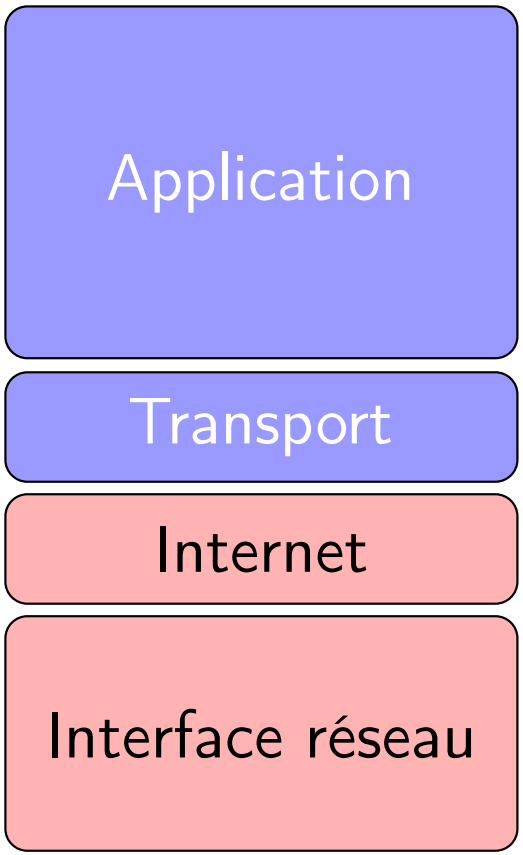
Introduction Protocoles DHCP et NAT

Sophie Robert, Université d'Orléans

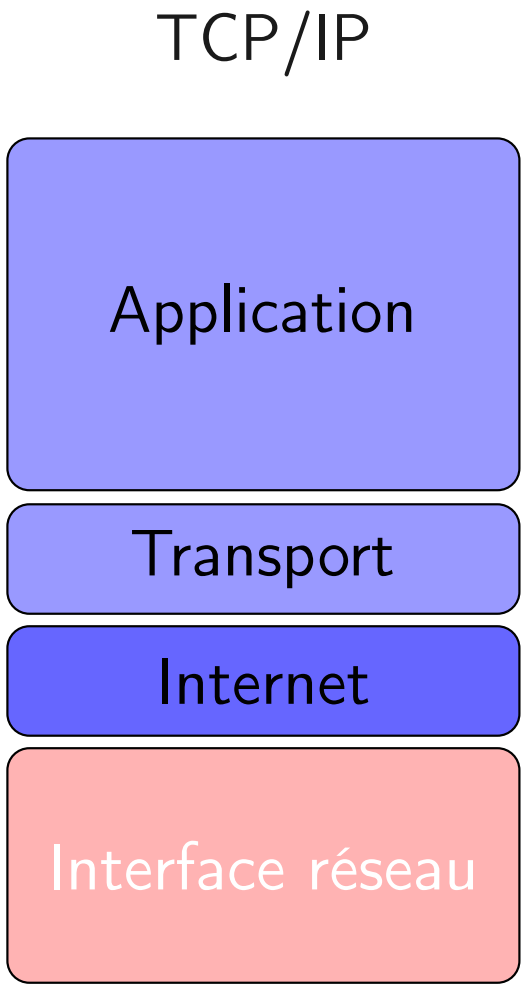
Réseaux 1 Semestre 5

TCP/IP



- Les couches basses : Ethernet, wifi
  - Le médium et les signaux
  - La couche liaison (protocole, accès au support, gestion collision)
- La couche Réseau : IP et routage
  - L'adressage IPv4 (CIDR) et la configuration statique
  - Le routage (static, dynamique : Bellman-Ford/Etats de lien)

Réseaux 2 Semestre 6



- Couche Réseau : complément Adressage et Configuration
  - DHCP
  - NAT
- La couche Transport
  - Fournir des communications de bout en bout entre applications
  - Les protocoles
  - TCP - UDP
- La couche Application
  - Interactions entre hôtes dans le but de délivrer un service à l'utilisateur
  - DNS, SMTP, HTTP (cf le 1er TP de Réseaux 1)

L'organisation de Réseaux 2

Les séances

- 4 CM
- 2 TD (Transport, Application)
- 5 TP (2h ou 3h)
- un lien avec la transition écologique

Les évaluations

- RNE : mixte (2CC + 1CT 1h30) avec 25%CC, 75%CT
  - CC1 : Questionnaire Celene DHCP-NAT après un devoir maison
  - CC2 : Questionnaire Celene Couche Transport - Couche Application (quelques éléments de préparation)
- RSE : 1 CT 1h30
- 2eme session : 1 CT 1h30

DHCP

Dynamic Host Configuration Protocol

Configuration dynamique

Dynamic Host Configuration Protocol

On veut permettre à chaque machine d'obtenir automatiquement :

- une adresse IP ;
- le masque du réseau local ;
- une passerelle par défaut ;
- le nom de domaine ;
- l'adresse d'un serveur DNS pour la résolution.

Serveur DHCP

Les `serveurs DHCP` doivent fournir ces renseignements tout en assurant plusieurs fonctionnalités :

- réutiliser les adresses délaissées ;
- autoriser certaines machines à avoir une IP fixe ;
- se coordonner avec d'éventuels autres serveurs DHCP dédiés au même réseau local.

DHCP est un protocole applicatif encapsulé dans des datagrammes UDP, port serveur 67, port client 68.

Le protocole, RFC 2131

Échange initial standard : **DORA**

- **DHCPDISCOVER** : le client cherche les serveurs DHCP, broadcast.
- **DHCPOFFER** : le serveur propose un bail au client, unicast.
- **DHCPREQUEST** : le client accepte l'offre, broadcast.
- **DHCPACK** : le serveur valide, unicast.

Dans le cas de plusieurs serveurs DHCP, un seul échange arrive à terme, le DHCPREQUEST en broadcast informe les autres serveurs d'un refus.

