Routage IP au niveau hôte

Philippe Langevin

Oct 2007, Nov 2008, Nov 2011.

Protocole IP

Interface réseau

Routeur

Livraison physique

Routage IP

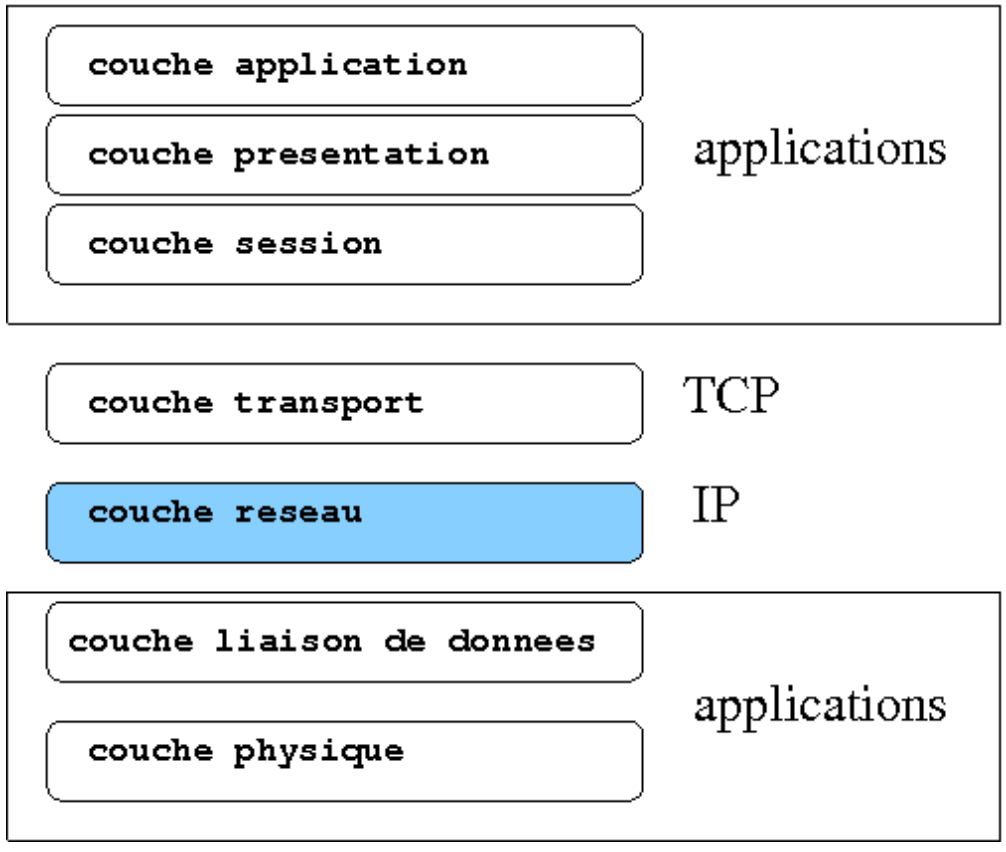
Adresse multicast

Filtrage et translation d'adresse

Algorithme de routage

Travaux-Pratiques

Modèle TCP/IP



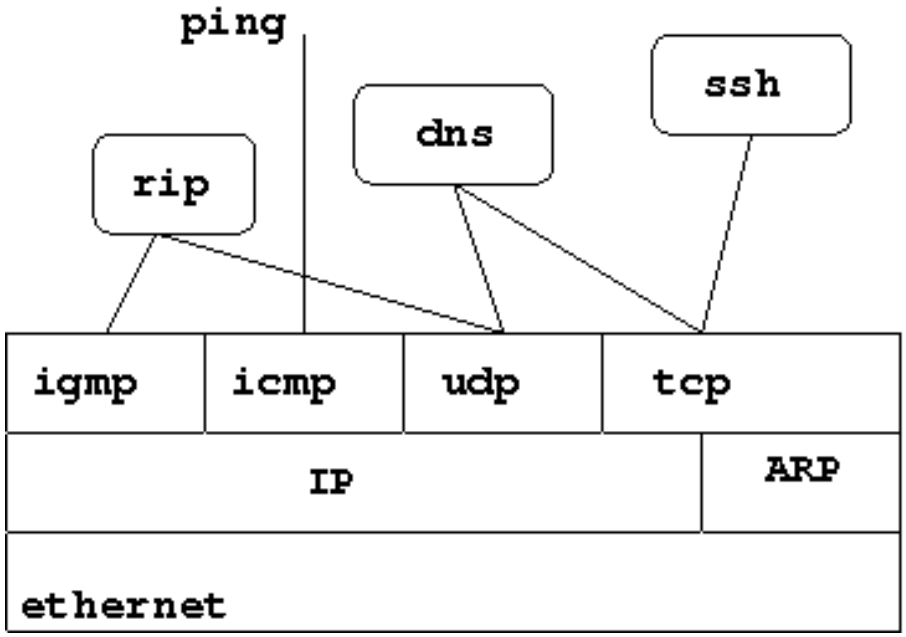
- Le protocole IP (Internet Protocole) constitue la couche réseau de TCP/IP. Il gère l'adressage et réalise l'acheminement des données.

objectif du cours

Il s'agit de se familiariser avec quelques points, outils et services.

- configuration réseau
- programmation réseau
- administration réseau
- parallèlisation

Architecture TCP/IP



- L'architecture TCP/IP comprend plusieurs protocoles et services.

adresse IP

Le protocole IP décrit le mode d'acheminement de paquets de données entre deux interfaces affectées d'adresses IP. Il prévoit trois modes :

- unicast
- broadcast (diffusion)
- multicast (groupe)

Classe d'adresses

Une adresse IP est formée de 4 octets:

192.168.0.253

Classe	Début	Numéros	Combinaison	Nb hôtes
A	1-126	1	255^3	16581375
B	128-191	2	255^2	65025
C	192-223	3	255	255
D	Multicasting			
E	Réservé			

$127.x.y.z$

désigne une adresse locale, la RFC1357 prévoit des adresses *privées*:

- En classe A : 10.0.0.0 à 10.255.255.255
- En classe B : 172.16.0.0 à 172.31.255.255
- En classe C : 192.168.0.0 à 192.168.255.255

Adresse Réseau

- L'adresse 192.168.128.52 ne fait pas partie du réseau 192.168.176.0 / 20 bits

Interface Réseau

Le périphérique dédié à la communcation réseau est une interface réseau. La commande `ifconfig` permet de configurer une interface réseau.

- `ifconfig` : voir une configuration.
- `ifconfig eth0 down` : désactiver l'interface.
- `ifconfig eth1 up` : activer l'interface.
- `ifconfig eth0 192.168.0.1 mask 255.255.255.0` : assigner une adresse IP.

