Travail en réseau sous Linux

Pierre Senellart

Semaine *Informatique pratique*, septembre 2023

Plan

La pile de protocoles d'Internet

Une pile de protocoles de communications.

Application	HTTP, FTP, SMTP, DNS	
Sécurité	SSL/TLS	(chiffrement, authentication)
Transport	TCP, UDP, ICMP	(fractionnement, fiabilité...)
Réseau	IP (v4, v6)	(routage, adressage)
Lien	Ethernet, 802.11 (ARP)	(adressage local)
Physique	Ethernet, 802.11 (physique)	

IP (Internet Protocol) [IETF(1981a)]

- Addressage de machines et routage sur Internet
- Deux versions du protocole IP : IPv4 (très répandue) et IPv6 (malheureusement encore pas suffisamment répandue)
- IPv4 : adresses de 4 octets affectés à chaque ordinateur, p. ex., 137.194.2.24. Les institutions obtiennent des gammes de telles adresses, qu'elles assignent comme elles le souhaitent.
- Problème : seulement 2^{32} adresses possibles (en fait, beaucoup moins, beaucoup d'entre elles ne peuvent pas être utilisées, pour de nombreuses raisons). Cela signifie que beaucoup d'hôtes connectés à Internet n'ont pas une adresse IPv4 et un système de network address translation (NAT) est utilisé.
- IPv6 : adresses de 16 octets ; espace d'adressage beaucoup plus grand ! Les adresses ont la forme 2001:660:330f:2::18 (ce qui signifie 2001:0660:0330f:0002:0000:0000:0000:0018). D'autres fonctionnalités intéressantes (multicast, autoconfiguration, etc.).

TCP (Transmission Control Protocol) [IETF(1981b)]

- L'un des deux protocoles principal de transport utilisé au-dessus d'IP, avec UDP (User Datagram Protocol)
- Contrairement à UDP, fournit une garantie de transmission des données (acknowledgments)
- Les données sont divisées en de petits datagrammes qui sont envoyés sur le réseau, et possiblement remis dans le bon ordre au point d'arrivée
- Comme UDP, chaque transmission TCP transmission indique un numéro de port (entre 0 et $2^{16} - 1$) pour la source et la destination afin de distinguer d'autres flux de trafic entre les mêmes machines
- Un client utilise un numéro de port aléatoire pour établir une connexion à un numéro de port fixé sur le serveur
- Le numéro de port du serveur identifie conventionnellement le protocole d'application au-dessus de TCP/IP : 22 for SSH, 25 for SMTP, 110 for POP3... voir `/etc/services`

DNS (Domain Name System) [IETF(1999a)]

- Les adresses IPv4 addresses sont difficiles à mémoriser, et un service donné (p. ex., le Web) peut changer d'adresse IP (p. ex., nouvel hébergeur ; voire changement d'IP fréquents en cas de pénurie d'adresse IPv4)
- Encore plus difficile de reternir des adresses IPv6 !
- DNS : un protocole UDP/IP qui associe des noms lisibles (p. ex., `www.google.com`, `weather.yahoo.com`) à des adresses IP
- Nom de domaine hiérarchique : `com` est un domaine de plus haut niveau (TLD), `yahoo.com` un domaine de deuxième niveau, etc.
- Résolution hiérarchique des noms de domaines : serveurs racines avec des adresses IP fixes qui connaissent les serveurs en charge d'un TLD, qui connaissent les serveurs en charge des sous-domaines, etc.
- Rien de magique avec `www.google.com` : juste un sous-domaine de `google.com`.

Internet et le Web

Internet : réseau physique d'ordinateurs (ou hôtes)

World Wide Web, Web, WWW : collection logique de documents

documents hyperliés

- contenu statique et dynamique
- Web public Web et Web privés
- chaque document (ou page Web, ou ressource) identifié par une URL

URL (Uniform Resource Locator) [IETF(1994)]