Le protocole, RFC 2131

No.	Time	Source	Destination	Protocol	Length	Info
12	11.25350…	0.0.0.0	255.255.255.2…	DHCP	342	DHCP Discover - Transaction ID 0x75f55
13	11.25392…	c6:e0:14:0b:d…	Broadcast	ARP	42	Who has 10.1.0.6? Tell 10.1.0.254
14	12.25532…	10.1.0.254	10.1.0.6	DHCP	342	DHCP Offer - Transaction ID 0x75f55
15	12.25574…	0.0.0.0	255.255.255.2…	DHCP	342	DHCP Request - Transaction ID 0x75f55
16	12.26333…	10.1.0.254	10.1.0.6	DHCP	342	DHCP ACK - Transaction ID 0x75f55

- broadcast DHCP : une adresse réservée 255.255.255.255
  - Attention adresse non routable (ne passe pas les routeurs)
- client avant configuration : 0.0.0.0
- unicast : communication vers une adresse offerte

Un contrat, Un bail

Côté Serveur

```
lease 10.1.0.6 {
  starts 1 2025/01/20 09:31:20;
  ends 1 2025/01/20 09:41:20;
  cltt 1 2025/01/20 09:31:20;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 12:73:6f:17:2b:ed;
  client-hostname "pc11";
}
```

- cltt client last transaction time
- Un bail de 10 minutes

Un contrat, Un bail

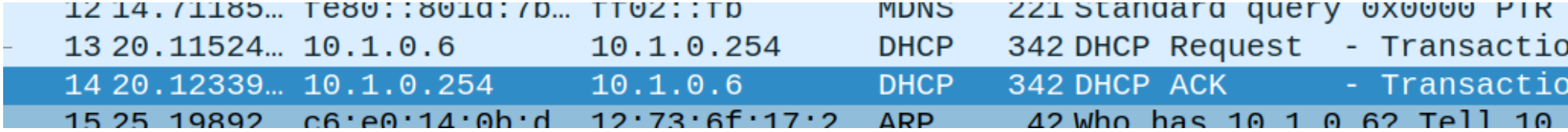
Côté Client

```
lease {
  interface "eth0";
  fixed-address 10.1.0.6;
  option subnet-mask 255.255.0.0;
  option routers 10.1.0.254;
  option dhcp-lease-time 600;
  option dhcp-message-type 5;
  option dhcp-server-identifier 10.1.0.254;
  option domain-name "example.org";
  renew 1 2025/01/20 09:36:12;
  rebind 1 2025/01/20 09:40:05;
  expire 1 2025/01/20 09:41:20;
}
```

- Les indications pour le renouvellement

Renouvellement

Continuité de service



- **DHCPREQUEST** : le client demande le renouvellement.
- **DHCPACK** : le serveur valide, unicast.

Auprès du serveur	sur le broadcast
moitié du bail	7/8ième du bail
A la fin du bail DORA	

Le protocole

Autres types de message

- **DHCPDECLINE** : le client refuse l'adresse IP (e.g. : le client détecte que l'adresse est déjà utilisée).
- **DHCPRELEASE** : le client met fin au bail de manière anticipée.
- **DHCPINFORM** : le client demande les paramètres de configuration (pas une adresse IP).
- **DHCPNACK** : le serveur informe le client de la fin de son bail.

Analyse des échanges

```
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Discover
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, Offer
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Reply 10.0.1.54 is-at 6e:5f:98:37:0c:07
IP 10.0.0.1 > 10.0.1.54: ICMP echo request, id 63671, seq 0
IP 10.0.1.54 > 10.0.0.1: ICMP echo reply, id 63671, seq 0
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
```

Analyse des échanges

```
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Discover
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, Offer
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Reply 10.0.1.54 is-at 6e:5f:98:37:0c:07
IP 10.0.0.1 > 10.0.1.54: ICMP echo request, id 63671, seq 0
IP 10.0.1.54 > 10.0.0.1: ICMP echo reply, id 63671, seq 0
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
```

Analyse des échanges

```
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Discover
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, Offer
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Reply 10.0.1.54 is-at 6e:5f:98:37:0c:07
IP 10.0.0.1 > 10.0.1.54: ICMP echo request, id 63671, seq 0
IP 10.0.1.54 > 10.0.0.1: ICMP echo reply, id 63671, seq 0
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
```

Analyse des échanges

```
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Discover
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, Offer
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Reply 10.0.1.54 is-at 6e:5f:98:37:0c:07
IP 10.0.0.1 > 10.0.1.54: ICMP echo request, id 63671, seq 0
IP 10.0.1.54 > 10.0.0.1: ICMP echo reply, id 63671, seq 0
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
```

Analyse des échanges

```
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Discover
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, Offer
IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Request who-has 10.0.1.54 tell 10.0.0.1
ARP, Reply 10.0.1.54 is-at 6e:5f:98:37:0c:07
IP 10.0.0.1 > 10.0.1.54: ICMP echo request, id 63671, seq 0
IP 10.0.1.54 > 10.0.0.1: ICMP echo reply, id 63671, seq 0
```

Première attribution

```
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
```

Premier renouvellement

```
ARP, Request who-has 10.0.0.1 tell 10.0.1.54
ARP, Reply 10.0.0.1 is-at 0e:ab:f8:0c:10:4b
IP 10.0.1.54.68 > 10.0.0.1.67: BOOTP/DHCP, Request from 6e:5f:98:37:0c:07, Request
IP 10.0.0.1.67 > 10.0.1.54.68: BOOTP/DHCP, Reply, ACK
```

Deuxième renouvellement

L'entête DHCP (BOOTP)

op	htype	hlen
hops	xid	secs
flags	ciaddr	yiaddr
siaddr	giaddr	chaddr
sname		
file		
options		

Table – Structure de l'en-tête DHCP sous forme simplifiée.

Les différents champs

Champ	Taille	Description
`op`	1 octet	Type de message : 1-Request/2-Reply.
`htype`	1 octet	Type de matériel (1 pour Ethernet).
`hlen`	1 octet	Lg de l'adresse matérielle (6 pour MAC).
`hops`	1 octet	Nombre de relais traversés (initialement 0).
`xid`	4 octets	Identifiant de transaction (choisi par le client).
`secs`	2 octets	Temps écoulé depuis le début de la requête DHCP (en secondes).
`flags`	2 octets	Indicateurs, incluant le flag `Broadcast` (0/1-unicast/broadcast).

Table – Structure de l'en-tête DHCP (basée sur BOOTP).

Les différents champs

Champ	Taille	Description
`ciaddr`	4 octets	Adresse IP du client (utilisée si le client connaît déjà son adresse).
`yiaddr`	4 octets	Adresse IP attribuée au client par le serveur DHCP.
`siaddr`	4 octets	Adresse IP du serveur DHCP (optionnel).
`giaddr`	4 octets	Adresse IP du relais DHCP (si utilisé).
`chaddr`	16 octets	Adresse matérielle du client (adresse MAC).

Table – Structure de l'en-tête DHCP (basée sur BOOTP).

