TCP/IP

Marc SCHAEFER

5 décembre 2002

Programme

- Motivations de ce cours
- Rappels sur TCP/IP
- Debugging
- Protocole ARP
- DNS
- DHCP Debian
- NTP
- Serveurs (daemons) TCP/IP
- netfilter
- IPv6
- Questions

Motivations de ce cours

TCP/IP est ...

- très répandu
- *ouvert*

il faut :

- connaître ses principes et limites
- savoir le dépanner

TCP/IP est un protocole aujourd'hui très répandu, dans sa version 4, en particulier sur les systèmes UNIX, mais aussi, récemment sur tous les systèmes d'exploitations. Grâce à *l'effet Internet*, presque toutes les applications actuellement développées le sont à l'aide de la suite de protocole ouverte TCP/IP (toutes les spécifications sont dans `/usr/share/doc/doc-rfc/` sur votre machine Debian, package `doc-rfc`).

On peut très bien utiliser journellement TCP/IP sans connaître ses détails d'implémentation : d'ailleurs une connaissance approfondie de TCP/IP pourrait faire l'objet d'un cours entier. Par contre, certains principes de base, orientés pratique, peuvent vraiment être utiles dans la mise en oeuvre de solutions UNIX, Linux ou mixtes Linux/Microsoft Windows.

Dans cette leçon, nous allons présenter les bases de TCP/IP d'un point de vue pratique, quelques outils de diagnostic pratiques comme `netstat`, `tcpdump(1)` et `arp(8)`. Nous allons également utiliser quelques outils vus dans des leçons précédentes et analyser leur comportement avec les outils de diagnostic.

Rappels sur TCP/IP

- projet du DARPA, 1969.
- ARPAnet, puis Internet.
- décrit dans les RFCs (Request For Comment)
- antérieur au modèle OSI (5 contre 7 couches)
- protocole très léger, **best-effort**

Projet initial Initialement un projet du Département de la Défense des Etats-Unis, département des projets avancés, le réseau ARPanet, dès 1969, s'occupe de la définition, de l'implémentation et du test conjoints de protocoles avancés d'interconnexion de réseaux distants. Très vite, les instituts de recherche, universités et centres de recherches commerciaux s'y connectent.

Protocole ouvert et documenté Tout le protocole et certains choix d'implémentation sont décrits dans les RFC (Request For Comments), dont certains sont érigés au rang de standards, ou dégradés en FYI (For Your Information). N'importe quel développeur ou utilisateur final peut soumettre, sous certaines conditions de forme similaire à celles d'un comité d'édition, un nouveau RFC. Aujourd'hui il y a près de 3000 RFCs dont la plupart sont en fait de nouvelles versions de précédents. On y retrouve également des protocoles connexes à TCP/IP, des encapsulations de protocoles et des thèmes avancés comme la sécurité, distribution de clés, etc. Ces standards sont, encore aujourd'hui, gratuitement transférables sur Internet.

Antérieur au modèle OSI TCP/IP ne s'inscrit pas dans le modèle OSI car il est antérieur : cependant, on peut relativement facilement intégrer TCP/IP dans un modèle restreint à 5 couches d'OSI.

Choix initiaux économiques Les choix initiaux de TCP/IP ont permis leur implémentation très répandue, notamment sous UNIX. En effet, l'adressage simple (32 bits d'adresse universelle), le peu de services de base, la simplification à l'extrême du routage, l'intégrité bout-en-bout (TCP) et l'absence de qualité de service (**best-effort**) offraient et offrent encore aujourd'hui une bonne plateforme universelle de compatibilité.

RFC-1, April 1969

```
Network Working Group                          Steve Crocker
Request for Comments: 1                        UCLA
                                               7 April 1969

                    Title:   Host Software
                   Author:   Steve Crocker
                     Installation:   UCLA
                     Date:   7 April 1969
          Network Working Group Request for Comment:   1
```


Rappel sur TCP/IP (2)

- concepts : **datagrammes**, **routage**, **passerelle**, **adresse**, **interconnexion**, **réseau**.
- adresses universelles à 32 bits
- **ICMP** : messages d'erreurs et gestion réseau limitée
- **UDP** : sans connexion ; sans garantie aucune
- **TCP** : orienté connexion ; retransmission et contrôle de flux
- pas d'encryption, signature, authentification, etc.