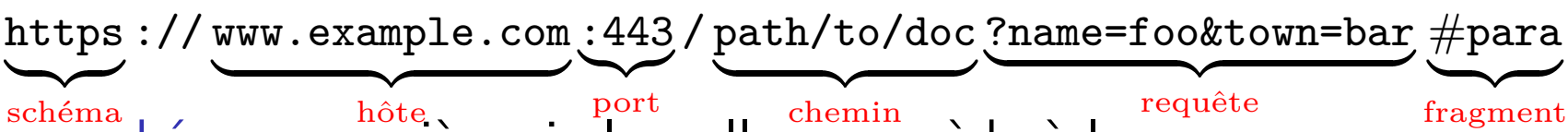

schéma : manière via laquelle on accède à la ressource, généralement http or https

nom d'hôte : nom de domaine de l'hôte (cf. DNS) ; possibilité d'hôtes virtuels.

port : TCP port ; defaults : 80 for `http` and 443 for `https`

chemin : chemin logique de la ressource

requête : paramètres additionnels optionnels

fragment : sous-partie optionnel de la ressource

URL relatives à un contexte (p. ex., l'URL ci-dessus) :

`/titi`	`https://www.example.com/titi`
`tata`	`https://www.example.com/path/to/tata`

HTTP (HyperText Transfer Protocol) [IETF(1999b)]

- Protocole d'application à la base du Web
- Version la plus couramment utilisée : HTTP/1.1 (mais HTTP2 existe, et HTTP3 est quasi-finalisé)

- Requête du client :

```
GET /MarkUp/ HTTP/1.1
Host: www.w3.org
```

- Réponse du serveur :

```
HTTP/1.1 200 OK
...
Content-Type: text/html; charset=utf-8

<!DOCTYPE html ...> ...
```

- Deux méthodes HTTP principales : GET et POST
- Des en-têtes additionnels, dans la requête et la réponse

Plan

Principes de base

- Les connexions réseaux se font via une série d'appels systèmes :

`socket`	pour créer un connecteur réseau
`bind`	pour associer la socket à une interface réseau (serveur)
`listen`	pour écouter sur un port (serveur)
`accept`	pour accepter les connexions (serveur)
`connect`	pour établir une connexion (client)
`select`, `read`, `write`, `poll`	pour lire, écrire, attendre des messages, etc.

- Les sockets ouvertes apparaissent comme des fichiers ouverts dans `/proc`
- Utiliser `strace` pour analyser les connexions faites par un processus !

netstat

- Affiche l'ensemble des sockets ouvertes : connexions en cours et serveurs en attente de connexion
- On peut sélectionner le type de socket :
  - `-t` pour TCP
  - `-u` pour UDP
  - `-x` pour Unix
- On obtient avec `-p` les processus en question
- On peut sélectionner les connexions en cours (par défaut), les serveurs en attente (`-l`) ou toutes (`-a`)

nc et socat

- `nc` : Outil léger et très pratique pour établir des connexions
- `nc -l` *p* : serveur TCP (ou UDP avec `-u`, socket Unix avec `-U`) sur le port *p*
- `nc` *h p* : client TCP (ou UDP, ou Unix) vers l'hôte *h*, port *p*
- Pour des besoins plus avancés (connexions multiples au même serveur, connecter deux serveurs, chiffrement, etc.), on peut utiliser `socat` :
  - `socat TCP4-LISTEN:p -`
  - `socat - TCP:h:p`

Connexion SSL/TLS

- Beaucoup de protocoles modernes utilisent une couche de chiffrement (HTTPS, SMTPS, etc.)
- Pour établir une telle connexion TCP vers hôte *h* port *p* :

```
openssl s_client -crlf -connect h:p
```

`dig` : client DNS très complet

- Permet d'effectuer des requêtes DNS arbitraires
- On peut spécifier un type d'entrée DNS (`-t`), telle que A, AAAA, CNAME, NS, MX
- On peut spécifier à quel serveur DNS on s'adresse avec `@serveur`
- Pour des requêtes plus simple : `host`

Capture du trafic réseau

tcpdump capture l'ensemble du trafic réseau local (pas seulement TCP) – très puissant ! nécessite en général les droits de superutilisateur

wireshark interface graphique permettant d'analyser ces résultats

mitmproxy proxy HTTP(S) pour capturer le trafic réseau entre l'ordinateur local et un site Web, en configurant l'utilisation de ce proxy

au sein d'un navigateur utiliser les extensions développeur, qui indiquent le trafic réseau réalisé par le navigateur

Mais aussi...