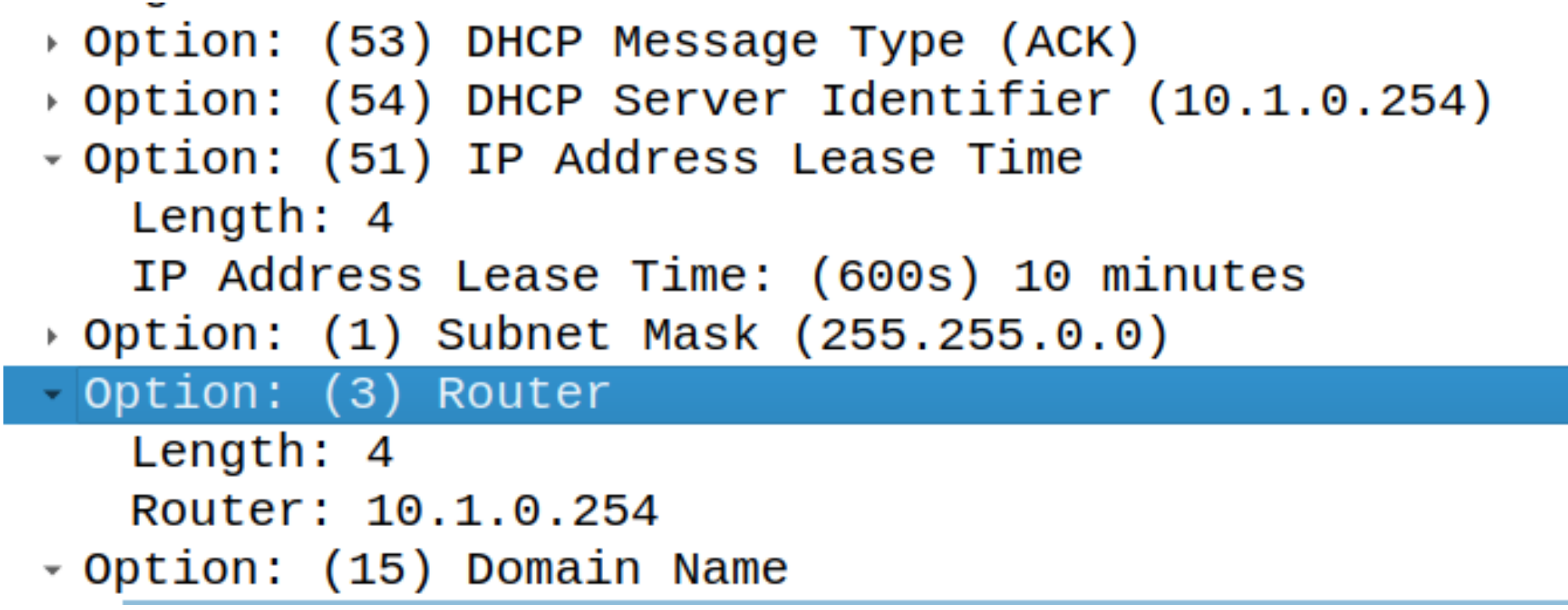
Les différents champs

Champ	Taille	Description
`sname`	64 octets	Nom du serveur DHCP (optionnel).
`file`	128 octets	Nom du fichier de démarrage (si utilisé pour BOOTP ou PXE).
`options`	Variable	Options supplémentaires (ex. masque de sous-réseau, passerelle, etc.).

Table – Structure de l'en-tête DHCP (basée sur BOOTP).

Adresse IP mais aussi

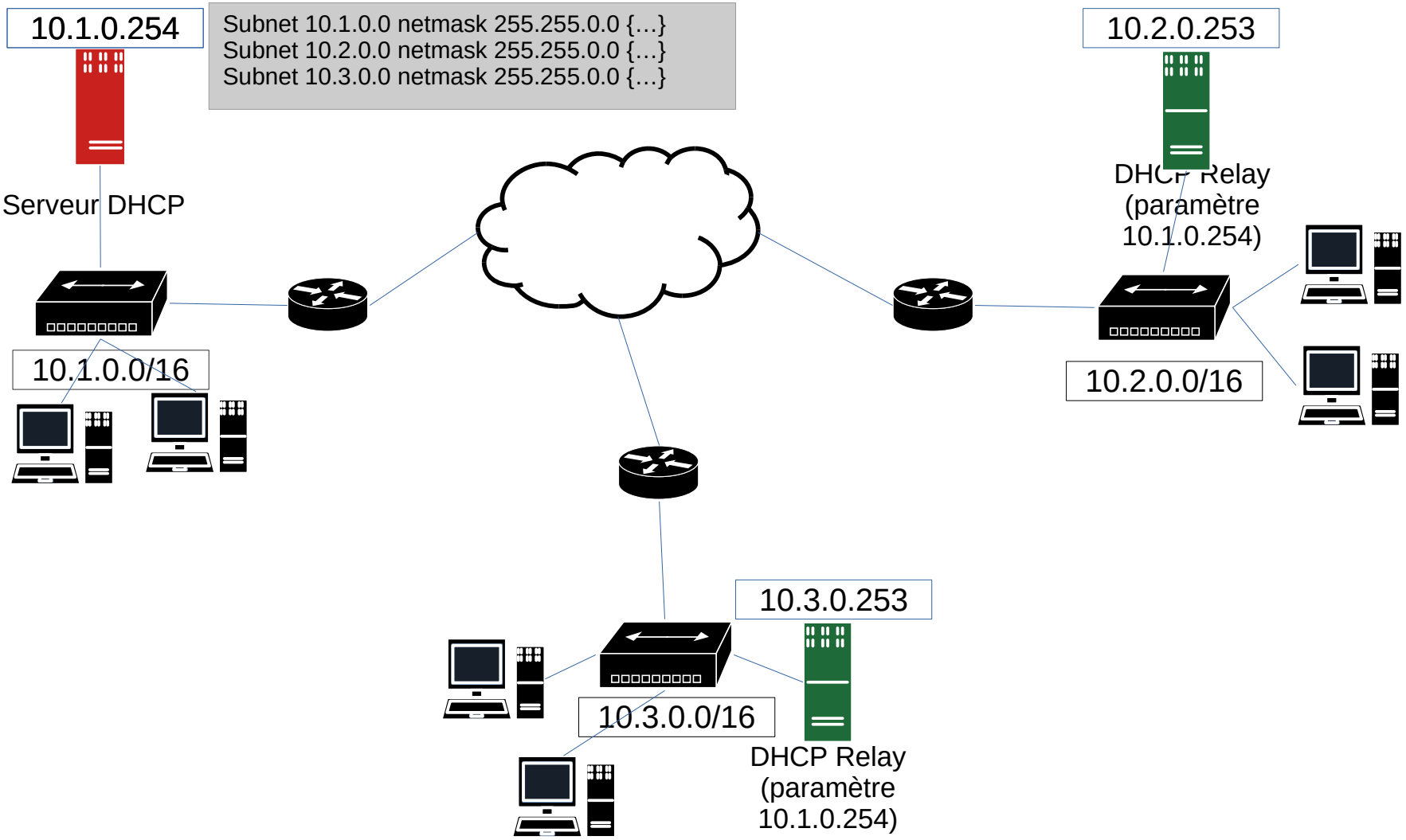
Quelques détails d'une offre



255.255.255.255 non routable ?

- Un serveur DHCP par sous réseau
- Mise en place d'un `dhcp relay`

255.255.255.255 non routable ?



Attribution d'adresses par DHCP

- Dynamique à partir d'une plage d'adresses
- Manuelle/Statique
  - Retrouver la même adresse IP à chaque fois.
  - Critère : son adresse MAC

Toujours DHCP ?

- Serveurs (web, imprimante,...) : attribution manuelle
- Serveur DHCP, Relais, Routeurs : configuration statique sans DHCP.