Services de base offerts TCP/IP offre deux types de services de base à l'usager :

TCP Transmission Control Protocol. Connexion bidirectionnelle fiable avec contrôle de flux, retransmission des paquets entachés d'erreur et réordonnancement.

UDP User Datagram Protocol. Sans connexion. Efficace, mais protocole laissé à la charge de l'implanteur.

à ces deux protocoles s'ajoute **ICMP**, Internet Control Message Protocol, qui permet de communiquer des erreurs ou des informations de routage/congestion entre deux piles IP.

Rappel sur TCP/IP (3)

classes

classe	début	plage	description
A	0	0.0.0.0 - 127.255.255.255	7 bits réseau / 24 bits hôte
B	10	128.0.0.0 - 191.255.255.255	14 bits réseau / 16 bits hôte
C	110	192.0.0.0 - 223.255.255.255	21 bits réseau / 8 bits hôte
D	1110	224.0.0.0 - 239.255.255.255	Multicast
E	1111	240.0.0.0 - 255.255.255.255	Réservé

notation de masque (subnetting, netmask)

plage	netmask	bits
10.0.0.0 - 10.255.255.255	/255.0.0.0	/8
192.168.4.56 - 192.168.4.59	/255.255.255.252	/30

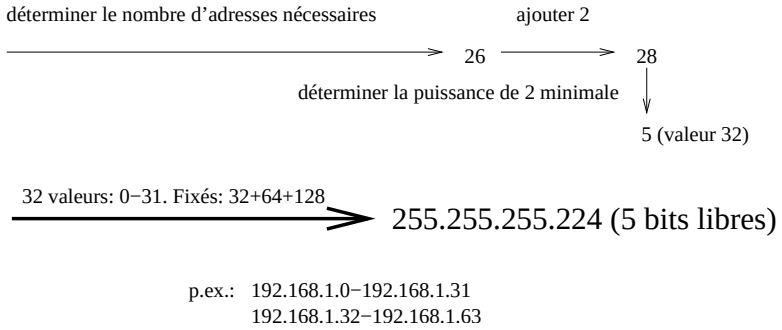
Rappel sur TCP/IP (4)

zones spéciales, RFC-1918

plage	description
127.0.0.0 - 127.255.255.255	loopback local
10.0.0.0 - **10**.255.255.255	Classes A réseau privé (/8)
172.16.0.0 - **172.31**.255.255	Classes B réseau privé (/16)
192.168.0.0 - **192.168.255**.255	Classes C réseau privé (/24)

Rappel sur TCP/IP (5)

calcul *subnetting*



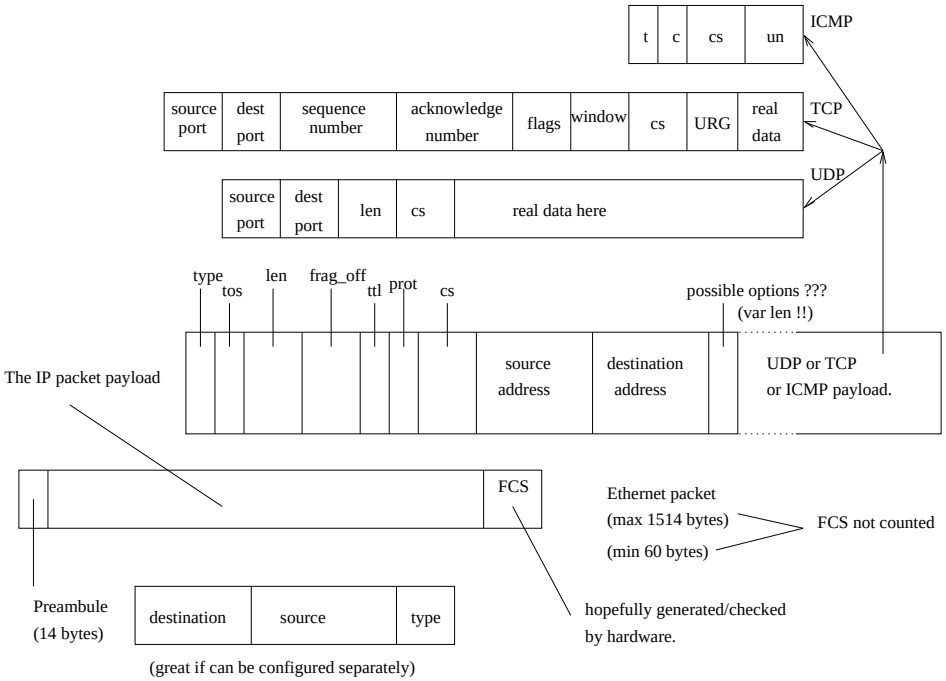
Méthode plus simple que ce calcul de bits, en particulier pour des réseaux plus petits (en plage) que `/24` (dernier octet varie partiellement) : soustraire la taille du subnet à 256. Par exemple, dans notre cas : $256 - 32 == 224$.

