* : par exemple sous `/home/demo/PARTAGE` si votre utilisateur est `demo`
2. créer ce répertoire par exemple avec `mkdir /home/demo/PARTAGE`
3. monter le partage sur ce répertoire, avec, sur une seule ligne :
   `sudo mount -t vboxsf -o uid=1000,gid=1000 PARTAGE /home/demo/PARTAGE` – dans la mesure où `id` vous indique bien que UID et GID valent 1000 pour votre utilisateur ; adaptez `demo` au nom de votre utilisateur, PARTAGE au nom du partage pour VirtualBox et /home/demo/PARTAGE au nom du répertoire sur votre Linux

Si vous avez des mises en garde côté *host* ou des problèmes, essayez d'abord de redémarrer la VM, puis si cela ne marche toujours pas supprimez et recréez le partage avec la VM arrêtée.

Le partage peut maintenant être accédé depuis le *guest* via le shell (ligne de commande) (`cd PARTAGE`, `ls`, ...) ou par l'interface graphique [6] : Répertoires → Dossier personnel puis PARTAGE. Les éventuels fichiers et répertoires se trouvant auparavant dans `/home/demo/PARTAGE` sont invisibles jusqu'au démontage.

Démonter avec `sudo umount /home/demo/PARTAGE`.

Montage automatique (persistant) Pour rendre le montage du partage automatique au démarrage, nous allons utilise la méthode classique UNIX qui met à disposition le montage au démarrage de la machine ; vous allez procéder par étapes :

1. le démonter s'il est actuellement monté (voir commandes `df` et `sudo umount /home/demo/partage` – adapter bien sûr !) et vérifier que son montage n'est pas automatique dans la configuration VirtualBox
2. ajouter ce montage au fichier système `/etc/fstab` (voir ci-dessous)
3. tester que ce montage fonctionne avec `sudo mount -a` (ce qui devrait faire l'équivalent de `sudo mount /home/demo/PARTAGE`, ce point de montage figurant désormais dans `/etc/fstab`) et `df`
4. seulement si le point précédent a fonctionné : redémarrer votre machine virtuelle Linux et contrôler avec `df` que le montage automatique est bien persistant.

Pour ajouter ce montage à `/etc/fstab`, faites comme suit, en faisant *attention à ce que vous faites*, car si ce fichier est corrompu, le démarrage du système sera interrompu et vous passerez en mode maintenance : éditez `/etc/fstab` via `sudo` et un éditeur en mode texte, comme `nano` ou `vi` si vous le maîtrisez, ou si vous préférez l'éditeur en mode graphique `pluma` ou `gedit` :

```
sudo pluma /etc/fstab
```

et vous y ajouterez une nouvelle ligne de la forme :

```
PARTAGE /home/demo/PARTAGE vboxsf defaults,uid=1000,gid=1000 0 0
```

6. dans de rares cas, il peut y avoir des erreurs lorsque l'on sauvegarde un fichier depuis `pluma` ou `gedit`, voir https://unix.stackexchange.com/questions/52951/gedit-wont-save-a-file-on-a-virtualbox-share-text-file-busy

La syntaxe utilisée est [7] : nom du partage dans VirtualBox (qui est par défaut le nom du répertoire dans Microsoft) ; point de montage dans Linux, type du système de fichiers (*VirtualBox shared folder*), options de montage et enfin deux nombres à zéro : adapter les chemins à votre configuration réelle !)

Attention : si vous renommez ou déplacez le répertoire de partage côté Microsoft, vous devez impérativement modifier la configuration de VirtualBox, voire le fichier `/etc/fstab` si le nom VirtualBox du partage a également changé, à défaut, vous passerez en mode maintenance au prochain démarrage de votre machine virtuelle.

Configuration de virtio-net

1. déterminez que votre VM Linux n'utilise pas les pilotes efficaces pour sa carte virtuelle Ethernet, mais un pilote réel lié à une émulation du matériel par VirtualBox (commande `lspci | grep Ether` qui liste le bus PCI, ici virtuel, où est branché le matériel de la machine : contrôleur graphique, réseau, etc)
2. arrêtez la VM Linux proprement
3. modifiez la configuration de cette VM dans VirtualBox sous Réseau → Carte réseau 1 → Avancé pour que VirtualBox n'émule pas du matériel mais propose une API [8] efficace au *guest*, le `virtio-net`
4. redémarrez la VM Linux
5. comparez la sortie de `lspci`
6. vérifiez que le réseau est configuré avec `ip addr show` et testez qu'il est fonctionnel par exemple avec un `ping 8.8.8.8` (terminer avec CTRL–c) puis par exemple `sudo apt-get update`

Montage d'un répertoire de la HE-Arc Pour cette partie, il faut être à la HE-Arc ou avoir lancé le VPN. Le chemin UNC [9] `\\intra.he-arc.ch\ORG\HE-Arc\Common` est le chemin DFS complet menant à un répertoire partagé de la HE-Arc. Toutefois, ce chemin DFS est un chemin logique. Déterminez à l'aide de la commande `smbclient` comme ci-dessous, ou de la FAQ DFS [10], de l'interface graphique Microsoft Windows Explorer, ou encore de la commande `net use` sous Microsoft Windows, le serveur et le partage concernés :

```
sudo apt-get install smbclient # laisser les configurations par défaut

# adapter le chemin DFS et les noms d'utilisateur
# (les backslashes ci-dessous annulent le saut de ligne, il sont
#  inutiles si vous tapez tout sur la même ligne)
smbclient //intra.he-arc.ch/ORG/HE-Arc/Common \
          -U EISI\\marc.schaefer \
          -c showconnect
```

7. le format complet du fichier `/etc/fstab` est documenté par la commande `man 5 fstab`
8. interface programmatique : *Application Programming Interface*
9. convention de nommage spécifique Microsoft, voir https://fr.wikipedia.org/wiki/Universal_Naming_Convention
10. voir aussi https://faq.he-arc.ch/dfs/ et notamment https://faq.he-arc.ch/dfs/dfsUTL.html

```
# ici la sortie est: //SRV-FS201.intra.eiaj.ch/ORGHE-ARCCommon
# (l'endroit réel où se trouvent les données de ce partage, soit
# sur SRV-FS201.intra.eiaj.ch, chemin ORGHE-ArcCommon)
```

Installez le package `cifs-utils` et utilisez un montage de type `cifs` (voir la documentation avec la commande `man mount.cifs` par exemple). Comme pour tout filesystem n'intégrant pas de gestion de droits POSIX, il vous faudra spécifier quel utilisateur devra avoir les droits sous Linux (votre utilisateur HE-Arc utilisé pour la connexion au partage n'a rien à voir avec l'utilisateur UNIX local) : cherchez [11] l'option `uid` dans la manpage et vérifiez, après montage, que les bons droits sont appliqués (p.ex. avec `ls -l`). Voyez l'exemple ci-après, à adapter en fonction de ce qui précède :

```
mkdir /home/schaefer/he-arc

# les backslashes ci-dessous annulent le saut de ligne, il sont
# inutiles si vous tapez tout sur la même ligne
sudo mount -o user=marc.schaefer,uid=1000,gid=1000 -t cifs \
           //SRV-FS201.intra.eiaj.ch/ORGHE-ARCCommon \
           /home/schaefer/he-arc
Password: <votre mot de passe HE-Arc>

ls -l /home/schaefer/he-arc

# pour démonter
sudo umount /home/schaefer/he-arc
```

Si vous le désirez, vous pouvez aussi vous documenter sur l'option `credentials` de la commande `mount.cifs` : cela vous permettra de ne pas taper votre mot de passe à chaque fois : dans ce cas il faudra protéger le fichier par des permissions restrictives (`chmod 600` par exemple) : on peut même imaginer automatiser le montage, mais ce n'est pas demandé.

11. avec un slash / et quittez avec `q`

Questions – 1.7

1. à quoi sert le *swap* ? quand sera-t-il utilisé par le système ?

2. on peut configurer un système Linux pour que l'accès administrateur (compte `root`) soit possible soit en se connectant sous `root`, ou en utilisant la commande `sudo`, voire même laisser ces deux possibilités : quels sont les avantages de bloquer l'accès au compte root et de ne laisser que l'accès via `sudo` ?

3. (BONUS) remplissez le tableau ci-après :

technologie	**est obsolète ?**	**parallèle ou série ?**
IDE (PATA)		
SATA		
SAS		
USB		
Firewire / IEEE-1394		

   (BONUS) quel est le point commun de ces technologies ?

4. quelle est la différence entre un média *live* (par exemple un *Live CD*) et un média d'installation ?

5. comparez la configuration réseau NAT à la configuration bridge (pont, switch) pour l'interface réseau d'une machine virtuelle (VM, *guest*) – n'hésitez pas à placer dans les couches du modèle OSI

6. donnez au moins 3 raisons différentes d'utiliser de la virtualisation (p.ex. une du domaine de l'administration système, une du développement logiciel et une de la sécurité) :

7. il y a plusieurs types de logiciels de virtualisation, qui diffèrent par leur niveau d'isolation entre le système hôte et le système virtualisé, la taille et le type de l'OS hôte, ainsi que par ce qui doit être virtualisé et ce qui peut être accédé directement par le système virtualisé : remplissez le tableau ci-dessous ; un élément est déjà mentionné pour vous orienter dans la bonne direction.
   Indication : cherchez une comparaison de la virtualisation sur Wikipédia.

type	**description courte**	**exemples de logiciels**
conteneur/isolateur		

8. la virtualisation normale c'est quand l'hyperviseur (ici VirtualBox) présente du matériel virtuel (simulé) au guest, qui lui croit accéder au vrai matériel comme s'il n'était pas virtualisé : c'est complexe pour l'hyperviseur et c'est lent ; qu'apporte la *para*virtualisation, en une ligne ?

9. dans les manipulations effectuées, quand a-t-on rendu possible la paravirtualisation ? et quand l'a-t-on utilisée ?

10. une entreprise, qui travaille principalement la journée et du lundi au vendredi, a virtualisé l'ensemble de ses 20 serveurs sur 20 machines virtuelles réparties sur 4 serveurs physiques de virtualisation ; proposez une idée pour économiser de l'énergie la nuit et le week-end :

11. Debian gère en sécurité les packages de la distribution : que doit-on faire si on installe manuellement des logiciels non packagés par Debian ? Autre cas similaire : vous installez du logiciel sous Microsoft Windows, non compris dans l'OS de base ou le store Microsoft : Microsoft Windows Update se chargera-t-il alors des mises à jour ? sinon, que faire ?

12. essayez puis expliquez la sortie de la commande Linux `lsblk`

13. donnez au moins une différence entre les architectures i386 et amd64

14. en tant qu'utilisateur de ce cloud, comparez un cloud privé (géré par votre entreprise) avec un cloud public (ou mutualisé, géré par un tiers), avantages et inconvénients (flexibilité, coût, confidentialité, ...) – il n'y a pas forcément de réponse absolue, elle dépendra du contexte :

Checklist – 1.8

base :

j'ai installé VirtualBox et je peux le lancer	☐
j'ai téléchargé le CD d'installation réseau de buster	☐
j'ai configuré la machine virtuelle buster	☐
j'ai fait l'installation de base de buster	☐
je me suis loggué comme un utilisateur normal	☐
je sais devenir super-utilisateur avec `su -` et remarquer quand j'ai ses privilèges	☐
j'ai installé quelques packages	☐
je sais lancer une commande sous root avec `sudo`	☐

normal :

je sais éditer la ligne de commande du shell, naviguer dans l'historique, retrouver une commande précédente	☐
j'ai installé puis vérifié que les *guest additions* sont installées	☐
j'arrive à faire du copier coller entre *host* et *guest*	☐
j'arrive à partager des fichiers entre *host* et *guest*	☐
j'ai déterminé que le pilote de la carte réseau du *guest* était au départ une émulation matérielle peu performante	☐
j'ai reconfiguré le type de carte réseau virtuelle dans VirtualBox en *virtio-net* et j'ai vérifié que Linux l'utilise	☐
j'ai répondu aux questions	☐

avancé :

j'ai testé (ajout à `/etc/fstab`) le montage automatique dans mon répertoire et sous mon utilisateur du partage avec le *host*	☐
j'ai accédé à un partage HE-Arc depuis la VM buster	☐

Evaluation des objectifs par l'enseignant – 1.9

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

2. Machine virtuelle Microsoft

Objectifs – 2.1

11

— installation d'un *guest* (VM) Microsoft Windows
— améliorer l'intégration à l'hôte avec les *guest additions*
— ajout d'un disque virtuel supplémentaire
— écrire des batches de login et de démarrage
— télécommander VirtualBox depuis le *host*
— répondre aux questions

allez à votre rythme : n'oubliez pas maintenir la checklist en fin de ce labo (page 37)

VM Microsoft Windows – 2.2

— installez une VM (*guest*) Microsoft Windows 10 64 bits nommée `MS-Windows-VM-ISC22-nomprenom`
— testez son fonctionnement de base
— installez les *guest additions* et testez qu'elles fonctionnent (copier-coller, redimensionnement de la fenêtre, etc) et le partage d'un dossier[a] avec le *host*
12 — ajoutez un nouveau disque virtuel en SATA et préparez le pour pouvoir l'utiliser pour y copier des fichiers
— montez en ligne de commande `CMD.EXE` un répertoire de la HE-Arc comme *lettre de lecteur* `S:\` de manière *non persistante*
— créez un script *batch* (`CMD.EXE`) qui s'exécute à chaque login de votre utilisateur et écrit quelque chose dans un fichier (p.ex. la date et l'heure)
— avancés : faites de même, mais au démarrage de la machine

a. pas le glisser-déposer

Image d'installation et licence VM Microsoft Pour installer le *guest* MICROSOFT WINDOWS il vous faut, comme pour Debian, une image ISO, que vous allez obtenir avec les outils *Microsoft Azure Dev Tools For Teaching*[1]. Créez aussi une clé de licence dans l'interface (*View key*) : elle est **personnelle** et ne peut être utilisée que dans le cadre de votre formation. L'image ISO à télécharger s'appelle `Windows 10 Education version 21H2 - 64 bits - Français`[2].

Après la création de la machine virtuelle de bon type, architecture et version de Microsoft Windows, associez l'image ISO d'installation téléchargée puis démarrez la machine virtuelle.

Lors de l'installation, n'oubliez pas de configurer le clavier *Français (Suisse)* ; puis vous devrez entrer[3] votre clé personnelle de licence obtenue ci-dessus. Acceptez le contrat de licence Microsoft. Choisissez ensuite *Installation personnalisée*. Il ne sera pas nécessaire de créer des partitions manuellement : faites *Suivant*. Une fois l'installation terminée, le système redémarre : ne tapez aucune touche et attendez la finalisation de l'installation : vous vous logguerez avec votre compte HES-SO `prenom.nom@hes-so.ch` (prénom et nom tronqués à 8 lettres)[4] ; ensuite vous choisirez notamment quel type et niveau de données vous allez envoyer à MICROSOFT.

1. avec votre compte HES-SO, voir https://faq.he-arc.ch/ sous Logiciels > Etudiants ; voir aussi la vidéo https://video.he-arc.ch/watch_video.php?v=OXDWKDWU7S4W
2. les versions XXH1 ont un support de 18 mois, et les XXH2 de 30 mois – nous allons utiliser cette VM très peu de temps ; éviter les versions N qui sont plus limitées
3. pas encore de *guest additions*, donc le *coller* ne fonctionne pas
4. annuler au moment où une adresse e-mail est demandée est possible : un compte local sera créé

Guest additions Insérer le CD des *Additions invités* (menu *Périphériques* de la VM). Le CD est alors en général disponible sous `D:\` (par l'explorateur de fichiers Microsoft). Y lancer `VBoxWindowsAdditions.exe`. Ensuite, l'installation commence (elle peut se cacher derrière l'explorateur de fichiers). Accepter de faire confiance à ORACLE CORPORATION (pour cette installation ou pour toutes, à vous de décider) et redémarrer (soit par la routine d'installation, soit en envoyant CTRL–ALT–DEL via le menu *Entrée* de la VM).

Montage de systèmes de fichiers SMB sous Microsoft Les systèmes de fichiers exportés par un serveur SMB (partage de fichiers Microsoft ou serveur Samba sous POSIX) peuvent être montés [5], sous Microsoft, en ligne de commande manuellement, voire au login, ou ou via l'interface graphique. Ils peuvent être montés de manière persistante (automatiquement à chaque login), ou unique.

La commande Microsoft `net use ?` vous donnera l'aide de cette commande. Pour monter un répertoire de la HE-Arc, vous devez utiliser l'utilisateur `EISI\prenom.nom` et votre mot de passe HE-Arc (voir aussi page 25).

Montage d'un disque supplémentaire Comme dans une machine réelle, il est possible d'ajouter du matériel à une machine virtuelle : plutôt que d'acheter un disque-dur supplémentaire, vous allez simplement créer un disque-dur virtuel dans le contrôleur SATA virtuel de la configuration de votre VM (il sera associé à un fichier dynamique de quelques GB maximum sur le host).

Vérifiez ensuite qu'il est accessible de votre machine virtuelle redémarrée, puis préparez-le à l'utilisation (partitionnement et création du système de fichiers) via les outils d'administration Microsoft : Gestion de l'ordinateur → Stockage) : initialisez le disque (GPT), puis partitionnez-le (*Nouveau volume*) et créez un système de fichiers (*Formater* [6]).

Vous pouvez ensuite y copier un fichier pour tester : sous Microsoft, la partition, puis le système de fichiers apparaîtront sous une lettre de lecteur, par exemple `E:\` [7]

5. Microsoft, contrairement à POSIX, lie fortement le montage de systèmes de fichiers distants à un utilisateur en particulier : POSIX, par exemple sous Linux peut monter, au démarrage de la machine un système de fichiers NFS, *Network File System*, accessible via les droits d'accès POSIX à tous les utilisateurs sans montage par utilisateur.

6. nom impropre, acceptable si l'on dit *formatage haut niveau* : en fait c'est une création d'un système de fichiers, tout simplement

7. sous Linux, le disque-dur brut serait nommé par exemple `/dev/sdb`, partitionné par exemple avec `fdisk`, la partition elle serait `/dev/sdb1`, elle serait initialisée avec p.ex. avec le système de fichiers `ext4` avec la commande `mkfs.ext4 /dev/sdb1` et serait montée ensuite en tant que système de fichiers dans un répertoire de l'arborescence du systèmes de fichiers, par exemple sous `/media/nouveau_disque` avec la commande `mount`, ou, au démarrage, directement dans `/etc/fstab` – similairement à ce que nous avons fait pour le partage ; consultez la page 16 pour plus d'informations sur la terminologie et les concepts.

Ecriture de scripts shell sous Microsoft Microsoft propose plusieurs langages de script shell : les scripts *batch* classiques (`CMD.EXE`), le PowerShell ou encore, récemment, le shell POSIX `bash`.

Ici, vous allez réaliser un ou deux petits scripts batch classiques, consultez par exemple `https://windows.developpez.com/cours/ligne-commande/?page=page_24`

Il vous faudra un peu chercher comment lancer un script à la connexion d'un utilisateur spécifique, voire au démarrage de la machine : vous pouvez d'abord tester les commandes individuelles, puis votre script, à la main, avant de l'ajouter comme tâche du système. Indications :

— on peut lier[8] ou déposer un exécutable ou un script *batch* dans le dossier de démarrage (*Startup Folder*)
— il est aussi possible d'utiliser l'ordonnanceur de tâches (*Scheduled Tasks*) ou l'éditeur de politiques (*Group Policy Management Console (GPMC)*, *gpedit*)

8. raccourci de l'interface graphique

Télécommander VirtualBox – 2.3

— on peut vouloir arrêter ou démarrer une machine virtuelle, quelque soit l'OS, automatiquement
— ou on peut vouloir suspendre un *guest*, en particulier pour la migration d'un *host* à un autre ou pour la sauvegarde automatisée par des snapshots

13

objectif : écrire un script de télécommande de VirtualBox (sur le *host* Microsoft) qui arrête proprement, puis redémarre une VM (Linux ou Microsoft, c'est égal : c'est un peu plus facile sous Linux, en particulier avant le login, sinon il faut configurer)

Télécommander VirtualBox Il peut être pratique par exemple chaque jour, d'automatiquement arrêter [1] une machine virtuelle, la sauvegarder, puis la redémarrer.

Si la VM (*guest*) est arrêtée, la sauvegarde peut être faite avec une simple copie du répertoire de la VM (sur le *host*), que cela soit avec un hôte Linux ou Microsoft.

Documentez-vous sur la commande
`"C:\Program Files\Oracle\VirtualBox\VBoxManage.exe"`

et notamment les opérations permettant d'arrêter proprement une VM dont on connaît le nom (indication : opération de contrôle `acpipowerbutton`), puis de la démarrer. Faites un script batch [2] Microsoft Windows, sur votre machine réelle (*host*) bien sûr, pour exécuter cette opération automatiquement. Faites l'essai déloggué du *guest* Linux.

Les avancés pourront écrire un script qui attend que le guest est bien arrêté : sinon on peut aussi simplement mettre un délai raisonnable.

Voir https://www.virtualbox.org/manual/ch08.html ainsi que l'aide de la commande concernée.

1. il y a des techniques qui permettent de sauvegarder un état fiable sans l'arrêter, mais en la suspendant (synchronisation puis snapshot) très peu de temps, ce que nous n'allons pas faire ici
2. lancez `CMD.EXE`

Questions – 2.4

1. quels seront les difficultés probables si l'on veut que le disque système (le premier disque) soit en VirtIO, à l'installation d'un Microsoft Windows virtualisé ?

2. cochez dans le tableau ci-dessous *vos* responsabilités en fonction du type de cloud choisi ; la colonne *vous gérez* (le cas où par exemple tout est dans votre infrastructure gérée par vous : cloud privé, serveur(s) chez vous) est déjà remplie :

responsabilités	**vous gérez**	**IaaS**	**PaaS**	**SaaS**
vos données	X			
applications	X			
environnement d'exécution (langages, runtime, middleware)	X			
OS	X			
virtualisation	X			
serveurs	X			
stockage	X			
réseau	X			

(*aaS : as a Service : Infrastructure, Platform & Software)

3. indiquez de quel type est chacun des exemples suivants :

cas	**vous gérez**	**IaaS**	**PaaS**	**SaaS**
votre PC pour accéder à de l'e-banking				
un serveur physique qui vous appartient dans un datacenter				
une machine virtuelle que vous gérez complètement chez Amazon Web Services (AWS), Exoscale, chez Microsoft Azure, ou chez Google Compute Engine (GCE)				
Heroku, OpenShift, AWS Elastic Beanstalk, Apache Stratos …				
Google Apps, Dropbox, Framadate, github, le webmail de l'École …				

4. qu'est-ce que la *Microsoft Windows Sandbox* ?

5. pour un *data center*, que signifie un indicateur d'efficacité énergétique de 1.2 ?

Checklist – 2.5

base :

j'ai installé une machine virtuelle (*guest*) Microsoft Windows	☐
j'ai testé le fonctionnement de base de cette machine virtuelle Microsoft Windows	☐

normal :

j'ai créé un deuxième disque virtuel accessible par la VM Microsoft Windows, j'ai créé son système de fichiers dans la VM et y ai déposé des fichiers	☐
j'ai pu télécommander VirtualBox depuis la machine réelle (*host*)	☐
j'ai répondu aux questions	☐

avancé :

j'ai créé un batch Microsoft dans la VM qui s'exécute au login de l'utilisateur ou au démarrage de la machine et qui effectue une action dans un fichier	☐
j'ai installé et testé les *guest additions* dans la VM Microsoft Windows	☐
j'arrive à partager des fichiers entre *host* et *guest* Microsoft via un dossier partagé	☐
j'ai monté un système de fichiers distant dans la VM Microsoft Windows	☐
j'ai effectué des éléments avancés	☐

Evaluation des objectifs par l'enseignant – 2.6

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

3. Emulation de réseaux couche 2

Objectifs – 3.1

14

- installer l'émulateur Netkit dans la VM buster avec le script fourni et comprendre ses concepts
- installer le labo `bridging` avec la commande fournie et expérimenter en pratique les concepts couche 2 vus au cours :
  - adresse MAC
  - capture réseau avec `tcpdump` et `Wireshark`
  - notion de domaine de diffusion et commutation
  - interaction couche 3 / couche 2 avec ARP
  - avancés : VLAN, STP

allez à votre rythme : n'oubliez pas maintenir la checklist en fin de ce labo (page 54)

Commutation ou "routage" couche 2 Formellement, le routage est une fonction de la couche 3 qui ne sera pas exercé dans ce laboratoire. Toutefois, on peut par abus de langage parler également de "routage en couche 2", même si l'on préférera parler de *commutation* (anglais : *switching* ou *bridging*, un *switch* étant un commutateur, et un *bridge* un switch à seulement deux ports, ou pont).

Ici on verra les notions de :
- mode inondation (par défaut)
- remplissage des tables (MAC source, port) des switches
- isolation en fonction de l'adresse MAC destination si elle figure dans la table
- diffusion (broadcast)
- différence entre switch et hub
- mode *promiscuous* de `tcpdump`

Un peu de couche 3 aussi Après un début en couche 2 seulement, nous allons également utiliser quelques concepts de la couche 3 et d'interaction avec la couche 2 (le protocole ARP, *Address Resolution Protocol*). Nous reviendrons sur la couche 3 dans la théorie prochaine Réseaux, et lors d'autres laboratoires.

Gardez une trace ! Faites des copies d'écran documentant les réponses aux questions.

Suspendre le laboratoire N'hésitez pas à suspendre la machine virtuelle Linux pour reprendre plus tard !

Netkit – 3.2

— émulateur en logiciel libre de réseaux et services quelconques
— destiné à la simulation d'apprentissage et de prototypage
— basé sur de la virtualisation relativement légère de type UML
— chaque machine émulée (PC, switch, routeur, serveur, etc) est un processus Linux contenant une nouvelle instance de kernel, un système de fichiers standard partagé en *copy-on-write* avec une installation très basique de Linux.
— les machines sont reliées entre elles par des switches virtuels
— il est possible d'interfacer du vrai matériel (téléphone IP, switch, ...) si désiré
— en pratique vous allez lancer des machines virtuelles UML dans votre machine virtuelle Debian *buster*.

Netkit Œuvre d'une université italienne [1], Netkit est intéressant également pour les nombreux exemples de laboratoires et de préconfigurations qu'il propose, pour le moment uniquement en anglais. Toutefois, nous allons dans le cadre des laboratoires utiliser un installateur et des laboratoires spécifiques et non *pas* ceux proposés sur le site de Netkit. Il existe une surcouche graphique à Netkit, permettant la définition de réseaux : https://www.marionnet.org/, nous n'allons pas non plus l'utiliser. Contrairement à d'autres logiciels comme CISCO PACKET TRACER, qui sont plutôt des simulateurs réseau, Netkit tourne du vrai logiciel dans des machines virtuelles de type UML [2] et est donc très flexible. Il peut aussi s'intégrer à de vrais réseaux.

Copy-on-write Il s'agit d'une optimisation où seules les modifications au disque virtuel sont stockées, séparément pour chaque machine virtuelle : un seul disque virtuel partagé existe quelque soit le nombre de VMs.

Cette technique est également utilisée dans les systèmes d'exploitation pour partager de manière efficace des pages mémoires.

1. qui est récemment passée à un nouvel outil basé Docker, Kathará, que je n'ai pas encore eu l'occasion de tester
2. User Mode Linux : comme des conteneurs, avec toutefois un kernel par VM : le kernel a été recompilé spécialement pour ne comprendre que les couches hautes (systèmes de fichiers, pile IP, scheduler, etc)

Installation de Netkit – 3.3

— téléchargez le script d'installation du GIT (`00_Org/deroule.md`, labo 3)
— exécutez-le avec `bash netkit-install` (pas sous root ! sous votre propre utilisateur) à l'Ecole ou avec le VPN activé
— si tout va bien, le script devrait dire OK
— effectuez le mini-test d'installation ci-dessous et répondez à ses questions directement sur la feuille.

16

Mini-test de Netkit Nous allons créer, lancer, arrêter et supprimer une machine virtuelle UML.