DHCP en pratique

Solution DHCP libre très répandue sous linux : `isc-dhcp-client`, `isc-dhcp-server` et `isc-dhcp-relay`.

Côté client :

- lancement du client avec dhclient
- configuration dans /etc/dhcp/dhclient.conf
- baux dans /var/lib/dhcp/dhclient.leases

```
send host-name = gethostname();
request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, domain-search, host-name,
        dhcp6.name-servers, dhcp6.domain-search, dhcp6.fqdn, dhcp6.
        netbios-name-servers, netbios-scope, interface-mtu,
        rfc3442-classless-static-routes, ntp-servers;
```

DHCP en pratique

```
send host-name = gethostname();
request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, domain-search, host-name,
        dhcp6.name-servers, dhcp6.domain-search, dhcp6.fqdn, dhcp6.sr
        netbios-name-servers, netbios-scope, interface-mtu,
        rfc3442-classless-static-routes, ntp-servers;
```

- On peut utiliser `require` à la place de `request`. Dans ce cas l'offre n'est acceptée que si les demandes ont une réponse dans l'offre.

DHCP en pratique

Côté serveur

- lancement du serveur avec /etc/init.d/isc-dhcp-server start
- configuration dans /etc/dhcp/dhcpd.conf
- baux dans /var/lib/dhcp/dhcpd.leases

Configuration serveur *isc-dhcp*

```
subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.50 192.168.0.100 ;
        option routers 192.168.0.1 ;
        option domain-name "green.net" ;
        option domain-name-servers 3.4.5.6 ;
}
```

Configuration serveur *isc-dhcp*

```
option domain-name "iiia.net" ;
option domain-name-servers 10.0.1.1 ;

subnet 10.0.0.0 netmask 255.0.0.0 {
        range 10.0.0.1 10.0.0.50 ;

host pc1 { hardware ethernet 6e :5f :98 :37 :0c :07 ; fixed-address pc1.iiia.net ; }
host pc2 { hardware ethernet 2e :18 :cc :d4 :8c :e7 ; fixed-address pc2.iiia.net ; }
host pc3 { hardware ethernet 2e :fe :a2 :81 :23 :ce ; fixed-address pc3.iiia.net ; }
host pc4 { hardware ethernet ce :53 :0c :0c :ef :61 ; fixed-address pc4.10.0.0.250 ; }
}
```

Dnsmasq

Sous linux, l'utilitaire `dnsmasq` fournit :

- serveur DHCP ;
- relais DNS ;
- serveur BOOTP, protocole d'amorçage ;
- serveur TFTP, protocole de transfert de fichier utilisé au démarrage (e.g. démarrer depuis une carte réseau).

Configuration serveur *dnsmasq*

Dans /etc/dnsmasq.conf

```
interface=eth0          # pour limiter l'écoute DHCP à cette interface
domain=green.net

dhcp-authoritative      # si un seul serveur DHCP
dhcp-leasefile=/tmp/dhcp.leases



# Plage DHCP
dhcp-range=192.168.0.50,192.168.0.100,12h
# Netmask
dhcp-option=1,255.255.255.0
# Route
dhcp-option=3,192.168.0.1
# Serveur DNS
dhcp-option=6,3.4.5.6
# IP fixe
dhcp-host=6e :5f :98 :37 :0c :07,192.168.0.27
```

NAT

Network Address Translation

Adresses IP privées ?

Non visibles de l'extérieur

Plusieurs blocs d'adresses privées réservés :

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

- 169.254.0.0/16 (Automatic Private IP Addressing)
  - non routable - échec de DHCP

Utiles pour économiser des adresses IPv4 !
Comment utiliser ces adresses sur internet ?
Exemple : le FAI ne fournit qu'une seule adresse publique et la box fait l'interface avec le réseau local.

Address Translation

Plusieurs options, mais dans tous les cas :

- il faut réécrire les adresses privées ;
- en utilisant la ou les adresse(-s) publique(-s) à disposition ;
- changer l'adresse source des paquets sortants ;
- changer l'adresse destination des paquets entrants.

Static NAT

Static Network Address Translation

Si l'on dispose d'autant d'adresses privées que publiques :

- table de correspondance $1-1$ fixe ;
- Dans les paquets sortants, l'adresse privée est remplacée par l'adresse publique qui lui est associée ;
- Dans les paquets entrants, l'adresse publique est remplacée par l'adresse privée correspondante ;
- Intérêts ?
  - Masquer son architecture interne
  - Indépendance des adresses publiques

Dynamic NAT

Dynamic Network Address Translation

Table de correspondance dynamique entre un pool de N adresses publiques et un ensemble de M adresses privées avec

- $N \leq M$
- Une IP publique est associée temporairement à une adresse privée
- L'attribution est dynamique

La translation d'adresse n'est basée que sur l'association simultanée d'autant d'adresses privées que d'adresses publiques (N au maximum).

NAT - PAT

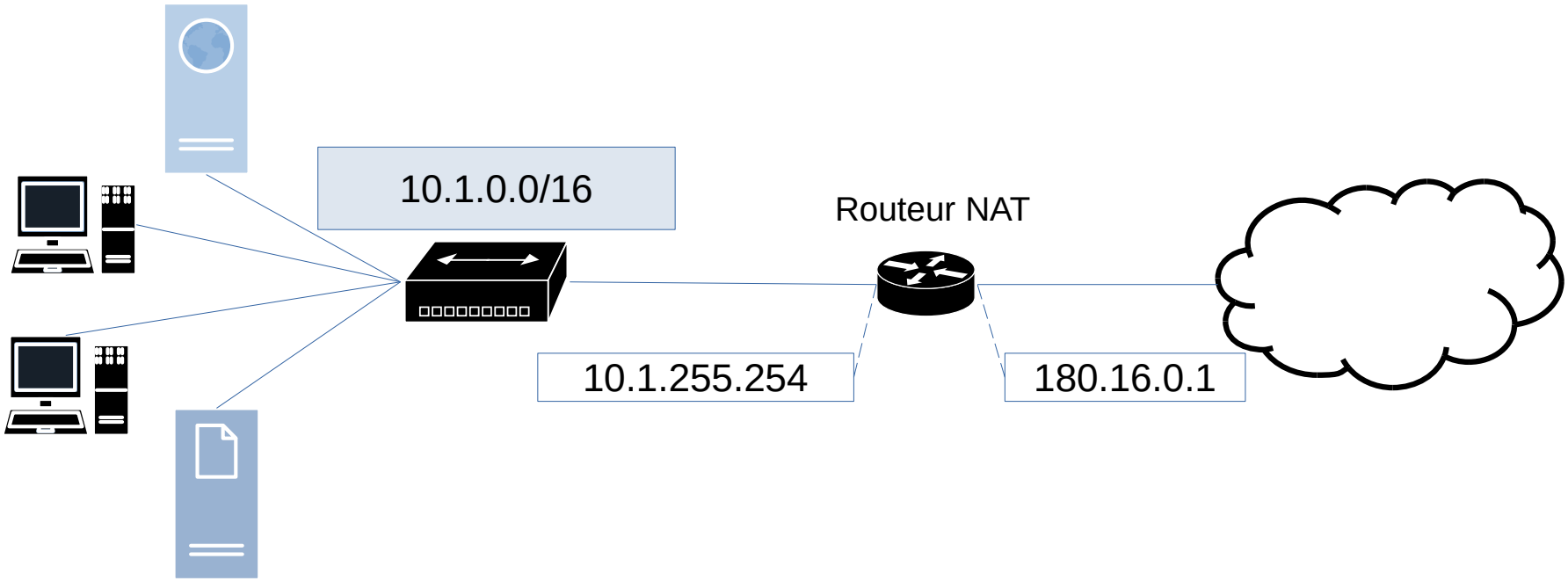
Port Address Translation

Si l'on dispose de moins d'adresses publiques, par exemple une seule :