Exercice : calcul des paramètres

Problème : pour un sous-réseau de 6 machines (p.ex. un DMZ), dont l'expansion n'est pas prévue choisir un sous-réseau dans une plage privée, donner :

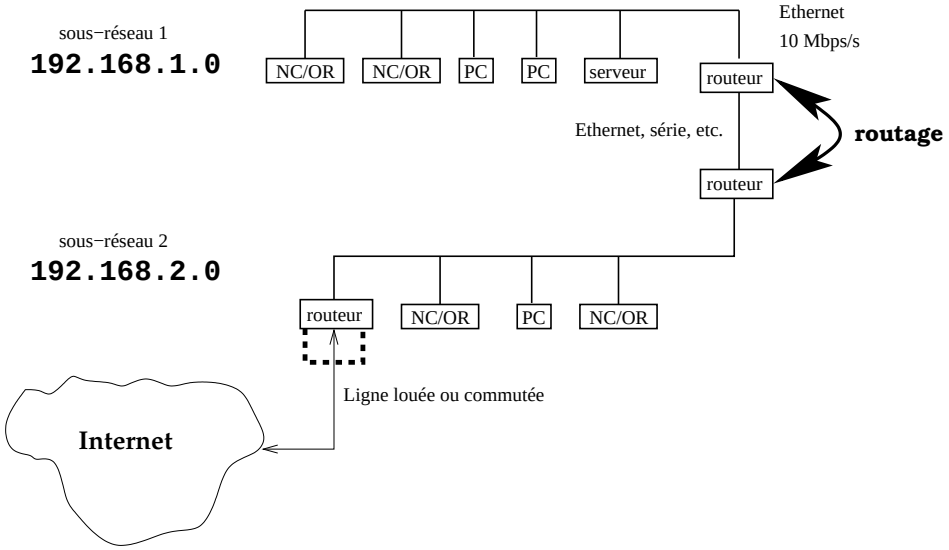
- la plage d'adresse utile
- le *netmask*
- la plage effective
- l'adresse de réseau
- l'adresse de *broadcast*
- éventuellement les adresses inutilisées dans la plage
- motiver les choix

Opérations offertes	Couches	Concepts offerts	Fonctionnalités
− connexion interactive, transfert de fichier, consultation de systèmes hypertextes, messagerie − gestion réseau	Application TELNET, FTP, HTTP, SMTP, NNTP DNS, SNMP, NTP	interaction utilisateur, concepts de haut niveau (fichier, objet multmédia)	définies par l'implanteur
read/write, close	Système d'exploitation	flots de données	uniformiser les accès aux périphériques/fichiers
socket, bind, listen, accept, sendto, recvfrom, shutdown		socket, port, protocole, datagramme	uniformiser les différents types de réseaux (TCP/IP, OSI, ...)
transmettre un message ou une suite de messages sur le réseau, choix parmi différentes qualité de service (TCP/UDP) à la connexion.	Protocoles (transport, gestion) TCP (connecté) / UDP (non connecté) / ICMP	TCP: flot bidirectionnel UDP: datagrammes TCP/UDP: ports ICMP: messages réseau	TCP: segmentation, validation des données et de l'ordre, bidirectionnel fiable. UDP: datagrammes, non fiable protocole défini par l'implanteur
amener à destination un paquet de données (avec gestion des messages d'erreur du réseau). Pas d'assurance de transmission effective.	Réseau IP	adresse IP, TTL	validation de header routage
envoyer une suite de bits sur un support physique Suivant l'implémentation: − détection de collisions − détection/correction d'erreurs − compression 'on the fly'	"Physique" Ethernet, FDDI, ATM, série, HAM, X25, ISDN, lignes louées/commutées, ...	routines d'émission et de réception de datagrammes IP	pilote de périphérique, encapsulation