ifconfig

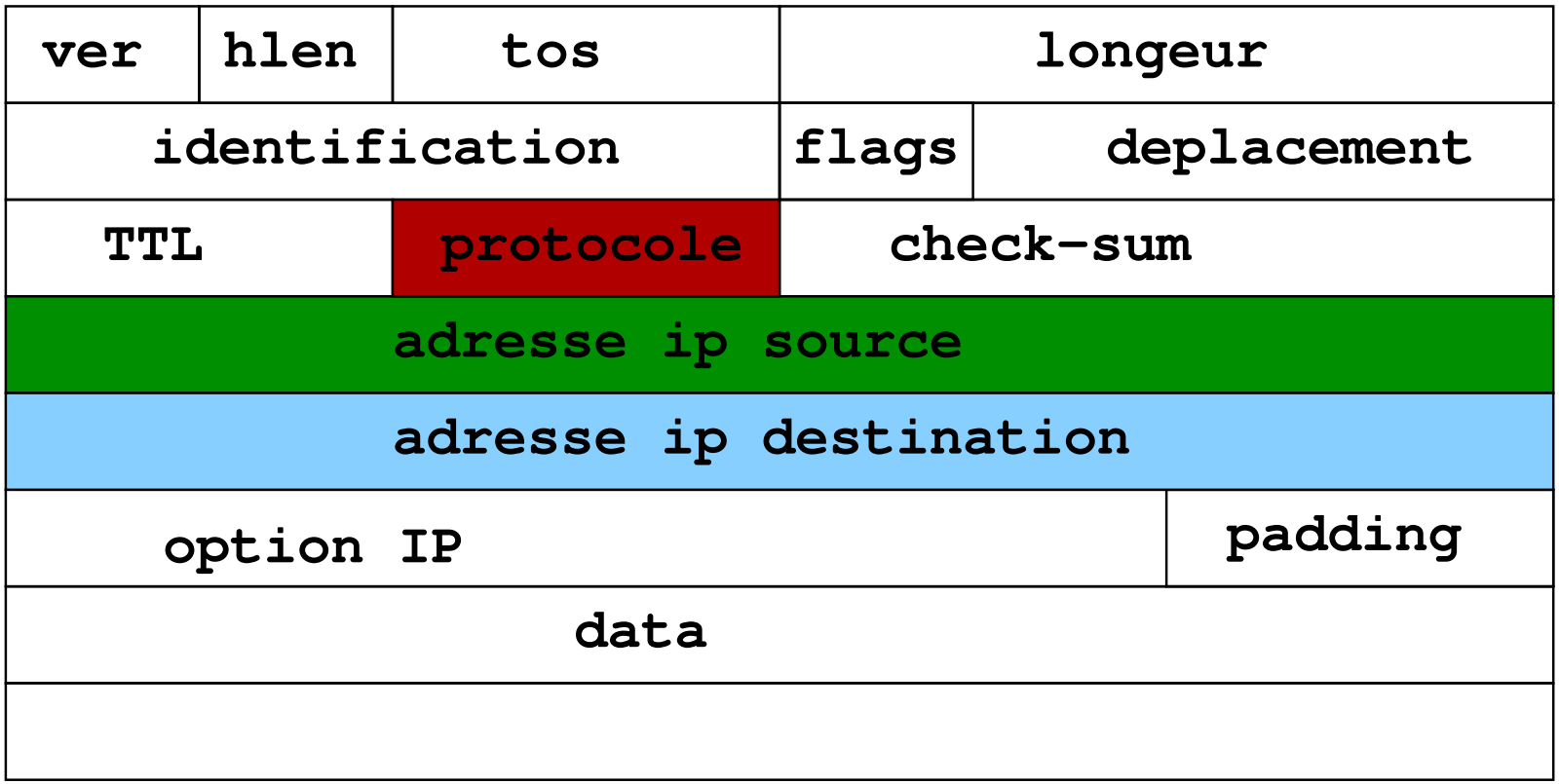
```
eth0 Link encap:Ethernet HWaddr 00:60:97:AA:04:0D
 inet adr:10.2.73.86 Bcast:10.2.79.255 Mask:255.255.248.0
 adr inet6: fe80::260:97ff:feaa:40d/64 Scope:Lien
 UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
 RX packets:230900 errors:0 dropped:0 overruns:0 frame:0
 TX packets:89477 errors:0 dropped:0 overruns:0 carrier:1
 collisions:12043 lg file transmission:1000
 RX bytes:74627279 (71.1 MiB)  TX bytes:55943233 (53.3 MiB)
 Interruption:10 Adresse de base:0xe800

lo  Link encap:Boucle locale
 inet adr:127.0.0.1  Masque:255.0.0.0
 adr inet6: ::1/128 Scope:Hote
 UP LOOPBACK RUNNING  MTU:16436  Metric:1
 RX packets:10794 errors:0 dropped:0 overruns:0 frame:0
 TX packets:10794 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 lg file transmission:0
 RX bytes:9488777 (9.0 MiB)  TX bytes:9488777 (9.0 MiB)
```

Routeur

Les réseaux sont reliés entre eux par des routeurs. Un routeur est un hôte qui joue le rôle de passerelle entre les réseaux sur lequel il est physiquement connecté.

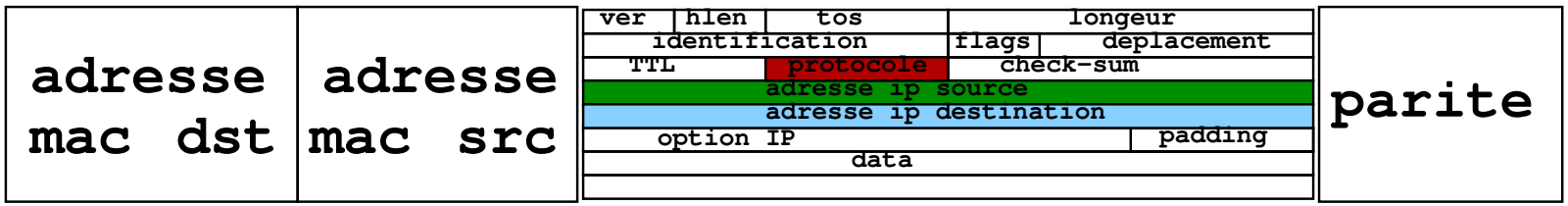
- une adresse physique par interface
- une adresse IP par réseau.
- autorisation des transfert d'interface :
  - `/proc/sys/net/ipv4/ip_forward`
  - `find /proc -name ‘‘*forward*’’`

datagramme


ver
hlen
tos
longeur
identification
flags
deplacement
TTL
protocole
check-sum
adresse ip source
adresse ip destination
option IP
padding
data

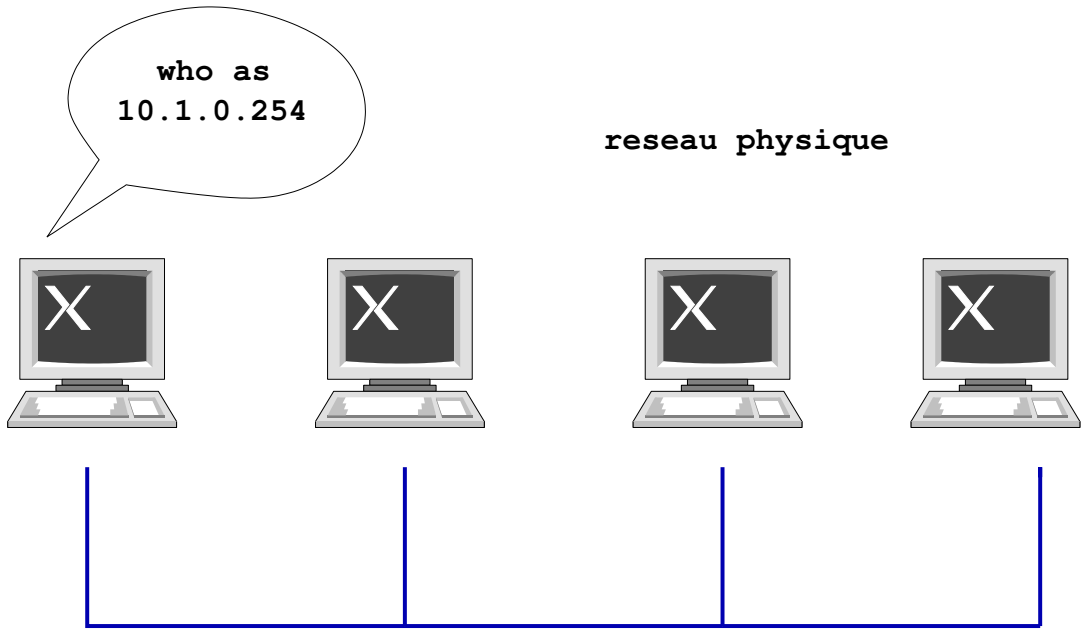
Encapsulation

Lors de la transmission physique, les datagrammes IP sont encapsulés dans des trames ethernet :

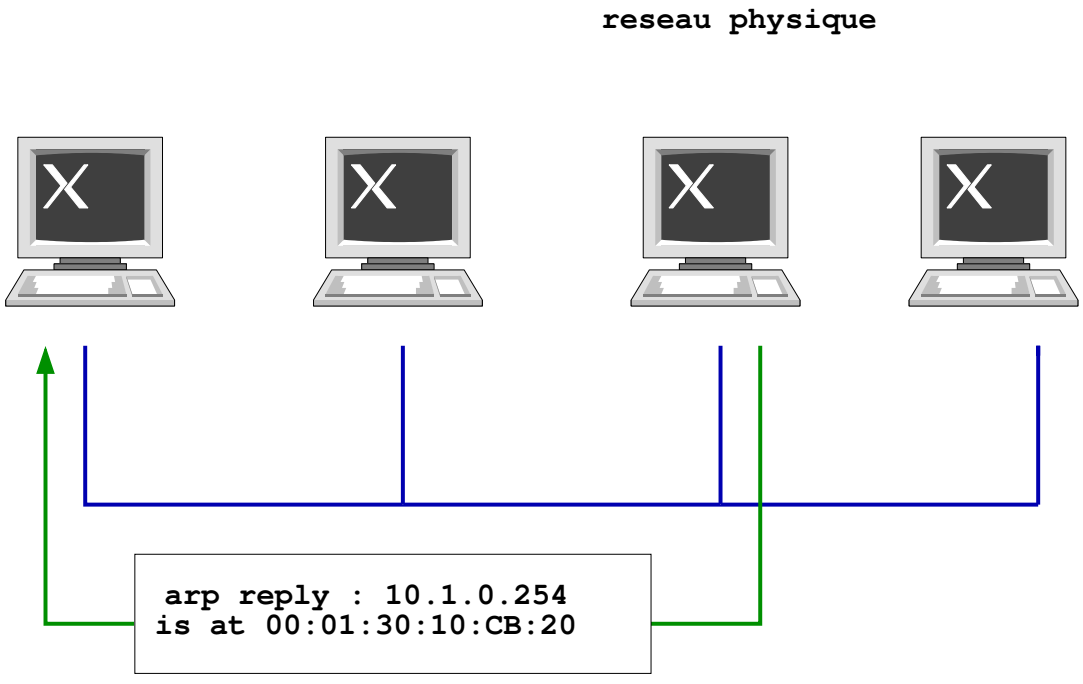


Ne pas confondre adresses mac et adresses IP !