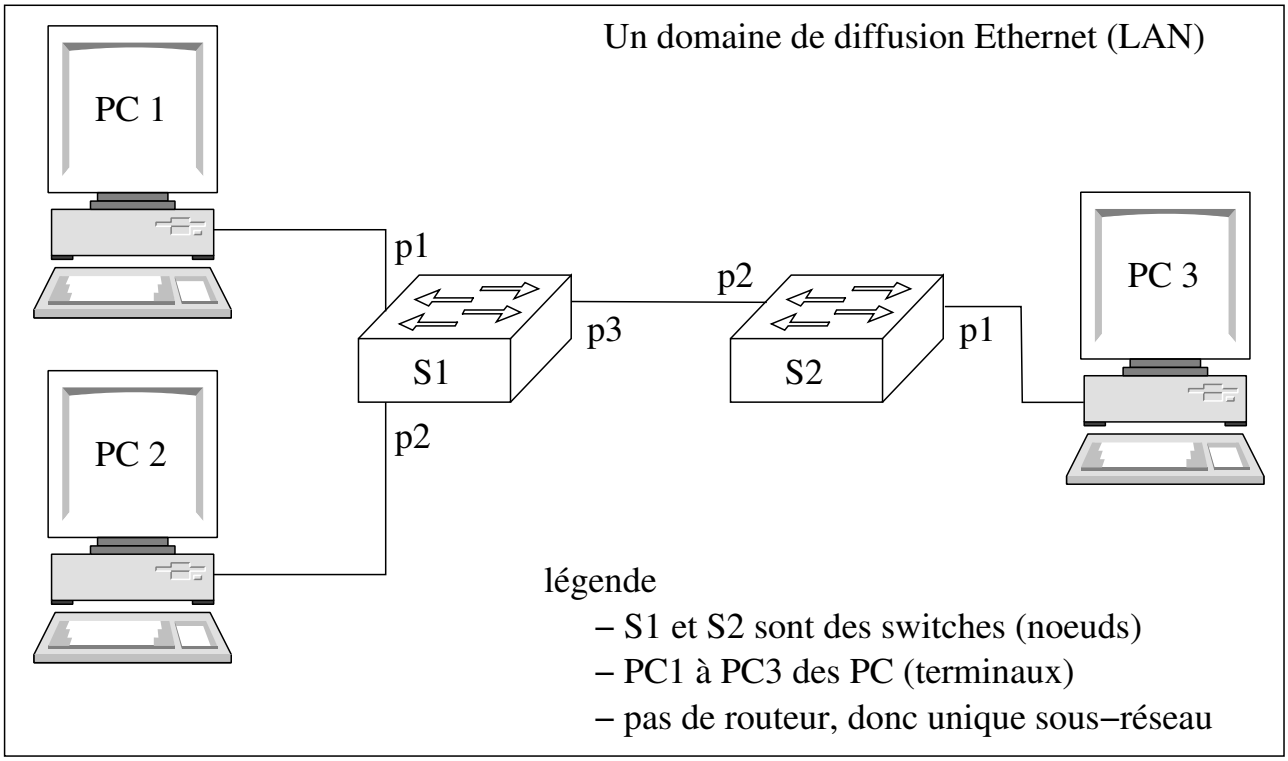
1. lancez un nouveau terminal
2. déplacez-vous dans la racine de NETKIT avec `cd ~/NETKIT`
3. créez l'arborescence `LABOS/labo-3` (indication : `mkdir -p`)
4. déplacez-vous dans `LABOS/labo-3`
5. lancez les instructions suivantes et observez :

```
# préparation de l'environnement pour utiliser netkit
# ne pas oublier avant d'utiliser netkit dans un bash (shell)
# (à chaque nouveau terminal ! optionnel: ajouter dans ~/.bashrc)
source ~/NETKIT/SOURCE_ME

# lancement d'une machine virtuelle UML pc1
vstart pc1

# avancés: pourquoi une différence si notable entre
# la sortie de ces deux commandes?
ls -lah *.disk
du -hs *.disk

# fermez la fenêtre de la machine virtuelle pc1
# puis effacez le(s) fichier(s) *.disk
rm *.disk
```

Un LAN (réseau local) ? Qu'est-ce que c'est ? – 3.4

— en couche 1 : des signaux, des **câbles**, des connecteurs, des tableaux de brassage (répartiteurs), voir évt. des concentrateurs (hubs) ou répéteurs électriques
— en couche 2 : des **switches**, et pour chaque interface réseau de chaque équipement (PC, imprimante, routeur, évt. switch *manageable*) une **adresse MAC** définie par le fabricant
17 — en couche 3 : des **adresses IP** à l'intérieur d'un sous-réseau bien défini, pour chaque équipement (PC, imprimante, routeur, ...) ; un **routeur** (passerelle par défaut) pour atteindre l'extérieur de ce LAN ou d'autres sous-réseaux de ce LAN
— en couche 4 : un ou plusieurs protocoles pour acheminer les données d'une application à l'autre : TCP, UDP
— en couche 7 : les applications (p.ex. mail, partage de fichiers, web, ...) : SMTP, NFS/SMB, HTTP, ...

Le petit LAN de ce labo Netkit (pas de routeur, une seule couche 2, un seul domaine de diffusion ; voir livret A5 chapitre 0 pour les conventions de schéma Cisco)

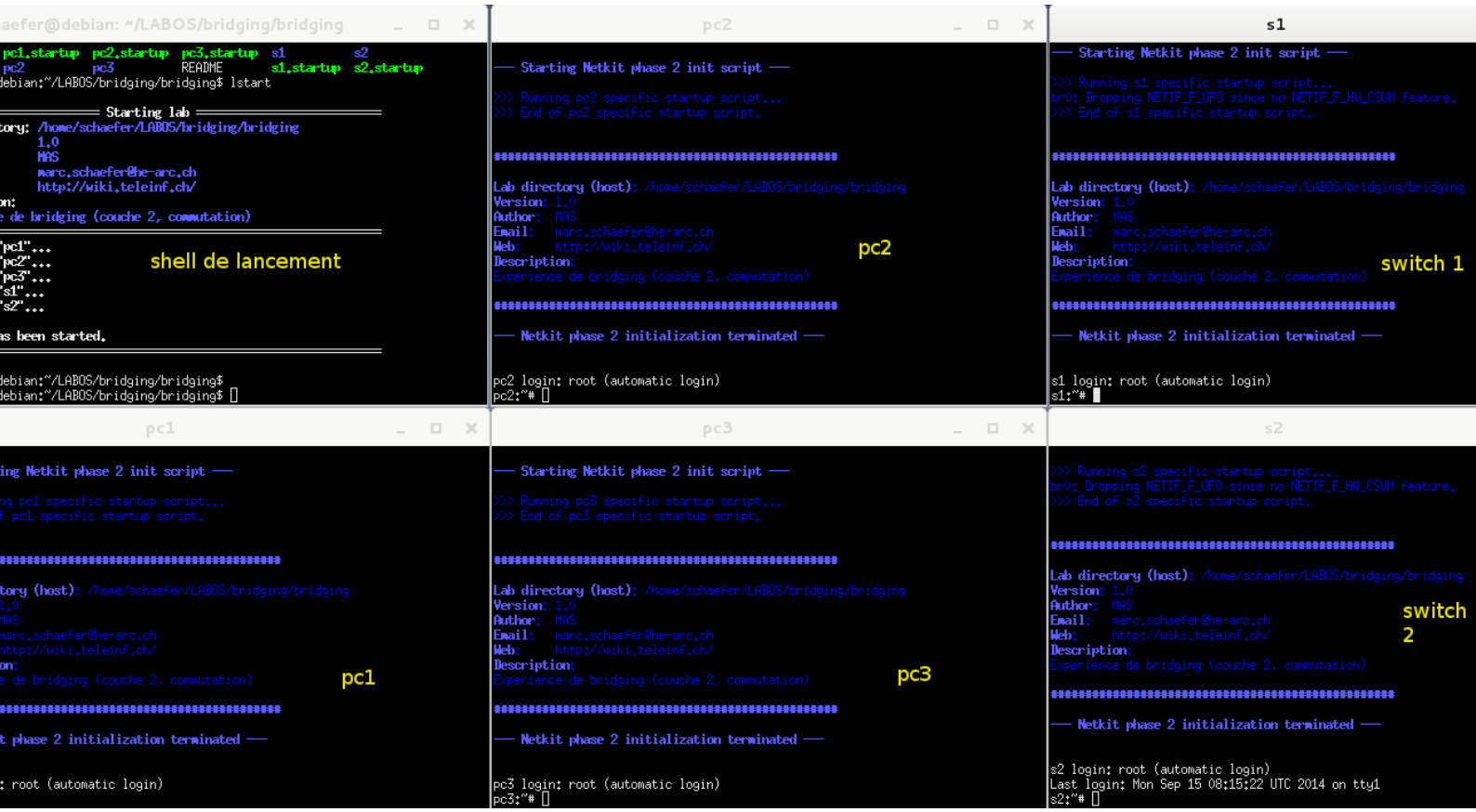


1er labo Netkit : bridging – 3.5

— n'oubliez pas de faire `source ~/NETKIT/SOURCE_ME` à chaque lancement d'un terminal Linux
— téléchargez ce labo avec `netkit-lab bridging`
— entrez dans le répertoire avec `cd briding`
— lancez le labo avec : `lstart -p0`
18 — arrêt et nettoyage du labo

1. CTRL–C dans la fenêtre de lancement (si nécessaire)
2. `lcrash` pour terminer les instances
3. `lclean` pour supprimer les systèmes de fichiers des instances
4. en cas de nécessité : `rm -rf ~/.netkit`

NB : un des avantages d'un tel émulateur est que (quasi) rien ne crée de trafic par défaut : en cas de difficulté, arrêtez tout, nettoyez et recommencez.



Commutation : tables des switches – 3.6

Démarrez le laboratoire (cf slide 18), puis effectuez les commandes et observations suivantes :

1. indiquez chacune des adresses MAC de chaque interface de chaque terminal (les PCs) sur le schéma du LAN en page 41
2. sans échange de trafic entre les PCs, observez-vous, à l'aide des commandes suivantes, des informations concernant l'emplacement des PCs 1 à 3, sur respectivement le switch 1 et le switch 2 ?
   — `s1# brctl showmacs br0`
   — `s2# brctl showmacs br0`
3. en conséquence, quels sont les modes actuels respectifs des deux switches, parmi *inondation* et *isolation* ? (entourez)

19

Adresses MAC Comme vu au cours, chaque interface a une adresse MAC. Ici, notamment les PCs ont chacun une adresse MAC (virtuelle). Consultez `ifconfig` ou `ip link show` pour obtenir l'adresse MAC de l'unique interface de chacun des PCs.

Commutation, switching ou bridging (parfois appelé par abus "routage couche 2") Originellement les tables des switches sont vides : les switches sont en mode *inondation*. Chaque switch apprend indépendamment en fonction des adresses *source* de chaque trame reçue sur chaque port et stocke l'information dans sa table propre sous forme d'un couple `(MAC, port)`. Lorsqu'un switch reçoit une trame à *destination* d'une adresse qui figure dans la table, elle peut être envoyée uniquement au port concerné (mode *isolation*) ! Sinon, la trame doit être envoyée sur toutes les interfaces du switch (sauf celle dont elle provient). Une entrée de table est purgée après un certain temps si aucun trafic avec l'adresse source correspondante n'est enregistré.

Donc, après quelques échanges, le trafic n'est plus envoyé sur les ports qui ne sont pas concernés, ce qui est plus efficace, voire plus sûr [1].

Explications sur la syntaxe des commandes La syntaxe `s1# brctl showmacs br0` signifie déplacer la souris dans la fenêtre du switch 1 (s1), activer le focus si nécessaire, et taper au prompt root (`#`) la commande `brctl showmacs br0`. Dans le cas présent, il s'agit de consulter la fameuse table de commutation (qui indique sur un port d'un switch donné quelles stations (PC) ont été vues) et qui sert à l'isolation (n'envoyer une trame que sur le bon port où se trouve sa destination).

1. des attaques en couche 2 permettent de rediriger le trafic malgré cela : ARP spoofing p.ex.

Commutation, switching, bridging – 3.7

Consultez les informations ci-dessous, puis effectuez les commandes indiquées **Manipulations ...** (page 46) et répondez en même temps aux questions ; faites des captures d'écran au besoin.

Thèmes :

- inondation (*flooding*) et isolation par les switches
- processus d'apprentissage par les switches (table)
- notion de *broadcast* (diffusion) au sein d'un domaine de diffusion couche 2

20

Domaine de diffusion Ethernet Un domaine de diffusion Ethernet est la portée couche 2 des *broadcasts* Ethernet. Ce domaine s'arrête au premier routeur (car couche 3). Ne pas confondre avec le *domaine de collision* qui est de taille nulle si les trois conditions cumulatives suivantes sont remplies : topologie étoile (ou étoile-arbre), switches au centre de chaque étoile, liaisons full-duplex entre stations et switches – ce qui est le cas ici.

Flooding (inondation) vs broadcast (diffusion) Il y a deux cas possibles où les switches vont envoyer une trame à tous leurs ports, sauf celui de réception :

1. l'inondation (flooding) est toujours liée à l'envoi à une adresse MAC destination (type *unicast*) particulière : dès lors que cette adresse MAC a émis une trame, les switches qui ont pu la recevoir apprennent le port où elle a été reçue et passent, pour cette adresse MAC, en mode isolation (envoi uniquement sur le bon port)
2. cela n'a rien à voir avec la diffusion (*broadcast*) qui s'exécute toujours pour une adresse destination MAC de type *broadcast* (tous les bits à un), voire pour les adresses de type *multicast* (envoi à plusieurs équipements d'un groupe).

Commande eth-send Cette commande – spécifique [1] à notre laboratoire – prend deux arguments : l'adresse MAC destination et l'interface réseau à utiliser. Elle permet d'envoyer une trame en couche

1. très avancés : code source C sous `src/eth-send.c`

2 à la destination. Dans la configuration de notre laboratoire, aucune réponse n'est générée.

Capture simple de trafic avec tcpdump Avec `tcpdump -i eth0 -n -e -vvvvv` on peut voir les trames reçues et envoyées sur `eth0`. Par défaut, une carte réseau ne va remonter au PC que les trames qui portent la bonne adresse destination (ou le *broadcast* `ff:ff:ff:ff:ff:ff`). `tcpdump` place l'interface en mode écoute (*promiscuous*) et permet de voir tout le trafic reçu [2].

On quitte la commande avec CTRL–C.

PS : le nom `tcpdump` est historique : la commande permet de voir la couche 3 et une partie des couches supérieures, mais également, avec l'option `-e`, la couche 2. N'hésitez pas à comparer la sortie hexadécimale `-e -XX` avec les octets de l'exercice de décodage couche 2 du cours Réseaux !

2. toutefois, les switches peuvent isoler !

Manipulations, observations, réflexion et analyse Le but de ces manipulations est de voir les modes des switches et leur processus d'apprentissage. Faites des captures d'écran quand c'est utile. Répondez directement sur la feuille.

NB : n'attendez pas trop longtemps entre les différents points, sinon il vous faudra recommencer au début (pourquoi ?).

1. lancez les commandes suivantes : (NB : option `-e` de tcpdump pour voir la couche 2 Ethernet)
   ```
   pc2# tcpdump -i eth0 -n -s 0 -vvvvv -e
   pc3# tcpdump -i eth0 -n -s 0 -vvvvv -e
   ```
   ainsi qu'une trame de `pc1` à l'adresse MAC de `pc2` :
   ```
   pc1# eth-send 00:00:00:02:00:00 eth0
   ```
   puis consultez les tables des switches :
   ```
   s1# brctl showmacs br0
   s2# brctl showmacs br0
   ```

   expliquez ce qui s'est passé et notamment pourquoi toutes les machines et tous les switches ont vu cette trame ; utilisez la notion d'inondation/isolation vue en cours :

   indiquez l'état actuel des tables des switches : (toutes les lignes ne sont pas forcément remplies sur chaque switch – uniquement les lignes concernant les PCs)

   s1

MAC	port

   s2

MAC	port

2. terminez le `tcpdump` sur `pc2` et lancez-en un sur `pc1`, laissez le `tcpdump` actif sur `pc3`, puis envoyez une trame de `pc2` vers l'adresse MAC de `pc1` :
   observez-vous cette trame sur `pc3` ? pourquoi ?

   le switch `s1` a-t-il appris quelque chose de nouveau ? pourquoi ?

   le switch `s2` a-t-il appris quelque chose de nouveau ? pourquoi ?

3. en laissant les `tcpdump` actifs sur `pc1` et `pc3`, envoyez une trame de `pc2` vers l'adresse MAC de `pc3` :
   observez-vous cette trame sur `pc1` ? pourquoi ?

   un switch sait-il quelque chose sur `pc3` ? pourquoi ?

4. envoyez une trame de `pc2` à l'adresse MAC `ff:ff:ff:ff:ff:ff`, pourquoi tous les équipements la reçoivent ? (indication : pas lié à la question précédente)

5. les trames envoyées par `eth-send` sont elles Ethernet (II) ou 802.3 ? quel protocole se trouve vraisemblablement dans les données de couche supérieure (*payload*) ? très (BONUS) avancés : est-ce une trame réelle de ce protocole ?

Lien entre couche 2 et couche 3 – 3.8

— chaque terminal source, puis routeur va communiquer avec un routeur ou le terminal de destination au-travers d'une technologie couche 2 spécifique
— pour acheminer en couche 2 une trame, il faut une adresse MAC : comment la conversion d'adresse couche 3 vers couche 2 se produit-elle ?
— différence entre switch et hub

21

voir les manipulations et observations ci-dessous

Commande ping La commande `ping` est une commande couche 3 qui permet de détecter si une machine est présente (par envoi d'un message couche 3 de type ICMP *Echo Request* et la *réception d'une réponse Echo Reply*). Nous utilisons ici les noms des machines (une configuration d'adresses IP (couche 3) a été effectuée mais les détails ne vous sont pas nécessaires ici – voir `/etc/hosts` si intéressé).

Manipulation et observations

1. capturez le trafic sur `pc1` et `pc3`; faites `pc2# ping -c 1 pc3`; sur `pc3` vous devriez voir au moins 4 trames, et 1 seule sur `pc1`
2. pour le protocole ARP, à quoi sert cette première trame ?

3. pourquoi cette première trame est visible partout, et pourquoi les 3 suivantes ne le sont pas ? (indication : quelle est l'adresse destination de la 1ère trame et que sait `s1` après celle-ci ?)

4. consultez le cache ARP (conversion d'adresses couche 3 vers couche 2) avec `arp -a` et indiquez ce que vous observez

5. (BONUS) très avancés : la réponse au ping de `pc3` vient tout de suite, mais quelques secondes après viennent une question et une réponse ARP, de quoi s'agit-il ? (indication : *gratuitous ARP*)

6. transformons s1 en hub : `s1# brctl setageing br0 0`, envoyez un nouveau ping et expliquez le changement (indication : que signifie *ageing* en anglais ?) (NB : vous ne verrez la première trame d'avant que si suffisamment de temps a passé, ce n'est pas le changement à observer ici !)

7. (BONUS) sur pc2 : refaites le ping, consultez le cache ARP, puis expérimentez avec la commande `arp -d pc3` : que fait-elle ?

Visualisation de captures de VLANs – 3.9

But :

1. démarrer une capture tcpdump dans un fichier (consultez sa *manpage* ou effectuez la procédure détaillée dans les notes ci-dessous)
2. générer du trafic (voir les notes ci-dessous pour configurer un VLAN et capturer le trafic de l'interface multiplexée – *trunk*)
3. transférer le fichier capture sur votre hôte Microsoft Windows (voir page suivante)
4. visualiser cette capture dans Wireshark sur votre hôte Microsoft Windows

22

Générer et capturer du trafic VLAN Le trafic VLAN peut être généré en effectuant les manipulations ci-dessous, dans le but de visualiser les VLANs dans Wireshark :

1. créez un VLAN sur `pc1` et `pc2` (les switches commuteront sans spécialités), dans leurs interfaces `eth0`
   — `pc1# vconfig add eth0 42`
   — `pc2# vconfig add eth0 42`
2. associez-y des adresses IP et des sous-réseaux (couche 3)
   — `pc1# ip addr add 198.51.100.1/24 dev eth0.42`
   — `pc2# ip addr add 198.51.100.2/24 dev eth0.42`
3. activez les interfaces réseau VLAN
   — `pc1# ip link set up dev eth0.42`
   — `pc2# ip link set up dev eth0.42`
4. lancez la capture `tcpdump` sur l'interface `eth0` de `pc2` – voir page 45
5. envoyez un datagramme couche 3, encapsulé dans un VLAN en couche 2, depuis `pc1`
   `pc1# ping -c 1 198.51.100.2`
   et remarquez la présence de tags 802.1q dans les trames captures sur `pc2`
6. terminez la capture `tcpdump` sur `pc2` avec CTRL–c puis relancez-la cette fois en ajoutant l'option `-w fichier.pcap` pour stocker dans un fichier plutôt que d'afficher à l'écran
7. transférez ce fichier sur votre machine Microsoft Windows et visualisez-le dans Wireshark (voir page suivante)

Attention : cette façon de faire des VLANs n'est pas celle qu'on utilise en pratique en entreprise : en effet, en général, les VLANs sont gérés au niveau des switches (des trunks interconnectent les switches du backbone) et les ports des switches sont configurés en mode access pour y connecter des PC ou serveurs.

Ici, c'est complètement autre chose : on configure les VLANs à l'intérieur des PC virtuels, et l'interface de sortie de chaque PC multiplexe plusieurs VLANs (*trunk*). Les switches ne font que de relayer l'ensemble du trafic sans tenir compte des VLANs.

NB : si vous capturez sur eth0.42, vous ne verrez pas les entêtes de VLAN, car ici l'interface est en mode *access*

Transfert du fichier sur le host pour visualisation Wireshark Il suffit ensuite de copier le fichier de capture sur votre hôte Microsoft Windows via votre partage permanent configuré au premier labo.

Astuce : comme vous avez monté ce partage dans votre répertoire utilisateur (*homedir*), alors vous pourriez passer les arguments `-w /hosthome/PARTAGE/fichier.pcap` pour directement écriture le fichier dans le host. En effet, dans netkit, `/hosthome` représente votre homedir et `/hostlab` le répertoire dans lequel vous avez lancé le labo.

Questions – 3.10

1. cochez les réponses justes :
   - ☐ un *broadcast*, ou diffusion en français, est lorsque l'émetteur d'une trame décide de l'envoyer à toutes les stations de la couche 2, grâce à l'adresse destination MAC `ff:ff:ff:ff:ff:ff`
   - ☐ une inondation, ou *flooding* est lorsqu'un switch relaie une trame à tous ses ports [1] car il n'a pas d'information sur le destinataire dans ses tables et ne peut donc pas isoler (filtrer) vers un seul port
   - ☐ un switch en mode isolation ne relaiera pas les trames *broadcast*
   - ☐ un switch apprend (complète ses tables) avec l'adresse destination d'une trame et isole avec l'adresse source
   - ☐ les entrées des tables s'effacent après un certain temps
2. a-t-on le droit d'utiliser les adresses MAC que l'on a configuré dans ce laboratoire en-dehors d'une simulation ou d'un test ? (indication : quel est leur O.U.I. ?)

   et en pratique, quand est-ce que cela pourrait poser problème ?
3. pourquoi a-t-on choisi ces plages d'adresses IP (couche 3) bizarres ? justifiez ! (indication : WHOIS)
4. lorsque des VLANs sont mis en exploitation, on distingue deux modes : *trunk* ou *access* ; l'interface eth0 de pc1 et pc2 est dans lequel de ces modes *dans notre cas* ? (indication : présente-t-elle des trames tagguées 802.1q ou non ?)
5. (BONUS) avancés : si le *copy-on-write* est une manière efficace de limiter l'espace disque utilisé par plusieurs machines virtuelles au départ basées sur la même image, en quoi la *déduplication* et le *copy-on-write* sont une optimisation avancée de la mémoire vive virtuelle (RAM) ? (indications : https://en.wikipedia.org/wiki/Kernel_same-page_merging et http://pfzuo.github.io/2016/03/16/Memory-Deduplication/ ainsi que http://vroomblog.com/vmware-mecanisme-memoire-esxi/)
6. (BONUS) avancés : l'*exponential backoff* permet d'ajouter des délais croissants en cas de problème (p.ex. collisions en raison de deux émissions simultanées sur un média partagé), choisis de manière aléatoire, de manière à augmenter la probabilité qu'un des émetteurs puisse émettre.

1. sauf celui d'où elle est venue

Il y a toutefois d'autres raisons de mettre en place une variante de cette algorithme, par exemple pour éviter la surcharge d'un service réseau ou d'une machine.

On va étudier ici une implémentation en `bash` d'un système qui redémarre un client d'un service réseau. Objectif : analyser, voire tester[2], le code `runner.sh` qui se trouve dans le déroulé (Gitlab) sous laboratoire 3 et répondre aux questions ci-dessous :

(a) si le service a fonctionné pendant plus de `$min_time` secondes, que se passe-t-il :

(b) sinon, que se passe-t-il la 1ère fois (indication : pourquoi parle-t-on d'*exponential* backoff ?)

(c) et la 2e fois :

(d) quel est le temps backoff maximum ?

(e) pourquoi y-a-t-il un délai aléatoire ?

7. (BONUS) très avancés : si cela vous intéresse, un laboratoire Spanning Tree Protocol (STP, arbre recouvrant) existe (installation avec : `netkit-lab STP`) – consultez son README !

2. pour terminer le `cat` faire CTRL–d

Checklist – 3.11

base :

j'ai pu installer Netkit sur ma VM	☐
j'ai pu configurer ce laboratoire Netkit	☐
j'ai pu lancer l'émulation Netkit	☐
je peux énoncer les principes de fonctionnement de Netkit	☐
j'ai pu consulter les fameuses tables de commutation qui associent l'adresse MAC source d'une machine au port du switch sur laquelle elle a été reçue	☐
j'ai pu observer l'inondation (*flooding*) effectuée par les switches lorsque la table ne contient pas d'information sur la destination ainsi que l'isolation une fois que les switches ont pu collecter suffisamment de trafic source	☐

normal :

j'ai pu observer la diffusion Ethernet (trame avec adresse destination *broadcast*) et je peux énoncer la différence avec l'inondation	☐
j'ai pu comprendre par l'expérience la différence entre hub et switch	☐
j'ai pu capturer des trames Ethernet avec `tcpdump`, les afficher à l'écran décodées et les sauvegarder dans un fichier au format pcap	☐
j'ai copié les fichiers pcap sur ma machine Microsoft Windows et je les ai visualisés avec `Wireshark`	☐
j'ai répondu aux questions non avancées	☐

avancé :

j'ai pu observer l'interaction entre la couche 3 et la couche 2 et notamment le protocole ARP	☐
j'ai pu configurer des VLANs	☐
j'ai pu capturer au moins une trame tagguée VLAN 802.1q	☐
j'ai répondu aux questions avancées	☐
j'ai pu constater le fonctionnement de base du spanning tree	☐

Evaluation des objectifs par l'enseignant – 3.12

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

4. Gestion des fichiers et processus

Objectifs – 4.1

23

— être à l'aise avec la ligne de commande (édition, historique) (voir page 19)
— consulter les pages de manuel lorsque nécessaire
— maîtriser le concept de jokers (wildcards) du shell
— décider quand mettre des guillemets ou apostrophes dans les arguments
— utiliser les commandes de base de gestion de fichiers, notamment : `echo`, `cd`, `pwd`, `ls`, `touch`, `cp`, `mv`, `rm`, `rmdir`, `diff`, `patch`, `sort`, `uniq`, `tar`, `chmod`, `tail`, `head`
— maîtriser le fonctionnement de base de `vi` (`vim`)
— créer et gérer les processus
— pouvoir résoudre un problème efficacement avec ces outils

allez à votre rythme : n'oubliez pas maintenir la checklist en fin de ce labo (page 68)

Pages de manuel Le manuel UNIX est organisé en sections : p.ex. la section 1 pour les commandes à taper par les utilisateurs, la section 8 pour les commandes d'administration, la section 5 pour documenter les configurations et fichiers, etc : voir la commande `man man`.

Aide du shell bash Les commandes internes de bash sont documentées avec l'outil `help`.

Conserver des traces de vos actions Documentez ce que vous faites dans votre fichier de notes, ou directement sur ce document. Vous pouvez aussi faire des copies d'écran, ou, encore plus automatique, sauvegarder l'entier d'une session shell avec la commande `script -a mon-journal` – mon-journal est le fichier de sortie. Cette commande lance un sous-shell, et il vous faudra le terminer avec `exit` à la fin. Si vous n'avez pas utilisé cette commande vous pouvez aussi sauvegarder une session ainsi : `history > ma-session` (ne stocke que les commandes tapées, la taille est limitée par la configuration du shell bash).

Jokers ou wildcards du shell et expansions par le shell Les caractères `*`, `?` et les ensembles `[]` et générateurs `{}` sont remplacés par le shell :

`fichier-[0-9]`	sera remplacé par la liste des fichiers `fichier-1` à `fichier-9`, pour ceux qui existent
`fichier-{0,1}`	*générera* la liste `fichier-0` et `fichier-1`, qu'ils existent ou non
`fichier-*`	sera remplacé par la liste tous les fichiers qui commencent par `fichier-`
`/usr/*/*.h`	sera remplacé par la liste des fichiers dont le nom se termine par `.h` et qui se trouvent dans n'importe quel sous-répertoire directement sous `/usr`
`*ic?ier*`	les noms de fichiers contenant `ic?ier` avec `?` représentant n'importe quel caractère remplaceront l'expression
`.*`	les fichiers *cachés* (cf `ls -a`) remplaceront l'expression – c'est le seul moyen de les sélectionner

NB : ci-dessus, le terme fichier peut être remplacé tout objet du système de fichiers (liens, répertoire, etc), et un chemin peut être préfixé (sinon la recherche sera faite dans le répertoire courant). S'il n'y a pas de correspondance, l'expression reste.

Attention : les wildcards ou jokers du shell sont différents et moins puissants que les *expressions régulières*[a] qui sont par exemple utilisées avec les commandes `grep`, `sed`, ou dans les langages de programmation pour le *pattern matching*[b] – ce n'est pas le sujet de ce laboratoire.

a. aussi appelées *expressions rationnelles*, voir https://fr.wikipedia.org/wiki/Expression_r%C3%A9guli%C3%A8re

b. https://fr.wikipedia.org/wiki/Filtrage_par_motif

Eviter l'expansion par le shell Le shell remplace[1] aussi d'autres séquences débutées par des caractères spéciaux (par exemple `$` pour les expansions de paramètres). Pour éviter ce comportement, il faut échapper ces caractères, ou les isoler dans des chaînes à apostrophes (les chaînes à guillemets sont plus permissives).