`arp -a` pour la table ARP locale

`ip` pour les paramètres réseaux locaux (interfaces réseaux, adresses IP, routage, etc.)

`ping` pour envoyer une requête ICMP echo vers une machine, manière la plus légère de tester son existence (mais elle peut ne pas répondre) et d'estimer la latence de la connexion

`traceroute` pour envoyer de multiples requêtes ICMP avec différentes expirations vers une machine afin d'estimer la topologie du réseau

`whois` pour accéder à la base Whois des informations sur les propriétaires de noms de domaine ou d'adresse IP

`iftop` pour visualiser le trafic réseau en temps réel

Plan

Secure Shell

- Protocole (et outils associés) permettant de se connecter à distance à un ordinateur en mode texte
- La connexion est chiffrée, l'authentification (par mot de passe ou par clef cryptographique) est gérée par SSH
- Une fois la connexion établie, SSH lance le shell de login de l'utilisateur
- Usages : utilisation d'un serveur de calcul, connexion à une machine distante, travail depuis un ordinateur sans installation Linux locale, etc.
- Un serveur SSH doit être installé sur la machine cible

Usage de base

- ssh login@machine pour se connecter à `machine` en tant que `login`
- Le fichier $HOME/.ssh/config est un fichier de configuration dans lequel on peut lister des alias de machines, des noms d'utilisateurs pour ces machines, des options de SSH, etc.
- Pour terminer la connexion, terminer le shell
- Possible de juste lancer une commande sans lancer le shell : `ssh login@machine commande` (ajouter `-t` si besoin d'un pseudo-terminal)
- Si connexion bloquée (problème réseau), interrompre la connexion avec « Entrée + ~ + . »

Clef SSH

- Plutôt que de s'auhtentifier par mot de passe, on peut répondre à un challenge cryptographique prouvant qu'on détient la clef privée associée à une clef publique communiquée au serveur
- Nombreux avantages : pas besoin d'accès par mot de passe sur la machine, possibilité d'avoir une clef toujours accessible ou d'avoir une clef protégée par mot de passe mais qui n'a besoin d'être déverrouillée qu'une fois par session de travail, possibilité de transmettre la clef en passant de serveur à serveur, etc.
- Pour créer une clef, `ssh-keygen` : génére une clef publique (`id_rsa.pub`) et une clef privée (`id_rsa`) dans `$HOME/.ssh/` ; mot de passe de protection de la clef privée choisi à la génération
- La clef publique doit être mise sur le serveur, dans `$HOME/.ssh/authorized_keys` (plusieurs clefs possibles, une ligne par clef)

Agent SSH

- Un agent SSH est un logiciel tournant sur la machine cliente et gardant en mémoire la clef privée déverrouillée
- Souvent installé par défaut dans les distributions modernes ; peut-être lancé à la main comme parent d'un shell avec `ssh-agent zsh` (sera accessible depuis ce shell)
- `ssh-add -l` pour lister les clefs en mémoire
- Clefs automatiquement ajoutées à l'agent à la première connexion si l'option de configuration `AddKeysToAgent` est positionnée à `yes` (dans `$HOME/.ssh/config`)
- Cette clef peut être transmise par l'agent au travers d'une connexion SSH pour être utilisée pour se connecter à une machine C via une machine B depuis une machine A détenant la clef ; option de configuration `ForwardAgent` positionnée à `yes`

Rebond

- Souvent impossible de se cnnecter directement en SSH à la machine C qui nous intéresse (par exemple, derrière un pare-feu) ; une connexion initiale via une machine B est nécessaire
- Ajouter au fichier `$HOME/.ssh/config` :

  ```
  Host C
  ProxyCommand ssh B "/bin/nc %h %p"
  ```

- Alors, `ssh login@C` passera automatiquement par `B` ; si un agent est bien configuré, pas de mot de passe demandé !

Proxy SOCKS

- SSH peut servir à établir un proxy SOCKS, qui permet de faire transiter le trafic (par exemple Web, via la configuration du proxy du navigateur) d'une machine A à une machine W via une machine B (comportement similaire à un VPN)
- Lancer `ssh -D 12345 B` pour démarrer le serveur SOCKS
- Configurer sur la machine locale le navigateur (ou autre logiciel) pour utiliser le proxy SOCKS sur la machine locale (`localhost`) et sur le port 12345
- Le trafic Web sera alors vu comme venant de la machine B !