Livraison physique



Livraison physique



ping mail.univ-tln.fr

```
[root@mnsnet]# /usr/sbin/tcpdump -n -i any


07:36.arp who-has 192.168.0.254 tell 192.168.0.30
07:36.arp reply 192.168.0.254 is-at 00:07:cb:1a:16:3d
07:36.IP 192.168.0.30.filenet-nch > 212.27.54.252.domain
         37820+ A? mail.univ-tln.fr.
07:36.IP 212.27.54.252.domain > 192.168.0.30.filenet-nch
         37820 1/0/0 A 193.49.96.2
07:36.IP 192.168.0.30 > 193.49.96.2: ICMP echo request,
         id 6933, seq 1, length 64
07:36.IP 193.49.96.2 > 192.168.0.30: ICMP echo reply,
         id 6933, seq 1, length 64
07:36.IP 192.168.0.30.filenet-nch > 212.27.54.252.domain
         52476+ PTR? 2.96.49.193.in-addr.arpa. (42)
```

Usurpation d'adresse mac

Le protocole `arp` n'est pas sécurisé :

- diffusion des requêtes.
- un hote pirate peut usurper une adresse mac.
- arp spoofing.

Commandes et fichiers

- `arp` - manipule la table ARP du système.
- fichier `/proc/net/arp`

```
[dm@msnet] cat /proc/net/arp
IP address      HW type Flags HW address          Mask  Device
192.168.0.254   0x1     0x2   00:07:CB:1A:16:3D *     eth
```

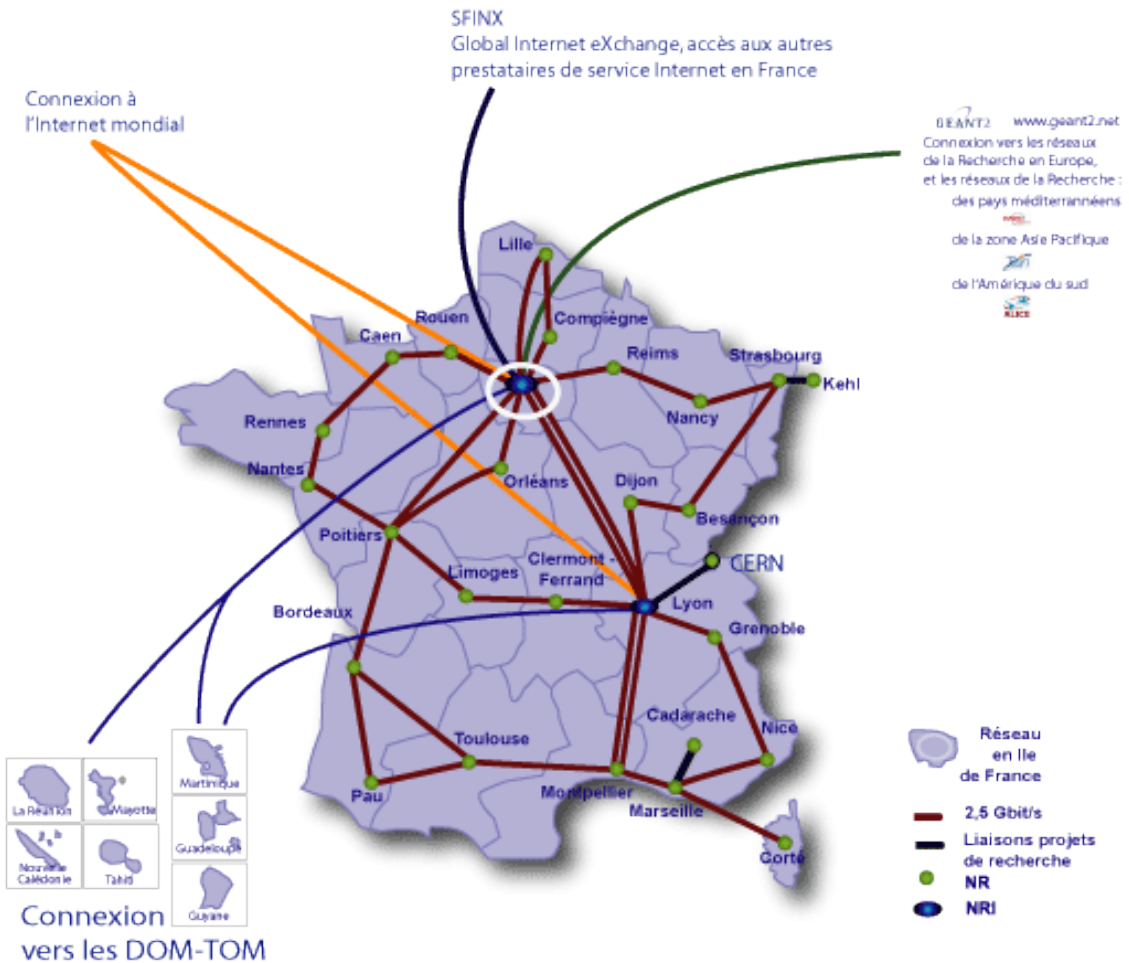
traceroute www.upf.fr

```
 1  10.2.72.1 3.980 ms  4.947 ms  6.344 ms
 2  c7200  6.716 ms  4.112 ms  4.858 ms
 3  194.214.66.30  14.135 ms  9.463 ms  7.697 ms
 4  193.50.108.93  7.448 ms  9.867 ms  6.292 ms
 5  193.49.2.14 8.079 ms  7.015 ms  11.694 ms
 6  marseille-g3-2-24.cssi.renater.fr 7.217 ms
 7  montpellier-pos2-0.cssi.renater.fr 17.109 ms
 8  lyon-pos15-0.cssi.renater.fr 16.135 ms
 9  ftld-lyon.cssi.renater.fr 20.408 ms  19.344 ms
10  po0-0.passe2.Paris.opentransit.net 24.689 ms
11  tengige0-3-0-0-1000.pastr1.Paris.opentransit.net
12  verio-4.GW.opentransit.net 28.795 ms  30.372 ms
13  as-0.r23.londen03.uk.bb.gin.ntt.net 33.649 ms
14  xe-3-1.r01.londen03.uk.bb.gin.ntt.net 50.833 ms
15  xe-3-1.r01.londen05.uk.bb.gin.ntt.net 32.742 ms
16  ge-2-15.c01.londen02.uk.wh.verio.net 32.899 ms
17  83.231.152.36  44.401 ms  32.400 ms  34.702 ms
18  eul0001112-pip.eu.verio.net 34.425 ms
```