1. notion de *shell expansion*, voir https://www.gnu.org/software/bash/manual/html_node/Shell-Expansions.html

Problème concret : imagettes – 4.2

— un développeur a conçu une petite application bash (URL dans le Wiki) qui sert à créer des pages Web avec des imagettes cliquables à partir d'images originales
— on vous demande de : créer un répertoire de travail, y télécharger l'archive et l'extraire, lire le README, lancer l'application depuis le répertoire d'images fournies, chercher des occurences d'un texte et le remplacer, recréer l'archive,
24 nettoyer (voir ci-dessous pour les étapes numérotées)
— vous utiliserez **exclusivement** la ligne de commande (à part `ALT-F2 xterm` ou `mate-terminal` et `firefox`)
— vous utiliserez de préférence les pages de manuel UNIX (`man`), voire Internet

Votre travail en détail (consultez aussi la page qui suit pour la fin des étapes et pour des indications sur divers éléments)

1. créer un répertoire de travail et y entrer
2. télécharger l'application dans ce répertoire, indications :
   — ne téléchargez pas avec le navigateur `firefox` mais en indiquant l'URL comme paramètre de la commande `wget` (avec option `-O archive.tar.gz` pour choisir la destination – l'option est sensible à la casse)
   — faites attention aux métacaractères du shell bash présents dans cette URL [1] : que faire ?
   — prenez une copie d'écran de cette commande et de son résultat
3. extraire et effacer l'archive
4. consulter son README et installer les packages manquants, s'il y en a
5. télécharger les exemples fournis de fichiers d'images dans ce même répertoire, se déplacer dans le sous-répertoire `images` et extraire
6. lancer correctement l'application, qui générera les imagettes dans un sous-répertoire `thumbnails`
7. tester le résultat avec p.ex. `firefox index.html`
8. déterminer dans les fichiers générés du répertoire courant et ses sous-répertoires où se trouve le texte `HE-Arc` (indication : `grep` récursif)

1. et qui, si ils sont laissés tels quels seront interprétés par le shell !

9. changer dans tous les fichiers générés le texte `HE-Arc` par `HE-Arc Ingénierie` indications :
   — ajouter une option à la commande précédente pour n'afficher que la *liste* des fichiers concernés, et non pas le contenu trouvé
   — apprenez à spécifier une *substitution* (chercher/remplacer) de l'entrée standard vers la sortie standard par exemple avec
   `echo HE-Arc | sed 's/MOTIF/REMPLACEMENT/'`
   — utilisez la commande `sed` et son option `--in-place` ou `-i` pour chercher et remplacer dans un ou plusieurs fichiers indiqués comme arguments : ces noms de fichiers doivent être générés par la commande du point précédent
   — le tout doit être fait en une seule ligne de commande, sans utiliser de script ni de pipe `|`, mais en utilisant la substitution de commande `$( )` [2]
10. recréer l'archive de l'application avec `tar` (consultez la page suivante pour un piège possible lors de la création d'archives) et contrôler son contenu
11. nettoyer / effacer le(s) répertoires de travail

Ergonomie du shell Rappelez-vous que vous pouvez éditer la ligne de commande avec les touches curseur et que la touche TAB vous permet de faire agir la fonction de complétion automatique du shell.

Téléchargement Un outil de téléchargement simple est `wget`, avec l'option `-O` pour indiquer la destination. `curl` est également utilisable mais attention, par défaut il envoie dans la sortie standard le contenu téléchargé, il faut rediriger ! Si l'URL contient des métacaractères (p.ex. `?`), il faut entourer la chaîne d'apostrophes (voire de guillemets pour les cas simples)

Archives De nombreux logiciels servent à construire des archives : `tar` en est l'un deux. Combiné à `gzip` ou d'autres logiciels de compression, il est une des façon d'archiver des fichiers.

On désarchive avec : `tar -xvzf archive.tar.gz`

On archive avec `tar -cvzf archive.tar.gz ARGUMENT`, où `ARGUMENT` est un répertoire (p.ex. `.`), une liste de fichiers ou de répertoires, etc.

`Attention au piège`: le premier argument est l'archive ! si vous y mettez un fichier existant, il sera écrasé.

Installer une commande Si une commande n'est pas disponible (erreur si on tape son nom), on peut installer le package qui la contient : vous pouvez chercher des mots-clés avec `apt cache search`, ou chercher un fichier particulier dans l'ensemble des packages de Debian : https://www.debian.org/distrib/packages#search_contents.

2. version plus lisible et *imbricable* des *backticks* (apostrophe inverse `) : par exemple, `echo start $(whoami) end` affiche la même chose que `echo start `whoami` end`, soit la même chose que le texte `start` puis le résultat de la commande `whoami`, puis le texte `end`; voir aussi https://www.gnu.org/software/bash/manual/html_node/Command-Substitution.html – en bref, cela permet de mettre sur la ligne de commande la sortie d'une autre commande

Composition de commandes du shell Le shell bash permet de combiner des commandes de diverses manières :

séquence on exécute les commandes les unes après les autres : `ls; id; date`

séquence conditionnelle on exécute `commande2` uniquement si `commande1` a réussi : `commande1 && commande2` ou respectivement échoué : `commande1 || commande2`

pipe (tuyau) `ls -la | sort -k 5,5 -n` – la sortie de la commande `ls` est triée numériquement selon sa cinquième colonne croissante (ici la taille du fichier ou du répertoire) : techniquement la sortie standard de `ls` est connectée à l'entrée standard de la commande `sort` et les deux commandes sont exécutées en parallèle comme filles du processus du shell ; la sortie d'erreur de chacune des commande reste sur l'écran.

résultat comme argument `ls -la $(ls -1 --sort=size | head -1)` – donne les détails du plus gros fichier ou répertoire du répertoire courant

On peut aller beaucoup plus loin avec cette notion de composition de commande : imaginons que l'on veuille créer une archive du répertoire `images` dans l'archive compressée `/tmp/archive.tar.gz`, mais que l'on veuille simultanément créer la version non compressée `/tmp/archive.tar` ainsi qu'un fichier séparé contenant un hachage sha256 (résumé cryptographique du contenu du fichier) :

```
tar cf - images \
   | tee /tmp/archive.tar >(sha256sum > /tmp/archive.tar.SHA256) \
   | gzip -9 > /tmp/archive.tar.gz

ls -la /tmp/archive.tar*
-rw-r--r-- 1 schaefer schaefer 6758400 Dec 19 15:28 /tmp/archive.tar
-rw-r--r-- 1 schaefer schaefer  329805 Dec 19 15:28 /tmp/archive.tar.gz
-rw-r--r-- 1 schaefer schaefer      68 Dec 19 15:28 /tmp/archive.tar.SHA2

sha256sum --check /tmp/archive.tar.SHA256 < /tmp/archive.tar
-: OK

diff /tmp/archive.tar <(gzip -d < /tmp/archive.tar.gz)
```

Ceci illustre la possibilité d'ouvrir des fichiers ou des pipes vers des commandes complexes avec bash avec `>(...)` (ou depuis, avec `<(...)`).

Encore plus fort : `diff <(ssh serveur1 cat /etc/motd) <(ssh serveur2 cat /etc/motd)`
Ceci compare (via SSH [3]) deux fichiers situés sur deux serveurs différents : en effet, cela lance les deux processus SSH, et ouvre un *pipe* [4] pour chaque, en lecture avec leur sortie (ici le résultat de la commande `cat`), puis passe les noms de fichiers [5] des pipes en argument à la commande `diff`. Cela marchera tant qu'il y a accès séquentiel, ce que font la plupart des *filtres* UNIX (`cat`, `diff`, `sed`, `awk`, `sort` ...)

3. connexion à distance sécurisée
4. tuyau, comme `|`
5. on peut le voir en préfixant la commande avec un `echo`

Processus – 4.3

— lancez la commande `sleep 3600` et constatez que le shell n'est plus accessible[a]
— lancez un autre terminal, et visualisez l'arborescence des processus avec `pstree`
— déterminez quel est le numéro de processus de la commande `sleep` et du shell qui la contient, avec les commande `ps`
— terminez la commande avec CTRL–D (fin de fichier dans un terminal) ou CTRL–C (terminaison avec signal `SIGINT`)
25 — lancez plusieurs fois cette commande, mais cette fois en y ajoutant un `&` à la fin
— utilisez la commande `jobs -l` pour obtenir les numéros de processus (global) et de job (du shell courant)
— amenez en avant-plan (clavier) un des jobs avec `fg %3`, si 3 est le numéro de job
— suspendez ce processus avec CTRL–Z
— mettez ce processus en arrière-plan avec bg
— tuez l'ensemble de ces processus avec `kill` (en utilisant soit l'identificateur de job p.ex. `%3`, soit le numéro de processus global)

a. le clavier de ce terminal est maintenant monopolisé par la commande `sleep`

Conservez des traces de vos actions – consultez la page 55.

Jobs et processus Un processus a un numéro de processus global pour le système, par exemple indiqué par la commande `ps x` :

```
schaefer@reliand:~$ ps x | grep sleep
 6653 pts/0    S      0:00 sleep 3600
 6654 pts/0    S      0:00 sleep 3600
 6655 pts/0    S      0:00 sleep 3600
 6665 pts/0    S+     0:00 grep sleep
```

(ici le dernier processus indiqué est un effet de bord du lancement en parallèle de la commande ps et du grep via un pipe, qui filtre les résultats qui nous intéresse)

ou les numéros de processus et les numéros de job `jobs -l` pour les processus liés au shell courant :

```
schaefer@reliand:~$ jobs -l
[1]   6653 Running                 sleep 3600 &
[2]-  6654 Running                 sleep 3600 &
[3]+  6655 Running                 sleep 3600 &
```

Ci-dessus les numéros de processus sont 6653, 6654 et 6655 et les numéros de jobs 1, 2 et 3.

Sélection de jobs Dans la liste de jobs précédente, le job 3 est marqué comme le prochain sélectionné par défaut par un simple `%` et le 2 le suivant dans la liste.

Les commandes qui agissent par défaut sur les jobs (comme p.ex. `fg` ou `bg`[1]) prennent le numéro de job, éventuellement préfixé par un pourcent `%`. Par contre, les commandes qui agissent normalement sur un numéro de processus, comme `kill` prennent par défaut un numéro de *processus*, ou éventuellement un numéro de job du même shell obligatoirement préfixé d'un `%`.

En effet, l'identificateur de job a une portée liée au shell concerné, et l'identificateur de processus une portée globale.

Suspension et besoin du clavier Lorsqu'un programme qui a besoin du clavier est mis en arrière plan, son exécution est automatiquement suspendue, comparons :

```
schaefer@reliand:~$ sleep 3600 &
[1] 7382
schaefer@reliand:~$ nano /tmp/test&
[2] 7384
schaefer@reliand:~$ jobs -l
[1]-  7382 Running                 sleep 3600 &
[2]+  7384 Stopped (tty output)    nano /tmp/test
```

Ici, la commande `%` ou `fg %2` remettra automatiquement le programme `nano` au premier plan et le suspendra (en anglais : stop qui ne veut pas dire terminer). On peut aussi suspendre et continuer n'importe quel programme via des signaux spécifiques :

```
schaefer@reliand:~$ kill -STOP %1

[1]+  Stopped                 sleep 3600
schaefer@reliand:~$ jobs -l
[1]+  7382 Stopped (signal)        sleep 3600
schaefer@reliand:~$ kill -CONT %1
schaefer@reliand:~$ jobs -l
[1]+  7382 Running                 sleep 3600 &
```

1. consultez également la page 20 (Avant-plan et arrière plan)

Traitement de texte – 4.4

— on vous fournit un log (fichier journal, voir Git) à télécharger qui contient en première et unique colonne une adresse IP : à chaque fois qu'une machine accède à un service, son adresse IP y est logguée.
— écrivez les lignes de commandes shell qui, sans utiliser de fichier intermédiaire :

1. comptent le nombre d'accès total
2. comptent le nombre d'adresses IP uniques
3. transforment la liste d'adresse IP en une sortie de plusieurs lignes contenant chacune le nombre d'occurences et l'adresse IP correspondante (indication : lisez bien la manpage de `uniq`, y compris la condition sur le tri des données), puis, dans la même séquence de commandes extraient les 2 lignes avec le nombre d'occurence le plus grand (indication : tri numérique d'une colonne spécifique)
4. extraient uniquement les adresses IP du point précédent (indication : sélectionnez la deuxième colonne avec `cut` ou, avancés, avec `awk`)

26

Réponses

1.

2.

3.

4.

Très avancés : il y a peut-être des lignes vides dans le fichier original, comment éviter des problèmes ?

diff et patch – 4.5

— on vous fournit trois codes sources simples en C (Wiki)
  — `un.c` qui contient une erreur de syntaxe
  — `deux.c` qui a cette erreur corrigée
  — `trois.c` qui est un programme *différent* mais avec la même erreur au même endroit

27 — on vous demande de :

1. comparer les deux fichiers avec la commande `diff` et comprendre le problème
2. générer un fichier *patch* de type unifié avec `diff -uP un.c deux.c`, nommé `correction.patch`
3. écrire un script shell avec les directives en page 64, vous patcherez et produirez un exécutable qui sera lancé
4. corriger manuellement avec `vi` le fichier `trois.c`, ajoutez un commentaire du C, sauvez et comparez avec `quatre.c`

Installation du compilateur Il vous faudra le package `build-essential`.

diff et patch Ces deux commandes sont des outils très utilisés par les développeurs pour respectivement produire des patches permettant de mettre à niveau une source voire même prendre des décisions de gestion d'un projet logiciel en intégrant (appliquant) ou non un *patchset* d'un contributeur en fonction de sa qualité.

Les systèmes de contrôle de version, comme par exemple `git`, font usage de cette notion de *patchset*. Par exemple :

```
git log doc/rapport.md

commit 036e3681c1fdb15590117aefcafeb088a299381e
[ ... ]

git diff 036e3681c1fdb15590117aefcafeb088a299381e
[ ... ]
-/etc/nginx/sites-avalaible et /etc/nginx/sites-enabled.
+/etc/nginx/sites-available et /etc/nginx/sites-enabled
```

Directives du script shell à écrire L'idée est de créer un script shell qui patchera un fichier source et compilera automatiquement [1] une application. Cela vous montrera aussi comment lancer un éditeur en mode GUI en arrière-plan (sans monopoliser le clavier de la fenêtre de terminal), comment appliquer des permissions et le fait que les extensions de nom de fichiers sous UNIX n'ont pas nécessairement d'importance.

Attention : si vous éditez ce script sous Microsoft, assurez-vous de sauver en UTF-8 *sans* BOM [2]. Sinon, le symptôme sera que la ligne `#!` produira une erreur d'exécutable non trouvé – et la commande `file` montrera qu'il y a un BOM.

1. lancez la commande `pluma` et constatez que le clavier, du shell ou terminal depuis lequel vous l'avez lancé, n'est plus disponible ; quittez `pluma` (CTRL–C dans le shell, ou menu de pluma)
2. créez un nouveau fichier, par exemple en lançant en *arrière plan* : `pluma mon-script &` et constatez que le clavier est toujours disponible dans le shell ; vous pouvez laisser pluma ouvert et simplement activer sa fonction de sauvegarde quand vous voudrez tester votre script
3. dans le script, écrivez une ligne de commentaire spécial *hash-bang* [3] `#! /bin/bash` tout en haut, puis vos commandes :
   — copier `trois.c` vers `quatre.c`
   — appliquer le fichier patch (avec la commande du même nom), `correction.patch` à `quatre.c`
   — compiler avec `gcc -Wall quatre.c -o quatre`
   — renommer l'application `quatre` en `application`
   — lancer `application` correctement
   Indications
   — pourquoi ne pas tester les commandes individuellement dans le shell resté ouvert et copier-coller les commandes qui marchent dans pluma ?
   — avancés : un `set -e` permet de faire planter le script à la première erreur qui suit.
4. testez ce script avec `bash mon-script` et corrigez-le s'il y a des problèmes.
5. testez ce script avec `./mon-script` et consultez l'erreur : appliquez un `chmod a+x mon-script` pour que le fichier soit exécutable et réessayez
6. avancés : remplacez, dans le script, la ligne `#! /bin/bash` par `#! /bin/cat` et lancez le script à nouveau et expliquez ce qu'il se passe

1. une façon plus classique est avec un Makefile, qu'on exercera dans un autre laboratoire
2. Voir couche 6 Réseaux, théorie et exercices
3. *hash* pour le dièse et *bang* pour le point d'exclamation ; elle permet à UNIX de savoir avec quel interprète exécuter ce script

Questions – 4.6

1. qui effectue l'expansion des jokers/wildcards, le shell (avant de lancer la commande) ou chaque commande après son lancement ? montrez sur un exemple :

   (si ce n'est pas évident, essayez la séquence suivante et expliquez les points commentés :

```
cd
mkdir tt
cd tt
echo *
touch echo
echo *
touch toto
echo *

# quelle commande sera exécutée avec quel argument ?
*

rm *
touch a.c
find ~ -name *.c -print

# qui étend le ~ ? qui étend le *.c ?
echo find ~ -name *.c -print

touch b.c

# pourquoi une erreur?
find ~ -name *.c -print
echo find ~ -name *.c -print
find ~ -name '*.c' -print

cd ..
rm -r tt
```

)

   est-ce une bonne chose ou pas que l'expansion des arguments se fassent ainsi ? discutez !

   très avancés : le `CMD.EXE` de Microsoft fait-il aussi ainsi ?

2. si je veux afficher une étoile avec `echo`, que dois-je faire ? (donnez au moins 3 idées)

3. comment [4] peut-on utiliser la syntaxe `grep` ou `sed` dans `vim`, par exemple pour chercher/remplacer ou supprimer des lignes contenant un motif ?

4. quels autres signaux peut-on envoyer avec la commande `kill` ? (indication : il n'y a pas que des signaux de terminaison ; voir pex. `help kill` pour une façon de les lister tous)

   entourez celui qui ne devrait servir qu'en cas de dernière extrémité, par exemple si le processus n'a pas réagi aux autres signaux de terminaison interceptables.

5. quel problème particulier posent les fichiers contenant des espaces aux commandes UNIX ? que faire ?

6. `cd` est-il une commande interne [5] de bash ou un exécutable ? (indication : `type cd` ; en fait sinon `cd` ne pourrait pas fonctionner) ?

4. http://stackoverflow.com/questions/509690/how-can-you-list-the-matches-of-vims-search et http://vimregex.com/#global

5. une commande interne de bash est exécutée dans le processus `bash` ; sinon les commandes sont exécutées comme processus séparé (fils), qui se termine à la fin de l'exécution de cette commande.

7. avancés : créez un répertoire `toto`, puis indiquez comment, avec une seule commande `chmod`, changer les permissions en : `drwxr-x--x`

   expliquez ensuite l'effet de ces permissions sur les *autres* (utilisateurs UNIX ni propriétaires, ni membres du groupe) : en particulier, si vous créez un fichier `toto.txt` dans ce répertoire, est-il affichable en indiquant son chemin complet (par un autre utilisateur), et le répertoire lui-même est-il listable ?

   déduisez-en une application pratique !

8. avancés : pourquoi la commande `sudo echo test=1 >> /etc/environment` ne fonctionne pas ? (indication : qui exécute quoi, qui ouvre quel fichier, éventuellement essayez la commande et des redirections à part)

   proposez une solution en une ligne.

Checklist – 4.7

base :

je suis à l'aise avec la ligne de commande UNIX (édition, historique …)	☐
je consulte les pages de manuels en-ligne et je trouve la réponse à mes questions	☐
je peux gérer mes fichiers en ligne de commande	☐
je sais appliquer des traitements simples à des fichiers textes, y compris à l'aide de *pipe*s	☐
je peux appliquer les opérations de base aux processus	☐

normal :

je sais quand utiliser les guillemets ou apostrophes dans les arguments de commandes	☐
je sais utiliser la commande `diff` pour comparer des fichiers textes	☐
je sais lancer une commande en arrière-plan	☐
je sais créer, activer et lancer un petit script `bash`	☐

avancé :

je sais utiliser certains jokers/wildcards du shell et les échapper	☐
je sais générer des patches et utiliser la commande `patch` pour les appliquer	☐

Evaluation des objectifs par l'enseignant – 4.8

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

5. Emulation de réseaux couche 3

Objectifs – 5.1

28

— expérimenter en pratique les concepts couche 3 vus au cours :
  — sous-réseau, routeur
  — portée (scope)
  — ARP
  — routage vs routage/NAT
— en apprendre un tout petit plus sur le fonctionnement de netkit (comment on implémente les liaisons virtuelles, comment on y configure des adresses et des routes)

allez à votre rythme : n'oubliez pas maintenir la checklist en fin de ce labo (page 79)

Gardez une trace Faites des captures d'écran pour documenter vos observations, voire des captures de trafic (avec `tcpdump`).

Un réseau couche 3 ? Qu'est-ce que c'est ? – 5.2

- une fédération de réseaux/sous-réseaux (un inter-réseau ou **internet**, par exemple le fameux Internet)
- des **terminaux** configurés à la main (statiquement) ou par **DHCP** (dynamiquement) : PCs, serveurs, imprimantes, équipements divers, ...
  - au moins une **adresse IP**, configurée au sein d'un sous-réseau donné
  - au moins une entrée de table de routage vers le routeur par défaut (**default route**)

29

- des **noeuds** couche 3 : les routeurs
  - possèdent une adresse IP par interface : chaque interface étant situé dans autre sous-réseau
  - assurent le **routage** :

    statique préconfiguré : **tables de routage statique**

    dynamique protocoles de routage internes ou externes, permettant de créer dynamiquement des **tables de routage** : p.ex. RIP, OSPF, EGP

Couche 3 Une couche 3, unique au travers d'un grand réseau comme Internet, utilise des couche 2 (et 1) intermédiaires qui peuvent être de technologies diverses. La portée de la couche 3 est universelle [1] et la portée de chaque couche 2 est identique à celle d'un *sous-réseau* de couche 3.

1. hors NAT

Définition de l'adressage – 5.3

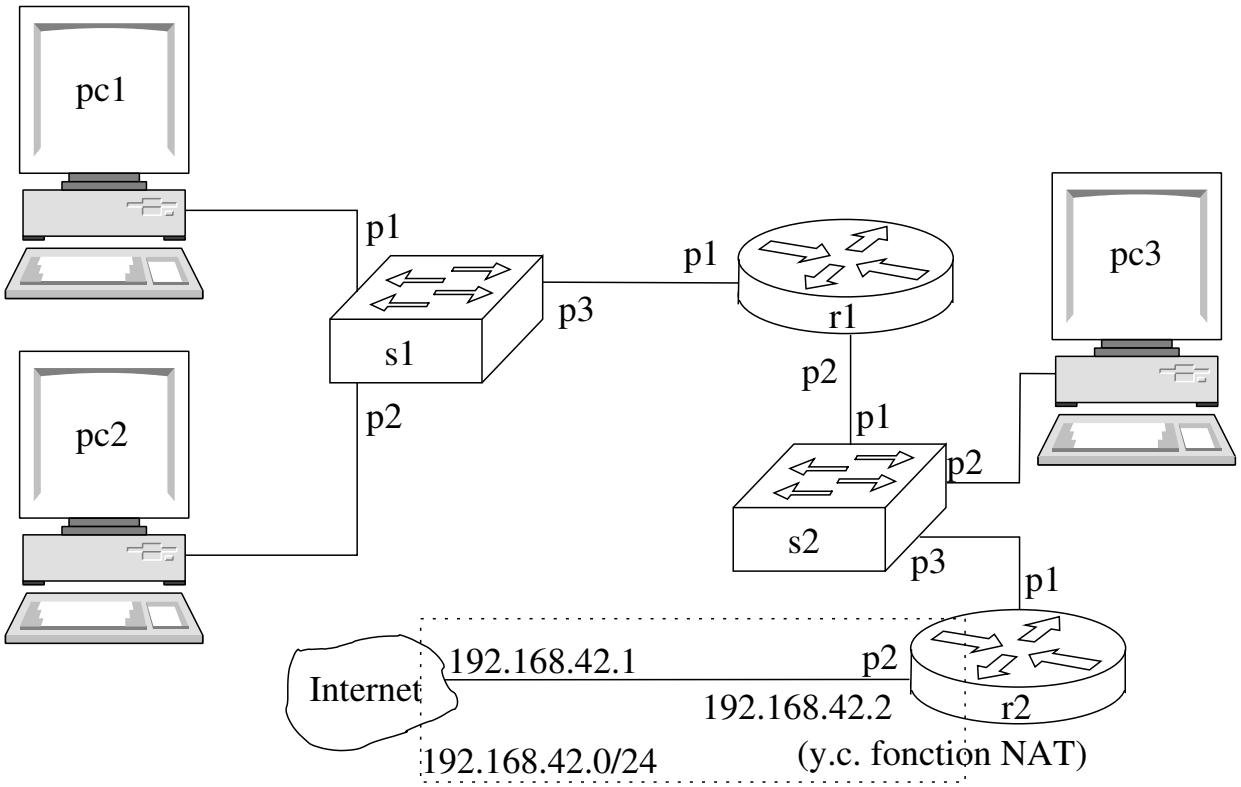
Complétez le schéma ci-dessous (toutes les adresses statiques) :

- entourez les sous-réseaux et indiquez pour chaque sous-réseau son adresse en notation CIDR
- indiquez ensuite les adresses et netmasks indispensables de *tous* les équipements qui en ont (couche 3 seulement) – équipements idéaux !
- l'interface `p2` de `r2` nous permettra d'aller sur Internet via (plusieurs) NAT

30

Directive : choisissez vos sous-réseaux dans des plages *privées* à choix : toutefois les sous-réseaux doivent être *disjoints* entre eux (aussi disjoint du sous-réseau de `r2.p2`. Un des sous-réseaux devra être de taille **/28** (et sa 1ère adresse 48) et un de taille **/24** (1ère adresse : 0). N'utilisez pas de sous-réseau dans les plages 10.0.0.0/8, 192.168.154.0/24 ni 192.168.156.0/24.

Schéma de notre réseau Netkit (couche 2 et 3)



Installation du labo Netkit – 5.4

```
netkit-lab routing
```

N'oubliez pas de sourcer le fichier de définition (voir page 73).

31

Structure d'un labo netkit Voici les principaux fichiers et répertoires :

lab.conf configuration générale et notamment topologie couche 1 du réseau décrit : chacune des interfaces de chaque équipement est nommée et affectée à un brin de câble qui porte un nom (usuellement une lettre)

répertoires chacun des équipements qui sera lancé sous forme de machine virtuelle dispose de son propre répertoire où netkit stockera l'état de cette machine virtuelle, sous forme différentielle (copy-on-write : uniquement les changements apportés au filesystem commun à toutes les VM netkit sont inscrits)

***X*.startup** chaque équipement *X* dispose de son propre fichier de configuration dans lesquels les interfaces sont activées, les adresses couche 2 (MAC) assignées et les adresses et routes, ainsi que toutes les configurations nécessaires, sont inscrites : on peut aussi exécuter des commandes shell UNIX qui le seront dans le contexte de la VM de l'équipement – ces fichiers seront à compléter avant le lancement.

Configuration du labo Netkit – 5.5

— suivez la logique de câblage virtuel (implémentant le réseau de la page 71), sous forme de lettres représentant des câbles du fichier `lab.conf`
— éditez les fichiers proposés `{pc,r}[1-3].startup` et reportez-y aux endroits indiqués les adresses IP et les netmasks en notation CIDR choisis par vous
— reportez sur les ports du schéma en page 71 les noms des interfaces réseau Linux (`ethX`) réellement utilisés
— démarrez ensuite le labo en suivant les instructions ci-dessous, puis faites les observations et manipulations de la page suivante

32

NB : toutes les adresses MAC sont déjà préconfigurées et les routeurs sont prêts à router, toutefois les adresses IP ne sont pas toutes configurées, et il manque des routes sur les routeurs **et** sur les PCs – pas de configuration automatique DHCP !

Attention : un objectif de ce labo est de *réfléchir* à toutes les routes nécessaires, y compris sur les PCs.

Rappel : installation du labo netkit routing

1. `source ~/NETKIT/SOURCE_ME`
2. `netkit-lab routing` (si pas déjà fait)

Démarrage du labo Démarrez le labo (rappel : `lstart -p0` pour démarrer, `lcrash; lclean` pour arrêter). Il y aura une demande `sudo` pour la création de la liaison au réseau réel (via pseudo-device UNIX *tap* et fonction NAT).

Idée : disposez les fenêtres `pc1`, `pc2`, `r1`, `pc3`, `r2` comme sur le schéma et minimisez les fenêtres `s1` et `s2`.

Routeur Une machine (terminal) n'est pas forcément un routeur (noeud). Pour l'activer, sous Linux : `echo > /proc/sys/net/ipv4/ip_forward 1`
(`/proc` est un pseudo-filesystem qui permet facilement de parler avec le kernel)

Dans notre cas, c'est *déjà* fait.

Trouver les problèmes Pour qu'une machine soit correctement configurée *dans* un sous-réseau donné, il faut que son adresse IP soit dans [1] le sous réseau, qu'elle soit unique au sein de ce sous-réseau et qu'elle ne soit pas réservée [2]. On peut bien sur *tester* en faisant des `ping` des autres machines du sous-réseau (y compris le routeur).