- la passerelle transforme l'adresse privée en l'adresse publique dans les paquets sortants en choisissant un numéro de port approprié ;
- processus inverse pour les paquets entrants : le numéro de port du paquet permet à la passerelle de choisir l'adresse privée de l'hôte concerné ;
- la passerelle maintient une table d'association.
- Quid des protocoles sans port ?

NAT - PAT

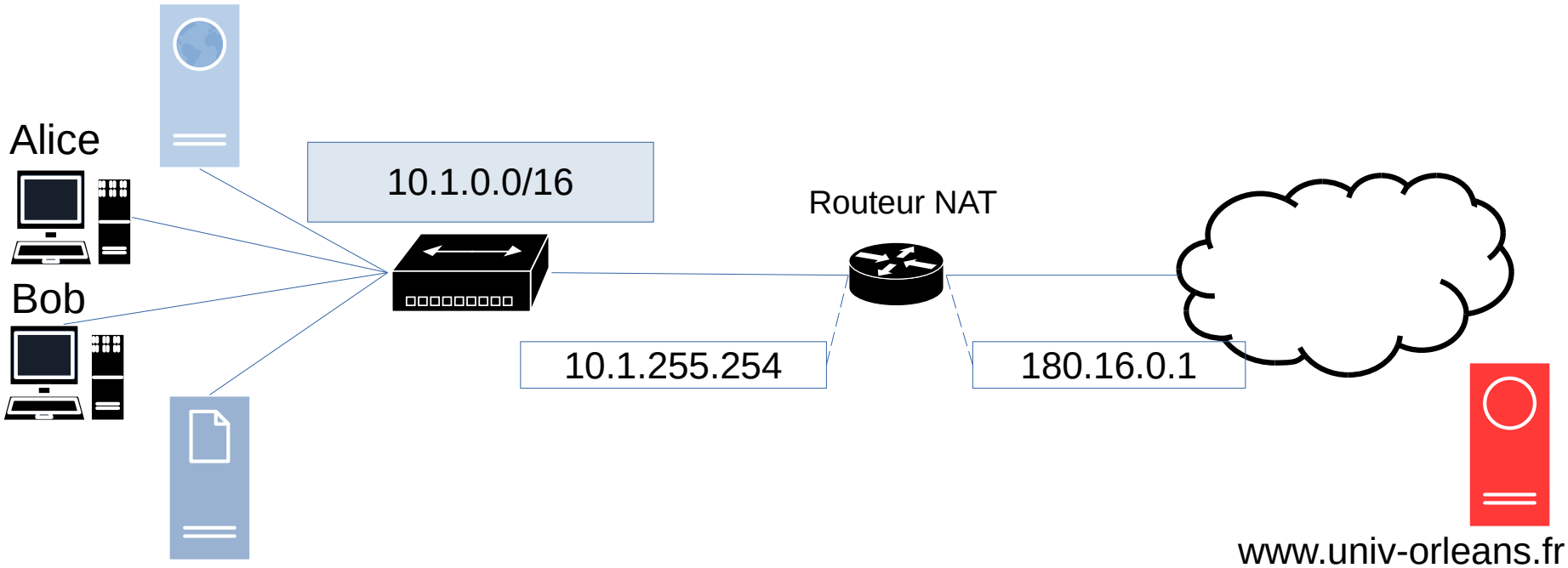
Port Address Translation



- Une table de Translation
- Qui initie la communication ?
- Comment gérer les serveurs (web, ...) ?

NAT - PAT

Port Address Translation

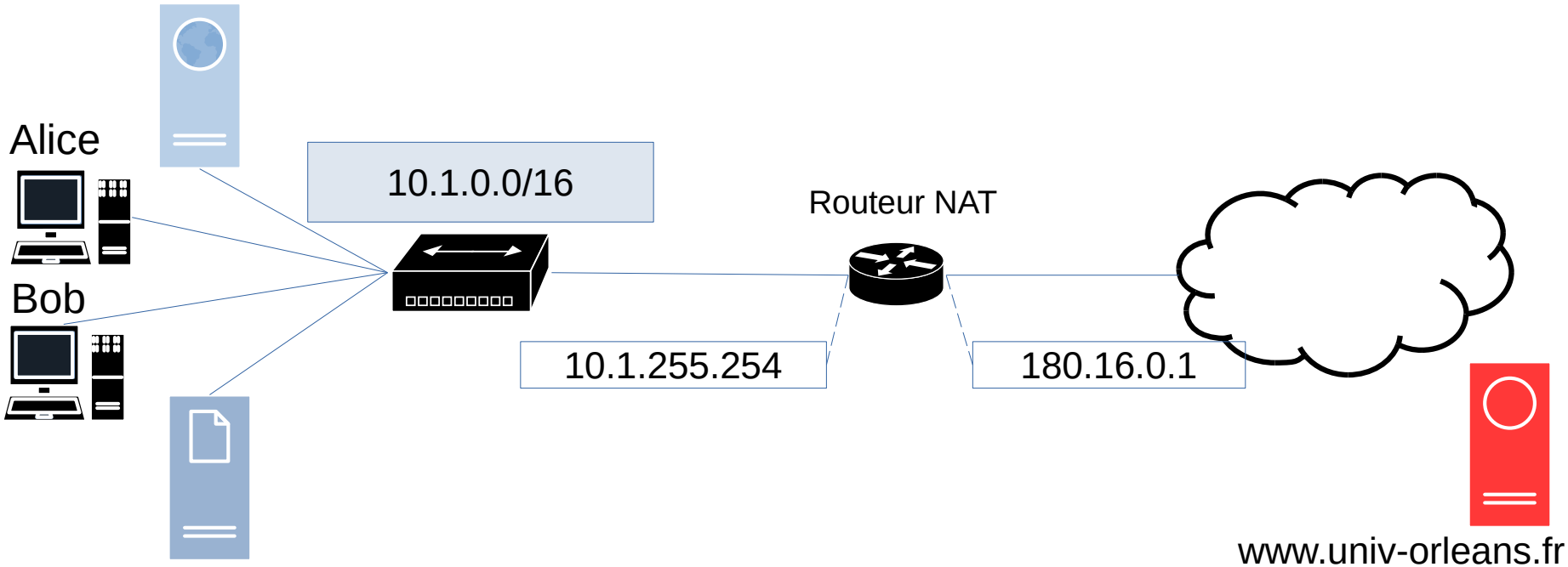


Alice `http://www.univ-orleans.fr`

Privée		Publique	
Adresse	Port	Adresse	Port
Alice : 10.1.0.1	3456	180.16.0.1	4455