Renater

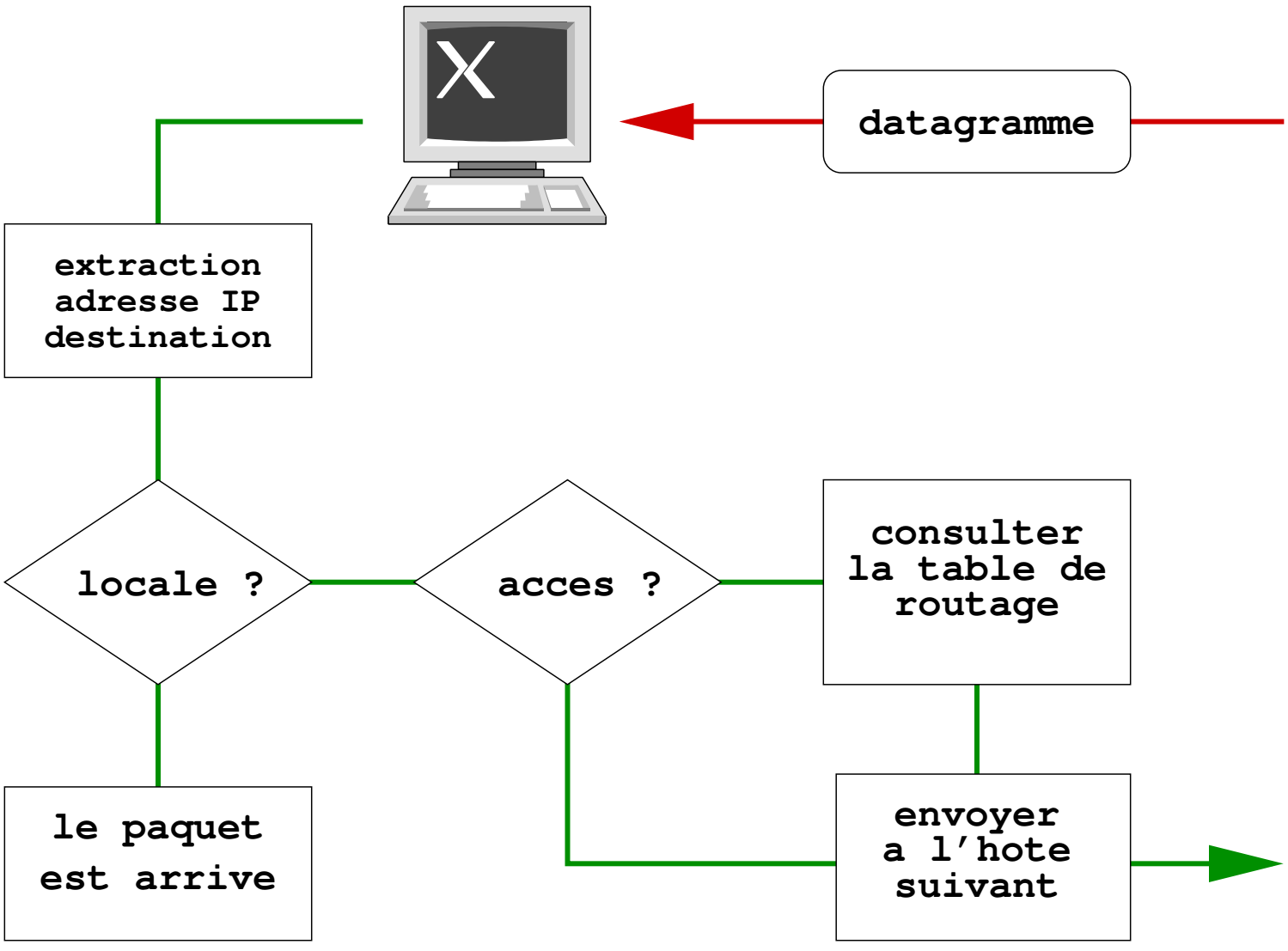
Réseau National de télécommunications
pour la technologie, l'enseignement et la Recherche

RENATER-4

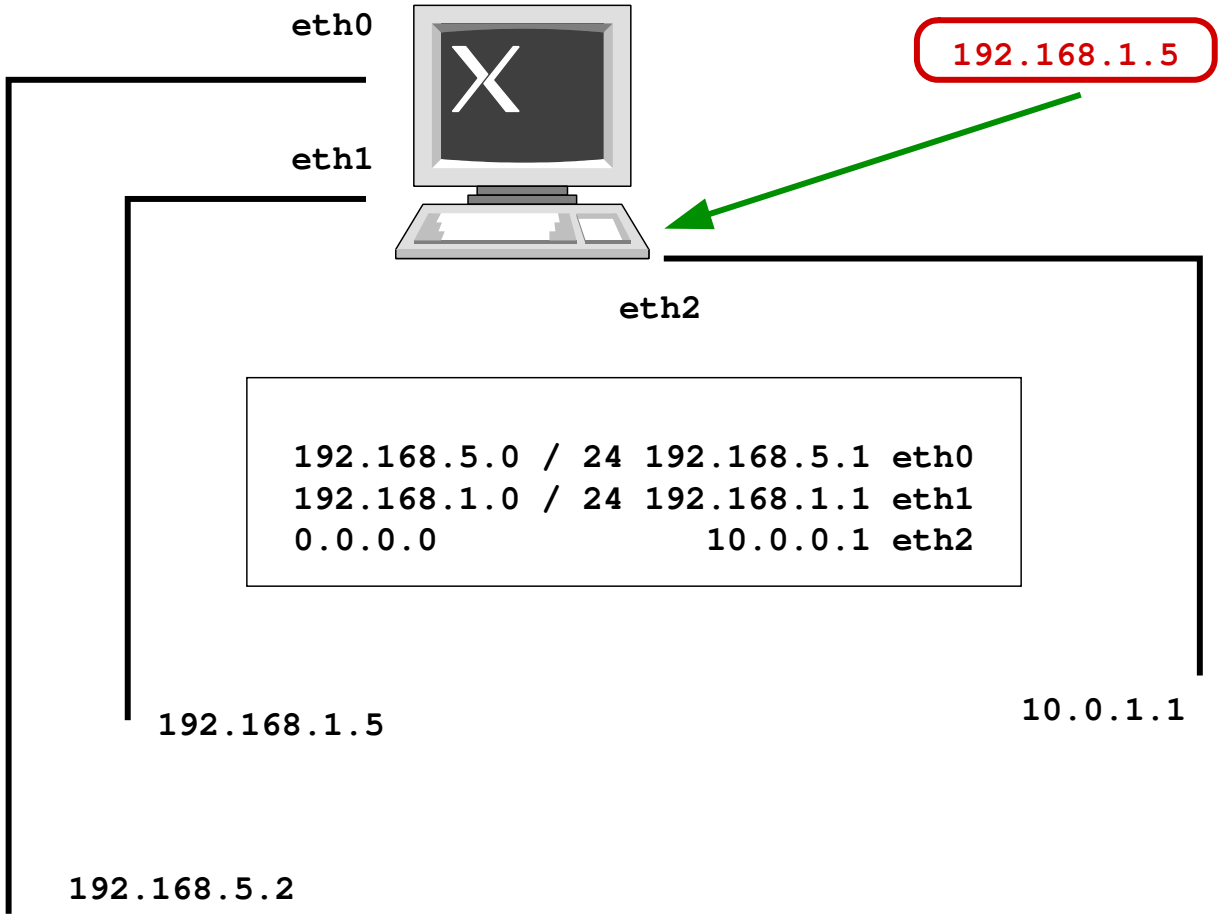


- Un maillage complet sur l'ensemble des points de présence du réseau

Routage IP


datagramme
extraction adresse IP destination
locale ?
acces ?
consulter la table de routage
le paquet est arrive
envoyer a l'hote suivant

Routage IP


eth0
eth1
eth2
192.168.1.5
192.168.5.0 / 24 192.168.5.1 eth0
192.168.1.0 / 24 192.168.1.1 eth1
0.0.0.0 10.0.0.1 eth2
192.168.1.5
10.0.1.1
192.168.5.2

Messages ICMP

Lorsque un paquet traverse une passerelle, le TTL du datagramme est décrémenté. Le routeur emet un message ICMP à l'adresse source IP du paquet:

- paquet perdu
- pas d'hôte sur le réseau
- pas de route pour la destination
- port fermé
- paquet filtré

Paquet filtré

```
[root@ou812 ~]# traceroute luminy.univ-mrs.fr.
traceroute to luminy.univ-mrs.fr (139.124.100.7), 30 hops m
 1  10.2.72.1 (10.2.72.1)  0.815 ms  0.860 ms  0.717 ms
 2  c7200 (192.168.6.2)  1.284 ms  2.705 ms  0.743 ms
 3  194.214.66.30 (194.214.66.30)  24.132 ms  28.485 ms  29
 5  193.50.108.138 (193.50.108.138)  29.207 ms  32.550 ms
 6  194.214.66.194 (194.214.66.194)  10.021 ms  4.012 ms  3
 8  vpn-internet-u2-Timone.phocean.fr (194.214.97.5)  21.96
 9  ad-u2-Luminy.phocean.fr (193.50.131.23)  20.335 ms  5.9
10  luminy.univ-mrs.fr (139.124.100.7)  10.074 ms  8.488 ms

[root@ou812 ~] # ping -c1 -t4 luminy.univ-mrs.fr
PING luminy.univ-mrs.fr (139.124.100.7) 56(84) bytes of dat
From (192.168.6.2) icmp_seq=0 Packet filtered

--- luminy.univ-mrs.fr ping statistics ---
1 packets transmitted, 0 received, +1 errors,
  100 % lost
```