Pour qu'une machine puisse sortir du sous-réseau, il faut qu'il existe au moins un routeur dans le sous-réseau, usuellement un routeur par défaut [3]. En capturant avec `tcpdump -i eth0 -e -n -s 0`, on verra alors que les datagrammes destinés à l'extérieur du sous-réseau sont envoyés *via* [4] l'adresse MAC (couche 2) du routeur indiqué dans la route par défaut.

Toutefois, et c'est que l'on fait dans ce laboratoire, on va généraliser ce concept à plusieurs sous-réseaux qui ne sont pas forcément atteignables par le routeur par défaut [5] !

En effet, la route par défaut est une configuration qui indique que pour atteindre tout Internet (sous-réseau `0.0.0.0/0`) il faut passer via le routeur indiqué pour cette route. En pratique, quand notre machine enverra un datagramme vers une adresse qui n'est pas dans une autre route plus précise (y.c. le sous-réseau auquel elle est branchée), elle enverra en *couche 2* à l'adresse MAC du routeur dont l'adresse IP est indiquée dans la route par défaut.

Or, il se peut aussi qu'il y ait des exceptions à cette route par défaut [6] : un sous-réseau d'envergure plus petite (donc avec un nombre de bits fixes plus grand) peut être routé par un autre routeur ! Une route supplémentaire donc donc être ajoutée : cela arrive en général plutôt sur des routeurs, mais dans ce laboratoire, cela devra être fait *aussi* sur certains PC en plus.

Procédure suggérée pour ce laboratoire :

1. tester que chaque routeur et chaque PC peut atteindre au moins une adresse IP de chaque sous-réseau hors de ses propres adresses, y compris Internet
2. si cela ne fonctionne pas, consulter l'éventuel message d'erreur (qui peut signaler p.ex. avec `Network unreachable` que le sous-réseau n'a pas de route pour l'atteindre, ou `Host unreachable` que le protocole ARP n'a pas fonctionné, rare si la machine destination est configurée et enclenchée)
3. s'il n'y a pas de message d'erreur, c'est que le datagramme est envoyé au mauvais endroit ou que c'est la réponse qui s'est perdue :
   (a) utiliser `tcpdump -i eth0 -e -s 0 -n` pour vérifier l'adresses couche 2 du routeur via lequel on a envoyé le datagramme (adapter éventuellement le nom de l'interface, ici `eth0`)
   (b) si c'est correct, c'est peut-être la destination qui n'arrive pas à acheminer correctement sa réponse car *sa* route n'est pas bonne (exclure ce cas en vérifiant l'adresse MAC de la réponse, ou en vérifiant éventuellement sur le ou les routeurs traversés)

1. donc que son adresse IP *AND* netmask == adresse du sous-réseau
2. la première et la dernière adresse d'un sous-réseau sont réservées respectivement à l'adresse de sous-réseau et au *broadcast*, et il a toujours au moins un routeur si l'on veut sortir, qui par convention utilise la 1ère adresse utilisable du sous-réseau
3. *default gateway*, ou passerelle par défaut – au sens couche 3
4. après conversion d'adresse couche 3 vers couche 2 (ARP) si pas déjà dans son cache ARP
5. un routeur trop intelligent pourrait envoyer, pour les datagrammes incorrectement routés via lui, des redirections ICMP qui pourraient être traitées ou non par nos PC, dans notre laboratoire, c'est désactivé
6. il y a de toute façon une route pour chaque sous-réseau branché, mais là l'envoi ne passe par par un routeur : l'envoi est directe à l'adresse couche 2 destination (obtenue par ARP)

Questions – 5.6

1. pour tous les PCs et routeurs
   (a) à l'aide de l'outil `ping` (vers des machines du même sous-réseau, des autres sous-réseaux et vers Internet p.ex. 8.8.8.8 [7]) vérifiez la connectivité des différents équipements de couche 3 ; s'il y a un problème, debuggez avec `tcpdump -i INTERFACE -n -v -e` notamment en réfléchissant à qui sont adressés les datagrammes IP une fois encapsulés en couche 2 (l'option -e visualise les adresses MAC) : réfléchissez, et **ajoutez les routes manquantes** (d'abord interactivement et après dans les fichiers de configuration de netkit – y compris sur les PCs dans ce cas – avec redémarrage – consultez également la section *Trouver les problèmes* en page 74)
   (b) visualisez la configuration réseau (`ifconfig` ou `ip addr show`) et les tables de routage (`netstat -r` ou `ip route show`)
   (c) voyez-vous des entrées implicites (automatiques) de table de routage qui ont été configurées automatiquement, sans utiliser de commande `route` spécifique dans le démarrage [8], mais en dérivant les sous-réseaux des configurations adresse IP et netmask ?

2. réflexions sur le routage
   (a) à quel sous-réseau correspond la *route par défaut* ? indiquez-le en *notation CIDR* (indication : comparez la sortie de `netstat -rn` et de `netstat -r`)
   (b) mettez-vous à la place du logiciel routeur qui décide sur quelle interface envoyer un datagramme, en fonction de l'adresse destination et de la table de routage : dans quel ordre faut-il traiter les entrées de la table de routage pour éviter des ambiguités (faites un exemple concret p.ex. avec `r2` ou `pc3`, puis essayez de généraliser à un principe)

7. serveur DNS Google
8. ici dans les fichiers `pcX.startup` par exemple

3. complétez le tableau ci-après pour un `ping` de `pc1` à `pc3`, en ne tenant compte que du datagramme ICMP ECHO REQUEST (aller), et en utilisant `tcpdump -i INTERFACE -n -v -e` :

	MAC		IP	
	src	dst	src	dst
capture sur `pc1`				
capture sur `pc3`				

   Indication : pour faire un `ping` en même temps qu'un `tcpdump`, deux possibilités :

   (a) `tcpdump -w fichier -s 0 -n -e &` (puis reprendre en avant-plan avec %, tuer avec CTRL-C, et consulter avec `tcpdump -r fichier -s 0 -n -e -v`)

   (b) multiplexer le terminal avec l'outil `screen`

   Au cours, nous avons vu que les adresses couche 2 ont une portée du domaine de diffusion (toute la couche 2 concernée), ou plus simplement que leur portée s'arrête aux routeurs. Par contre, la couche 3 a une portée universelle.

   Expliquez en quoi le tableau de la page précédente est conforme à cette théorie de la portée (scope) de la couche 2 ?

   Ici, les adresses de couche 3 ont-elles *vraiment* une portée universelle ? sinon, pourquoi ?

4. à l'aide de l'outils `mtr` (meilleur traceroute) vérifiez que vous sortez bien par le(s) routeur(s) prévus lorsque vous tentez une trace vers p.ex. `8.8.8.8` depuis `pc1` ; expliquez la ligne bizarre entre votre réseau virtuel et le réseau de la HE-Arc : de quoi s'agit-il ? (indications : faites `ip addr show` sur votre machine virtuelle Linux et rappelez-vous dans quel mode réseau elle a été configurée dans VirtualBox)

5. dans notre réseau, lister le(s) routeur(s) qui font du routage *avec* fonction NAT :

   (avancés : et dans le mtr ci-dessus ?)

6. la commande `ipcalc`[9] permet de faire des calculs de netmask : essayez `ipcalc 157.26.77.42/24` et `ipcalc 157.26.89.7/22`

9. `sudo apt-get install ipcalc` dans votre Linux

7. qu'est-ce que le *Software Defined Networking* (SDN), en quelques mots ?

8. (BONUS) avancés : comparez les traces visibles sur `eth1` de `r2` et sur l'interface réseau externe du host Linux [10] et identifiez la fonction NAT : utilisez pour cela un `ping` d'une adresse IP extérieure à votre émulateur (p.ex. l'adresse IP Ethernet de votre PC, 8.8.8.8, ou 157.26.77.13 depuis la HE-Arc).

9. avancés : que pensez-vous de la configuration de notre réseau ? spécifiquement, est-ce une bonne idée de ne pas avoir de routeur central, en ce qui concerne la configuration des terminaux ? (on voudrait n'avoir qu'une route par défaut, p.ex. diffusée par DHCP, sur chacun des terminaux – PCs). Dessinez le réseau ainsi simplifié (couche 3 uniquement, mais en conservant les 3 sous-réseaux !)

10. avancés : pourquoi vous demande-t-on le mot de passe `root` pendant le lancement de r_2 en page 73 ?

11. avancés : quel est le principe du joker/wildcard du shell en page 73 ?

12. (BONUS) avancés : en quoi ARP est un risque de sécurité ? que proposez-vous pour le limiter ?

10. vous pouvez utiliser `netstat -rn`, `ip route` ou `ip link` – vous pouvez aussi lancer Wireshark sur votre Linux et choisir la bonne interface

13. très avancés : essayez le laboratoire *dynrouting*[11] (routage par échanges entre routeurs RIP)

11. netkit-lab dynrouting

Checklist – 5.7

base :

j'ai pu définir les sous-réseaux nécessaires et affecter des adresses et netmask	☐
je comprends le principe général d'une configuration d'un labo netkit	☐
j'ai pu configurer ce laboratoire Netkit en fonction des sous-réseaux et adresses définies	☐
j'ai pu lancer l'émulation Netkit	☐

normal :

j'ai pu comprendre les routes qui manquaient et ajouter celles-ci	☐
la portée des couches, sur un exemple, est claire pour moi	☐
j'ai pu capturer le protocole ARP et comprendre la notion de cache ARP	☐
j'ai répondu aux questions	☐

avancé :

j'ai vu le fonctionnement du NAT/PAT	☐
j'ai de meilleures notions du placement des routeurs dans le but de simplifier la configuration des terminaux	☐
j'ai pu mettre en place du routage dynamique	☐
j'ai répondu aux questions avancées	☐

Evaluation des objectifs par l'enseignant – 5.8

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

6. Outils d'analyse réseau

Objectifs – 6.1

33

- approfondir l'outil de capture Wireshark
- appliquer la théorie de couche 2 et 3 vue en cours Réseaux
- se documenter sur certaines commandes réseaux
- résumer le fonctionnement du protocole DHCP
- expliquer le NAT sur un cas concret
- aller plus loin en s'informant sur IPv6 sur un cas concret

Directives : ce laboratoire est en fait le laboratoire à rendre de Réseaux.

Directives
A rendre sous forme d'un rapport de laboratoire Réseaux par groupe de 2 (ou 3 si nombre impair), au délai indiqué, à l'enseignant correspondant.

Rapport court, avec illustrations expliquées et liens à la théorie et références (cours, Internet, ...) ; consultez les Recommandations pour les rapports de laboratoire [1] ainsi que les indications dans chacune des parties ci-après.

En résumé : observer-expliquer-relier à la théorie.

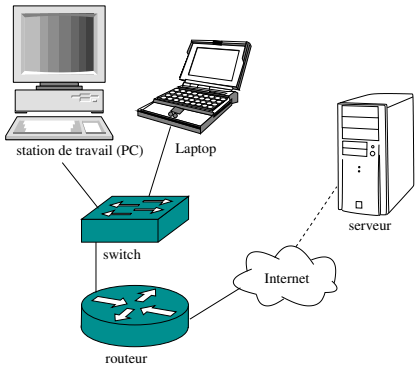
Lieu Ce laboratoire sera commencé lors du cours Réseaux, puis vous pourrez le terminer au laboratoire durant le laboratoire Découverte OS et Réseaux. Si nécessaire, vous pouvez également le terminer chez vous : il suffit d'adapter les adresses IP à la configuration réelle de votre réseau.

Captures Par captures, on peut entendre une simple copie d'écran, mais aussi les fichiers en format `pcap` de Wireshark : pour ces derniers, vous avez intérêt de les conserver pour pouvoir choisir comment vous allez les présenter dans le rapport.

1. https://gitlab-etu.ing.he-arc.ch/isc/documentation/docs/-/wikis/recommandations-rapports-labo

Un réseau classique – 6.2

Un réseau classique (au laboratoire, chez vous) est le suivant :



34

— *laptop* et *PC* sont dans le même sous-réseau
— *routeur* est le routeur par défaut de ce sous-réseau (*passerelle par défaut*), et a donc aussi une interface réseau à l'intérieur de ce sous-réseau
— *serveur* est situé dans un autre sous-réseau, accessible à travers un ou plusieurs routeurs

Fonctions idéales en pratique Toutefois :
— les fonctions de *routeur*, serveur DHCP, voire *switch* peuvent être combinées dans un seul équipement
— si le sans-fil est utilisé, le principe est similaire : un access point (AP) peut combiner ou non les fonctions ci-dessus
— des fonctions supplémentaires comme de la réécriture d'adresse (NAT/PAT) ou un petit firewall sont aussi possibles

La vérité est parfois ailleurs Attention : les outils de capture (`tcpdump`, `wireshark`) ne vous montreront pas toujours la réalité :
— le calcul des checksums [1] (p.ex. IP) peut être effectué par la carte réseau (parfois, d'où des lignes en rouge dans Wireshark)
— suppression du CRC [2] par la carte réseau (toujours)
— suppression des informations 802.1q/p (VLAN [3]/priorité) : parfois, suivant les cartes et pilotes
— fusion de trames supérieures au MTU [4] : le travail de décomposition et de calcul des checksum TCP est laissé à la carte ! (optimisation LSO/TSO [5])

1. somme de contrôle de la détection d'erreur (couche 3 : IPv4 entête IP, IPv6 : aucune ; couche 4 : TCP/UDP : entêtes et données)
2. code de redondance cyclique, qui sert à détecter les erreurs en rafales en couche 2
3. réseaux virtuels couche 2
4. *Maximum Transmission Unit* côté ordinateur, découpé par la carte réseau : la taille maximum des trames en couche 2 par exemple
5. https://serverfault.com/questions/459844/server-sending-out-packets-bigger-then-mtu/630533#630533

Quelques particularités Microsoft En cas de duplicat d'adresse, Microsoft Windows 8 n'affiche plus de message mais l'adresse est marquée comme *doublée* dans `IPCONFIG/ALL`.

Sur la table de routage Microsoft, *onlink* signifie qu'il n'y a pas de passerelle (adresse directement *sur le lien* (la liaison, couche 2)), voir http://superuser.com/questions/59996/what-does-on-link-mean-on-the-result-of-route-print-command.

Configuration d'une machine, ARP et portée des adresses MAC – 6.3

objectifs

- analyser la configuration du sous-réseau (adresse de sous-réseau, adresse dynamique de votre machine, adresse IP du routeur, adresse(s) IP du/des serveurs DNS)
- exercer la configuration manuelle (statique) d'une machine
- décrire la notion de routage à l'intérieur et vers l'extérieur d'un sous-réseau (relier les observations à la théorie)

N'oubliez pas de reconfigurer votre laptop en dynamique à la fin du labo (DHCP, adresse & DNS dynamique)

Lieu Ce labo peut être terminé chez vous, de préférence en filaire : il suffit d'*adapter* les adresses IP : il vous faut au moins une adresse IP à l'intérieur de votre réseau (p.ex. celle de l'interface interne de votre routeur, à trouver par exemple en consultant la route par défaut avec `netstat -rn`) ainsi qu'une adresse IP extérieure (p.ex. 8.8.8.8, le DNS de Google).

Le labo peut aussi être fait en WiFi plutôt qu'en Ethernet (filaire). Pour la toute première partie (config manuelle), configurez, faites les captures de la configuration ipconfig/all en recopiant simplement la configuration dynamique dans la manuelle, puis continuez en *remettant* la configuration dynamique !

Matériel et logiciel Lorsqu'effectué dans la salle 304, par groupe de deux étudiants :

- un câble jaune connecté sur le sous-réseau du laboratoire 157.26.77.0/24 (prise **jaune**)
- un ordinateur portable étudiant (laptop) sous Microsoft Windows avec une interface réseau *filaire* 10/100 ou 1000
- logiciel Wireshark

Manipulations

1. désactivez l'interface sans-fil du laptop
2. connectez le laptop au sous-réseau 77 (jaune)
3. lancez Wireshark sur l'interface Ethernet filaire, aidez-vous des captures pour expliquer vos observations

4. configurez votre laptop *manuellement* (via l'interface graphique) comme ci-dessous :
   — sous Microsoft Windows : Panneau de configuration, Afficher l'état et la gestion du réseau, Connexion au réseau local, Propriétés, Protocole IPv4, Propriétés ; faites une copie d'écran
   — adresse IP : 157.26.77.x (avec x = 40 + numéro de poste du PC, de 1 à 15, soit de 41 à 55)
   — netmask 255.255.255.0
   — passerelle IP (gateway IP, routeur par défaut) : 157.26.77.1
   — serveur DNS : 157.26.77.13
5. vérifiez que votre configuration est correcte avec la commande `ipconfig /ALL` ; faites une copie d'écran
6. remplissez le tableau ci-dessous en expérimentant avec la commande `ping` et en capturant avec Wireshark : ces résultats devraient être *cohérents* avec la théorie !

   Attention, s'il y a quelque chose déjà dans le cache ARP (commande `arp -a`) effacez-le, ou essayez avec d'autres adresses dans le même sous-réseau que votre machine ou à l'extérieur (possible aussi chez vous à la maison même en WiFi). Si possible, identifiez (par exemple via le cache ARP) à qui appartiennent les adresses MAC (la machine destination, le routeur du sous-réseau, ...)

adresse IP	même sous réseau ?	requête ARP ?	IP dest	MAC dest
157.26.77.11, 157.26.77.13 ou 157.26.77.15				
8.8.8.8				

   Les deux dernières colonnes de cette table correspondent à l'entête IP et MAC du *datagramme IP* ICMP (pas la requête ARP, s'il y en a eu une).

Indications

— vous pouvez filtrer dans Wireshark avec `arp || icmp` (champ en haut, puis *Apply*)
— si vous ne voyez pas de requêtes ARP, vous pouvez effacer le cache ARP avec

Microsoft Windows `arp -d *` : nécessite les droits administrateur : le plus simple : Menu Démarrer, chercher `cmd.exe`, bouton droite sur l'icône, exécuter en tant qu'administrateur ; sur des versions récentes plutôt effacer avec `arp -d IP`, avec IP p.ex. 157.26.77.13

système standard POSIX (GNU/Linux, Mac OS X) `sudo arp -d 1.2.3.4`

— le rapport doit contenir au minimum :
   — copies d'écrans commentées
   — captures Wireshark expliquées, échanges décrits
   — explications et justifications des observations par rapport à la théorie de routage interne (ARP) et hors du sous-réseau vue en cours
   — tableau complété

traceroute et TTL – 6.4

objectifs

— décrire le fonctionnement de la commande `traceroute` et le lien au champ *TTL* de l'entête IP

— dessiner un diagramme d'interaction décrivant les échanges

36

peut être fait sous Microsoft Windows, Mac OS X ou GNU/Linux, à choix – adaptez les adresses si vous faites chez vous.

Sous Microsoft Windows Capturez les échanges produits par `TRACERT -d 8.8.8.8` et par `TRACERT -d 8.8.8.8` sur l'interface de commandes et par Wireshark et expliquez (essayez aussi sans l'option `-d`).

Sous système standard POSIX (GNU/Linux, MAc OS X, systèmes embarqués... Capturez les échanges produits par `traceroute -n 8.8.8.8` et par `traceroute -d 8.8.8.8` sur l'interface de commandes et par Wireshark et expliquez (essayez aussi sans l'option `-n`).

En option, comparez avec la commande `mtr`.

Indications Le rapport doit contenir, au minimum :

— copies d'écrans commentées

— captures Wireshark commentées, échanges décrits (*diagramme d'interaction*) et expliqués

— explications des observations par rapport à la théorie vue en cours (notamment la manipulation du champ TTL et les messages d'erreur ICMP)

DHCP – Dynamic Host Configuration Protocol – configuration dynamique des adresses – 6.5

objectif

— décrire le fonctionnement du protocole DHCP

37

Aussi possible en WiFi (restrictions).

Aussi possible en WiFi Peut être fait en WiFi plutôt qu'en Ethernet (filaire), toutefois sous Microsoft il n'est pas possible de capturer toutes les interfaces (pas d'interface virtuelle `any`), donc il se peut qu'en WiFi vous ne voyiez pas les échanges initiaux (l'interface étant déconfigurée juste avant l'échange DHCP) – dans tous les cas vérifiez avec Wikipedia si vous avez vu tous les messages. Vous pouvez aussi comparer avec votre VM Debian.

Manipulations

1. désactivez l'interface sans-fil de votre laptop
2. lancez Wireshark sur l'interface filaire
3. configurez votre laptop pour prendre automatiquement une adresse (DHCP) si pas déjà remis
4. analysez la prise automatique d'une adresse (protocole DHCP) avec Wireshark
   — messages échangés (et leur contenu)
   — adresses couche 2 et couche 3

Indications

— si vous la ratez la première fois, faites (Microsoft) : `ipconfig /renew` : attention, cela ne fera pas une demande complète depuis le début : le plus simple pour une demande complète est d'alterner entre réseau jaune et réseau rouge, ou évt. un `ipconfig /release` puis `ipconfig /renew`

- vous pouvez filtrer, dans Wireshark, avec `bootp` (pour ne garder que les trames DHCP), voire `bootp || arp`
- vous pouvez utiliser l'affichage *Flow graph* (menu Statistiques) pour rapidement voir les messages échangés
- sous Linux, on peut capturer l'interface réseau `any` dans Wireshark pour capturer même si le câble réseau est enlevé et remis, p.ex.
- pour le rapport :
  - captures de ce que vous avez vu avec Wireshark et ipconfig
  - description des messages et échanges principaux
  - n'hésitez pas à vous documenter sur Internet pour expliquer vos observations et le fonctionnement du DHCP, et pour vérifier que vous avez vu tous les messages

Commandes – 6.6

objectifs

- utiliser et interpréter la sortie de quelques commandes de base du réseau
- avancés : déterminer la partie du délai RTT [a] qui est due au temps de transmission et en déduire le débit

38

a. *Return Trip Time* : délai aller-retour, `ping`

Commandes multiplateformes Documentez-vous et remplissez le tableau ci-dessous (la ligne Microsoft est déjà remplie) :

plateformes	**ping**	**traceroute**	**visualiser les interfaces réseaux ou les configurer**	**routes**	**réobtenir adresse IP**
GNU/Linux					
Mac OS X					
Microsoft Windows	`PING.EXE`	`TRACERT.EXE`	`IPCONFIG`	`NETSTAT -rn`	`IPCONFIG /RENEW`

WHOIS Montrez, grâce à la base de données WHOIS [1], pour le sous-réseau `193.72.186.0/24` :

- qui en est le propriétaire ?
- quelle adresse e-mail contacter en cas d'abus ?
- quel système autonome (AS) annonce les routes pour ce sous-réseau ?

1. en utilisant la commande `whois` sous GNU/Linux ou un outil web à trouver

— de quel organisme fait partie ce système autonome et qu'est-ce qu'un LIR ?
— comment est branché ce sous-réseau à Internet (indication : `mtr 193.72.186.6` ou traceroute `193.72.186.6`)

RTT Le *Return Trip Time* (RTT) est le délai aller-retour entre deux machines sur Internet. Il peut se mesurer à l'aide de la commande `ping` (qui utilise les messages ICMP *ECHO REQUEST* et *ECHO REPLY*.

Quel est le délai minimal, maximal et la variation de délai entre votre laptop et `46.140.72.222` (estimez avec `ping`) ?

Calcul de débit (avancés) Comment pourriez-vous estimer le débit minimum entre votre laptop et `46.140.72.222`, sachant que le RTT mesuré précédemment contient à la fois un délai dû au réseau et au traitement, mais également le temps de transmission lié au débit minimum sur le chemin.

Idée : envoyer plusieurs pings de taille différentes pour annuler le délai dû au réseau, que l'on supposera constant.

Montrez comment vous procédez sur un exemple et faites un petit calcul de débit.

NAT / explication de captures – 6.7

objectifs

— décrire le fonctionnement d'un routeur avec fonction NAT, dans un cas particulier

— avancés : classifier le type de NAT

39

Cette partie n'a aucun rapport avec les autres parties.

Analyse de captures / NAT On a capturé les deux fichiers ci-dessous (ils se trouvent dans le wiki Découverte OS et Réseaux sous le labo 6) :

`outside.pcap` capture sur l'interface extérieure du routeur (p.ex. ADSL ...)

`inside.pcap` capture sur l'interface intérieure du routeur (p.ex. Ethernet)

Ces captures sont en format `pcap` : elles ont été réalisées avec l'outil embarqué `tcpdump` [1].

Chargez ces captures dans Wireshark et expliquez ce que fait le routeur et pourquoi.

Pour rappel : la couche 3 est de terminal à terminal, les routeurs se bornent à acheminer les datagrammes et à détecter les boucles. Un datagramme voit donc ses adresses couche 3 (IP) source et destination inchangées à travers le réseau. Sauf dans le cas de NAT ! C'est ce que vous allez analyser ici.

Avancés : de quel type de NAT [2] s'agit-il ?

1. malgré son nom, il capture des trames Ethernet contenant par exemple des paquets IP et la suite de l'encapsulation jusqu'à la couche 7
2. https://fr.wikipedia.org/wiki/Network_address_translation#Diff%C3%A9rents_types_de_NAT

Indications

- la capture a été effectuée directement sur un routeur (avec plein d'autres fonctions), style de ceux que vous pourriez avoir chez vous
- on s'intéresse à la **couche 3** *uniquement* ici !
- seules les 3 premières lignes de chaque capture sont importantes
- dans le cas présent, l'interface outside est Ethernet et l'interface inside est virtuelle, mais vous n'avez pas besoin de le savoir pour résoudre cette partie

Si vous ne comprenez rien, réfléchissez à ce qui se passe chez vous, si votre laptop est branché derrière un routeur en A/VDSL ou CATV (UPC) :

- quelle est l'adresse IP de votre laptop ?
- quelle est l'adresse IP du routeur par défaut, vu du laptop ?
- sont-ce des adresses d'un type particulier ?
- quelle est l'adresse IP du routeur, vue de l'extérieur (p.ex. allez sous `https://www.whatismyip.com/` ou `https://www.whatismyipaddress.com/`)
- en conclusion, que fait le routeur de spécial ici ?

Recherche documentaire IPv6 (avancés) – 6.8

objectifs

— chercher dans la documentation en-ligne

— expliquer un problème avancé de réseau

40

Cette partie n'a aucun rapport avec les autres parties.

Le problème Marc a besoin d'un routeur ADSL/VDSL supportant IPv6 *nativement* (pas de tunnel). Il se renseigne, et un vendeur lui dit que le TD-8816 de TP-Link, assez bon marché, est compatible.

Or, à la mise en exploitation, dans le syslog (journaux) du routeur, il y a l'erreur suivante, erreur *émise par ce routeur*, suite à une trame *reçue* du fournisseur (FAI) :

```
Protocol-Reject unknown protocol 0x8057
```

Pourquoi ? Que faire ?

Indications A réfléchir en fonction de ci-dessus, se renseigner (rechercher sur Internet) et motiver les réponses aux questions suivantes :

— l'opérateur choisi propose-t-il vraiment IPv6 natif ? (indication : des trames IPv6 arrivent-elles au routeur ?)

— est-ce que routeur ne supporte pas vraiment IPv6 ? (justifier) proposez une correction

— avancés : est-ce que cela peut marcher aussi si l'on suppose que l'on a en plus un petit PC embarqué avec Linux connectable à cet équipement pour faire de l'IPv6 natif ? (indication : la session couche 2 PPPoE d'ADSL/VDSL peut être terminée [1] soit sur le modem/routeur ADSL/VDSL, soit sur le petit PC Linux embarqué

1. le routeur agit alors en mode modem uniquement et perd toute gestion de la couche 3 ; la liaison entre le modem et le petit PC Linux embarqué peut être Ethernet (protocle PPPoE), ou USB par exemple ; des packets IPv4 et IPv6 y sont alors encapsulés de manière quasi transparente

Évaluation – 6.9

Les poids peuvent varier légèrement du tableau ci-contre :

parties	**poids**
structure	10%
6.3	15%
6.4	20%
6.5	20%
6.6	15%
6.7	15%
6.8	5%

41

Consultez les directives au début du chapitre et les indications dans chaque section.

7. Applications réseaux socket

Objectifs – 7.1

42

- utiliser les outils de développement classiques
  - créer et expliquer des `Makefile` simples
  - tracer en *boîte noire* les actions d'un logiciel, y compris réseau
  - simuler des clients et serveurs TCP ou UDP avec `netcat`
- développer des applications réseaux bas niveau, utilisant les socket POSIX
  - client TCP puis HTTP simple
  - serveur TCP multiprocessus
  - serveur TCP multiplexé (avancés)

allez à votre rythme : n'oubliez pas de maintenir la checklist en fin de ce labo (page 118)

Ce laboratoire Dans ce laboratoire, les éléments avancés ne sont pas nécessaires pour obtenir la note maximum.

NB : certains éléments de ce labo seront réalisés pendant le cours RéseauxISC1, notamment les pages 97 à 111 – en 2022-2023 c'est encore à préciser.

Boîte noire / boîte blanche Le concept de boîte noire ou blanche vient du test logiciel : on parle de tests en boîte noire lorsqu'on s'intéresse uniquement aux fonctionnalités du logiciel, sans tenir compte de leur implémentation.

Ici, on ne s'intéressera qu'aux interactions entre le programme et les appels systèmes (section 2 du manuel UNIX, commande de traçage `strace`) voire les appels de bibliothèques (section 3, commande `ltrace`), grâce à des programmes de traçage.