NAT - PAT

Port Address Translation

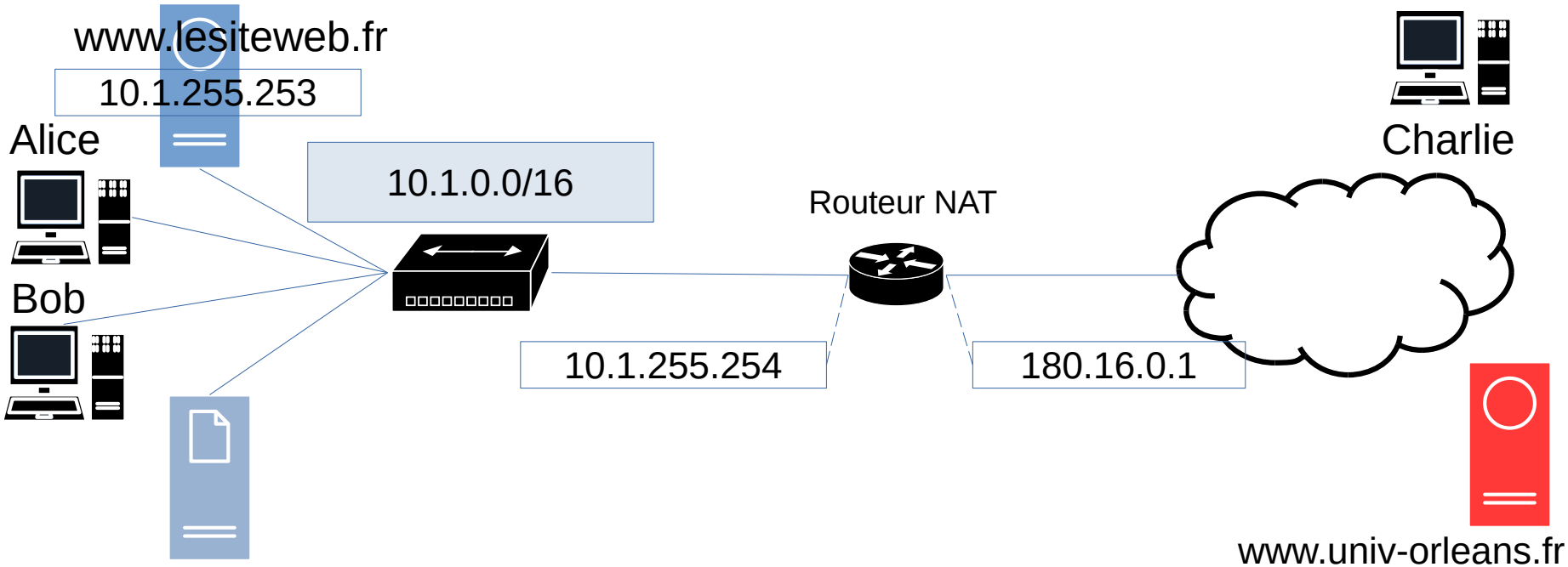


Alice et Bob `http://www.univ-orleans.fr`

Privée		Publique	
Adresse	Port	Adresse	Port
Alice : 10.1.0.1	3456	180.16.0.1	4455
Bob : 10.1.0.2	3457	180.16.0.1	4456

NAT - PAT

Port Address Translation

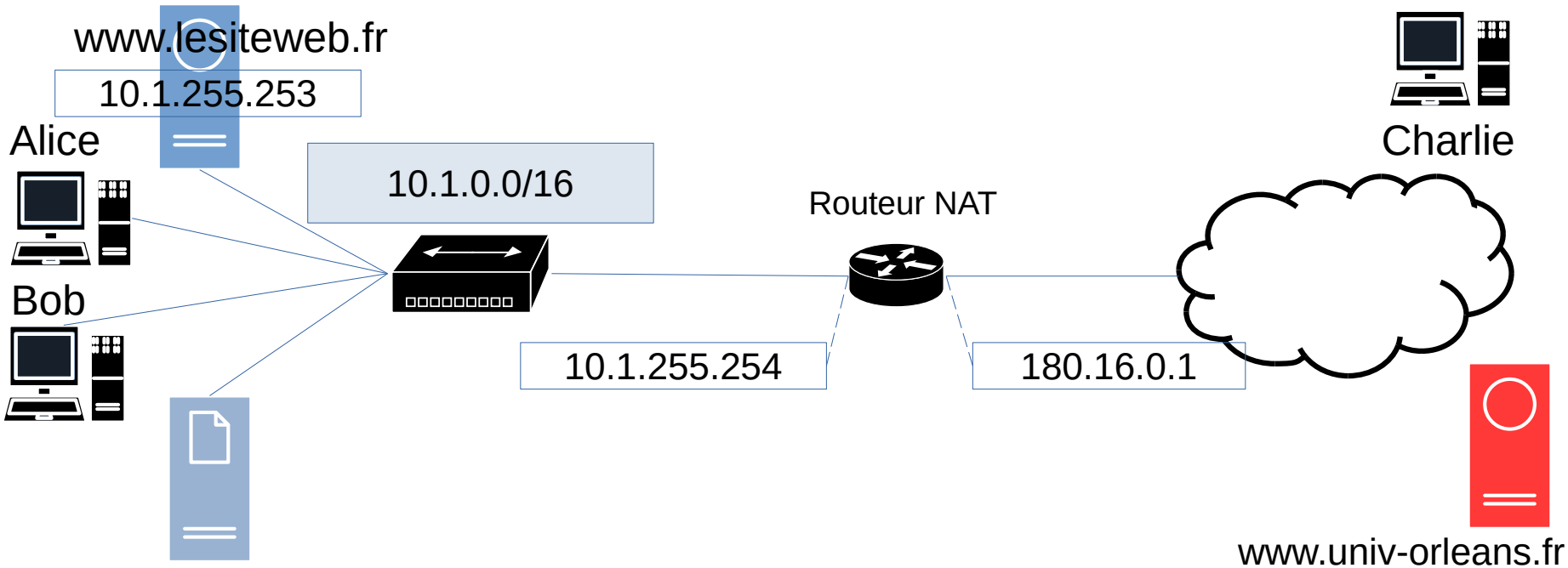


Charlie `http://www.lesiteweb.fr` ?????