Time to live exceeded

```
[pl@localhost $ traceroute www.univ-tln.fr
traceroute to www.univ-tln.fr (193.49.96.34), 30 hops max,
 1  192.168.0.254 (192.168.0.254)  2.538 ms  4.685 ms  5.32
 2  82.244.173.254 (82.244.173.254)  25.973 ms  29.212 ms
 3  78.254.6.222 (78.254.6.222)  31.299 ms  31.923 ms  32.6
 4  tga83-1-v902.intf.nra.proxad.net (78.254.254.113)  33.8
 5  tsm83-1-v900.intf.nra.proxad.net (78.254.254.109)  36.6
 6  tdv83-1-v902.intf.nra.proxad.net (78.254.254.105)  48.8
 7  tro83-1-v900.intf.nra.proxad.net (78.254.254.101)  22.5
 8  tca83-1-v902.intf.nra.proxad.net (78.254.254.97)  26.37
 9  lse83-1-v900.intf.nra.proxad.net (78.254.254.93)  28.43
10  sf283-1-v902.intf.nra.proxad.net (78.254.254.89)  33.99
11  ban83-1-v900.intf.nra.proxad.net (78.254.254.85)  37.79
12  lbe83-1-v902.intf.nra.proxad.net (78.254.254.81)  22.95
13  scy83-1-v900.intf.nra.proxad.net (78.254.254.77)  23.60
14  cio13-1-v902.intf.nra.proxad.net (78.254.254.73)  27.55
15  au213-1-v900.intf.nra.proxad.net (78.254.254.69)  32.45
16  au113-1-v902.intf.nra.proxad.net (78.254.254.65)  36.06
```

Time to live exceeded

```
[pl@localhost]$ ping -c1 -t8  www.univ-tln.fr
PING www.univ-tln.fr (193.49.96.34) 56(84) bytes of data.
From tca83-1-v902.intf.nra.proxad.net
  (78.254.254.97) icmp_seq=1 Time to live exceeded

--- www.univ-tln.fr ping statistics ---
1 packets transmitted, 0 received, +1 errors,
  100% packet loss, time 65ms
```

Destination Host Unreachable

```
[root@ou812 ~]# host port-aci
port-aci.univ-tln.fr has address 10.2.73.93

[root@ou812 ~]# ping -c1 10.2.73.93
PING 10.2.73.93 (10.2.73.93) 56(84) bytes of data.
From 10.2.73.86 icmp_seq=0 Destination Host Unreachable

--- 10.2.73.93 ping statistics ---
1 packets transmitted, 0 received, +1 errors
```

Destination Net Unreachable

```
[root@ou812 ~]# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 193.49.2.14 icmp_seq=7 Destination Net Unreachable
From 193.49.2.14 icmp_seq=8 Destination Net Unreachable
From 193.49.2.14 icmp_seq=9 Destination Net Unreachable
From 193.49.2.14 icmp_seq=10 Destination Net Unreachable

--- 10.0.0.1 ping statistics ---
11 packets transmitted, 0 received, +4 errors, 100
```

Port unreachable

```
[langevin@ou812 ~] host port-aci maitinfo7
;; connection timed out; no servers could be reached

[root@ou812 ~]# tcpdump -t icmp or port domain
tcpdump: verbose output suppressed, use -v or -vv for full
listening on eth0, link-type EN10MB (Ethernet), capture siz

IP ou812.univ-tln.fr.32831 > maitinfo7.univ-tln.fr.domain:
        18541+ A? port-aci.univ-tln.fr. (38)
IP ou812.univ-tln.fr.32832 > mail.univ-tln.fr.domain:
        21309+ PTR? 223.185.9.10.in-addr.arpa. (43)
IP maitinfo7.univ-tln.fr > ou812.univ-tln.fr:
        icmp 74: maitinfo7.univ-tln.fr
     udp port domain unreachable
```

Manipulation des tables de Routage

- `route` affiche la table de routage.
- `route add` permet d'ajouter une route.
- `route del` permet de supprimer une route.

Exemple

```
[root@ou812] route

Table de routage IP du noyau
Destination     Passerelle      Genmask         Iface
10.2.72.0       *               255.255.248.0   U eth0
169.254.0.0     *               255.255.0.0     U eth0
default         10.2.72.1       0.0.0.0         UGeth0

[root@ou812] route add 192.168.0.1 reject

[root@ou812] route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Iface
192.168.0.1     -               255.255.255.255 !H -
10.2.72.0       *               255.255.248.0   Ueth0
169.254.0.0     *               255.255.0.0     Ueth0
default         10.2.72.1       0.0.0.0         UGeth0
```

Correct ou pas ?

destination	masque	passerelle	interface	?
10.1.30.0	255.255.0.0	10.1.30.9	10.1.30.1	
10.1.40.0	255.255.255.0	10.1.40.2	10.1.40.1	
10.1.50.0	255.255.255.0	10.1.50.1	10.1.50.1	
10.1.60.0	255.255.255.0	10.1.60.1	10.1.40.1	
10.1.70.0	255.255.255.0	10.1.40.8	10.1.40.1	
10.1.80.1	255.255.255.0	10.1.50.9	10.1.50.1	
10.1.90.1	255.255.255.255	10.1.50.9	10.1.40.1	
0.0.0.0	0.0.0.0	10.1.50.9	10.1.40.1	

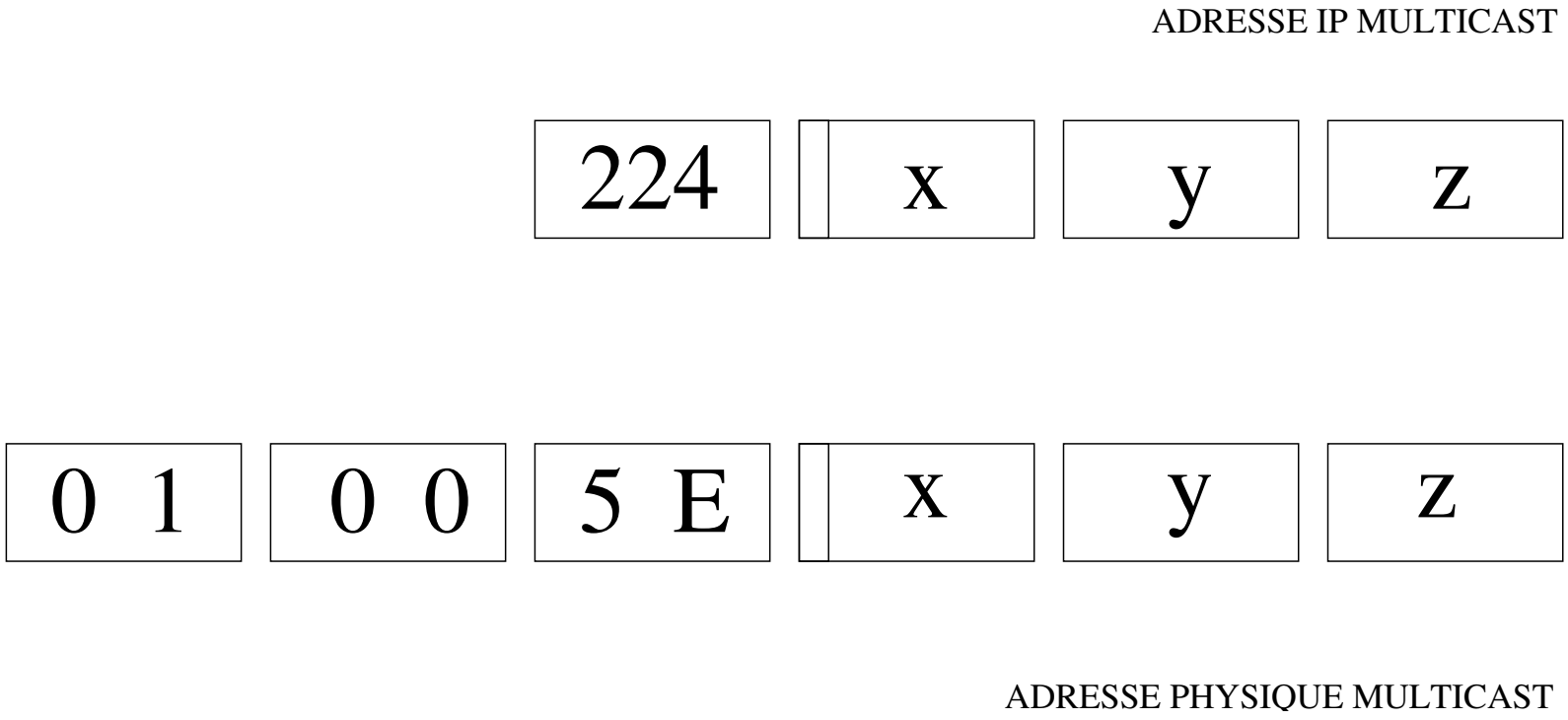
Adresse multicast

- L'adresse 224.0.0.1 identifie tous les hôtes d'un sous-réseau. Tous les hôtes ayant des capacités multicast dans un sous-réseau doivent faire partie de ce groupe.
- L'adresse 224.0.0.2 identifie tout routeur multicast dans un réseau.
- Le champ d'adresse 224.0.0.0 - 224.0.0.255 est alloué pour les protocoles bas niveau. Les datagrammes envoyés dans cette plage d'adresse ne seront pas routés par des routeurs multicast.
- La plage d'adresse 239.0.0.0 - 239.255.255.255 est allouée à des fins administratives. Les adresses sont allouées localement pour chaque organisation mais elles ne peuvent exister à l'extérieur de celles-ci. Les routeurs de l'organisation ne doivent pas pouvoir router ces adresses à l'extérieur du réseau de l'entreprise.