On parle de test en boîte blanche lorsqu'on a le code source complet et donc que l'on peut adapter ce que l'on instrumente en fonction, et se concentrer sur des éléments qui nous intéresse (p.ex. pour vérifier la sécurité d'une partie du code, spécifiquement).

Makefile – 7.2

principes

- rôle : construction de cibles à partir de dépendances, par exemple pour la compilation
- syntaxe :
  `cible: dépendances`
  `TABULATION commandes`
- notion de dépendances
- la première cible du fichier est exécutée par défaut (sinon la/les cible(s) sont les arguments du `make`)
- règles implicites (notamment pour le C)
- langage interne simple ; extensions par le shell
- version *luxe* : GNU `make` avec un langage puissant

Exercices : Makefile

1. `sudo apt-get update`
   `sudo apt-get install build-essential strace ltrace tcpdump`
2. assurez que `make` est installé et que c'est GNU make :

```
$ which make
/usr/bin/make

$ dpkg -S /usr/bin/make
make: /usr/bin/make

$ dpkg -s make | grep GNU
 GNU Make is an utility [ ... ]
```

3. téléchargez l'archive `makefile.tar.gz` du GIT et désarchivez-là, par exemple dans `~/labo-9/makefile`
4. affichez le `Makefile` avec `cat --show-tabs Makefile`
5. avancés : déterminez le jeu de caractères du fichier Makefile avec la commande `file -i` et réencodez en UTF-8 avec la commande `recode` (package `recode`)
6. en fonction de ce que vous voyez et de votre expérimentation, expliquez en quelques phrases ce qui se passe quand on tape `make` (équivalent ici à `make all`) – vous pouvez aussi utiliser l'option `-d` pour voir le debug des décisions de make :

7. grâce à https://www.debian.org/distrib/packages#search_contents déterminez dans quel package se trouvent les commandes `pngtopnm` et `pnmtojpeg`, puis installez-le si nécessaire
8. si vous relancez `make`, pourquoi les fichiers JPEG ne sont pas créés à nouveau ? (indication : appelez la cible `clean` pour repartir de zéro)

9. après avoir observé que `make` ne recrée pas `jura.jpeg` s'il existe déjà et est aussi ou plus récent que `jura.png`, faites `touch jura.png` et relancez `make`, qu'observez-vous et pourquoi ?

10. à quoi sert le `@` en début de commande dans un Makefile ?

11. ce Makefile est suboptimal, car il a des répétitions (D.R.Y. [1]) : vous voyez bien que la cible `neuchatel.jpeg` est très similaire à la cible `jura.jpeg` ; consultez Wikipedia [2] et créez une *règle générique* transformant tout fichier finissant par `.png` en `.jpeg` : il vous faudra employer la variante utilisant les pourcents
12. règles implicites :
    (a) que se passe-t-il si vous créez un fichier `a.c` (p.ex. avec `touch`) puis que vous faites `make a.o` ?

    (b) essayez ensuite d'influencer les arguments du compilateur (variable `CFLAGS` à définir dans le fichier texte `Makefile`, pas avec un paramètre de la commande `make`), essayez par exemple d'ajouter l'option `-Wall` et testez
13. avancés : il est dommage qu'on doive lister les cibles manuellement dans la variable `DEST_IMAGES`, d'ailleurs on y a oublié le Canton de Berne (`berne.png`) : il serait plus intéressant de générer cette liste dynamiquement en fonction des fichiers `*.png` du répertoire : consultez à nouveau Wikipedia (section *Fonctions*) et modifiez la définition de la variable de manière à ce qu'elle appelle le shell pour obtenir la liste des fichiers JPEG à générer en fonction des fichiers PNG existants (indication : testez d'abord vos commandes à la main dans le shell)
14. avancés : consultez la documentation de GNU make [3] pour éviter l'appel au shell ; vous aurez besoin de *wildcards* et de *patsubst*

NB : *conservez votre Makefile complété (fichier ou imprimé) pour le rendu !*

1. don't repeat yourself – une règle de base en programmation : ne vous répétez pas
2. http://fr.wikipedia.org/wiki/GNU_Make
3. http://www.gnu.org/software/make/manual/make.html

Applications réseaux bas niveau : les sockets – 7.3

- interface standard **socket(7)** POSIX
- existe partout, plus ou moins conformément au standard
- variantes
  - BSD sockets : version originale, très répandue
  - POSIX sockets : version standardisée, quasi égale aux sockets BSD
  - Microsoft Winsock : plateforme Microsoft, proche des BSD sockets avec quelques différences, limitations et ajouts spécifiques
- tous les langages offrent une interface plus ou moins directe aux sockets : Java, Perl, PHP, Python, .NET ...

en plus haut niveau :

- bibliothèques implémentant déjà certains protocoles couche 7 (HTTP, FTP, ...)
- Web Services (SOAP, REST, XMLRPC, ...) : appel de procédures distantes
- services distribués (partage de fichiers, cloud de calcul ou de stockage, ...)

44

Microsoft Winsock A part des noms de structures et de fonctions différentes, l'API Microsoft Winsock se distingue notamment par les éléments suivants, qui sont finalement liés à la conception même de l'OS hôte :

- contrairement à POSIX, il n'est pas possible de promouvoir un socket en un fichier `FILE *` pour y utiliser p.ex. les fonctions de la bibliothèque C standard (comme `fprintf(3)`) : les fonctions bas niveau (`send(2)` et `sendto(2)`, `recv(2)` et `recvfrom(2)`) doivent être utilisées
- Microsoft Winsock supporte en plus la notion de socket asynchrone, implémentée par des callbacks : POSIX implémente, lui, des I/O asynchrones de plusieurs types, généralisées à tous les descripteurs de fichiers (socket ou non).
- le multiplexage d'I/O offert par `select(2)` sous POSIX est implémenté de manière suboptimale sous Microsoft : on y recommandera donc l'approche multiprocessus (ou, plus clairement, multithread).
- les raw sockets (qui permettent de générer ou d'analyser du trafic arbitraire en couche 3 ou 2) sont plus limités sous Microsoft Winsock

La note Microsoft suivante donne quelques éléments quant au portage de logiciels BSD sockets vers Microsoft Winsock : https://msdn.microsoft.com/en-us/library/ms740096%28VS.85%29.aspx

Attention, il existe plusieurs implémentations différentes de Microsoft Winsock, par Microsoft et par des tiers, avec des différences notamment sur les protocoles supportés et quelques éléments de détail.

Principes des sockets – 7.4

- les sockets TCP (mode *stream*) sont utilisés en mode caractère (octet) : on y écrit un certain nombre d'octets qui sont lisibles dans le socket partenaire
  - **attention** : à cause du découpage par TCP et du mode *périphérique caractère* d'UNIX, on ne peut pas écrire autant qu'on veut à la fois, et on ne va pas forcément recevoir tout ce que l'on veut à la fois !
  - par défaut les sockets sont bloquants : d'autres modes permettent d'éviter le blocage, comme par exemple :
    - mode non bloquant (`NDELAY`)
    - multi-processus
    - multiplexage (via `select(2)` ou `poll(2)`)
    - asynchronous POSIX I/O
- les sockets UDP ou *raw* (bruts) sont eux utilisés en mode message (*dgram*) avec les méthodes `sendto(2)` et `recvfrom(2)`
- les sockets IP ne sont pas les seuls existants : p.ex. les sockets locaux UNIX

45

Concept de socket ou prise réseau L'abstraction programmatique est basée sur le concept de sockets (RFC-147). Un socket est une prise qui permet à une application d'envoyer et de recevoir des données.

Une connexion TCP est identifiée par 4 valeurs (*nexus* : adresses IP source et destination, ports TCP source et destination). Un socket est identifié par une valeur spécifique au système d'exploitation (sous UNIX : un entier, comme les descripteurs de fichiers usuels que l'on ouvre avec `open(2)`).

TCP est client-serveur : la gestion du socket est différente dans les premiers étapes selon que l'on soit un client (ouverture *active*) ou un serveur (ouverture *passive* en attente de connexion). Une fois la connexion établie, les entrées/sorties sont faites de la même façon (appels systèmes `read(2)` et `write(2)`).

UDP n'a pas de phase d'ouverture de connexion : il permet l'envoi de datagrammes via des appels systèmes différents : `recvfrom(2)`, `sendto(2)`.

La documentation système UNIX (commande man) est fort utile pour décrire l'usage de ces appels systèmes et fonctions. Par exemple :

man 2 socket pour la documentation de l'appel système socket

man 7 socket pour les informations concernant les sockets, en général

Types de socket En plus du *domaine* d'un socket (IPC local UNIX, IPC distant IPv4 ou IPv6 via UDP/TCP, ...), on distingue le *type* : flux (STREAM, p.ex. TCP) ou messages (datagrammes, p.ex. UDP). De plus, en mode flux (TCP), il y a N + 1 sockets côté serveur : un socket en écoute passive qui attend des connexions, et N instances de sockets liés à des connexions déjà actives avec N clients. En effet, chaque instance de connexion est un *nouveau* socket distinct.

Cycle de vie des sockets TCP Le cycle de vie des sockets TCP comporte des états (dans les ellipses) et des transitions (sur les arcs). Les états sont les états internes des sockets (voir aussi l'automate TCP du cours), et les transitions sont sous forme d'appels systèmes (voir la section 2 du manuel UNIX) qui sont utilisés pour créer des sockets, les lier à un port connu, accepter des connexions TCP, échanger des données et fermer la connexion (voir figure 1 *cycle de vie des sockets serveur TCP*). Dans cette figure, les mots en majuscules sont des états internes TCP, et les mots en minuscules des appels systèmes.

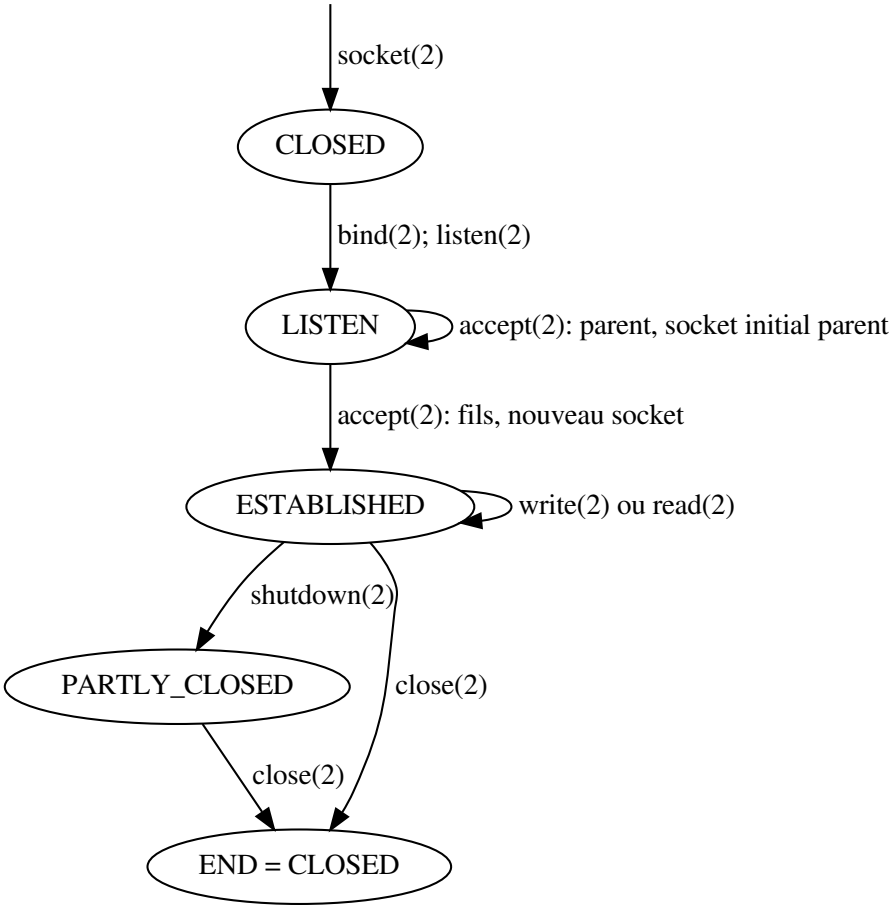


FIGURE 1 – Cycle de vie des sockets : serveur TCP

L'état final `END` correspond en fait à un socket à nouveau fermé (`CLOSED`).

Rappel : dans le cas d'un serveur, chaque nouvelle connexion produit un *nouveau socket* (en plus du socket de base qui permet d'accept(2)er des connexions). Sur le schéma, nous avons supposé un modèle de serveur multiprocessus (le processus père gère le socket de base et les fils gèrent les sockets de connexion ; l'appel système fork(2) permet de dupliquer le père à chaque connexion – voir page 112).

En mode client TCP, après la création du socket, on effectue une connexion TCP [1] (voir figure 2 *cycle de vie des sockets client TCP*).

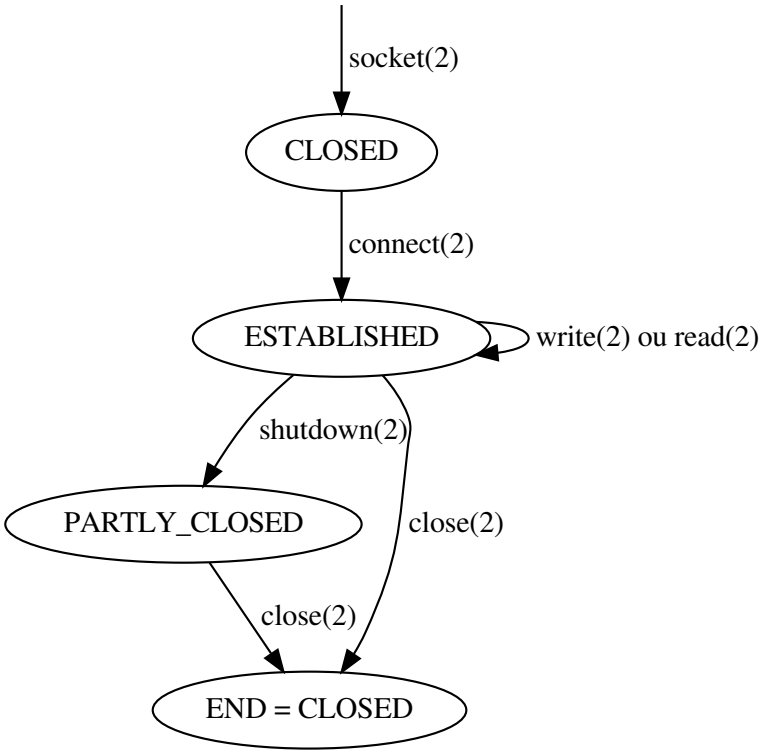


FIGURE 2 – Cycle de vie des sockets : client TCP

Il y a une correspondance entre les cycles de vie des socket et le fameux diagramme d'état du protocole TCP, avec les états représentés ici en majuscules.

En TCP, une fois la connexion établie, les appels systèmes `read(2)` ou `write(2)` peuvent être utilisés, en se rappelant que comme tout *périphérique de type caractère*, on n'arrive pas forcément en une seule fois à écrire tout ce que l'on veut, et qu'en lecture, les appels systèmes retournent dès que des données sont disponibles : cela correspond à devoir utiliser ces appels systèmes ainsi, ce qui est un peu compliqué :

```
/* écrire le buffer de longueur len */
while (len) {
   ssize_t actual = write(socket, buffer, len);
   if (actual > 0) {
      len -= actual;
      buffer += actual;
   }
   else {
      break; /* erreur */
   }
}
```

```
/* lire exactement N octets */
while (len) {
   ssize_t actual = read(socket, buffer, len);
   if (actual > 0) {
      len -= actual;
      buffer += actual;
   }
   else {
      break; /* EOF ssi (actual == 0), ou erreur sinon */
   }
}
```

On peut simplifier le code en utilisant les fonctions de la bibliothèque C standard à la place, ce qui est décrit ci-après.

1. notons qu'il est possible d'utiliser le `bind(2)` avant le `connect(2)` si l'on désire un port spécifique plutôt qu'un port fantôme (aléatoirement alloué) : cela n'est en général pas nécessaire.

Promotion d'un socket en FILE * de la libc Sur une connexion TCP ouverte, en alternative aux entrées/sorties bas niveau avec `write(2)` et `read(2)`, POSIX permet de promouvoir un descripteur de socket (qui est aussi un descripteur de fichier UNIX) en un descripteur de fichier (`FILE *`) de la bibliothèque C standard (libc), à l'aide de la fonction `fdopen(3)` :

```
FILE *f = fdopen(socket, "r+");
```

On pourra alors ensuite utiliser les fonctions habituelles de la bibliothèque C standard (libc) comme (`fwrite(3)`, `fread(3)`, `fprintf(3)`, `fgets(3)`, `fputs(3)`, `fflush(3)`, `fclose(3)`), qui ont notamment l'avantage de simplifier la gestion des envois et réceptions de données : un buffering (tampon interne) est effectué par la libc, qui se charge de recevoir et d'envoyer les données proprement dites, par exemple ligne par ligne :

```
if (fgets(buffer, sizeof(buffer), f)) {
}
else {
   /* EOF ssi feof(), sinon consulter ferror() */
}
```

On peut aussi obtenir le descripteur de socket (numéro de fichier UNIX) avec `fileno(F)`, si F est un FILE *. Il n'est pas recommandé de mélanger les I/Os de bas niveau (read(2), write(2), ...) avec les I/O de la libc (fread(3), fwrite(3), fprintf(3), fgets(3), ...) notamment en raison du buffering offert par la libc.

Traçage et simulation – 7.5

— on vous fournit dans le GIT une application compilée (amd64, voire i386)
— votre but est de déterminer ce que fait cette application, en traçant son exécution
— vous
  — utiliserez les commandes de traçage ci-dessous et les schémas de cycle de vie dès la page 98
  — consulterez les manuels (sections 2 : appels systèmes, 3 : libc, etc)
  — simulerez un, voire plusieurs clients, avec `netcat`
— si pas déjà fait : `sudo apt-get update`

```
sudo apt-get install build-essential strace ltrace
sudo apt-get install tcpdump netcat-traditional
sudo apt-get install net-tools
sudo update-alternatives --display nc (doit être traditional)
sudo apt-get install emacs atril
```

voir exercice en page 103.

46

Commandes de traçage En plus du debugger (`gdb`) qui ne nous sera pas utile ici, citons :

netstat quels sockets existent ? dans quels états sont-ils ?
  — `netstat -an` (voir tous les sockets)
  — `netstat -an --inet` (sockets IPv4)
  — `netstat -an --inet6` (sockets IPv6 : suivant les applications parfois IPv4 aussi)
  — `netstat --inet --listen -n` (sockets en écoute – état LISTEN, IPv4)
  — `netstat --inet6 --listen -n` (sockets en écoute – état LISTEN, IPv6)
  — avec option `-p` (sous `sudo`) : identification du processus lié

strace traçage des appels systèmes – section 2 du manuel UNIX

ltrace traçage des fonctions de la libc – section 3 du manuel UNIX

Wireshark l'outil de capture réseau que l'on ne présente plus et qui sous POSIX fonctionne aussi sur l'interface loopback

Sous Microsoft, on retrouve des outils similaires (p.ex. `netstat -ab`) ainsi que la commande open source `dtrace`, qui y a été portée récemment, voir https://github.com/microsoft/DTrace-on-Windows (dès Microsoft Windows 18980) et qui permet de voir notamment les appels systèmes. Pour son architecture et sa relation aux sous-systèmes spécifiques Microsoft, consulter https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/dtrace.

Rappel : espace utilisateur vs espace kernel Les logiciels tournent en espace utilisateur, et appellent des fonctions (p.ex. de la bibliothèques C standard, section 3 du manuel, comme `printf(3)`). Toutefois, pour certaines opérations, par exemple privilégiées, il faut effectuer des *appel systèmes*, qui s'exécutent en espace kernel.

Exercice : que fait cette application ?

1. téléchargez, décompressez et vérifiez que l'application a la permission d'exécution
2. lancez [1] l'application (`./a-tracer`), laissez ce terminal occupé par la commande
3. sur un autre terminal, déterminez le numéro de processus de la commande que vous venez de lancer (p.ex. avec `ps x`, à piper à `grep a-tracer` par exemple), puis, avec `netstat --inet --listen -n` [2] déterminez quels ports sont en écoute ; vous pouvez (sous `root`) lancer la commande avec l'option `-p` pour déterminer celui que l'application utilise
4. terminez l'application (CTRL–C dans son terminal) et vérifiez que le port n'est plus en écoute
5. lancez l'application mais cette fois comme `strace -f ./a-tracer` [3] et répondez aux questions suivantes :
   — s'agit-il d'un serveur ou d'un client et pourquoi ?
   — quels sont les états du cycle de vie d'un socket qui sont passés ? (voir page 99)
   — quel est l'état actuel du cycle de vie qui est actif ? (indication : nom de l'appel système en cours que montre `strace` ou état du socket selon commande `sudo netstat --inet -anp | grep a-tracer`)
6. à l'aide de netcat [4] connectez-vous en TCP à l'application et répondez aux questions suivantes :
   — que fait l'application au moment où netcat s'y connecte ?
   — l'application est-elle toujours composée d'un seul processus ?
   — quel processus application écoute toujours sur le socket de base, et quel processus application écoute sur la connexion elle-même ? (indication : comme les connexions viennent de la machine locale, `sudo netstat -anp --inet | grep 12345` donnera trois lignes sur ce port ! identifiez ces trois lignes)

1. en cas d'erreur lié à une adresse occupée lors du `bind(2)`, vérifiez que vous n'avez pas lancé l'application par erreur – par exemple en cliquant dessus – et obtenez son numéro de `PROCESSUS` avec `ps x` et tuez-la avec `kill PROCESSUS`
2. `netstat --inet6 --listen -n` pour IPv6
3. vous pouvez spécifier les appels systèmes tracés avec l'option -e (par défaut tous) ; vous pouvez stocker la sortie de strace avec une redirection de la sortie d'erreur ou avec la commande `script`
4. commande : nc localhost PORT

— si vous créez une deuxième connexion, combien de processus `a-tracer` existent maintenant ?
— quel appel système a permis de créer ces processus depuis le processus original ?
— que fait l'option `-f` de `strace` ?
— comment attachez-vous un `strace` à un numéro de processus spécifique ?

7. visualisez le cycle de vie côté client en traçant netcat dès le départ [5]
8. envoyez un datagramme UDP d'un socket à un autre :

```
xterm1% strace -e socket,bind,recvfrom nc -l -p 12345 -u localhost
xterm2% date | strace -e socket,bind,write,connect nc -u localhost 12345
```

(NB : la notion de client/serveur est plus lâche en UDP : dans le cas de protocoles P2P, les N partenaires se lient (bind) tous à un port particulier et communiquent entre eux avec des recvfrom() et des sendto() : dans notre cas ci-dessus, le client fait un connect() et le serveur un bind(), ce qui permet au client de faire un simple write() comme il n'est associé qu'à un serveur UDP via le connect() – un peu un abus vu qu'UDP n'est pas en mode connecté)

9. avancés : quand on termine la connexion TCP, le processus fils associé à cette connexion termine : toutefois, vous observez des processus *zombies* (état de processus Z, processus "defunct") : expliquez quel est le bug dans l'application.

5. `strace -e socket,connect nc ...` pour limiter la quantité de traçage aux deux appels systèmes utiles

Client TCP simple – 7.6

— téléchargez dans votre Linux le PDF du laboratoire [a] et l'archive `simple-client.tar.gz` du Git, désarchivez-la et consultez le `Makefile` et le code source C
— le package `build-essential` contient ce qu'il faut pour compiler (voir aussi la page 102 pour tous les autres outils)
— vous pouvez
  — soit éditer avec n'importe quel éditeur (pluma, vi, nano, emacs) et lancer la compilation en ligne de commande avec `make`
  — soit utiliser Emacs et le menu Tools > Compilation [b] pour compiler via `make`
  — soit utiliser codeblocks : `sudo apt-get install codeblocks` sous Debian buster
— à l'aide des indications ci-dessous et ci-après, créez un client TCP qui puisse se connecter à un serveur local simulé par netcat

47

a. pour pouvoir, depuis le visionneur atril, facilement copier&coller
b. `ESC-x compile`

Recommandations

— consultez systématiquement les manpages (notamment pour savoir quels fichiers d'include utiliser, les types, etc)
— traitez les erreurs (voyez les sections RETURN VALUE et ERRORS des manpages), p.ex. affichez-les avec `perror("ma fonction a plante car");` permet de mieux comprendre ce qui se passe en cas d'erreur.
— compilez avec le Makefile proposé (ou alors n'oubliez pas d'ajouter l'option des warnings -Wall à vos arguments de compilation)

Exercice : Client TCP simple

1. téléchargez et désarchivez le squelette du GIT
2. le but est de compléter ce squelette pour créer un petit client TCP qui se connecte à une adresse IP (donnée sous forme de nom de machine) et un port donnés en ligne de commande, par exemple :

```
$ ./simple-client localhost 7000
```

3. dans la fonction `tcp_connect()`, implémentez, en fonction des exemples qui suivent et des pages de manuel :
   (a) la résolution du nom *hostname* en adresse IP, numéro de port, et protocole (IPv4 ici), via la fonction `getaddrinfo(3)`

   (b) l'affichage des informations d'adresse IP et de numéro de port
   (c) la création d'un socket
   (d) la connexion à un serveur distant avec l'appel système `connect(2)`
4. compilez, corrigez les éventuelles erreurs de syntaxe
5. lancez un serveur TCP netcat sur le port 7000 : `nc -l -p 7000` dans un terminal
6. lancez votre client sur localhost port 7000 dans un autre terminal
7. si cela ne fonctionne pas, vérifiez avec `strace` sur le client si tous les appels systèmes sont appelés avec les bons paramètres
8. expliquez pourquoi l'affichage direct du numéro de port sans passer par `getnameinfo(3)` ne fonctionne pas (voir ci-dessous *La structure générique sockaddr*)
9. le but final est de lire la 1ère ligne envoyée par un serveur :
   (a) ajoutez la conversion du socket en un `FILE *` permettant d'utiliser les fonctions de la bibliothèque C standard comme `fgets(3)`, `fputs(3)`, `fprintf(3)`, ...– voir ci-dessous
   (b) lisez la 1ère ligne envoyée par le serveur dans un tampon avec `fgets(3)` et affichez-le à l'écran avec `puts(3)`
   (c) testez p.ex. avec le serveur `gandalf.teleinf.labinfo.eiaj.ch` port 25 (serveur de mail SMTP) – voir description protocole en figure 3 en page 108.

Résolution d'adresse du service Les deux paramètres en ligne de commande sont *hostname* et *port*. Pour pouvoir se connecter, l'appel système `connect(2)` prend une structure générique de type `struct sockaddr *`. Il faut donc la remplir : c'est le rôle de `getaddrinfo(3)` :

```
int s;
struct addrinfo hints;
struct addrinfo *result, *rp;

hints.ai_family = AF_UNSPEC; /* IPv4 or v6 */
hints.ai_socktype = SOCK_STREAM; /* TCP */
hints.ai_flags = 0;
hints.ai_protocol = 0; /* any protocol */

if ((s = getaddrinfo(hostname, port, &hints, &result))) {
   fprintf(stderr, "getaddrinfo(): failed: %s.\n", gai_strerror(s));
}
else {
   /* getaddrinfo(3) retourne une liste d'adresses, essayons chacune
    * jusqu'à ce que la connexion fonctionne!
    */
   for (rp = result; rp != NULL; rp = rp->ai_next) {
      /* ici on peut afficher les adresses, puis créer le socket
       * et tenter une connexion !
       */
   }
```

```
   freeaddrinfo(result);
}
```

Afficher une adresse générique Pour chaque structure d'adresse `rp` retournée par getaddrinfo(3), on peut afficher en clair les informations comme ci-dessous :

```
char ipname[INET6_ADDRSTRLEN]; /* support IP v6 et v4 */
char servicename[6]; /* "65535\0" */

if (!getnameinfo(rp->ai_addr,
                 rp->ai_addrlen,
                 ipname,
                 sizeof(ipname),
                 servicename,
                 sizeof(servicename),
                 NI_NUMERICHOST|NI_NUMERICSERV)) {
   printf("Trying connection to host %s:%s ...\n",
          ipname,
          servicename);
}
```

Création du socket et connexion Et pour chaque adresse `rp`, tester une connexion :

```
/* socket creation */
if ((s = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol))
    == -1) {
   perror("socket() failed");
}

/* connection */
if (connect(s, rp->ai_addr, rp->ai_addrlen) != -1) {
   ...
}
else {
   perror("connect()");
}
```

La structure générique sockaddr Dans les détails, le type `struct sockaddr` n'est jamais réellement utilisé : en fonction de la famille de socket utilisée (IPv4, IPv6, UNIX domain socket, ...), des structures spécifiques sont utilisées en interne. Par exemple la structure `struct sockaddr_in`. Aujourd'hui, il ne faudrait plus y accéder directement : c'est justement le but des fonctions `getaddrinfo(3)` et `getnameinfo(3)` d'abstraire de ces éléments spécifiques à l'implémentation.

Toutefois, pour voir si vous avez bien compris le cours *couche 6* nous allons ajouter un affichage brut de la valeur du champ *sin_port* dans le cas d'une `struct sockaddr_in` (IPv4) :

```
   if (rp->ai_family == AF_INET) {
      printf("HACK: raw port number: %d\n",
             ((struct sockaddr_in *) rp->ai_addr)->sin_port);
   }
```

Testez et expliquez pourquoi le numéro de port indiqué n'est pas celui passé en paramètre.