- Charlie ne connaît que 180.16.0.1 :80 pour sa requête
- Il faut au niveau du routeur remplacer 180.16.0.1 :80 dans le paquet entrant par 10.1.255.253 :80.

NAT - PAT

Port Address Translation



Charlie `http://www.lesiteweb.fr`

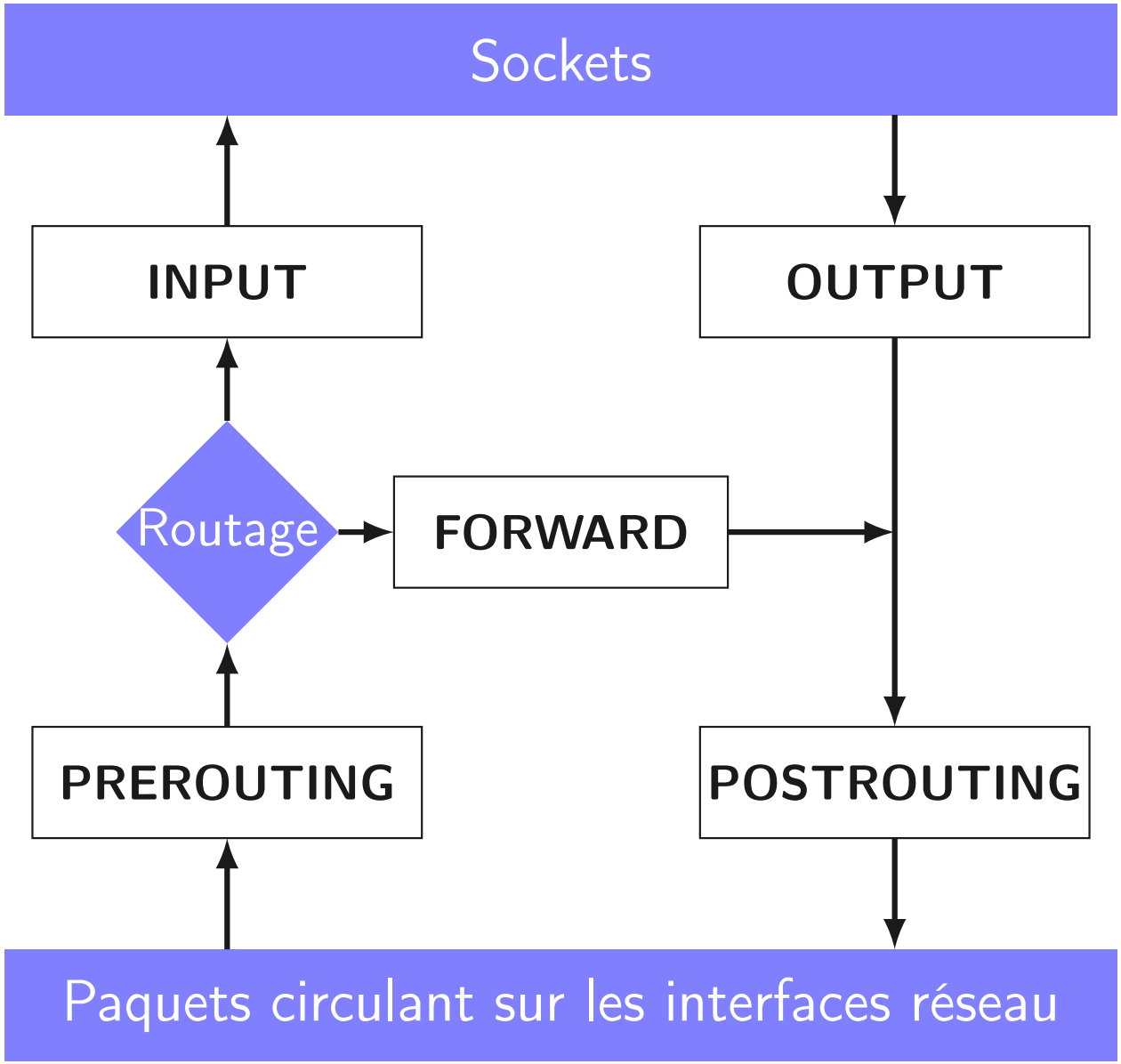
Mettre en place du Port Forwarding pour rediriger des requêtes entrantes.

NAT en pratique : iptables

Utilitaire linux (`http://netfilter.org`).

- Permet de faire du filtrage réseau au niveau du noyau linux.
- Implémente NAT-PAT mais aussi des pare-feux.
- paquet `iptables`.
- Fonctionnement par tables qui regroupent des ensembles de règles, e.g. : NAT.

Tables iptables



Commandes iptables

iptables -t TABLE { -A, -I, -D } CHAINE REGLE

- TABLE : contexte, par exemple nat, filter,...
- CHAINE : chaque table est une collection de chaînes.
- REGLE : chaque chaîne est une liste ordonnée de règles (conditions + cible).
- -A/I/D

  -A pour ajouter,
  -I *i* insérer la règle à la position *i* dans la chaîne,
  -D supprimer.

Construire un filtre

Possibilité de :

- créer ou supprimer des tables, des chaînes.
- envoyer vers une cible -j CIBLE, en particulier
  - ACCEPT
  - REJECT (avec message d'erreur ICMP)
  - DROP (sans message d'erreur)
  - ou une autre chaîne.
- utiliser des conditions pour appliquer la règle sur l'adresse ou le port (src ou dst), le protocole, l'interface (entrée ou sortie), le type de message,...

Exemple SNAT

NAT source

iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 180.16.0.1

- Remplacer l'adresse source (réseau privé) par l'adresse passée en paramètre.
- En POSTROUTING pour connaître l'adresse IP de l'interface de sortie (e.g. routeur à 3 interfaces).
- La box (ou routeur NAT) écrit dans une table l'association effectuée, si un paquet réponse arrive sur le port choisi, l'adresse destination sera substituée.