Régions multicast

scope	TTL	Plage d'adresse
Noeud	0	
Lien	1	224.0.0.0 - 224.0.0.255
Département	< 32	239.255.0.0 - 239.255.255.255
Organisation	< 64	239.192.0.0 - 239.195.255.255
Global	< 255	224.0.1.0 - 238.255.255.255

Conversion IP multicast adresse physique



ping -c1 -t1 224.0.0.1

```
64 bytes from 10.2.73.86: icmp_seq=0 ttl=64 time=0.133 ms
64 bytes from 10.2.73.84: icmp_seq=0 ttl=64 time=0.578 ms (
64 bytes from 10.2.73.103: icmp_seq=0 ttl=64 time=0.657 ms
64 bytes from 10.2.73.87: icmp_seq=0 ttl=64 time=0.718 ms (
64 bytes from 10.2.73.100: icmp_seq=0 ttl=64 time=0.780 ms
64 bytes from 10.2.73.75: icmp_seq=0 ttl=64 time=0.797 ms (
64 bytes from 10.2.73.113: icmp_seq=0 ttl=64 time=1.08 ms (
64 bytes from 10.2.73.181: icmp_seq=0 ttl=64 time=1.08 ms (
64 bytes from 10.2.73.178: icmp_seq=0 ttl=64 time=1.18 ms (
64 bytes from 10.2.73.74: icmp_seq=0 ttl=254 time=1.19 ms (
64 bytes from 10.2.73.187: icmp_seq=0 ttl=255 time=1.39 ms
64 bytes from 10.2.73.111: icmp_seq=0 ttl=64 time=1.39 ms (
64 bytes from 10.2.75.5: icmp_seq=0 ttl=128 time=1.49 ms (I
64 bytes from 10.2.73.48: icmp_seq=0 ttl=60 time=2.30 ms (I
64 bytes from 10.2.73.43: icmp_seq=0 ttl=60 time=2.56 ms (I
64 bytes from 10.2.73.206: icmp_seq=0 ttl=64 time=6.79 ms (
64 bytes from 10.2.73.183: icmp_seq=0 ttl=255 time=6.81 ms
64 bytes from 10.2.73.78: icmp_seq=0 ttl=64 time=7.12 ms (I
```

setsockopt - Lire et écrire les options dune socket.

```
#include <sys/types.h>
#include <sys/socket.h>
int setsockopt(int s,int level,int optname,void* optval,
        socklen_t optlen);
```

`setsockopt()` manipule les options associées une socket. Il s'agit de préciser le niveau visé et le nom de loption. Au niveau socket, `level` prend la valeur SOL_SOCKET. Pour tous les autres niveaux, il faut fournir le numro de protocole approprié.
Les paramètres `optval` et `optlen` sont utilisés pour déterminer les options.
`optname` et toute autre option sont passées sans interprétation au protocole approprié, pour quil linterprète lui-même.

Socket multicast : main

```
int  main( int argc, char* argv[] )
{
  int sock;

  ADDRMULTICAST = inet_addr( "224.1.2.3");

  if ( (sock = socket(AF_INET, SOCK_DGRAM, 0 )) < 0
)
    perreur("socket");

  if ( ! args( argc, argv ) ) exit( 1 );

  if ( RECEPT ) reception( sock );
  if ( SEND   )  emission( sock );

  close(sock);
  return 0;
}
```

Socket multicast : args

```
int args(int argc, char* argv[])
{
  char *optliste = "a:p:rs:t:h";
  int   opt;
  while ( ( opt = getopt(argc, argv, optliste )) >=0 ) {
            switch ( opt ){
            case 'a' : ADDRMULTICAST  = inet_addr(optarg
            case 's' : SEND   = atoi(optarg);
break;
            case 'r' : RECEPT = 1;
break;
            case 't' : TTL =    atoi(optarg);
break;
            case 'p' : PORT   = atoi(optarg);
break;
            case 'h' : printf("\nusage %s : %s", argv[0]
            default  : return 0;
            }
          }
```

Socket multicast : reception

```
void  reception( int sock )
{ char bfr[ 1024 ];
  int  nb;
  struct ip_mreq rm;
  struct sockaddr_in adr;
  memset( & adr, 0, sizeof( struct sockaddr_in) );
  adr.sin_family          = AF_INET;
  adr.sin_addr.s_addr     = htonl( INADDR_ANY);
  adr.sin_port            = htons(PORT);

  if ( bind(sock,  (struct sockaddr*) & adr, len ) < 0 )
       perreur("bind");
  memset( & rm, 0, sizeof( struct ip_mreq) );
  rm.imr_multiaddr.s_addr = ADDRMULTICAST;
  rm.imr_interface.s_addr = htonl (INADDR_ANY );
  if ( setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, &
    perreur("opt");
  while ( 1 ) {
      nb = recv(sock, bfr, 1024, 0 );
```

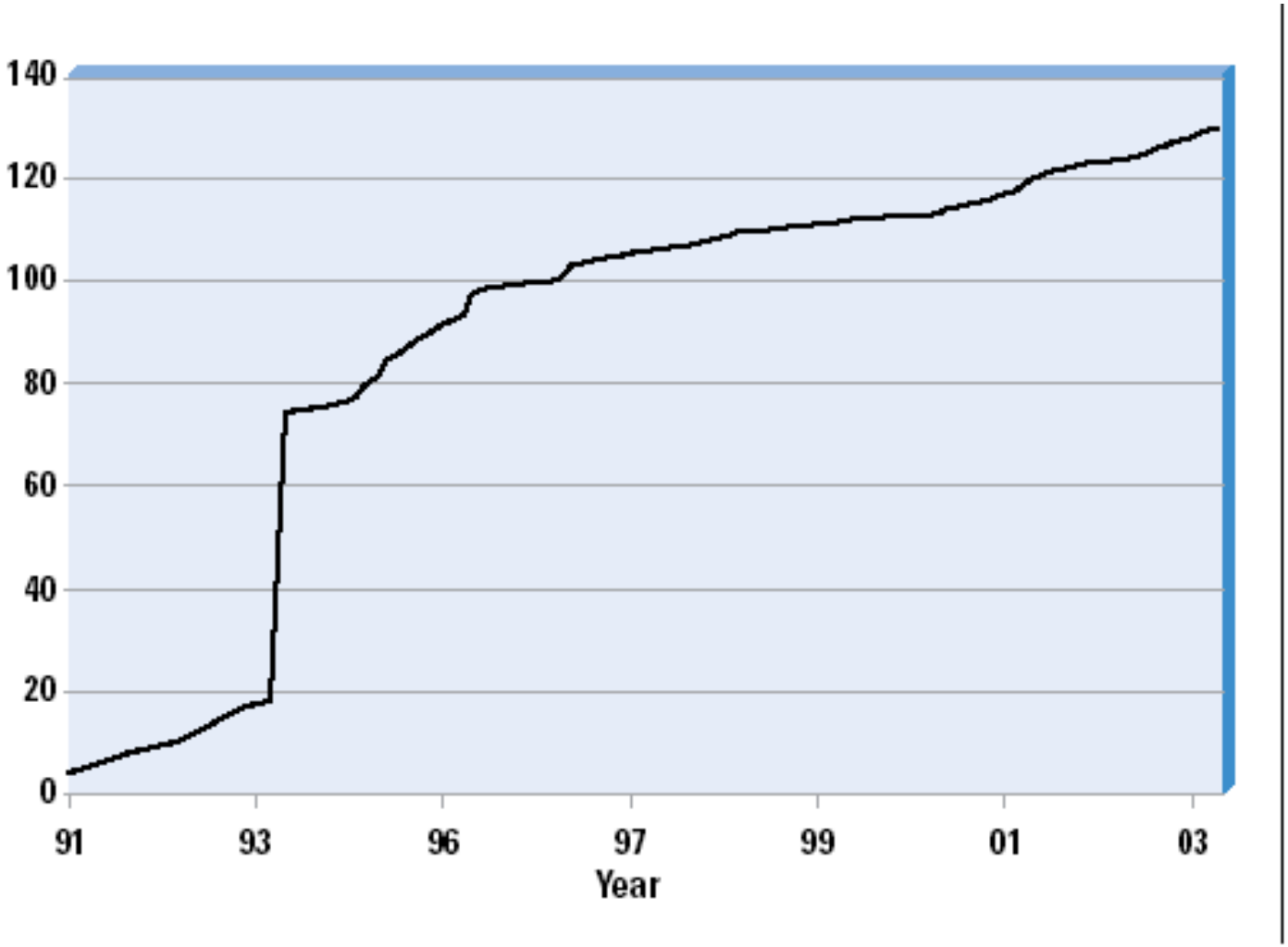
Socket multicast : emission

```
void  emission( int sock )
{  char bfr[ 1024 ]="\nhello !";
   int  nb;
   struct sockaddr_in adr;
   memset( & adr, 0, sizeof( struct sockaddr_in) );
   adr.sin_family          = AF_INET;
   adr.sin_addr.s_addr     = ADDRMULTICAST;
   adr.sin_port            = htons(PORT);
   setsockopt(sock, IPPROTO_IP, IP_MULTICAST_TTL, &TTL, s
   while ( SEND-- ) {
      nb = sendto(sock, bfr, 1024, 0, (struct sockaddr *)&
         if ( nb <= 0 ) {
            if ( nb ) perror("send");
            break;
         }
         sleep(1);
   }
}
```