Promotion du socket en FILE * La conversion du socket en FILE * de la bibliothèque C standard se fait comme suit :

```
/* associer le descripteur de fichier du socket s à
 * un fichier FILE * f de la bibliothèque C standard,
 * pour simplifier les entrées/sorties.
 */
if ((f = fdopen(s, "r+"))) {
   /* depuis ici, vous pouvez utiliser les fonctions de la libc comme
    * fprintf(3), fgets(3), fputs(3), ... sur f
    */
   ...
}
```

Protocole SMTP Le protocole SMTP – que nous étudierons en couche 7 – précise que le serveur va toujours envoyer une ligne (*banner*) au début de la connection : c'est pratique.

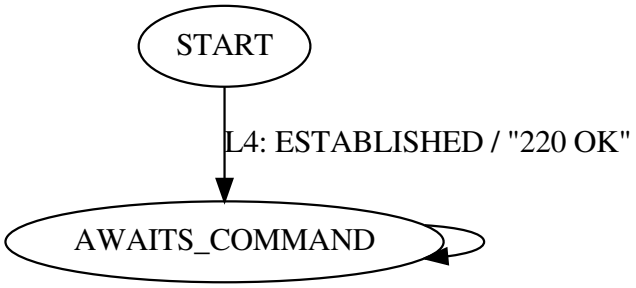


FIGURE 3 – Début du protocole couche 7 SMTP (point de vue du serveur)

Ce schéma se lit comme l'automate TCP : lorsque le serveur reçoit un événement de connexion de couche 4, il envoie la chaîne "220 OK" (et une fin de ligne), puis il attend commande (sous forme d'une ligne) de la part du client.

Informatif : méthode obsolète pour la résolution d'adresse Certaines très anciennes implémentations n'ont peut-être pas les fonctions `getaddrinfo(3)` ou `getnameinfo(3)`. On fera alors un peu différemment : c'est moins générique et moins sûr.

Pour IP version 4, il faut utiliser une structure de type `struct sockaddr_in *` (à convertir ensuite par typecast du C en `struct sockaddr *`) cette structure doit être définie puis remplie comme suit pour se connecter à un port (à définir) de la machine locale 127.0.0.1 :

```
struct sockaddr_in sa; /* the remote TCP server endpoint */

sa.sin_family = AF_INET;
sa.sin_port = htons(port); /* conversion host to network short */
sa.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
```

La fonction `htons(3)` est indispensable car l'ordre des bytes (*endianness*, big endian) du réseau n'est pas forcément la même que pour votre machine (Intel : little endian).

Pour se connecter ailleurs que sur `INADDR_LOOPBACK` (127.0.0.1), on passera par le DNS : il faudra donc convertir le nom de machine passé en argument en adresse IP via la fonction `gethostbyname()` et la structure `hostent`, le remplissage devient alors

```
struct hostent *hp; /* remote host entity pointer */
struct sockaddr_in sa;

sa.sin_family = AF_INET;
sa.sin_port = htons(port);
if ((hp = gethostbyname(hostname))) {
   memcpy(&sa.sin_addr, hp->h_addr, sizeof(sa.sin_addr));
   if (connect(s, (struct sockaddr *) &sa, sizeof(sa)) == 0) {
      ...
   }
}
```

Client HTTP simple – 7.7

— testez le protocole HTTP/0.9 avec netcat en TCP sur `gandalf.teleinf.labinfo.eiaj.ch` port 80 (HTTP)
— implémentez un client HTTP/0.9 simple sur la base de votre client TCP en C
  — connectez-vous à `gandalf.teleinf.labinfo.eiaj.ch` port 80
  — sans attendre une ligne du serveur, envoyez votre demande de document `/` (racine)
  — affichez à l'écran tout ce que dit le serveur jusqu'à la coupure de la connexion
— avancés : faire en protocole HTTP/1.1 plutôt que HTTP/0.9 (voir ci-dessous)

48

HTTP Ce protocole est à la base du Web et nous l'étudierons en couche 7. Il a deux variantes principales : une ancienne (HTTP/0.9) qui est plus simple et une plus moderne (HTTP/1.1), plus flexible. Les automates intégrés sont à la figure 4 en page 111.

Protocole HTTP/0.9 Le protocole HTTP/0.9 est une ancienne variante du protocole actuel (HTTP/1.1), qui a l'avantage d'être très simplifiée, sans supporter de notion de *sites virtuels*. Il vous suffira de vous connecter au serveur HTTP distant, et de demander le document racine `/` comme dans la transcription suivante :

```
$ echo -n -e 'GET /\r\n' | nc gandalf.teleinf.labinfo.eiaj.ch 80
<html><body><h1>Serveur par défaut</h1>

<ul>
   <li><a href="http://gandalf.teleinf.labinfo.eiaj.ch">gandalf</li>
</ul>

</body></html>
```

(le serveur déconnecte immédiatement)

Protocole HTTP/1.1 C'est le protocole HTTP actuel [1], qui supporte la notion de code de résultat, et surtout d'entêtes clients et serveurs, permettant par exemple de différencier entre *sites virtuels* sur la même adresse IP et le même port.

```
$ (echo -ne 'GET / HTTP/1.1\r\n'
   echo -ne 'Host: gandalf.teleinf.labinfo.eiaj.ch\r\n\r\n') \
   | nc gandalf.teleinf.labinfo.eiaj.ch 80
HTTP/1.1 200 OK
Date: Wed, 04 Mar 2015 11:56:17 GMT
Server: Apache/2.2.22 (Debian)
Content-Length: 334
Content-Type: text/html

<html><body><h1>Serveur gandalf</h1>
<p>Ceci est gandalf [157.26.77.13], le serveur de virtualisation
du laboratoire de téléinformatique.</p>

[ ... ]
```

(la connexion ne termine pas tout de suite, sauf si le client a déjà fermé sa direction avec `shutdown(2)` [2], car le serveur attend une commande suivante)

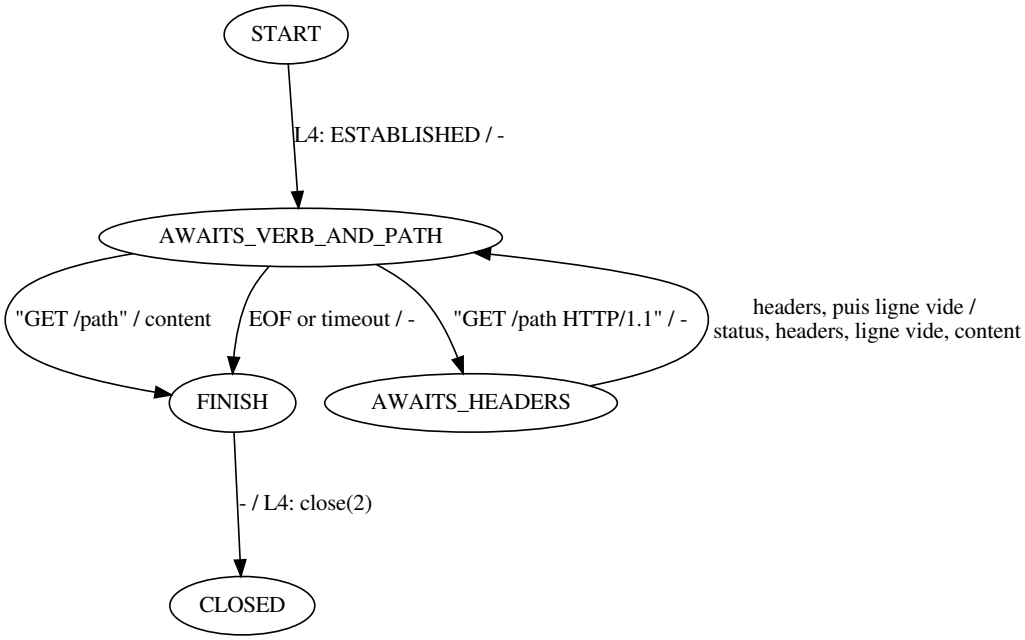


FIGURE 4 – Protocoles couche 7 HTTP/0.9 et HTTP/1.1 (point de vue du serveur)

1. HTTP/2.0 est en test
2. ce qui produira un EOF, *End Of File* côté serveur, voir l'automate en figure 4

Serveur TCP multiprocessus – 7.8

— implémentez un serveur TCP, similairement au logiciel que vous avez tracé (voir page 102)
— ce serveur doit lancer un processus par connexion (voir `fork(2)` ci-dessous)
— utilisez le cycle de vie du serveur (voir figure 1) et/ou les traces que vous avez documentées
49 — de préférence, ce serveur devrait écouter sur un port spécifié en ligne de commande
— avancés : documentez-vous pour éviter la création de zombies à la terminaison des processus fils

voir aussi l'exemple de serveur [a] dans la manpage de `getaddrinfo(3)`.

a. UDP, pas TCP, adaptez !

Serveur multiprocessus Le principe de ce type de serveur est (selon le cycle de vie en figure 1, page 99) que le processus père accepte les connexions, et pour chaque connexion crée un processus fils. Le fils n'a plus besoin du socket que le père utilise comme argument de l'`accept(2)` : il peut donc le fermer immédiatement ; le père n'a plus besoin du socket de connexion retourné par `accept(2)` et qui est uniquement utilisé par ce fils-là, il peut donc le fermer également immédiatement. Dès que cette connexion spécifique se termine, le fils se termine également.

Dupliquer un processus : fork(2)

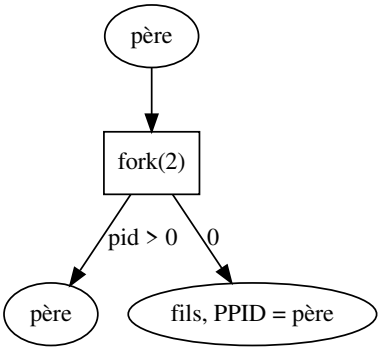


FIGURE 5 – Duplication d'un processus père avec fork(2)

L'appel système `fork` crée un processus fils du processus courant (désormais appelé le père) et duplique l'ensemble de l'adressage : code, constantes, données, pile d'exécution.

Après l'exécution, les deux processus continuent l'exécution au code qui suit le `fork(2)`. Les deux processus sont initialement exactement les mêmes sauf : leur numéro de processus (PID), leur numéro de processus parent (PPID), et le code de retour de l'appel système `fork(2)` (le fils voit 0, le père le numéro de processus du fils) qui permet de définir le rôle père ou fils pour la suite.

Les données et la pile des processus vont ensuite diverger [1] en fonction du code exécuté par les deux processus.

Cette technique est la technique privilégiée de création de processus POSIX [2].

Exemple général de fork

```
#include <unistd.h>
#include <stdio.h>

int main(int argc, char **argv) {
   pid_t pid = fork();

   /* le père et le fils exécutent ce code */
   if (pid == 0) {
      /* seul le fils exécute ce code, à compléter */
      printf("child: %d.\n", getpid());
   }
   else if (pid > 0) {
      /* seul le père exécute ce code, à compléter */
      printf("parent: %d.\n", getpid());
   }
   else {
      /* en cas d'erreur (père seulement: le fils n'est pas créé) */
      perror("fork() failed");
   }

   /* le père et le fils exécutent ce code: ici une simple attente
    * pour bien voir que les processus existent
    */
   usleep(60*1000000);

   return 0;
}
```

exécution :

```
$ gcc -Wall fork.c -o fork
$ ./fork
parent: 10956.
child: 10957.
[ attente d'une minute ]
```

1. en fait, les pages de mémoire virtuelle correspondantes sont initialement partagées, ce qui est économique, toutefois, dès qu'un des processus modifie quelque chose, la page est automatiquement dupliquée et le partage supprimé : c'est le mécanisme *copy-on-write*

2. sous Linux, `fork` est émulé par un appel système plus général, `clone`, qui permet de définir ce qui est partagé entre les deux processus (de l'isolation complète normale au multithread où l'on partage tout sauf la pile d'exécution)

Serveur TCP multiplexé (très avancés !) – 7.9

— il s'agit d'un paradigme de programmation *événementiel*
— plutôt que de lancer un processus par connexion TCP, investiguez la possibilité d'ouvrir tous les sockets de connexion dans votre processus principal
— l'appel système `select(2)` peut être utilisé pour déterminer lesquel(s) des fichiers doivent être traités en I/O ou erreur
50 — il est recommandé de ne pas mélanger select(2) avec les I/O de la bibliothèque C standard sans précautions particulères : utilisez read(2) et write(2)

Multiplexage avec select Le principe est p.ex. de stocker tous les descripteurs de fichiers (ou de sockets) dans un tableau, puis d'informer `select(2)` sur lesquels nous intéressent pour lecture, écriture ou erreur. On appelle `select(2)` avec un délai maximum optionnel (timeout) et l'appel système retourne soit si un ou plusieurs descripteurs peuvent être traités, ou si le délai est dépassé. Il suffit alors de déterminer lesquels sont actifs, les traiter, puis réappeler `select(2)`.

On utilise des champs de bits et des macros spéciales pour préparer les structures de données.

```
fd_t myfds[N]; /* les N connexions ouvertes, qui ont dû être
                * affectées à des sockets de connexion
                * d'une manière ou d'une autre
                */
fd_t maxfd, j;
fd_set readset; /* permettra d'indiquer à select(2) tous les
                 * descripteurs qu'on aimerait pouvoir lire
                 */

// initialisation
FD_ZERO(&readset);
maxfd = 0;
for (j=0; j<N; j++) {
   FD_SET(myfds[j], &readset);
```

```
   maxfd = (maxfd>myfds[j])?maxfd:myfds[j];
}

// ici, pas de délai maximum
                          // read   write err   timeout
result = select(maxfd+1, &readset, NULL, NULL, NULL);
/* retourne en cas d'erreur
 * ou si un des descripteurs peut être lu sans bloquer
 */
if (result == -1) {
   // erreur à traiter
}
else {
   for (j=0; j<N; j++) {
      if (FD_ISSET(myfds[j], &readset)) {
         // myfds[j] est lisible
      }
   }
}
```

Référence : http://developerweb.net/viewtopic.php?id=2933

Questions – 7.10

1. quel est le rôle de shutdown(2) : consultez la manpage et la capture pour expliquer

2. à quoi sert l'option `SO_REUSEADDR` ?

3. à quoi servirait la limitation de l'adresse d'écoute (mettre autre chose qu'`INADDR_ANY` ?)

4. il n'est pas possible de faire écouter le service sur le port 80, en lançant l'application sous votre utilisateur, pourtant aucun service n'écoute par défaut sur ce port : quel est le problème et que faut-il faire pour que cela soit possible ? (indication : aidez-vous éventuellement de `perror(3)` lors du traitement d'erreur du `bind(2)`)

5. avancés : on observe les permissions suivantes :

   ```
   schaefer@reliand:~$ ls -l $(which passwd) /etc/shadow
   -rw-r----- 1 root shadow  2988 May 18 10:08 /etc/shadow
   -rwsr-xr-x 1 root root   51096 May 25  2012 /usr/bin/passwd
   ```

   on sait que la commande `passwd` permet à un *utilisateur normal* de changer son mot de passe et donc de modifier le fichier `/etc/shadow`; expliquez comment elle obtient ces droits spécifiques durant son exécution

6. avancés : on veut lancer le serveur TCP, depuis votre utilisateur, ainsi : `./a-tracer 80` : comment faites-vous pour qu'il ait le droit d'écouter sur le port 80 ? (sans rediriger avec le firewall/NAT-PAT de Linux)

   similairement à la question sur les permissions ci-dessus, proposez une autre méthode ne nécessitant pas une invocation particulière :

pourquoi un programme qui possède ces permissions spéciales les perd lorsqu'il est lancé sous `strace` ou `ltrace` ? (indication : voir la permission spéciale avec `ls -l $(which sudo)` et le message d'erreur de `strace sudo`) :

7. avancés : comparez la variante serveur TCP multiprocessus avec la variante multiplexée.

Checklist – 7.11

base :

je peux créer et expliquer des `Makefile` simples	☐
je peux analyser le fonctionnement d'un logiciel avec les outils de traçage des appels systèmes et du réseau	☐
je peux simuler des clients et serveurs TCP et UDP avec netcat	☐
j'ai pu développer le client HTTP et le tester	☐

normal :

j'ai pu développer le serveur TCP multiprocessus et le tester et le tracer	☐
j'ai répondu aux questions (voir page 116)	☐

avancé :

j'ai pu développer le serveur TCP multiplexé, le tester et le tracer	☐

Evaluation des objectifs par l'enseignant – 7.12

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

8. SNMP, Python & Docker

Objectifs – 8.1

51

— mettre en pratique quelques éléments de langage *Python* vus au semestre de printemps
— exercer la virtualisation sous forme de *conteneurs Docker* (voir ci-dessous)
— appliquer à un cas concret : le protocole SNMP (dès page 121)
  — interrogation simple en ligne de commande avec outils UNIX
  — petite application Web Python interrogeant et visualisant des informations SNMP

dans ce laboratoire, les parties avancées ne sont *pas nécessaires* pour atteindre la note maximale.

Les conteneurs

Bref historique des conteneurs L'appel système UNIX `chroot(2)` de 1979 permet de définir une racine (`/`) par processus, permettant ainsi une première version de confinement du système de fichiers : par exemple le serveur DNS BIND peut utiliser cette fonctionnalité : même une attaque non patchée, exploitée (p.ex. *buffer overflow*), mènera l'attaquant à des droits non-root et sans accès au système de fichiers de la machine. Il pourra toutefois consulter les processus, effectuer un déni de service sur la mémoire, etc. `chroot(2)` est aussi utilisé pour facilement compiler des logiciels avec toutes leurs dépendances ou pour éviter un *dependency hell* avec des logiciels incompatibles. Tous les OS UNIX disposent de cet appel système. Contrairement à la virtualisation classique, l'isolation est faible, mais la performance grande [1]. Debian l'utilise par exemple pour créer automatiquement une instance de système d'exploitation de base, éventuellement d'une version différente du système d'exploitation hôte [2].

Les *Jails* (prisons) sont une évolution du chroot et ont été développées fin des années 1990 dans le monde UNIX/BSD. Leur compartimentage est meilleur que chroot. Le monde propriétaire UNIX a

1. *A performance comparison of linux containers and virtual machines using Docker and KVM*, 2016 : https://link.springer.com/content/pdf/10.1007/s10586-017-1511-2.pdf
2. via la commande `debootstrap`

également développé des solutions comme p.ex. les zones SOLARIS.

Deux mouvements contraires : plus ou moins d'isolation Dans le monde Linux, de nombreuses solutions ont été développées dans les années 2000 pour la containerization, dans le but d'aller vers une meilleure *isolation* (réseau, processus, mémoire, I/O, mémoire virtuelle, liste d'appels systèmes autorisés, etc) : citons p.ex. OpenVZ. Ensuite, les fonctions communes à plusieurs systèmes ont été intégrés dans le kernel sous forme des *cgroups*, ce qui a permis le développement de solutions de haut niveau comme Docker et lxd. Ces solutions gardent la performance de `chroot(2)` en offrant de la gestion (migration de conteneur en *live*, ...) et de la sécurité étendue.

Parallèlement, des solutions plus simples et moins *isolantes* ont été développées, par exemple la commande `fakeroot(1)` qui permet de lancer une commande dans un environnement simulé, par interception (préchargement `LD_PRELOAD`) des fonctions de la bibliothèque C standard : elle permet facilement la compilation, la préparation et le packaging de logiciels avec les bons droits sans devenir `root`.

Comme autre exemple de confinement d'application simplifié, citons Firejail[3], qui, en se basant sur les fonctions du kernel, permet le lancement d'une application dans un contexte de sécurité défini, sans nécessiter la création d'un système de fichiers spécifique.

Sous plateforme Android, le logiciel Knox de SAMSUNG permet une isolation supplémentaire des applications – les applications sont déjà isolées par défaut entre elles comme elles tournent chacune sous un UID différent et SELinux est utilisée comme couche de contrôle d'accès complémentaire.

Sous plateforme MICROSOFT, à part Docker récemment porté en natif, il existe également des solutions légères de *sandboxing* (littéralement : bac à sable à applications) d'application légères, comme p.ex. Sandboxie.

Le panorama des solutions d'isolation ne serait pas complet sans présenter les logiciels de confinement du système de fichiers basés modèles d'accès MAC (*Mandatory Access Control*) comme SELinux ou app-armor, ce dernier étant utilisé par lxc et Docker.

Container cloud et IoT : Thèmes actuels Le thème des containers légers est très actuel, notamment dans le domaine du *cloud*, où des notions comme le déploiement facilité d'applications complexes ou l'exécution de code de tiers de manière sécurisée sont très importants.

Le focus se déplace actuellement des infrastructures (cgroups, lxc, ...) vers les systèmes de *packaging* d'applications comme conteneur, si possibles mutualisées ou communautaires, comme les templates lxc, lxd ou Docker.

Dans la direction du déploiement d'application confinées, que cela soit pour les serveurs, l'IoT[4], ou le desktop, il existe par exemple le concept de *snaps*[5], qui, similairement à Android – mais sans Java – permet d'installer des applications dans leur propre contexte, sans problèmes de compatibilité de bibliothèques ni packaging fort, avec du confinement et des règles de communication entre chaque *snap* et l'environnement.

3. https://firejail.wordpress.com/
4. https://fr.wikipedia.org/wiki/Internet_des_objets
5. https://doc.ubuntu-fr.org/snap#qu_est_ce_qu_un_snap

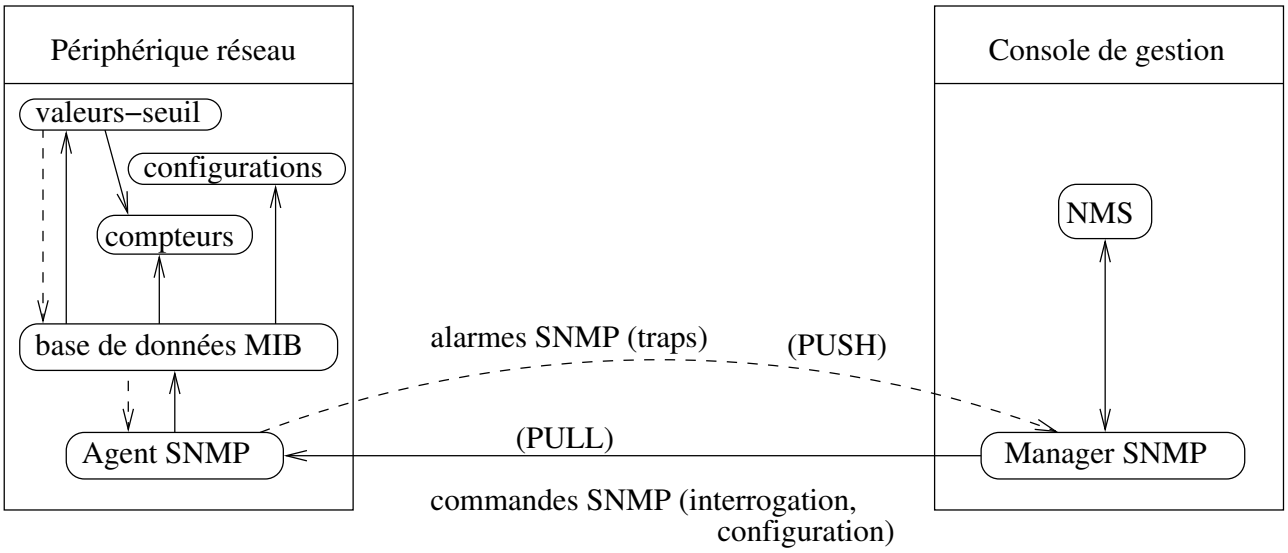
Simple Network Management Protocol

SNMP en quelques mots SNMP (*Simple Network Management Protocol*, RFC-1157) est un protocole de couche application permettant de consulter (surveiller) des états et de modifier des configurations d'équipements réseaux, du routeur à l'imprimante, en passant par les équipements de bus industriels ou des serveurs.

SNMP a été conçu pour être simple et extensible, et largement compatible avec le modèle de gestion d'OSI. Il a été très nettement révisé depuis sa spécification initiale, et la structure de données transportée (MIB, *Management Information Base*) est elle-même très extensible (RFC-1156, RFC-1155).

Toutefois, comme ce protocole est déjà assez ancien, il n'utilise pas des formats modernes textuels (XML p.ex.) mais *présente* les données dans un format binaire structuré télécom classique. Sur le réseau circule la syntaxe de transfert BER (*Basic Encoding Rules*), convertie de et vers la syntaxe locale (adaptée) de chaque plateforme, OS et langage par des *stubs* (routines de conversion), elles-mêmes générées automatiquement depuis une syntaxe abstraite et universelle ASN.1 (*Abstract Syntax Notation One*) [6].

L'architecture de SNMP



On distingue deux partenaires au sein de SNMP :

- les systèmes ou stations de gestion (NMS, *network management stations*)
  - exécutent les applications de gestion qui surveillent et contrôlent des équipements réseaux
  - forment des consoles à travers laquelle les administrateurs peuvent réaliser des tâches d'administration.
  - ce sont des clients (mais voir l'exception des modèle PUSH ci-après)
- les agents (*management agents*) situés sur les équipements réseaux (*network elements*), qui sont interrogés par les NMS ci-dessus (ce sont donc des serveurs, exception : modèle PUSH ci-après) ; exemple : ordinateurs, routeurs, imprimantes, serveurs, ...

Le protocole SNMP définit le protocole de communication entre les stations de gestion et les agents.

6. voir livret A5 Réseaux, section couche 6, figure 6.3 Interaction des syntaxes, page 91

L'architecture du protocole est principalement client-serveur (PULL), avec les agents interrogés par le systèmes de management. Toutefois, des *notifications* peuvent être demandées et délivrées de manière asynchrone par les serveurs vers les clients (PUSH).

La MIB La MIB est une base de données en couche présentation (couche 6 du modèle OSI) qui permet de traduire entre représentation textuelle et codes numériques utilisés dans le protocole. Elle définit les structures de données retournées, voire transmises en fonction du type d'information. Chaque domaine d'application (graphage de trafic réseau, configuration, etc) a sa propre MIB.

Pour parcourir la MIB et les données concrètes associées, on peut utiliser différents outils :

— `snmpwalk` (shell) ou `tkmib` (GUI simpliste) sous UNIX
— Open Eyes sous Java
— getif sous MICROSOFT WINDOWS
— `http://www.oidview.com/mibs/detail.html` en Web

Fonctionnalités de SNMP SNMP offre les fonctionnalités suivantes :

modèle client-serveur (GET)	modèle notification (PUSH) : traps
— consulter des informations (p.ex. compteur de trafic, de pages, configuration d'interfaces, etc) — modifier des informations (configurer) — configurer des conditions d'alerte à notifier de manière asynchrone (*trap*)	— alerter de manière asynchrone (trap) un NMS lorsque des conditions configurées par ce NMS sont réunies

SNMP en pratique L'utilisation la plus fréquente de SNMP est le *graphage* de statistiques réseau, comme par exemple avec l'outil MRTG (ou munin, ou `http://www.zabbix.com/`) avec la base de données sérielle (*timeseries*) RRDtool. Une autre utilisation est la surveillance de systèmes, par exemple avec l'outil Nagios (qui permet à la fois la surveillance active et la surveillance passive/push de traps). Enfin, la configuration et la maintenance d'équipements est possible également par SNMP, avec certaines précautions : toutefois elle se fait souvent avec d'autres outils, ajoutant souvent un contrôle de version, par exemple : Gerty, Ansible (via SSH, y compris pour des machines Linux et Microsoft), FAI ou des solutions maison.

Outils dans différents langages En plus des outils d'interrogation UNIX, il y a des interfaces programmatiques pour différents langages, citons par exemple :

— Perl – package Debian/Ubuntu libnet-snmp-perl (ou libsnmp-perl pour une version en C)
— libsnmp-dev (C)
— Agent++ (C++)
— SNMP4j (Java)

Références

https://www.frameip.com/snmp/ Introduction à SNMP

https://traffic.lan.switch.ch/ Exemple de graphes chez SWITCH

http://www.onlamp.com/pub/a/bsd/2000/07/27/Big_Scary_Daemons.html Walk the SNMP walk

https://makina-corpus.com/blog/metier/2016/initiation-a-snmp-avec-python-pysnmp Initiation à SNMP en Python avec la bibliothèque PySNMP

http://www.reedbushey.com/124Essential%20SNMP%202nd%20Edition.pdf Livre Essential SNMP, O'Reilly

http://irp.nain-t.net/doku.php/215snmp:start Une vision différente du rôle de SNMP

http://archive.oreilly.com/pub/a/perl/excerpts/system-admin-with-perl/twenty-minute-snmp-tutorial.html SNMP en Perl

Premiers pas avec SNMP – 8.2

objectifs

— interroger un agent SNMP avec des outils UNIX (modèle PULL client-serveur classique)

— concepts de base de la MIB (voir page 122)

52

Manipulations et observations

1. dans votre machine GNU/Linux, installez les outils nécessaires [1] :

```
# outils d'interrogation SNMP (Manager SNMP)
sudo apt-get install snmp

# ajouter les sources d'installation non-free (si pas
# déjà présentes dans /etc/apt/sources.list)
sudo sed --in-place 's/contrib$/contrib non-free/g' \
                    /etc/apt/sources.list

# mettre à jour la liste des paquets
sudo apt-get update

# installer les MIBs
sudo apt-get install snmp-mibs-downloader

# activer les MIBs installées
sudo sed --in-place 's/^mibs/#mibs/' /etc/snmp/snmp.conf
```

1. vous trouvez des explications sur les termes utilisés et le fonctionnement général de SNMP dès la page 121 ; pour Debian en particulier voir aussi https://wiki.debian.org/SNMP

2. capturez le trafic avec Wireshark ou `tcpdump` dans votre machine virtuelle (première interface réseau Ethernet de votre VM Linux, voir `ip link show`)
3. expérimentez une interrogation SNMP de l'agent situé sur un serveur du laboratoire de téléinformatique [2] avec, par exemple :

```
snmpget -v 1 -c public 157.26.77.13 system.sysName.0

# essayez aussi sans l'argument system.sysName.0 pour
# voir le `walk' dans la capture réseau Wireshark
snmpwalk -v 1 -c public 157.26.77.13 system.sysName.0

snmpdf -v 1 -c public 157.26.77.13
```

4. questions
   (a) selon le schéma en page 121, identifiez chacun des partenaires (qui est celui qui interroge ? qui est celui qui répond ?)