Exemple DNAT

Port Forwarding NAT destination

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
                              -j DNAT --to 10.1.255.253
```

- Remplacer l'adresse de destination (adresse publique de la box) par l'adresse dans le réseau privé.
- Utile par exemple pour héberger un serveur web dans un réseau privé (derrière un pare-feu).
- Un paquet est transmis uniquement s'il contient du TCP sur le port 80.

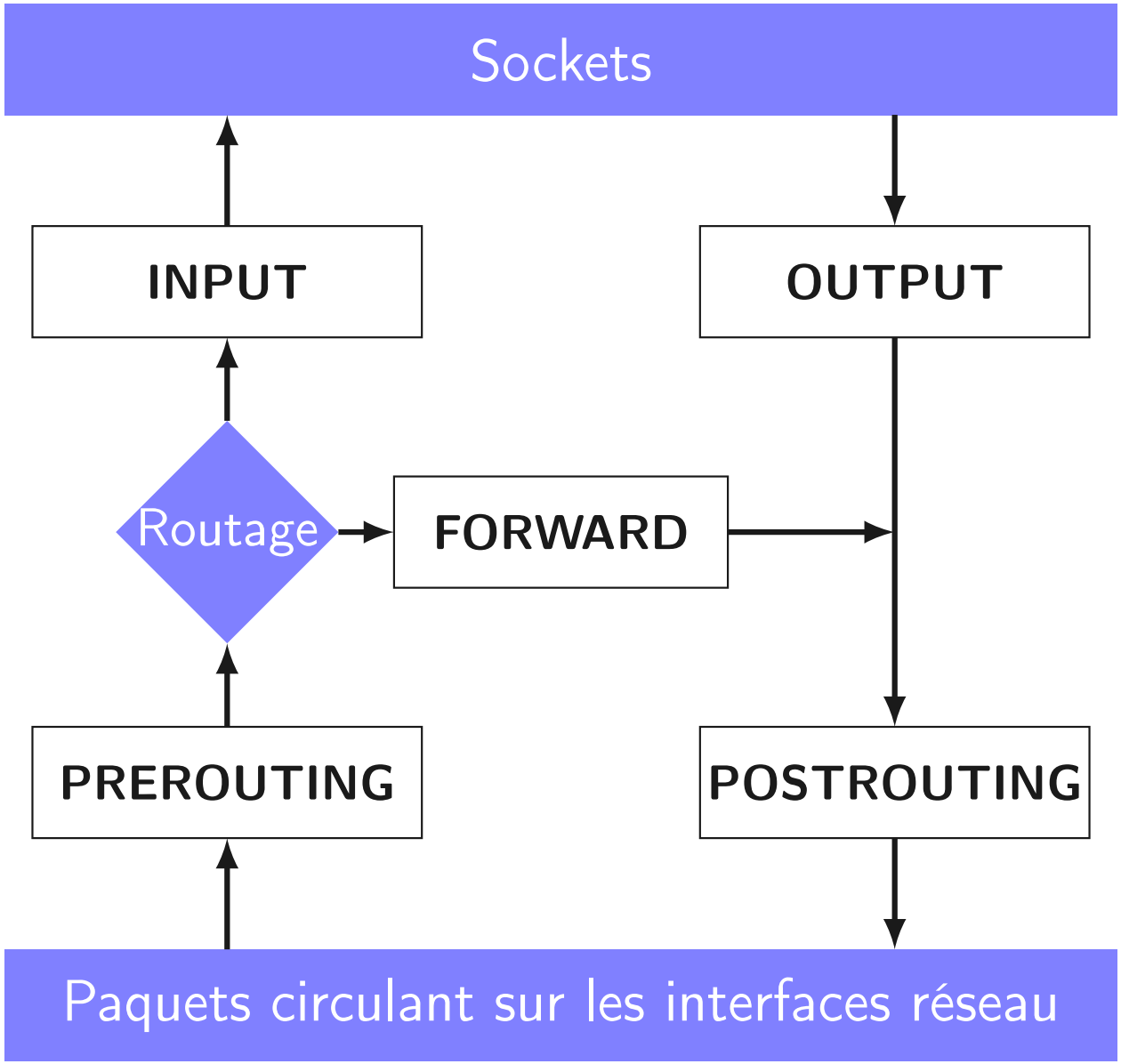
Pare-feu

La table principale `filter` (table par défaut) pour configurer le pare-feu :

- Chaîne `INPUT` pour les paquets entrants (après décision de routage).
- Chaîne `OUTPUT` pour les paquets sortants.
- Chaîne `FORWARD` pour les paquets à transférer (après décision de routage).

Politique de contrôle et sécurité. La décision du sort des paquets est prise après lecture des entêtes de couche 2, 3 et 4.

Pare-feu avec iptables



Autres usages

DMZ

Configurer un routeur/pare-feu à 3 pattes avec :

- eth0 vers l'extérieur (internet) ;
- eth1 vers la DMZ qui contient un serveur HTTP ;
- eth2 vers le réseau d'entreprise qui contient un serveur PostgreSQL accessible au serveur HTTP sur le port TCP 5432.

On supposera que l'extérieur n'accède qu'au serveur HTTP, le réseau d'entreprise à tout l'internet et la DMZ à l'extérieur et au serveur PostgreSQL uniquement.

Autres usages

DMZ

Préparer le terrain.

```
iptables -t filter -F FORWARD
iptables -t filter -F INPUT
iptables -t filter -F OUTPUT
iptables -t nat -F PREROUTING
iptables -t nat -F POSTROUTING
```

Autres usages

DMZ

Règles par défaut

```
iptables -t filter -P FORWARD DROP
iptables -t filter -P INPUT DROP
iptables -t filter -P OUTPUT DROP
```

Autres usages

DMZ

Traffic local.

```
iptables -t filter -A INPUT -i lo -j ACCEPT
iptables -t filter -A OUTPUT -o lo -j ACCEPT
```

Autres usages

DMZ

Autoriser les connections SSH sur le routeur.

```
iptables -t filter -A INPUT -i eth2 -p tcp --dport 22 -j ACCEPT
iptables -t filter -A OUTPUT -o eth2 -p tcp --sport 22 -j ACCEPT
```

Autres usages

DMZ

NAT source : le LAN accède à internet.

```
iptables -t nat -A POSTROUTING -s 192.168.20.0/24 -o eth0 -j SNAT
--to 1.2.3.4
```

```
iptables -A FORWARD -i eth2 -o eth0 -j ACCEPT
```

```
iptables -A FORWARD -i eth0 -o eth2 -j ACCEPT
```

Autres usages

DMZ

NAT destination : internet peut se connecter au serveur web.

```
iptables -t nat -A PREROUTING -d 1.2.3.4 -p tcp --dport 80 -j DNAT
--to-destination 192.168.10.2:80
```

```
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -m state
--state NEW,ESTABLISHED -d 192.168.10.2 -j ACCEPT
```

```
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --sport 80 -m state
--state ESTABLISHED -s 192.168.10.2 -j ACCEPT
```

Le serveur web ne peut pas initier une communication.

Autres usages

DMZ

Le serveur web peut interroger le serveur SQL.

```
iptables -A FORWARD -i eth1 -o eth2 -p tcp -s 192.168.10.2 --dport 5432
-m state --state NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A FORWARD -i eth2 -o eth1 -p tcp -d 192.168.10.2 --sport 5432
-m state --state ESTABLISHED -j ACCEPT
```

Autres usages

DMZ

Le LAN peut demander la page web.

```
iptables -A FORWARD -i eth2 -o eth1 -p tcp -d 192.168.10.2 --dport 80
-m state --state NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A FORWARD -i eth1 -o eth2 -p tcp -s 192.168.10.2 --sport 80 -m
state --state ESTABLISHED -j ACCEPT
```

Droits

Merci à l'équipe historique Réseaux en L3 Nicolas Ollinger, Martin Delacourt et Mathieu Liedloff.