Affichage graphique

- SSH peut aussi faire en sorte que des programmes graphiques X11 lancés sur la machine distante apparaissent sur la machine locale (on parle d'export X11)
- Lancer avec `ssh -X` ou option de configuration `ForwardX11`
- Peut dépanner, mais assez lent
- Alternative : VNC, protocole de connexion graphique à distance, divers serveurs et clients

Transfert de fichier

- SSH permet aussi le transfert de gros volumes de données, telles que des fichiers
- Diverses variantes et outils : `scp`, `sftp`, etc.
- Recommandé : rsync, un outil de transfert (et synchronisation) d'une hiérarchie de répertoires d'une machine à une autre – utilise SSH (et les alias, la configuration) par défaut
- `rsync -avzP repertoire serveur:chemin` transfère de machine locale au serveur
- `rsync -avzP serveur:chemin repertoire` transfère du serveur à la machine locale
- Options : `-a` transfère récursivement, en préservant quasiment tout ce qu'il est possible de préserver ; `-v` affiche chacun des fichiers transférés ; `-z` compresse à la volée pour gagner du temps ; `-P` permet le transfert en plusieurs fois de gros fichiers

Depuis une machine non Linux

- Clients SSH pour de nombreux systèmes :

  Windows : putty (et WinSCP pour transfert de fichier)

  Android : ConnectBot (et AndFTP pour transfert de fichier) ou Termux (terminal et environnement Linux traditionnel sous Android)

  Mac OS X, autres Unix : ssh en ligne de commande

- Pour l'export X11, il faut un serveur X11, tel que Xming ou VcXsrv pour Windows

Plan

Multiplexeurs de terminaux

- Logiciels tournant au sein d'un terminal, qui sont eux-mêmes des émulateurs de terminaux
- Serveur en arrière plan qui maintient une collection de pseudo-teminaux, avec des processus lancés dans chacun
- Client permettant à tout moment de récupérer l'accès à chacun de chacun de ces terminaux (s'attacher), ou d'en redonner le contrôle au serveur (se détacher)
- Permet en particulier (mais pas seulement !) de lancer des commandes potentiellement interactives dans un terminal et de se déconnecter de la machine sans que ces commandes perdent leur terminal
- Idéal pour du travail à distance
- Plein d'autres fonctionnalités, en particulier gestion de fenêtres multiples

`screen`

- Multiplexeur de terminal historique, présent sur beaucoup de systèmes
- Créer une session : `screen` (on peut lui donner un nom avec `-S`)
- Lister les sessions : `screen -ls`
- S'attacher à une session : `screen -r` suivi du PID ou du nom de la session (avec `-d`, on détache une session éventuellement en cours) session éventuellement en cours ; avec `-x`, on peut s'attacher à une session déjà attachée)
- Se détacher d'une session : CTRL+a d

`tmux`

- Multiplexeur de terminal plus moderne que `screen` avec interface plus agréable
- Créer une session : `tmux new` (on peut lui donner un nom avec `-s`)
- Lister les sessions : `tmux ls`
- S'attacher à une session : `tmux attach -t` suivi du PID ou du nom de la session (avec `-d`, on détache une session éventuellement en cours ; on peut s'attacher à une session déjà attachée)
- Se détacher d'une session : CTRL+b d

Bibliography I


IETF.
Request For Comments 791. Internet Protocol.
http://www.ietf.org/rfc/rfc0791.txt, September 1981a.


IETF.
Request For Comments 793. Transmission Control Protocol.
http://www.ietf.org/rfc/rfc0793.txt, September 1981b.


IETF.
Request For Comments 1738. Uniform Resource Locators (URLs).
http://www.ietf.org/rfc/rfc1738.txt, December 1994.

Bibliography II

IETF.
Request For Comments 1034. Domain names—concepts and facilities.
http://www.ietf.org/rfc/rfc1034.txt, June 1999a.

IETF.
Request For Comments 2616. Hypertext transfer protocol—HTTP/1.1.
http://www.ietf.org/rfc/rfc2616.txt, June 1999b.