Routage

- de proche en proche
- table de routage IP
- différence routeur/switch
- commandes UNIX :
  - consulter : `netstat -rn`
  - modifier : `route`



Routeur/switch Un routeur se charge d'acheminer des paquets IP depuis un réseau donné jusqu'à un autre réseau ou un autre routeur, en examinant les entêtes IP (adresse source et destination). Les broadcasts ne traversent pas les routeurs. Un routeur peut travailler avec des connexions de types différents.
Un switch travaille au niveau Ethernet : c'est une espèce de répéteur, mais qui travaille au niveau Ethernet (MAC) plutôt qu'électrique. il transmet les broadcasts. Parfois il peut éviter d'envoyer les paquets non broadcast sur des interfaces qui ne contiennent pas la destination Ethernet (MAC) d'un paquet pour optimisation, ce qui est une fonctionnalité (performance, sécurité) supplémentaire à l'isolation bas niveau Ethernet/électrique.

Rôle du netmask Le netmask spécifie les adresses à considérer comme locales sur l'interface considérée. Par exemple, `192.168.1.56/30` signifie que tout paquet pour la plage 192.168.1.56-192.168.1.59 doit être envoyée sur cette interface. Le reste doit être envoyé via d'autres règles de routage, par la route par défaut ou rejeté.

Exemples de commandes de routage Le routage est en général simple : dès le moment où l'on a plusieurs interfaces, ou plusieurs sous-réseaux, cela peut se com-

pliquer.
– exemples :
  – `route add default gw 192.168.1.1`
  – `route add default dev eth0`
  – `route add -net 192.168.1.56 netmask 255.255.255.252`
– Ne pas oublier, si la machine doit faire du *gateweying* (routage de trafic entre interfaces) `echo > /proc/sys/net/ipv4/ip_forward 1`

Problèmes et futur de TCP/IP

- sécurité des données
- pas d'encryption, signature, authentification, etc.
- QoS, bandwidth reservation
- 'bientôt' (?) IPng
- solutions en attendant

Sécurité de données avec TCP On n'insistera jamais assez que l'intégrité de données offerte par **TCP**, le 'meilleur' service IP dans ce domaine, est assez basique. Il s'agit en effet d'un simple checksum. En raison du fait qu'en général, les couches inférieures offrent des protections (CRC-32 en Ethernet, ou en PPP, mais rien en SLIP par exemple), ce n'est pas si important. Plus ennuyeux sont l'absence de chiffrement des données (n'importe qui sur le parcours du paquet peut le consulter, voire dans certains cas altérer, ou s'insérer dans, les données), l'absence de signatures électroniques rendant impossible la vérification de provenance et la non-répudiabilité, etc. On le montrera grâce à l'outil `tcpdump`.

Nouvelle version future TCP/IP possède encore de nombreux autres défauts, que l'on corrige au coup-par-coup, ou qui seront corrigés dans la version suivante, actuellement en test, **IPv6** ou IPng, qui sera bientôt utilisé largement (je disais déjà cela en 1995 ... l'inertie, en informatique, n'est pas à négliger).

Solutions pour aujourd'hui Tous ces défauts peuvent (doivent !) être corrigés au niveau application : `ssh`, **SSL**, `md5sum`, **GPG** sont des protocoles ou outils qui peuvent être déployés dans ce sens. **IPSEC** est une solution interopérable de chiffrement et

authentification de paquets IP au plus bas niveau. Ces protocoles sont, comme auparavant, décrits dans les **RFC**.

Exercice : RFCs

Problème : dans quels RFCs sont décrits les protocoles IPSEC ? A quoi cela peut-il servir ?

Debugging

- `strace`, p.ex. `strace -e connect`
- messages du kernel, logs, compteurs
- `/proc`
- outils : `ping`, `traceroute`, `ifconfig`
- connexions ouvertes et serveurs : `netstat -anp`
- capture de paquets : `tcpdump`
- analyse de paquets, génération de paquets, graphes de traffic

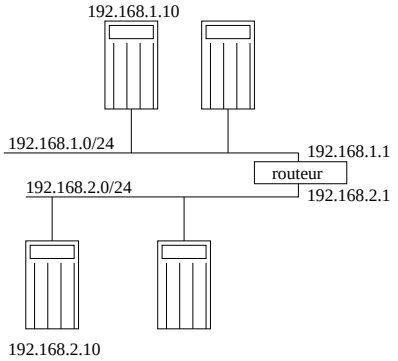
Debian Un paquet Debian très intéressant : `netdiag`, qui comprend de nombreuses commandes comme trafshow, strobe, netwatch, statnet, tcpspray et tcpblast. Enfin, `iptraf` contient un programme de statistiques réseau assez performant.