tcpdump -n igmp or dst 225.0.0.31

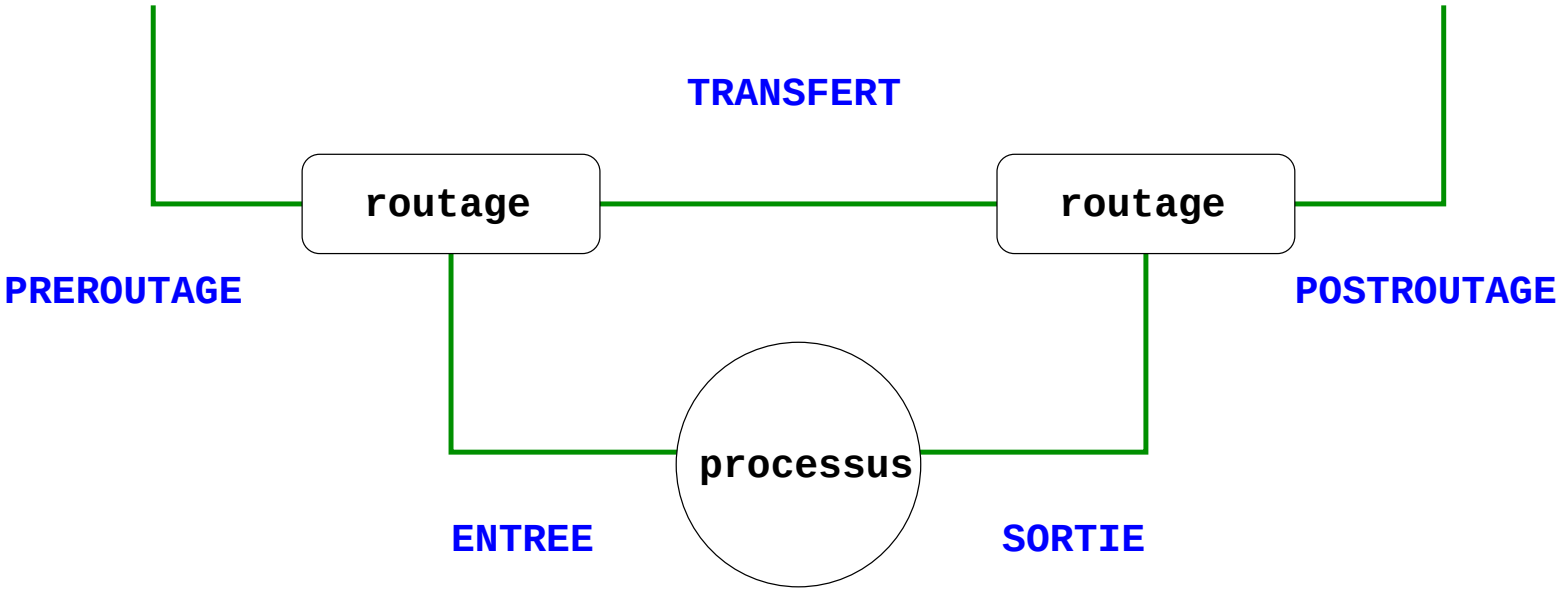
```
listening on eth0, link-type EN10MB (Ethernet), capture siz
16:49:31.014253 IP 192.168.0.1 > 224.0.0.1: igmp query v2
# ./igmp.exe -r
IP 10.2.73.86 > 224.0.0.251: igmp v2 report 224.0.0.251
IP 10.2.73.86 > 225.0.0.31: igmp v2 report 225.0.0.31
IP 10.2.73.86 > 225.0.0.31: igmp v2 report 225.0.0.31
IP 10.2.73.86 > 225.0.0.31: igmp v2 report 225.0.0.31
# ./igmp.exe -s3 -t1
IP 10.2.73.86.32809 > 225.0.0.31.31415: UDP, length 1024
IP 10.2.73.86.32809 > 225.0.0.31.31415: UDP, length 1024
IP 10.2.73.86.32809 > 225.0.0.31.31415: UDP, length 1024
#  killall igmp.exe
IP 10.2.73.86 > 224.0.0.2: igmp leave 225.0.0.31
IP 10.2.75.5 > 225.0.0.31: igmp query v2 [gaddr 225.0.0.31]
IP 10.2.75.5 > 225.0.0.31: igmp query v2 [max resp time 10]
IP 10.2.75.5 > 225.0.0.31: igmp query v2 [max resp time 10]
IP 10.2.75.5 > 224.0.0.1: igmp query v2
IP 10.2.73.86 > 224.0.0.251: igmp v2 report 224.0.0.251
```

Evolution du nombre d'hôtes



En février 2011, la réserve de blocs libres d'adresses publiques IPv4 de l'Internet Assigned Numbers Authority (IANA) est arrivée à épuisement.