   (b) quel est le protocole de couche 4 utilisé ? quel est le numéro de port de l'agent, et quel est le numéro de port du manager ?

   (c) avez-vous eu besoin de vous identifier (donner un login) ? de vous authentifier (donner un mot de passe) ? les données sont-elles en clair ? existe-t-il un moyen d'améliorer cela avec un agent compatible ? (indication : `man snmp.conf`, en particulier la section SNMPv3 SETTINGS ; consultez également la capture SNMP)

   (d) quelles sont les principales requêtes et réponses du protocole ?

   (e) dans quels champs de la MIB (quels OIDs – voir page 122) *HOST-RESOURCES* est-ce que la commande `snmpdf` ci-dessus va chercher ? montrez un exemple avec la commande `snmpwalk` indications :
   — analyser par exemple avec `tkmib` (voir page 122), remplacer la cible `localhost` par `157.26.77.13` pour la conversion nom/adresse – les résultats vont dans la zone texte juste en-dessous, qu'il faut peut-être redimensionner

2. accessible en WiFi HEARC-SECURE, filaire ou VPN, aussi avec une VM en NAT

— alternative, ce site : `http://www.oid-info.com/` (même s'il fait des warnings, il décrit en texte les OIDs SNMP)

5. avec un logiciel à choix parmi la liste en page 122, déterminez, en n'oubliant pas de configurer la destination (le nom de l'agent interrogé), ici 157.26.77.13 :
   (a) le numéro d'identification de l'entrée pointant sur le nombre d'interfaces réseaux sur l'agent 157.26.77.13 (indication : sous `iso.org.dod.internet.mgmt.mib-2.interfaces`), puis obtenez les noms de ses interfaces réseau
   (b) les informations de l'objet `iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable .ifEntry.ifInOctets`

Conception d'une application simple Python – 8.3

— but de l'application : faire un petit graphe Web de statistiques d'un équipement réseau via le protocole SNMP
— bibliothèques logicielles python version 3 nécessaires :
  — `PySNMP` – voir page 137
  — `http.server`
53 — `Matplotlib` – voir page 137
— l'application interrogera un équipement par protocole SNMP
— l'application répondra sur le port 80 en HTTP et supportera une seule fonction, retourner un document au format MIME `image/svg+xml` qui sera affiché par le navigateur

— `deliverable` : architecture de l'application envisagée et diagrammes d'interaction avec le client HTTP et l'équipement SNMP et code voir dès chapitre 8.7 (page 135).

Architecture de l'application Faites un petit schéma de comment vous allez concevoir votre application : les composants logiciels, les interactions entre composants.

Diagrammes d'interaction Faites un petit schéma plaçant le client web, le serveur web en python, et l'agent SNMP et décrivez les questions et réponses échangées.

SVG Le format SVG est un format basé XML (un des formats d'échange de données et de stockage de documents standard du Web). C'est un fichier texte, que l'on peut générer manuellement, avec une bibliothèque de *template* ou avec une bibliothèque de plus haut niveau. Beaucoup de logiciels supportent ce format vectoriel, y compris les navigateurs actuels.

D'autres exemples de formats basés XML : le format télécom ASN.1/XER, le format de document *Open Document Format* (ODF) standardisé ISO d'OpenOffice et de LibreOffice, le format de document récent de Microsoft Office, le format XHTML, etc.

Docker – 8.4

— multi-plateforme, open-source ; version commerciale existe (EE)
— permet de décrire des ensembles de services *packagés*, déployables dans un conteneur
— aspect communautaire : conteneurs préparés par d'autres (utilisation voire héritage) [a]
— vers la notion d'*applications encapsulées* et sécurisées dans le cloud

avantages

— par rapport à l'installation classique de logiciels : plus de problèmes de compatibilité lors de déploiement d'applications (versions d'OS, de bibliothèques, etc) ; meilleure sécurité, par confinement
— performance bien meilleure que la virtualisation classique, même sur Microsoft dès Windows 10
— facilité et performance de déploiement et d'auto-configuration, permettant la réutilisation

inconvénients

— sécurité : ne pas utiliser n'importe quel conteneur préparé d'Internet sans vérifications
— ne supporte que des applications utilisant l'ABI [b] kernel Linux

a. p.ex. sur https://registry.hub.docker.com/
b. Application Binary Interface

54

Fonctionnement technique de Docker Sous GNU/Linux, Docker fait usage des fonctionnalités standards kernel de conteneurs (cgroups, namespaces, unionFS, ...) et partage le kernel de base de l'OS avec chacun des containers. Toutefois, comme fonctionnalité supplémentaire par rapport aux solutions classiques de conteneurs sur cette plateforme (lxc, OpenVZ), il ajoute une couche de gestion, créant son propre standard de *packaging* de logiciel – un peu comme si VirtualBox exportait non pas une image disque plus un fichier de configuration XML des éléments simulés (réseau, etc), mais une archive tar.gz des fichiers et un fichier de configuration, y compris l'interaction éventuelle avec d'autres containers.

Sous MICROSOFT WINDOWS et sous MAC OS X, Docker fonctionnait initialement en machine virtuelle VirtualBox GNU/Linux et donc souffre d'une performance comparable à la virtualisation classique d'une seule VM.

Toutefois, MICROSOFT a récemment développé *Windows Subsystem for Linux* (WSL) [1], ce qui améliore les performance sous ce système d'exploitation depuis MICROSOFT WINDOWS 10.0.18917.

1. https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux – il en existe deux versions : WSL1 émule et remplace un kernel Linux, exactement comme WINE remplace l'API MICROSOFT Win32 sous Linux ; WSL2 tourne en arrière-plan un kernel Linux modifié, dans une VM Microsoft HyperV, dans laquelle les conteneurs sont lancés : voir https://www.docker.com/blog/docker-hearts-wsl-2/

Mise en place de Docker – 8.5

— installation de Docker Community Edition : sous votre VM GNU/Linux, ou en VM VirtualBox sous Microsoft ou Mac OS X ; ou encore, expérimentalement en natif avec HyperV sous Microsoft (dès Microsoft Windows 10 Professional avec WSL)

55

— test rapide avec une démonstration Docker

— `deliverable` : Docker est installé et fonctionnel sur votre machine (en VM ou, optionnellement, en natif)

Installation de Docker Community Edition

— recommandé : sous votre machine virtuelle GNU/Linux [1] Debian buster :

1. recommandé : mise à jour avec
   `sudo apt-get update && sudo apt-get -u dist-upgrade`
   `sudo apt-get clean` (faire de la place)
   puis redémarrer la machine virtuelle
2. installation de quelques packages utiles :

```
# transport HTTPS pour apt, certificats racines, logiciel de
# téléchargement curl
sudo apt-get install apt-transport-https ca-certificates \
             curl software-properties-common
# rappel: le backslash signifie qu'il faut tout taper, sans
# backslash, sur une seule ligne.
```

3. installation de la clé de signature de Docker CE :

```
curl -fsSL https://download.docker.com/linux/debian/gpg \
   | sudo apt-key add -
```

1. voir aussi https://store.docker.com/editions/community/docker-ce-server-debian

```
# vérification que c'est la bonne empreinte de clé
sudo apt-key fingerprint 0EBFCD88

# devrait sortir quelque chose qui indique que la clé
# de signature GPG/PGP 0EBFCD88 a l'empreinte
#    9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
```

4. ajout de la source d'installation de *Docker stable* à Debian :

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/debian \
   $(lsb_release -cs) \
   stable"
```

5. mise à jour des listes de packages :
   `sudo apt-get update`
6. installation de Docker CE :
   `sudo apt-get install docker-ce`
7. optionnel : nettoyage des packages téléchargés :
   `sudo apt-get clean`
8. démarrage manuel :
   `sudo service docker start`
9. test rapide : `sudo docker run hello-world`
   si cela fonctionne vous devriez observer le téléchargement de l'image et l'affichage d'un message de bienvenue en anglais ; en option vous pouvez aussi tester le container Ubuntu proposé dans le message [2] : vous serez alors dans un shell root dans un container Ubuntu, vérifiez avec `cat /etc/apt/sources.list` et terminez avec `exit`.
10. vous pouvez supprimer les containers et les images du/des container(s) ci-dessus avec :

```
sudo docker rm $(sudo docker ps -a -q)
sudo docker rmi $(sudo docker images -q)
```

— en alternative : sous Mac OS X ou Microsoft Windows, en *mode VirtualBox*
voir : https://docs.docker.com/toolbox/overview/

— en meilleure alternative : en mode *natif*[3] si votre version de Microsoft Windows 10 Professional est au moins le build 10586, en mode Microsoft Hyper-V et avec la couche de compatibilité ABI Linux (WSL)
voir : https://www.docker.com/docker-windows (activation de WSL nécessaire voir p.ex. https://msdn.microsoft.com/en-us/commandline/wsl/install_guide)

2. avec `sudo` car nous n'avons pas autorisé votre utilisateur à utiliser Docker directement, voir https://docs.docker.com/engine/installation/linux/linux-postinstall/ si cela vous intéresse
3. possible aussi sur Mac OS X avec l'Apple Hypervisor

Déploiement d'une VM Docker – 8.6

— but : déployer une VM Docker avec les outils nécessaires pour un serveur web minimal en Python, le plus automatiquement possible

56

— moyen : le `Dockerfile` proposé ci-dessous permet de construire le container, d'y copier un programme python instanciant un serveur Web exportant les fichiers du container et de l'exécuter en exportant un port TCP vers le host

— deliverable : démo de la VM via Firefox sur votre Linux

Création d'un Dockerfile Créez un répertoire, entrez dedans, puis créez un fichier appelé `Dockerfile` avec ce contenu :

```
# partir d'une image de base Debian
FROM debian

# installer des packages supplémentaires
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update \
    && apt-get -y install python3

# nettoyage des caches de packages
RUN apt-get clean

# installation par copie de notre application dans le template
# (nécessaire de spécifier le chemin complet de la destination)
COPY simple-http-server.py /src/simple-http-server.py

# le port 8001 sera ouvert (et mappé sur le host, probablement
# sur un port dynamique, suivant comment on instanciera la
# machine)
EXPOSE 8001
```

```
# exécuter python3 avec le script indiqué
CMD ["python3", "/src/simple-http-server.py"]

# BUGS
#    - installe pas mal de choses inutiles dans l'image Docker
```

Petit serveur HTTP en Python A créer dans le même répertoire que le `Dockerfile`, sous le nom `simple-http-server.py` :

```
#! /usr/bin/python3

import http.server
import socketserver

port=8001

# cette version exportera tous les fichiers
# du répertoire courant (/ de la VM): pour un
# meilleur contrôle, utiliser pour votre application
#    http.server.BaseHTTPRequestHandler
# (risque?)
handler = http.server.SimpleHTTPRequestHandler

httpd = socketserver.TCPServer(("", port), handler)
print("serving at port", port)
httpd.serve_forever()
```

Création de l'image Docker, lancement d'une instance et test Commandes à taper :

```
# entrer si nécessaire dans le répertoire où vous
# avez créé les deux fichiers Dockerfile et
# simple-http-server.py

# construire une image de base avec les instructions du Dockerfile
# (situé dans le répertoire courant)
# le point à la fin de la ligne représente le répertoire courant
sudo docker build -t my-python-snmp .

# lancer le conteneur en avant-plan: le conteneur python-test
# sera basé sur l'image de base my-python-snmp
#    -rm   supprime le conteneur et son image s'ils existent
#    -P    utilise des ports externes aléatoires
# (la commande sera bloquante tant que le service web python
#  tournera)
sudo docker run --rm -P --name python-test my-python-snmp
```

```
# donne le(s) numero(s) de port(s) aleatoire(s)
# (exécuter dans un autre terminal)
sudo docker port python-test

# tester avec firefox: remplacer le 32771 avec la valeur ci-dessus
# (observer)
firefox http://localhost:32771/

# NOTE: on peut se passer de sudo ci-dessus si l'utilisateur normal
# a les droits docker, on peut les ajouter ainsi:
   sudo groupadd docker # existe éventuellement déjà
   sudo service docker restart
# si l'utilisateur est schaefer (il doit se relogguer après!)
   sudo addgroup schaefer docker
```

Codage et test de la solution – 8.7

— but : coder en Python l'interrogateur SNMP et le visualiseur SVG retourné en HTTP, puis tester

— avancés : test dans le container Docker (non avancés : sur votre Linux normal)

57

— deliverable : code Python, et démo interactive depuis un navigateur

Déroulement proposé (voir ci-après pour des exemples et références)

1. implémenter un petit serveur Web en Python, avec `http.server`, qui retourne soit un fichier (ou un here document) SVG (avec le bon entête `Content-type` pour le SVG soit `image/svg+xml`), soit un fichier (ou un here document) HTML; testez ce petit serveur Web depuis un navigateur – voir page 136
2. implémenter le code Python qui affiche à l'écran un compteur SNMP, lu par le protocole SNMP via la bibliothèque PySNMP (voir page 137) et tester (p.ex. page 126, pour un compteur de trafic d'une interface réseau, testez avec `snmpwalk` puis utilisez le bon identifiant dans votre programme)
3. compléter le point 2 pour que les valeurs lues soient stockées (historisées) dans un fichier texte sous la forme `nombre de secondes depuis une référence ESPACE valeur`
4. créer un script de génération d'un SVG en fonction de l'historique des points, vers un fichier de sortie p.ex. `graph.svg` – voir page 137
5. on supposera que le script d'acquisition des compteurs (qui ajoute au fichier historique) est lancé régulièrement, et que le script qui génère le SVG est également lancé régulièrement (p.ex. avec `cron`, mais cela n'est pas demandé)
6. modifiez votre serveur Web pour que le fichier SVG généré soit retourné (soit fichier statique, soit génération à la demande) et testez avec le navigateur

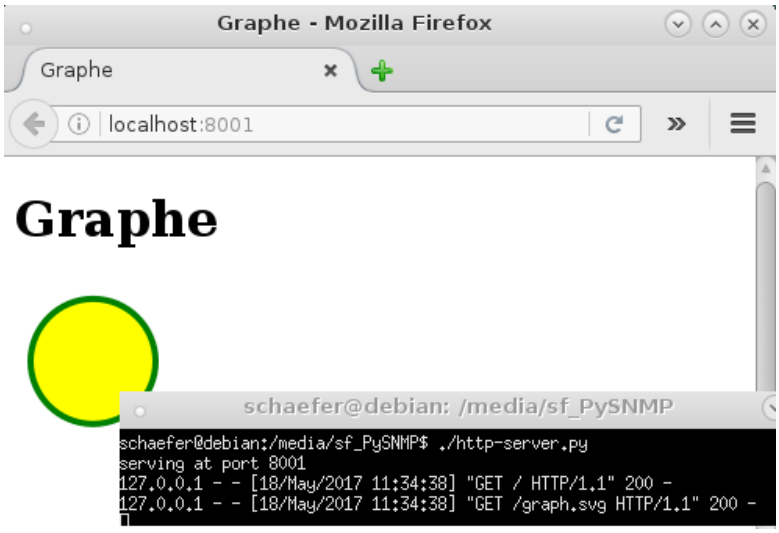
Serveur WWW avec Content-type : paramétrable en Python Dérivez une classe `my_http_server` de `http.server` et redéfinissez la méthode `do_GET` pour configurer une réponse 200 OK ; puis si le chemin est `/graph.svg` retourner un Content-type correct pour le SVG et un contenu SVG de test comme suit :

```
<svg version="1.1"
     baseprofile="full"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xmlns:ev="http://www.w3.org/2001/xml-events">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4"
          fill="yellow" />
</svg>
```

Sinon retourner l'HTML suivant, qui réfère le chemin du SVG ci-dessus :

```
<!DOCTYPE html>
<html>
   <head>
      <title>Graphe</title>
   </head>
   <body>
      <h1>Graphe</h1>
      <p><object data="/graph.svg" type="image/svg+xml">
         </object>
      </p>
   </body>
</html>
```

Le résultat devrait ressembler à :



Afficher un compteur SNMP en Python Après l'installation des packages `python3-pysnmp4 python-pysnmp4-doc python-pysnmp4-apps python-pysnmp4-mibs`, consultez par exemple le fichier `examples/v1arch/asyncore/manager/cmdgen/fetch-scalar-value.py` sous `/usr/share/doc/python-pysnmp4-doc`.

Simplifiez le programme pour faire la requête sur 157.26.77.13, en UDP sur IPv4 seulement. Adaptez le numéro d'objet en vous aidant soit de la MIB, soit p.ex. de `snmpwalk` pour lire un des compteurs d'interface réseau – dans le but que les valeurs changent.

Génération d'un graphe Sur la base d'un fichier d'historisation du compteur du genre :

```
42 500000
43 501000
44 503000
```

créer le code python qui génère un fichier SVG qui représente un graphe X/Y.

Outils envisageables :

Matplotlib voir cours Python : avantage : intégrable dans une application python 3

pychart bibliothèque `svg.charts` (package `python-pychart`, maintenu par Debian, mais bientôt obsolète car python 2.7, voir `http://home.gna.org/pychart/doc/pychart.html` et notamment `http://home.gna.org/pychart/doc/node2.html` et `http://home.gna.org/pychart/doc/module-theme.html`, notamment `output_format` et `output_file`) – ne peut être utilisé que dans une approche multi-programmes : un programme pour générer le graphe en python 2.7, et le serveur web en python 3

Checklist – 8.8

base :

j'ai pu faire des requêtes simples SNMP et analyser le protocole	☐
j'ai conçu l'application selon les directives	☐
j'ai pu installer et tester Docker	☐
je peux déployer une machine Docker spécifique	☐
j'ai codé l'application en Python (mini serveur Web SVG/HTML)	☐
j'ai codé l'application en Python (affichage SNMP simple)	☐
j'ai codé l'application en Python (création SVG du fichier statique)	☐
j'ai testé l'application complète avec un navigateur	☐

avancés : (non nécessaire pour la note maximale)

j'ai pu tester mon application Python avec Docker	☐

Evaluation des objectifs par l'enseignant – 8.9

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

9. Labo Services

Objectifs – 9.1

— déployer un petit sous-réseau IPv4 (avec PCs, routeur, serveur DHCP, DNS, Web, ...)
58
— exercer le déploiement automatisé de services (*deployment industrialisation*)
— instancier 3 fois supplémentaires le sous-réseau (en adaptant les adresses), interconnecter les 4 sous-réseaux à l'aide de routeurs au sein d'un *backbone* et router avec un protocole de routage dynamique
— déployer des services supplémentaires

les parties avancées ne sont *pas nécessaires* pour atteindre la note maximale ; ce labo peut être fait comme proposé avec netkit, ou vous pouvez le réaliser dans ses grandes lignes avec Docker, à choix !

Déploiement de services Plusieurs approches existent :

images ou archives on construit manuellement, ou par scripts, un système de fichiers, puis on archive celui-ci (tar.gz, zip) ou l'on crée une image brute du système de fichiers, d'une partition, d'un disque-dur (`dd`, Ghost, CloneZilla, ...)

scripts à la création on installe automatiquement la machine dans un environnement d'auto-installation et on applique les modifications souhaitées, puis on exploite la machine ainsi créée ou sous forme d'image dupliquée : FAI [1], Debian preseeding, Red Hat Kickstart, etc

scripts à l'exécution sur la base d'une machine déjà créée, on applique des modifications incrémentales : mises à jour, installations, configurations, ... ; par exemple : Ansible [2], système de packaging de l'OS, installeurs d'applications, etc

Ces approches peuvent bien sûr être combinées. L'avantage des méthodes basées images est leur relative facilité de création (on clique un peu partout jusqu'à ce que cela marche, puis on sauve l'image qui, enfin marche). L'approche basée génération par des scripts a un avantage à long terme (contrôle de version, flexibilité, création de sous-systèmes par dépendances, ...) mais elle est plus réfléchie et demande un apprentissage à la fois de la technologie de génération et des logiciels utilisés.

Un exemple d'approche combinée est le système de virtualisation d'applications sous forme de containers Docker : il désarchive des images tar.gz contenant les divers composants logiciels à installer, puis

1. Fully Automatic Installation
2. autoconfiguration via SSH après installation de systèmes GNU/Linux, Microsoft Windows, Cisco, ...

exécute des scripts de configuration spécifiques à chaque composant et à l'ensemble. Les composants logiciels de base auront été pré-créés sous forme d'archives, mais peuvent avoir été préparés par des scripts automatique, par exemple en `chroot`. Exemple : on peut installer un mini-Debian dans un répertoire avec la commande `debootstrap`.

En cas d'absence de documentation sur les formats des fichiers de configuration d'une application donnée, une approche plus manuelle est obligatoire (scripting probablement non adapté).

Dans ce laboratoire, l'on va combiner ces diverses approches (images de base de netkit, génération de configurations de démarrage à partir du premier réseau).

Services Au sein d'un petit sous-réseau, les services suivants sont très souvent déployés :

DHCP adresses IP dynamiques dans une plage, fixage d'adresse en fonction de l'adresse MAC

DNS forward et reverse ; notion de master et slave, forwarder, lien avec le DHCP

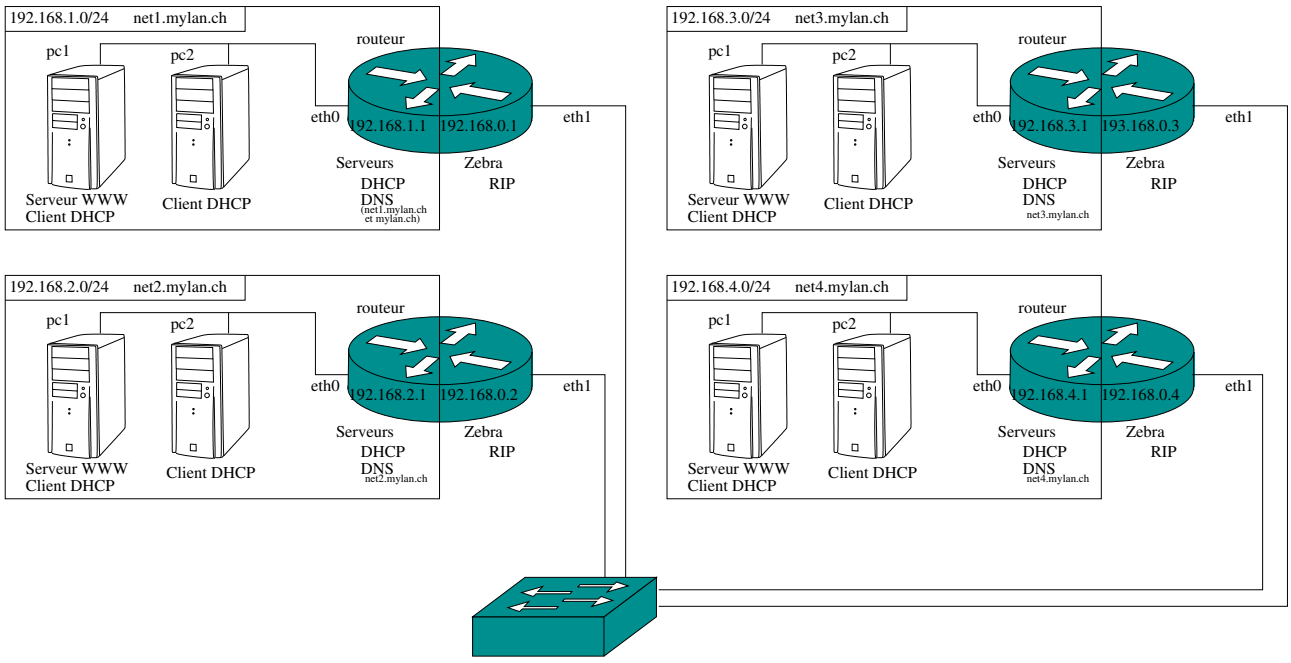
NTP serveur de temps, notion de strate

Web petit serveur Web statique ou avec langages côté serveur

En plus, il n'est pas rare en entreprise de déployer des systèmes de routage dynamique.

Quatre éléments ne seront pas traités ici : NAT/PAT, sécurité (firewall), accès Internet et NTP (en option).

Objectif final Après la première étape (partie I, voir page 141) où l'on va préparer la configuration du premier sous-réseau, la partie II (voir page 152) dupliquera la partie I en quatre sous-réseaux pour les faire inter-agir. Le schéma ci-dessous documente le résultat final.



NB : on travaillera en routage normal, sans NAT, même si l'on utilise des adresses privées.

Partie I : déploiement d'un petit sous-réseau – 9.2

objectifs

— installez le labo comme d'habitude :

```
source ~/NETKIT/SOURCE_ME
netkit-lab services
```

— consultez le schéma en page 140, ainsi que dans le répertoire `~/NETKIT/LABOS/services/services` le README pour le concept, les machines définies et les liaisons configurées dans le `lab.conf` ainsi que les fichiers `*.startup`

59

— effectuez les manipulations et observations ci-dessous

— vous pouvez soit manuellement changer les configuration dans les machines virtuelles netkit (mais perdu au `lclean`), ou modifier le script shell de démarrage du routeur et redémarrer depuis zéro (`lcrash; lclean; lstart -p0`), ce qui est recommandé pour le déploiement de services

— ici, on configurera la zone `net1.mylan.ch` et son *reverse*

Manipulations et observations

1. consultez la documentation sur le DHCP (voir page 142, cherchez *dhcpd* dans `n1-routeur.startup`) et configurez l'ajout dynamique au fichier `/etc/dhcp3/dhcpd.conf` dans le script `n1-routeur.startup`
2. consultez la documentation sur le DNS [1] puis complétez la création des fichiers de zone `net1.mylan.ch.zone` et `1.168.192.in-addr.arpa.zone` dans le script `n1-routeur.startup`
3. testez vos modifications en démarrant le laboratoire avec `lstart -p0` : 3 fenêtres s'ouvrent (deux PC en client DHCP et 1 routeur avec serveur DHCP et DNS)
4. vérifiez que `n1-pc2` a bien pris une adresse IP dans la plage prévue avec `ip addr show`, sinon relancez-y `dhclient eth0` en consultant éventuellement le fichier de log du routeur `tail -f /var/log/daemon.log` – consultez [2] aussi les messages d'erreurs du démarrage dans la fenêtre netkit correspondante
5. vérifiez que le serveur DNS fonctionne depuis `n1-pc2`
   (a) `host -t ns net1.mylan.ch` devrait retourner le serveur DNS de votre petit sous-réseau, ici le routeur `dns.net1.mylan.ch`

1. voir page 144 et notamment les éléments concernant les zones en page 146, ainsi que les fichiers `168.192.in-addr.arpa.zone` (zone inverse) et `mylan.ch.zone` (zone)
2. texte en bleu, utiliser p.ex. la molette de la souris

(b) `host -t a dns.net1.mylan.ch` devrait retourner l'adresse IP 192.168.1.1

6. consultez la documentation sur le lien entre DNS et DHCP (voir page 150) puis implémentez le lien entre serveur DHCP et serveur DNS de manière à ce que si l'on lance `dhclient eth0` sur `n1-pc2`, on puisse ensuite faire :
   (a) `host pc2.net1.mylan.ch` devrait retourner l'adresse IP dynamique de pc2
   (b) `host 192.168.1.101`, en remplaçant 192.168.1.101 par la vraie adresse IP dynamique de pc2, devrait retourner l'adresse textuelle `pc2.net1.mylan.ch` (reverse DNS)
   (c) pour le rapport : le champ `TXT` a aussi été ajouté automatiquement : que contient-il ? de plus, où sont stockées les modifications dynamiques de la zone ?

   en cas de problème, faites par exemple `tail -f /var/log/daemon.log` (ou `syslog`)
7. consultez la documentation sur le DHCP de type DHCP static (voir page 144) et ajoutez dans la configuration DHCP du routeur (dans `/etc/dhcp3/dhcpd.conf`) une configuration permettant d'assigner à la machine `pc1` une adresse IP fixe 192.168.1.42 mais assignée par DHCP (avec son adresse MAC) ; n'oubliez pas également de configurer le reverse (`PTR`) dans la zone reverse ; testez avec la commande `host`
8. configurez une entrée DNS `A` pour `www.net1.mylan.ch` pointant sur l'adresse IP de `pc1`, et comme un serveur Apache est démarré automatiquement sur `n1-pc1` (sinon `/etc/init.d/apache2 start`), testez depuis `n1-pc2` avec `lynx http://www.net1.mylan.ch/`
9. ajoutez à la configuration DNS de la zone `net1.mylan.ch`
   (a) un champ MX de priorité 0 pointant sur `pc1.net1.mylan.ch` et la même chose en priorité 10 sur `pc2.net1.mylan.ch`
   (b) un champ CNAME [3] liant `routeur.net1.mylan.ch` vers `dns.net1.mylan.ch`

DHCP

Introduction

Un serveur DHCP (*Dynamic Host Configuration Protocol*) permet de configurer automatiquement des clients DHCP (ou plus anciennement BOOTP). Cela comprend par exemple :

- l'adresse IP (allouée dynamiquement dans une plage ; allouée statiquement selon des critères comme p.ex. l'adresse MAC couche 2 Ethernet)
- le masque de sous-réseau
- la passerelle ou route par défaut (default gateway)
- le serveur DNS
- des services additionnels
  - serveur de temps
  - image (kernel, système d'exploitation) à charger (démarrage réseau PXE p.ex.)
  - systèmes de fichiers réseau (machines sans disque, diskless)
  - serveur WINS (serveur de nom spécifique partage de fichiers Microsoft)
  - autres.

3. le type d'enregistrement CNAME est assez dangereux, les règles pour l'utiliser sont strictes, voir page 145

Installation (inutile pour netkit)

Sous GNU/Linux :

- installation du package : `sudo -E apt-get install isc-dhcp-server`
- limitation à une interface particulière
  - sudo gedit /etc/default/isc-dhcp-server
  - modifier pour avoir p.ex. INTERFACES="eth1" (remplacer eth1 par l'interface du sous-réseau 192.168.40.0/24, par exemple eth1 ou eth0)
- après un changement de configuration, faire sudo service isc-dhcp-server restart

Sous Cisco, voir
http://www.it-connect.fr/configurer-le-service-dhcp-sur-un-routeur-cisco/

Sous Microsoft Windows, voir par exemple http://www.laboratoire-microsoft.org/articles/network/conf_dhcp_win2k3/

Stratégie d'allocation dans un sous-réseau

Introduction

On suppose que notre sous-réseau 192.168.1.0/24 contient trois types de machines :

- des machines à adresses IP fixe, configurées en dur (p.ex. serveurs, équipements comme imprimantes), sans utiliser le serveur DHCP : **hard static**(véritablement fixes)
- des machines à adresses IP fixe, mais configurées dynamiquement par DHCP (selon l'adresse MAC ; p.ex. serveurs, équipements divers) : **DHCP static** (réservées spécifiquement dans le DHCP)
- des machines à adresses IP dynamiques, allouées dans un *pool* (une plage d'adresse) : **DHCP dynamic** (dynamiques)

(NB : cette classification (et en particulier les noms donnés) n'est pas une norme ni un standard industriel)

Plages dynamiques (type DHCP dynamic)

Prenons par exemple le cas suivant : deux sous-réseaux (probablement sur deux interfaces différentes)