tcpdump

Outil de trace, principalement au niveau **IP**.

```
defian:/home/schaefer# tcpdump -i eth0 -n -S -t
tcpdump: listening on eth0
193.72.186.8.1517 > 193.72.186.6.80: S 211947184:211947184(0)
   win 32120 <mss 1460,sackOK,timestamp 3538752 0,nop,wscale 0>
   (DF) [tos 0x10]
193.72.186.6.80 > 193.72.186.8.1517: S 3731483051:3731483051(0)
   ack 211947185 win 32736 <mss 1460>
193.72.186.8.1517 > 193.72.186.6.80: . ack 3731483052 win 32120
   (DF) [tos 0x10]
3 packets received by filter
0 packets dropped by kernel
```


Protocole ARP



192.168.1.10 veut contacter 192.168.2.10.

route par défaut, routeur 192.168.1.1

ARP−request pour 192.168.1.1 dans le réseau physique Ethernet. Réponse de 192.168.1.1.

Envoi du paquet au routeur. Routeur route sur sous−réseau 192.168.2.0/24. ARP−request pour 192.168.2.10. Réponse et envoi du paquet à destination.

Address Resolution Protocol: à quelle adresse Ethernet faut−il acheminer un paquet IP ?

Comme on le voit, pour pouvoir écouter tout le trafic d'une interface, il faut lancer la commande `tcpdump` sous l'utilisateur privilégié `root`.

Filtres Si l'on n'est intéressé que par un certain type de trafic, `tcpdump` dispose d'un système de filtre programmable assez puissant dont nous verrons quelques exemples par la suite.

Address Resolution Protocol : **mapping** adresse IP vers adresse Ethernet.

Exemple où `tcpdump` peut aussi travailler au niveau ‘physique’ :

```
defian:/home/schaefer# tcpdump -i eth0 -n -S -t -e 'arp'
tcpdump: listening on eth0
0:c0:c:3:6f:50 ff:ff:ff:ff:ff:ff 0806 42: arp who-has 193.72.186.7 tell 193.72.186.8
0:60:8:bd:7b:2c 0:c0:c:3:6f:50 0806 60: arp reply 193.72.186.7 is-at 0:60:8:bd:7b:2c
defian:/home/schaefer# arp
Address                 HWtype  HWaddress           Flags Mask            Iface
pc2.alphanet.ch         ether   00:60:08:BD:7B:2C   C                     eth0
vulcan.alphanet.ch      ether   00:40:05:16:08:20   C                     eth0
defian:/home/schaefer# arp -d pc2.alphanet.ch
```

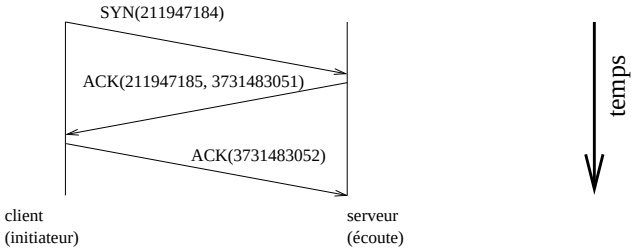
Réseau physique ARP n’est utile qu’à l’intérieur d’un réseau physique. Les adresses Ethernet ne sortent pas d’un sous-réseau (ne passent pas les routeurs). On peut cependant parfois sur une machine Linux vouloir rendre public des adresses de liaisons non Ethernet sur le réseau Ethernet pour simplifier la configuration des clients : voir par exemple l’option de `pppd` **proxy arp**. Certains routeurs ou switches mémorisent l’adresse ARP : remplacer un serveur ou une carte en panne n’est pas toujours aisé !

Requêtes et messages ICMP

Exercice : utiliser `ping`, pour voir quelques messages ICMP, comme : ICMP Echo Reply, Echo Request, et erreur de routage.

Modèle client/serveur TCP

Ouverture en 3 phases :



Attaque difficile si les numéros de séquence ne sont pas prédictibles.

Attaques possibles Si les numéros de séquence sont prédictibles, un attaquant peut s'insérer, sous certaines conditions, dans une connexion TCP, voire en créer une monodirectionnelle depuis une adresse fausse.