chaines de routage

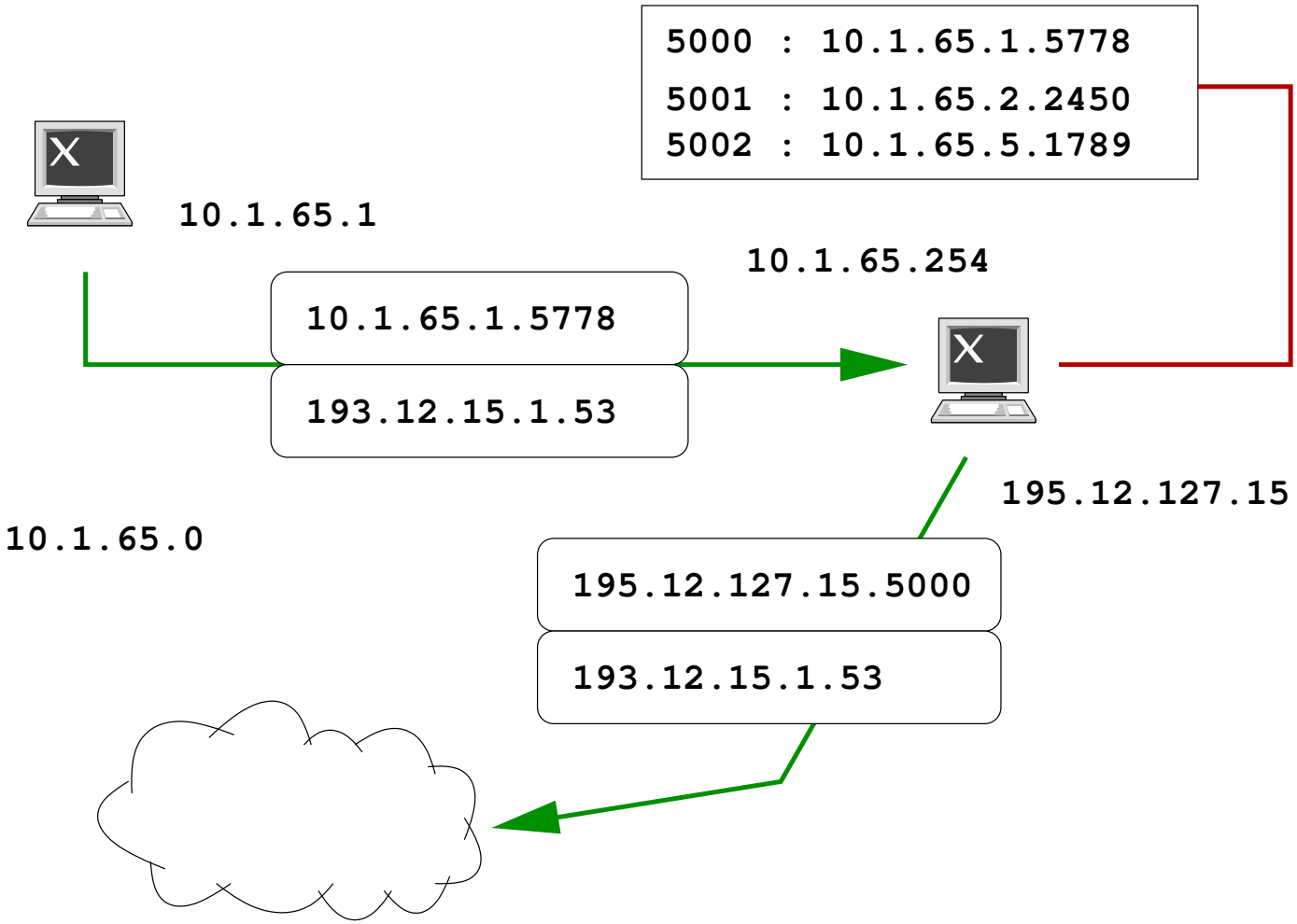


- modification des adresses : 5 zones critiques.
- `iptable`

iptables -L

```
Chain INPUT (policy ACCEPT)
target     prot opt   source  destination
ACCEPT     all  --  anywhere  anywhere
                         state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere  anywhere
ACCEPT     tcp  --  anywhere  anywhere
                         state NEW tcp dpt:ssh
REJECT     all  --  anywhere  anywhere
                         reject-with icmp-host-prohibited
Chain FORWARD (policy ACCEPT)
REJECT     all  --  anywhere  anywhere
                    reject-with icmp-host-prohibited
Chain OUTPUT (policy ACCEPT)
target     prot opt source      destination
```

Translation de ports

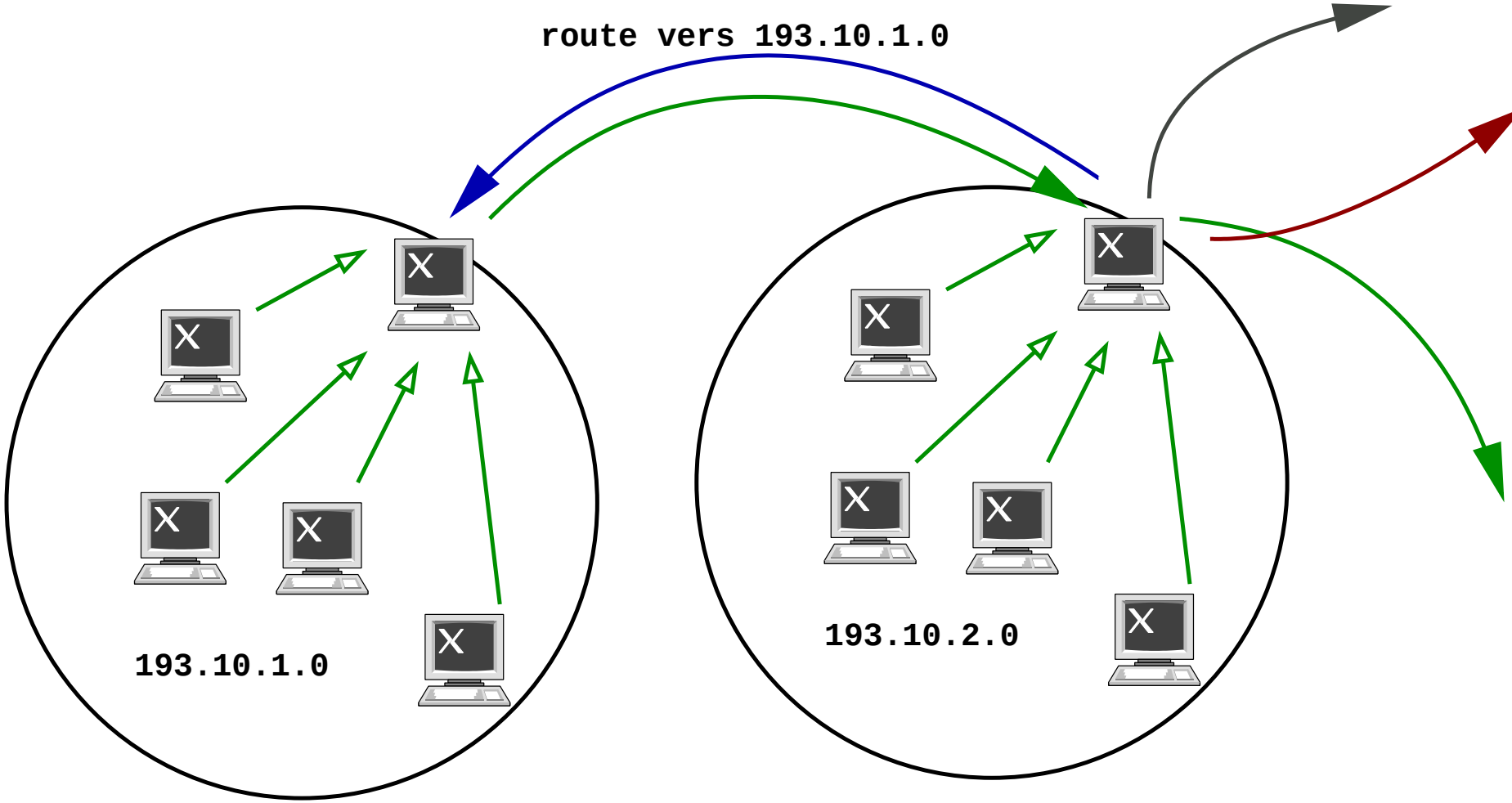


- `iptables -t nat -A POSTROUTING -s 10.1.65.0/24 -o eth0 -j MASQ`

Table de routage

- Un routeur décide de la destination (passerelle, gateway) d'un paquet par consultation de la table de routage et de l'adresse IP du datagramme.
- La maintenance des tables de routage est une opération fondamentale. Elle peut être manuelle, statique ou dynamique.

Algorithme de la patate chaude



- La notion de route par défaut permet la connexion d'un réseau local à un réseau global.
- `route add default gw ip`

Configuration réseau

1. A quoi sert la commande **ipcalc** ?
2. Utiliser les commandes **ifconfig**, **ip**, **netstat** pour déterminer la configuration des interfaces réseau de votre poste.
3. Le répertoire **/etc** contient traditionnellement des informations configurations. Trouver les numéros de port des services : **ftp**, **ssh**, **DNS**.
4. Quels sont les serveurs DNS utilisés par l'hôte. Quels sont les domaines de complétion par défaut.
5. Les serveurs DNS sont ils sur le réseau correspondant à l'interface **eth0** ?
6. Utiliser la commande **arp** pour déterminer l'adresse MAC de la passerelle par défaut.
7. Sur quelle machine est hébergé votre systeème de fichiers ?

ssh

1. Consulter le manuel de la commande **who**.
2. Utiliser la commande **ssh-keygen** pour construire un jeu de clés.
3. Consulter le manuel de la commande **ssh-keygen** pour savoir ce que vous devez faire de ces clés !
4. Ecrire un script pour déterminer la liste des utilisateurs connectés sur les machines du département d'informatique.

ping

1. Consulter le manuel de la commande **ping**
2. Envoyer un ECHO REQUEST à destination du serveur DNS.
3. Envoyer un ECHO REQUEST à destination du serveur DNS, avec un TTL de 1. Commenter.
4. Quelle est la passerelle du site de l'université connectée au réseau publique ?
5. Ecrire un script basé sur **ping** pour tracer les routes.
6. Comparer les routes obtenues à destination du serveur DNS avec celles qui sont données par **traceroute**.

netcat

1. Consulter le manuel de la commande **nc**.
2. Lancer **nc** en écoute sur le port 31415 de l'interface eth0 de votre poste. Il s'agit d'un serveur!
3. Utiliser la commande **netstat** pour vérifier qu'il existe un processus en écoute sur le port 31415.
4. Utiliser **nc** à partir d'une autre machine pour vous connecter à ce port. Il s'agit d'un client !
5. Ecrire un script **cmd.sh** qui lit et exécute des commandes sur l'entrée standard, une commande par ligne, les résultats seront placés sur la sortie standard.
6. Utiliser **nc** pour exécuter **cmd.sh** à distance : **nc -l 31415 | ./cmd.sh**, tout simplement. Le résultat des commandes reste sur la machine distante :-(
7. Utiliser un tube de communication (**mkfifo**), pour renvoyer le résultat des commandes vers le client **nc** : **nc -l 31415 < /tmp/pipo | ./cmd.sh > /tmp/pipo**.