```
subnet 192.168.42.0 netmask 255.255.255.0 {
   range 192.168.42.200 192.168.42.250;
   option routers 192.168.42.8;
   option broadcast-address 192.168.42.255;
   option domain-name-servers  80.83.47.10,  80.83.47.157;
   option domain-name "net1.mylan.ch";
}

subnet 192.168.52.0 netmask 255.255.255.0 {
   range 192.168.52.100 192.168.52.199;
   option routers 192.168.52.1;
```

```
   option broadcast-address 192.168.52.255;
   option domain-name-servers  192.168.52.1;
   option domain-name "net2.mylan.ch";
}
```

Assignation dynamique d'adresses IP basées sur l'adresse MAC (type DHCP static)

Par exemple :

```
host tournedix {
   hardware ethernet           00:C0:4F:78:8E:7F;
   fixed-address                    192.168.1.43;
}
```

DNS

Introduction

Le DNS (Domain Name Service) associe des noms de domaine à des informations, comme par exemple des adresses. Chaque entrée (*record*) possède un type, une durée de validité, et des attributs.

Le DNS est organisé hiérarchiquement :

- racine `.`
- *top level domains*[4] : `ch.`, `fr.`, `uk.`, `de.`, `us.`; `org.`, `net.`, `com.`, `arpa.`
- sous-domaines : `alphanet.ch`, `wikipedia.org`, `he-arc.ch`, `eiaj.ch`
- sous-sous-domaines : `www.alphanet.ch` (feuille), `fr.wikipedia.org`, `labinfo.eiaj.ch`
- sous-sous-sous-domaines : `teleinf.labinfo.eiaj.ch`
- etc

Un sous-domaine peut être *délégué* à un autre serveur DNS (type `NS`). Un système de cache utilisant est utilisé, avec une durée de validité, à chaque niveau. Certains serveurs ne répondent qu'aux questions concernant les données pour lesquels ils sont autoritaires (*authoritative*), d'autres peuvent effectuer des requêtes pour des clients (*recursive*) : un serveur peut se référer à un autre pour ce faire (*forwarder*).

Lorsqu'un serveur (ou un client) ne contient pas l'information demandée (pas dans le cache, pas *authoritative*) et qu'il travaille en mode *recursive* et n'a pas de *forwarder*, il remonte dans l'arborescence jusqu'à éventuellement remonter aux serveurs racine (*root servers*), dont la référence est préconfigurée.

4. de premier niveau, TLD

Redondance du DNS

Un serveur secondaire est comme un serveur principal (maître), autoritaire et donc non distinguable du point de vue du client.

Toutefois, il stocke les informations de zone temporairement en les transférant de temps en temps d'un serveur maître (primaire) par un protocole de transfert de zone (AXFR) ou autre (copie de fichiers p.ex.). Un protocole PUSH (primaire vers secondaire) existe également.

La mise en place d'un ou plus d'un serveur secondaire (sous forme d'entrée NS pour le domaine chez le délégateur, sans aucune différence entre secondaire et primaire) sert à la fiabilité et à la performance.

Un numéro de version est associé à chaque zone pour permettre la synchronisation avec les serveurs secondaires (*secondary*), qui sont autoritaires mais contiennent une copie de la zone provenant du *master*.

Contenu du DNS

Le DNS est un arbre. Chacun des nœuds peut contenir des informations et éventuellement des nœuds fils. Les informations (*record*, enregistrement) sont typées :

type	**description**
SOA	start-of-authority : l'origine du domaine et des paramètres d'expiration (cache, secondaire, etc)
NS	délégation serveur de nom
A	correspondance nom vers adresse IP4
AAAA	idem, IPv6
MX	Mail eXchanger (serveur SMTP)
PTR	correspondance adresse IP vers nom (via pseudo-domaine inversé `in-addr.arpa`)
CNAME	aliases
TXT	informations (clés DKIM ou enregistrements SPF pour l'antispam p.ex.)
NAPTR	informations ENUM (téléphonie E.164)
SRV	localisation de services (p.ex. SIP et XMPP), RFC-2782

Quelques règles spéciales résumées :

— il est possible d'avoir plusieurs entrées d'un type pour un nom donné, dans ce cas, l'idée est de les traiter en *round-robin* (tourniquet) ; sauf pour le MX qui contient une priorité en lui-même (valeur plus basse plus prioritaire)
— un MX ne peut pointer sur un CNAME, en particulier quand cela signifierait de faire une 2e requête DNS
— les firewalls Cisco sont souvent configurés pour interdire systématiquement les CNAME de CNAMEs, en particulier quand cela signifierait de faire une 2e requête DNS : le CNAME est en général à éviter et à remplacer par un champ A
— tous les caractères ne sont pas légaux dans un nom (p.ex. _)
— l'absence de . final peut provoquer l'ajout du domaine par défaut suivant le nombre de . dans le nom.

— une entrée *wildcard* (*) peut correspondre à plusieurs nœuds. Si le nœud existe, le wildcard ne s'applique pas.

Exemple de déclaration de domaines autoritaires

On déclare ici deux zones, dont le contenu se trouve dans les fichiers référencés (et décrits dans les sections suivantes) : une zone pour la résolution normale et une pour la résolution inverse. Cela signifie que notre serveur DNS est autoritaire pour ces deux domaines et est délégué (`NS`) depuis les domaines de plus haut niveau.

```
zone "alphanet.ch" IN {
   type master;
   file "/var/lib/bind/alphanet.ch.zone";
};

zone "72.140.46.in-addr.arpa" IN {
   type master;
   file "/var/lib/bind/72.140.46.in-addr.arpa.zone";
};
```

(ceci se fait dans un des fichiers de configuration de BIND, voir le script `n1-routeur.startup` qui fait cela dans le fichier `named.zone.conf`, pour ce dont on aura besoin dans le laboratoire)

Exemple d'un fichier de zone

```
; nom: alphanet.ch
; "@" représente la zone courante
$TTL 14400
$ORIGIN @
; les 2 champs après SOA sont: le nom du serveur autoritaire et
; l'adresse e-mail pour contacter l'administrateur: les deux
; sont relatifs au domaine courant si pas de "." final
@               IN SOA ns1.alphanet.ch. hostmaster (
                       2014082701      ; serial
                       3H              ; refresh
                       1H              ; retry
                       1W              ; expiry
                       1D )            ; minimum

; on peut soit indiquer le FQDN (avec point final), ou
; un raccourci (le domaine actuel sera ajouté)
@       IN      NS      ns1.alphanet.ch.
@       IN      NS      ns2

ns1 IN A 46.140.72.222
```

```
ns2.alphanet.ch. IN A 159.69.145.25

@       IN      A       46.140.72.222
www     IN      A       46.140.72.222

@       IN      MX      10      smtp.alphanet.ch.
```

Exemple d'un fichier de zone inverse (reverse DNS)

```
; nom: 72.140.46.in-addr.arpa.
$TTL 14400
@       IN      SOA     ns1.alphanet.ch.        hostmaster.alphanet.ch. (
                                2014123002      ; serial (UPDATE!)
                                1H              ; refresh
                                1H              ; retry
                                1W              ; expiry
                                1D)             ; minimum

@       IN      NS      ns1.alphanet.ch.
@       IN      NS      ns2.alphanet.ch.

222     IN      PTR     www.alphanet.ch.
```

Requêtes DNS

Les commandes les plus pratiques sont `host`, `nslookup` et `dig`.

Par exemple :

```
$ host -t ns sunrise.ch.
sunrise.ch name server ns3.sunrise.ch.
sunrise.ch name server ns1.sunrise.ch.
sunrise.ch name server ns4.sunrise.ch.
sunrise.ch name server ns2.sunrise.ch.

$ host -t any ns1.sunrise.ch.
ns1.sunrise.ch has address 212.98.37.132
ns1.sunrise.ch has IPv6 address 2001:1700:a00::2

$ host -t aaaa ns1.sunrise.ch.
ns1.sunrise.ch has IPv6 address 2001:1700:a00::2

$ host 46.140.72.222
222.72.140.46.in-addr.arpa domain name pointer shakotay.alphanet.ch.

# en fait, cette requête est traduite ainsi:
```

```
$ host -t ptr 222.72.140.46.in-addr.arpa
222.72.140.46.in-addr.arpa domain name pointer shakotay.alphanet.ch.

# illustration du délai de péremption du cache (TTL DNS)
$ dig alphanet.ch | grep SOA
alphanet.ch. 896 IN  SOA ns1.ecoweb.ch. dns.ecoweb.ch. 2016032001
    18000 3600 604800 43200
$ dig alphanet.ch | grep SOA
alphanet.ch. 893 IN  SOA ns1.ecoweb.ch. dns.ecoweb.ch. 2016032001
    18000 3600 604800 43200
$ dig alphanet.ch | grep SOA
alphanet.ch. 892 IN  SOA ns1.ecoweb.ch. dns.ecoweb.ch. 2016032001
    18000 3600 604800 43200

# qui est le NS de he-arc.ch., demande à ns1.he-arc.ch
$ host -t ns he-arc.ch. ns1.he-arc.ch
Using domain server:
Name: ns1.he-arc.ch
Address: 157.26.64.6#53
Aliases:

he-arc.ch name server ns1.he-arc.ch.
he-arc.ch name server ns2.he-arc.ch.

# qui est le NS de he-arc.ch., demande à ns1.he-arc.ch
$ dig -t ns he-arc.ch. @ns1.he-arc.ch

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> -t ns he-arc.ch. @ns1.he-arc.ch
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29978
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 4

;; QUESTION SECTION:
;he-arc.ch.                     IN      NS

;; ANSWER SECTION:
he-arc.ch.              1800    IN      NS      ns1.he-arc.ch.
he-arc.ch.              1800    IN      NS      ns2.he-arc.ch.

;; ADDITIONAL SECTION:
ns1.he-arc.ch.          1800    IN      A       157.26.64.6
ns1.he-arc.ch.          1800    IN      AAAA    2001:620:417:f0::1006
ns2.he-arc.ch.          1800    IN      A       157.26.64.66
ns2.he-arc.ch.          1800    IN      AAAA    2001:620:417:f0::1066

;; Query time: 1 msec
;; SERVER: 157.26.64.6#53(157.26.64.6)
```

```
;; WHEN: Tue Apr 26 13:52:37 2016
;; MSG SIZE  rcvd: 151

$ host -t ns .
. name server l.root-servers.net.
. name server c.root-servers.net.
. name server e.root-servers.net.
. name server m.root-servers.net.
. name server j.root-servers.net.
. name server f.root-servers.net.
. name server b.root-servers.net.
. name server g.root-servers.net.
. name server a.root-servers.net.
. name server i.root-servers.net.
. name server d.root-servers.net.
. name server k.root-servers.net.
. name server h.root-servers.net.

$ host -t ns ch.
ch name server h.nic.ch.
ch name server a.nic.ch.
ch name server b.nic.ch.
ch name server d.nic.ch.
ch name server c.nic.ch.
ch name server f.nic.ch.
ch name server e.nic.ch.

$ host -t ns alphanet.ch.
alphanet.ch name server ns2.imp.ch.
alphanet.ch name server ns2.alphanet.ch.
alphanet.ch name server ns1.ecoweb.ch.
alphanet.ch name server ns1.imp.ch.
```

L'outil `nslookup` est obsolète, mais c'est le seul qui est installé par défaut sous Microsoft Windows. On entre dans l'outil en tapant `nslookup`. Ensuite, on peut configurer le type de requête (`set querytype=MX`) puis entrer des requêtes (p.ex. `sunrise.ch`). On peut aussi utiliser les arguments de la ligne de commande.

Sécurité

Le DNS, comme vu précédemment, est

- en clair
- sans signature électronique (pas de garantie d'*intégrité*)
- ne permet pas d'éviter le flooding (de manière à fausser une réponse) : le challenge-response est trop petit
- certains serveurs mal conçus acceptent des informations non reliées à une question (*cache poisoning*)

— un point d'entrée d'attaque

C'est pourtant un des protocoles fondamentaux d'Internet !

Plusieurs pistes à considérer pour la sécurisation du DNS :

— Secure-DNS : signature électronique basée sur une chaîne de confiance SSL/TLS (DNSSEC), avec des champs pour les clés publiques et certificats, garantit l'intégrité des données, par exemple :

```
$ dig +dnssec teleinf.ch
[ ... ]
;; AUTHORITY SECTION:
teleinf.ch.
   871     IN      SOA     test1.teleinf.ch.
      root.teleinf.ch. 2017020201 10800 3600 604800 86400

teleinf.ch.             871     IN      RRSIG
   SOA 8 2 14400 20180202111143 20170202091143 63999 teleinf.ch.
   HqraZGx4HlfpR2nwgp5yvcN7RMTLKkA82SNYg9TDUU+k9u/KSUovuxwH
   mcrHQBnEpovXmevAPIoznVjeqZUozy8PYxvZie6O27mtETnElx5G9Vg2
   UUqzl9PTtO5wBsux/EKbMVZ7MZXfIDFh1IKQcsx0oPXaIKidE9ECPSbA
   YraHqaTFMFDaV6P5nmVW1d+Zo4qbx2JxoiELI7OvElxleB6CwpXs4QSB
   Co5sdOeQzDclSNKO+CUBN6U8gNJQVnwxYb9i6+Mxvkqw0CwuZsOobyuf
   tQEVstxvMoSaFxO4Q85OoA0Ba8xHBBQ4bqBWTr9d7H2Invt9IsTpUMZJ
   dgDoPQ==

teleinf.ch.             871     IN      NSEC
    ipv6.teleinf.ch. NS SOA RRSIG NSEC DNSKEY

teleinf.ch.             871     IN      RRSIG
   NSEC 8 2 86400 20180202111143 20170202091143 63999 teleinf.ch.
   p/WDXvCkWRbB3V21Av6EjiRp7Dwl5PwUADBEMXx4SYmPlFomeqvNGmqQ
   ol9bCTZ1ANJj27VBmH8GanQY6mcvXVKsc65I/cajnzFJAWyTaimpa0E+
   0auuOFkI+33WGkTonZwg/Rdl7iZkw9YQj4SQzM6IDYWG8QnNQD8LJVgE
   Ms7M1PrthyV4vY2rRD88i3Ir1X0GhyWPIkNGvHJL7Lbts0R6cvfI5/M2
   aNCZxz511kxSpCSdJciv2htBwH27rZpYJuB1pDpWiO+swAMn+6Kqdvs6
   kt3ce8jZ9FCBIQN1+nQ1TDxwb8OIa+S6fd06PB4A5TR8L5FZ5mGt7lX1
   Zq5UXQ==
```

— protocole sécurisé d'échange entre maître et esclave possible.
— chroot : isolation de service (confinement)
— clients communiquant avec votre serveur DNS de manière sécurisée, par exemple en DNS sur HTTPS avec https://developers.cloudflare.com/1.1.1.1/dns-over-https/

Lien entre le DHCP et le DNS Dans certains cas, il peut être intéressant de mettre à jour le DNS automatiquement suite à une configuration DHCP. L'idée est d'introduire dans le DNS le nom de la machine communiquée par le client DHCP (ou celle configurée dans le serveur) et l'adresse IP concernée. Cela permet p.ex. de simplifier la gestion des journaux ou de renoncer au protocole Microsoft WINS dans un réseau local pour la résolution de nom.

Voir par exemple http://www.debian-administration.org/articles/343

Pour l'activer, il suffit d'ajouter l'option suivante dans `/etc/dhcp3/dhcpd.conf` :

```
ddns-update-style interim;
```

La modification ci-dessus peut être faite manuellement avec un éditeur de texte, ou mieux, car automatisable, via un `sed` dans le fichier `n1-routeur.startup` comme suit :

```
sed --in-place \
    "s/^ddns-update-style none;/ddns-update-style interim;/" \
    /etc/dhcp3/dhcpd.conf
```

et dans les déclarations de domaine DNS concernés (forward et reverse, fichier `named.zone.conf`), à ajouter dans la création des fichiers dans `n1-routeur.startup` :

```
allow-update { 192.168.1.1; }; // IP of DHCP server
```

(attention : pas dans les fichiers de zone !)

De plus, le client DHCP doit également communiquer son nom (comportement par défaut sous Microsoft, option à configurer dans `/etc/dhcp/dhclient.conf` sous Linux – déjà fait dans notre cas).

Signalons que les changements dynamiques à la zone seront effectués dans un *journal* incrémental et non pas sur le fichier de zone lui-même.

Partie II : déploiement de 4 sous-réseaux – 9.3

objectifs

- en partant de la configuration de base du premier sous-réseau, dupliquez vers les sous-réseaux 2 à 4
- le principe est que les sous-réseaux sont configurés en `192.168.X.0/24`, avec chaque routeur avec deux adresses IP : `192.168.X.1` sur l'interface reliée aux pc1 et pc2 et `192.168.0.X` sur l'interface *backbone*
- les routeurs sont reliés entre eux par le backbone, au sein de leur propre sous-réseau `192.168.0.0/24`
- les routeurs échangeront ensuite les routes et tous les sous-réseaux seront accessibles de partout
- effectuez les manipulations et observations ci-dessous

Manipulations et observations

1. consultez le script `duplicate-subnet.sh` qui sert à copier les fichiers `n1-*.startup` vers les fichiers `nX` correspondants ; comprenez-en le principe et essayez de le lancer
2. observez les répertoires et fichiers startup ainsi créés, ainsi que les modifications au fichier `lab.conf`
3. testez le démarrage du laboratoire : 12 machines netkit devraient être créées
4. testez que chacun des sous-réseaux fonctionne individuellement (ping et host)
5. consultez les informations sur le logiciel Zebra (voir page 153) et adaptez[1] la configuration pour activer zebra et le routage automatique RIP
6. consultez les informations Zebra sur `n3-routeur` ainsi :

```
telnet localhost zebra
# utilisez le mot de passe "zebra" (sans les guillemets)
#
# en cas d'erreur faire:
#    for i in stop start; do /etc/init.d/zebra $i; done
```

1. le plus simple est de modifier `n1-routeur.startup`, de lancer `./clean.sh` et de relancer `duplicate-subnet.sh`

```
# si nécessaire, passez en mode privilégié:
enable

# voir les routes
Router# show ip route
```

vous devriez constater qu'il y a des routes statiques (*directly connected*) et des routes échangée automatiquement avec les autres routeurs

7. testez que les machines du sous-réseau 3 peuvent accéder au sous-réseau 1, y compris la résolution DNS normale et inverse (attention : une machine configurée en DHCP statique ne met pas à jour le DNS : peut-être devriez-vous configurer le DNS en statique)
8. testez un `traceroute` ou un `mtr` à travers le réseau (p.ex. de `n1-pc1` à `n4-pc1`).
9. capturez les échanges RIP sur le *backbone* dans le répertoire général de ce laboratoire, par exemple avec `tcpdump -w /hostlab/capture -s 0 -i eth1 -n`; il peut être utile de faire un `/etc/init.d/zebra restart`; consultez la capture avec `wireshark capture &` sur le Linux (pas la VM netkit) après vous êtes déplacé dans le répertoire du laboratoire services.

Zebra Zebra est un *daemon*[2] qui permet d'implémenter du routage dynamique sous Linux. Il supporte notamment les protocoles RIP et OSPF (*Internal Gateway Protocol*, pour les réseaux d'entreprises) et BGP (*Border Gateway Protocol*, routeur de bord de réseaux) pour l'annonce globale de systèmes autonomes (AS, *Autonomous Systems*) sur Internet.

Pour activer un routeur Zebra dans notre laboratoire, il suffit d'ajouter ce qui suit à `n1-routeur.startup`, et le script de duplication fera le reste pour les autres routeurs :

```
sed --in-place "s/^ripd=no/ripd=yes/" /etc/zebra/daemons

cat > /etc/zebra/ripd.conf <<EOF
hostname $(hostname)
password root
enable password root
router rip
redistribute connected
network 192.168.0.0/24
log file /var/log/zebra/ripd.log
EOF

/etc/init.d/zebra start
```

2. Disk And Execution MONitor

Partie III : autres services – 9.4

faites au moins le 1er point (sinon, toute cette partie est optionnelle)

objectifs

- configurez un *slave* DNS pour la zone `net2.mylan.ch` et son *reverse*
- installez un serveur NTP sur le host Linux (PC labo) et consultez la hiérarchie des serveurs de temps (strates)
- comment synchroniser le temps de manière plus précise que NTP à travers un réseau Ethernet, sans synchronisation en couche physique ?

61

DNS slave Ici, `n3-routeur` sera le *slave* et `n2-routeur` est (déjà) le `master`.

Le but est que la zone net2.mylan.ch soit modifiée pour définir deux NS. Il faut aussi que la zone mylan.ch (gérée par le routeur de net1) délègue le sous-domaine net2.mylan.ch aux *deux* NS situés sur les routeurs des sous-réseaux correspondants.

La commande `host` permet d'interroger un serveur DNS spécifique (ci-après le serveur DNS `XXX`) : testez avec `host -t ns net2.mylan.ch XXX` avec `XXX` valant successivement 192.168.0.2, 192.168.0.3 et 192.168.0.1.

Appliquez également à la zone *reverse* de 192.168.2.0/24.

NB : lorsqu'on modifie une zone, on prend l'habitude de modifier le *serial* de la zone.

NTP Cette partie marche peut-être mieux sur le réseau rouge (ADSL, VLAN 16), en fonction des firewalls de l'Ecole.

Exemple :

```
apt-get install ntp

# vérifiez que le daemon est lancé
ps uw -C ntpd
```

```
# consultez /etc/ntp.conf
# (notamment les serveurs de référence qui y ont été configurés par
#  défaut)

# testez avec un client ntp
# (attendre un peu après le lancement)
ntpdc
# commandes à essayer
#    listpeers (liste des serveurs sources de synchronisation)
#    dmpeers  (p.ex. déterminer le délai estimé, la distance à une
#              horloge de référence (strate), la compensation calculée,
#              etc)
```

Autres idées

- ajouter un NAT et une liaison à Internet sur un des routeurs
- configurer un firewall pour considérer un des sous-réseaux comme une DMZ
- tunnel L2 ou L3 entre machines virtuelles sur VirtualBox ou netkit
- stockage distribué (ipfs, glusterfs, iSCSI, hadoopfs, ...)

Questions – 9.5

1. quel est le rôle du mot clé *authoritative* dans la configuration du serveur *DHCP* (pas DNS !) ?

2. pour quelle raison aurait-on tendance à configurer une durée de bail DHCP courte (p.ex. 1h) ou longue (p.ex. une semaine) ? cas concret : pour un hall d'aéroport avec un /24

3. peut-on imaginer un réseau dans laquelle aucune adresse de type *hard static* n'existe ?

4. quel est l'avantage de configurer des imprimantes, machines virtuelles ou éventuellement serveurs en adresse de type *DHCP static* ?

5. peut-on mettre les plages d'adresses *DHCP dynamic* là où se trouvent des adresses *hard static* ?

6. qu'est-ce que Cisco appelle des *manual bindings* ?

7. voici deux exemples de configurations DNS particulières : pour chacune, expliquez le concept qui se cache derrière (indication : lancez les commandes plusieurs fois pour voir si quelque chose change)

   (a) `host -t a cvs.alphanet.ch.` (à quoi cette configuration sert-elle ?)

(b) `dig -t a shakotay.dyn.alphanet.ch.` (quel est ce cas particulier ? indication : TTL DNS)

8. consultez sur le serveur DHCP le fichier `/var/lib/dhcp/dhcpd.leases` : que contient-il ? qu'en déduisez-vous sur la volatilité des adresses IP dynamiques ?

9. capturez et expliquez ce qu'il se passe avec :
`host big-entry.alphanet.ch` (sur machine réelle, ou `nslookup`) en particulier concernant le(s) protocole(s) de couche 4 utilisé(s) pour la requête

10. expliquez la résolution inverse (adresse IP vers nom), son fonctionnement technique interne

11. comment un serveur de nom trouve-t-il quels serveurs gèrent la racine `.` ?

12. à quoi sert l'option DNS (bind/named) `forwarders` ?

13. vous modifiez une zone sur le master DNS, pour qu'elle se propage correctement sur le(s) slave(s) DNS, que devez-vous absolument changer ?

14. que pensez-vous de la sécurité du protocole DNS ? quelles possibilités existent pour l'améliorer ?

15. vous venez de configurer deux serveurs DNS pour la zone `mondomaine.ch`, et ils sont associés à un *registrar* : expliquez ce que le *registry* [1] doit faire pour que cela fonctionne, techniquement (indication : `whois alphanet.ch` ; de plus, voir les entrées `NS` sur `n1-routeur` de `mylan.ch` vers les sous-domaines)

16. pourquoi est-ce important d'avoir des champs PTR existants et cohérents ?

17. avancés : à quoi servent les champs de type SRV, TXT et NAPTR ? (donnez un exemple pour chacun)

18. avancés : a-t-il été nécessaire de changer le protocole DNS pour supporter les domaines accentués, ou seul une modification du client a suffi ? (indication : *punycode*, exemple : `linux-neuchâtel.ch` dans votre navigateur)

19. avancés : à quoi sert le DNSSEC ?

1. le *registry* gère la base de données du *top domain*, ici `ch.`, ainsi que les installations techniques, comme les serveur de la zone `ch.`, le *registrar* est une entreprise commerciale qui offre une interface d'achat de domaines et transmet des mutations de masse au *registry*

Checklist – 9.6

base :

j'ai pu installer un petit réseau avec netkit	☐
j'ai pu configurer un service DHCP	☐
j'ai pu définir une stratégie d'allocation d'adresses et l'implémenter	☐
j'ai pu configurer un service DNS	☐
j'ai pu configurer le lien entre DHCP et DNS	☐
j'ai pu répondre aux questions	☐

avancés : (non nécessaire pour la note maximale)

j'ai pu déployer automatiquement 4 sous-réseaux	☐
j'ai pu configurer le routage dynamique entre sous-réseaux	☐
j'ai pu répondre aux questions *avancées*	☐

Evaluation des objectifs par l'enseignant – 9.7

dépassés	☐
atteints	☐
proches	☐
non atteints	☐

Recommandations :

10. Où trouver de la documentation

— site officiel Oracle VirtualBox, http://www.virtualbox.org
— manuel d'installation de Debian stable amd64, https://www.debian.org/releases/stable/amd64/index.html.fr
— manuel d'administration de Debian complet : http://debian-handbook.info/
— ressources d'administration Debian : https://wiki.debian.org/fr/SystemAdministration, https://www.debian-administration.org/
— ressources netkit : http://wiki.netkit.org/