Si les numéros de séquence ne sont pas prédictibles, un attaquant situé sur le chemin des paquets (Ethernet : tout le sous-réseau) peut s'insérer et capturer les données.

Dans un certain sens, grâce à son ouverture en trois phases, **TCP** peut dans certains cas offrir une meilleure sécurité que **UDP**. Mais pas forcément une sécurité suffisante.

Impact sur la performance Il est évident que le protocole TCP, par son protocole d'ouverture à 3 phases, a une grande latence : c'est pour cela que HTTP/1.1 a introduit la notion de connexions persistantes.

DNS

Problème

– correspondance nom vers adresses et retour
– savoir qui gère un service donnée

Solution DNS

– base de données hiérarchique, délégable
– cache
– redondance par serveurs secondaires
– types de données

Round-robin Il s'agit bien *des* addresses : cela est utilisé notamment pour un pool de serveur en *round-robin*.

Types de données Le DNS contient notamment l'indication de qui gère le mail pour un domaine (enregistrements MX), et les correspondances d'adresses (A), y compris pour la correspondance adresse IP vers nom (résolution inverse, à travers le pseudo-domaine `.IN-ADDR.ARPA`), voire des informations supplémentaires (HINFO, etc).

Hiérarchique Le DNS est hiérarchique car il permet de déléguer la gestion d'arborescences. Par exemple, un domaine sous `alphanet.ch` (p.ex. `test.alphanet.ch`, dont une machine serait `www.test.alphanet.ch` par exemple) peut très bien être géré par un serveur distinct. Cela correspond par exemple à l'organisation d'une société en départements. Il existe d'autres systèmes de nommage : par exemple le service **WINS** de Microsoft, qui n'est pas hiérarchique.

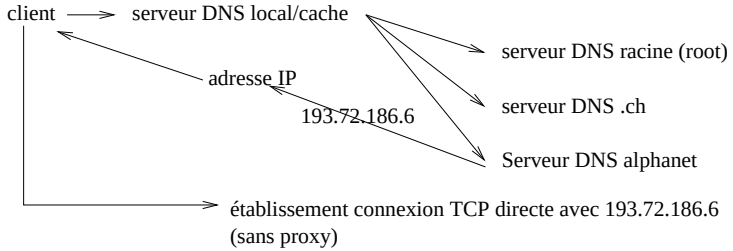
Implémentation de référence L'implémentation de référence est **BIND** ou named de l'Internet Software Consortium (`http://www.isc.org/`), c'est celle qui est no-

tamment utilisée pour les serveurs root et la plupart des serveurs de noms actuels. Pour de petits serveurs moins complexes et pour une meilleure sécurité, le logiciel `djbdns` est recommandé.

Sécurité Le DNS n'a pas de sécurité intégrée : diverses attaques sont possibles (*cache poisoning*). Des systèmes de signatures existent, mais aucun n'est pour le moment utilisé à large échelle. Tout cela ne fait qu'ajouter au fait qu'utiliser des adresses pour l'authentification est une mauvaise idée (et qu'une adresse DNS est encore plus mauvaise qu'une adresse IP).

DNS : Exemple

Processus de résolution hiérarchique (DNS)



www.alphanet.ch

Requête DNS

Exercice :

- faire une requête DNS, p.ex. avec `nslookup`
- analyser la trace
- quel est le protocole IP utilisé ?
- impact sur la sécurité ?

Notons que les requêtes **DNS** peuvent aussi se faire parfois en TCP (longues requêtes principalement).

Succession de nslookup Apparemment, dans les versions ultérieures de Debian, la commande **nslookup** sera marquée comme *obsolète* : la commande **host** la remplace :

```
schaefer@search:~$ host www.esnig.ch
www.esnig.ch            CNAME   hermes2inter.cpln.ch
hermes2inter.cpln.ch    A       157.26.161.10

schaefer@search:~$ host www.dilog.ch
www.dilog.ch            A       194.148.8.243

schaefer@search:~$ host -t A www.dilog.ch
www.dilog.ch            A       194.148.8.243

schaefer@search:~$ host -t MX www.dilog.ch
www.dilog.ch            MX      10 vulcan.alphanet.ch
```


Autre outil : dig L'outil dig permet de déterminer plus en détail la configuration d'un domaine (délais, etc) d'un serveur DNS.

Debian Le serveur DNS conseillé sous Debian est `bind9`, en mode `chroot`. Pour de petites installations, un serveur DNS simplifié comme `djbdns` peut être recommandé.

DHCP

Problème

– allouer des adresses IP dynamiquement

Solution DHCP

– serveur DHCP répondant aux *broadcasts*
– possibilité de figer IP sur adresse ARP
– possibilité de spécifier d'autres options (serveurs DNS, kernel à charger, etc)
– ancêtre : **BOOTP**
– associé à **TFTP** et/ou NFS permet de démarrer une station sans disque (*diskless*)

Debian

```
apt-get install dhcp3-server
```

clients DHCP : `dhcpcd`, `pump`, etc.

Microsoft Windows et adresses automatiques Lorsqu'une machine Microsoft Windows n'arrive pas à contacter un serveur DHCP, il peut arriver qu'elle s'attribue automatiquement une adresse dans la plage `169.254.0.0`. A ma connaissance, aucun standard ouvert ne définit cela, seul un vague *draft* IETF jamais diffusé.
Les derniers 16 bits sont générés aléatoirement, ensuite l'adresse est construite, quelques requêtes ARP sont envoyées pour vérifier que personne n'a cette adresse, puis l'adresse est prise.
Il est évident que ce protocole n'est pas sans faille. Dans un petit réseau sans serveurs ni adresses fixes autres, cela peut cependant pallier à l'absence de configuration IP fixe ou de serveur DHCP.

NTP

Problème

– maintenir un temps cohérent, voire correct dans un réseau

Solution NTP

– *daemon* qui adapte lentement l'heure
– serveur(s) de référence
– horloge matérielle éventuelle
– solution simple : `ntp-simple`

Capture de données

Exercice : Capturer quelques trames IP dans un fichier :

```
# tcpdump -i eth0 -s 1600 -w /tmp/packets_log
```

Analyse données

– avec outils divers UNIX
– avec `tcpdump`
– avec `epan`

Serveurs (daemons) TCP/IP

Deux types :

– *stand-alone*
– `inetd`

Qui écoute :

– `netstat -anp`

On peut rapidement analyser les données avec `strings`, qui nous donne les séquences ressemblant à du texte dans un fichier. On peut aussi utiliser l'utilitaire `hexdump`, ou le mode hexadécimal de **Emacs**.

Pour une analyse un peu plus poussée, on peut réutiliser `tcpdump` avec son option `-r`. Dans ce cas on n'a plus besoin de tourner sous `root`. Par exemple si résoudre les noms DNS prend trop de temps dans la phase de capture, on peut laisser cela à la phase de lecture. Ne pas négliger les options `-v`, avec autant de v que nécessaire. L'option `-X` permet d'avoir une sortie hexadécimale mais aussi ASCII.

Protocoles particuliers, statistiques Pour analyser des protocoles particuliers (`NFS`, `SMB`, etc), on peut utiliser `epan`. Pour bénéficier de la fonctionnalité complète de statistiques, utiliser sur la Debian potato : `export PATH=${PATH}:/usr/lib/epan/` avant de lancer (Debian bug #30997).

inetd

écoute et lancements de daemons

– `/etc/inetd.conf`
– `/etc/init.d/inetd reload`
– `update-inetd`

syntaxe :

```
service_name socket_type protocol wait|nowait[.max] user[.group]
   server_program arguments ...
```

exemple :

```
netbios-ns      dgram   udp     wait    root    /usr/sbin/tcpd
   /usr/sbin/nmbd -
```


netfilter

packet filter

– stateful packet filter (connection tracking)
– Network Address Translation support, including transparent proxying
– QoS and policy routing, load balancing, fail-over
– remplace ipfwadm (2.0.x) et ipchains (2.2.x)
– très avancé
– trop complexe pour ce cours !

IPv6

IPng

– successeur à IPv4
– adressage étendu (128 bits), qualité de service, authentification et chiffrement (optionnels), mobilité.
– mais : avantages repris dans IPv4 ! (IPsec, QoS, NAT/Masquerading)
– problème principalement politique (USA vs monde)
– test : IPv6 en tunnel IPv4.
– support Linux fonctionnel : y.c. firewalling.

ssh

- connexion interactive, commandes distantes, copie de fichiers
- originellement remplacement de telnet/rlogin/rcp/rsh/rexec
- encryption, authentification RSA/IDEA/etc.
- clé publique
- tunnel/redirection de port

Résoud certains problèmes de TCP/IP

Remplacement SSH est un remplacement pour `telnet` et `rlogin`/`rcp`. Il permet de se connecter à des serveurs distants, avec authentification par mot de passe, adresse, clé publique d'utilisateur ou clé publique d'équivalence hôte.

Exécution de commandes à distance Il permet aussi d'exécuter des commandes à distance (remplaçant ainsi `rsh`). Un exemple un peu farfelu, qui décompresse localement un fichier, envoie le fichier ainsi décompressé à une commande distante de compression, et sauve le nouveau fichier comprimé localement.

```
gzip -d < fichier.gz | ssh -e none remote 'bzip2 -9' > fichier.bz2
```

Redirections de ports En plus du transfert de fichier sécurisé (via la commande `scp`), on peut également créer des redirections sécurisées (dont les données passent par une connexion encryptée et authentifiée) :

- des ports en écoute sur la machine serveur, qui sont redirigés à des ports en écoute sur la machine cliente, option `-L`.
- des ports en écoute sur la machine cliente qui sont redirigés sur des ports en écoute soit sur le serveur, soit sur d'autres machines accessibles depuis le serveur, option `-R`.

Comptes restreints On peut également configurer le serveur SSH (`sshd`) pour n'autoriser le lancement que d'une commande spécifique : p.ex. pour autoriser un accès sécurisé à CVS, stocker des données de backup, ou faire des redirections de port, sans avoir un vrai accès shell.

En cas d'utilisation des fonctionnalités d'authentification par clé, il faut mettre un mot de passe sur sa clé privée.

Ne pas confondre Ne pas confondre SSH, un protocole permettant une connexion sécurisée entre deux machines, ainsi que quelques fonctionnalités qui dépassent ce cadre, avec **SSL**, *Secure Socket Layer*, une surcouche à TCP permettant de transférer des données, par exemple HTTP, sécurisées.

Application : courrier sécurisé

But : utiliser les services usuels SMTP et POP en encryptant les données via un tunnel SSH.

```
ssh -L 10110:pop.isp.ch:110 -L 10025:smtp.isp.ch:25
   user@server
```


On configurera alors les clients mail locaux pour utiliser le port 10110 local pour la lecture POP-3, et le port 10025 pour l'envoi SMTP. Ces numéros sont choisis arbitrairement. Notons que si `smtp.isp.ch` n'est pas la même chose que `server`, les données circuleront en clair entre ces deux serveurs.

On remarquera que les versions récentes de ssh (notamment OpenSSH) n'écoutent sur ces ports que pour l'adresse de **loopback**, diminuant l'impact de sécurité.

Peut-on faire pire ? Oui, largement. On peut faire un VPN encrypté via une connexion ssh. Regardez ici :

```
computera# chmod 600 /dev/ptyzf && chown root.root /dev/ptyzf
computera# slattach /dev/ptyzf & # choisir une valeur inutilisée (!)
computera# ifconfig sl0 192.168.4.1  # assuming sl0 was free before
computera# route add 192.168.4.2 dev sl0
computera# ssh < /dev/ttyzf > /dev/ttyzf -e none -t root@computerb
    'slattach 'tty''
computera# ssh root@computerb 'ifconfig sl0 192.168.4.2; route add
    192.168.4.1 dev sl0'
```


(une meilleure version est décrite dans le VPN-HOWTO, mais ce qui précède permet de comprendre le concept).

Exercice : serveur via inetd(8)

But : créer un service qui affiche un texte déterminé sur un port particulier.

Pour prolonger la discussion ...

`http://www.alphanet.ch/~schaefer/some_files/protocoles_internet.ps`
Description plus détaillée des protocoles Internet ainsi que de IPv6.

`http://www.alphanet.ch/~schaefer/linux_event/presentations/internet_intranet/` Autre présentation.

`0-201-63346-9` TCP/IP Illustrated, Volume 1

`http://www.faqs.org/rfcs/` Les RFCs

Linux IPv6 HOWTO

`http://www.netfilter.org` netfilter/IPtables

`http://people.debian.org/~csmall/ipv6/` IPv6 sur Debian GNU/Linux

`$Id: tcpip.tex,v 1.11 2002/12/03 09:15:37 schaefer Exp $`