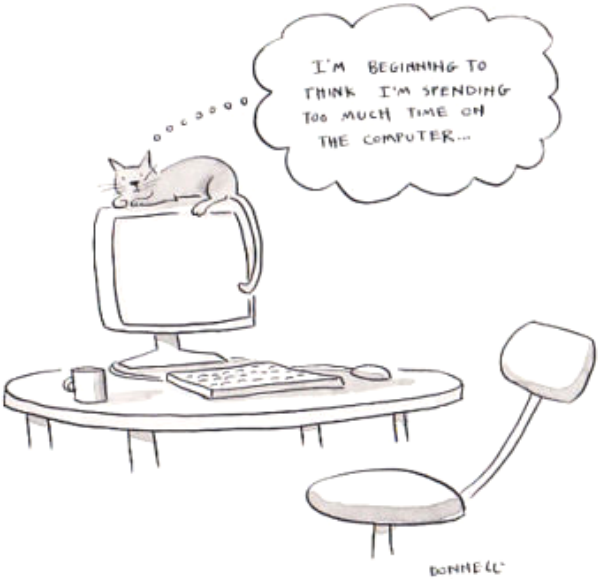
Faculté des Sciences & Techniques

Université de Limoges

Master 1ère année



Infrastructure Réseaux — P-F. Bonnefoi

Version du 5 février 2024

Table des matières

1 Internet et routage dynamique 4
La préhistoire du routage 7
Internet et routage : la notion d'AS 14
InterNet : une collection d'AS organisées en «*transit*», «*Peer*» ou «*Customer*» 23
Le routage dynamique : routage interne ou externe aux AS 33
Algorithme de routage : vecteur de chemin, «*Path Vector Routing*» 36
BGP, «*Border gateway Protocol*», RFC 4271 40
Algorithme de routage : vecteur de distance, «*vector-distance*» 48
RIPv2, «*Routing Internet Protocol*», RFC 2453 58
Algorithme de routage : par état de lien, «*state-link*» 60
OSPF, «*Open Shortest Path First*», RFC 2328 68
Faire le point 73
Linux comme routeur : RIP, OSPF, BGP... 74
Sécurité et BGP : Sécuriser les sessions extérieures de BGP : eBGP 77
2 Routage, Règle de routage et Politique de routage 79
La «triade» des fondamentaux 82
Routage & Linux 84
«*Routing Policy*» & le firewall NetFilter 92
Un exemple : fusionner deux réseaux identiques 95
3 VPNs & Tunnels 101

Quelques protocoles de niveau 2 104
Tunnel de niveau 3 : GRE, «*Generic Routing Encapsulation*» 108
Tunnel de niveau 3 : IPSec 110
Pourquoi un VPN et Quel VPN ? 115
Les avantages d'OpenVPN 118
WireGuard 119
Socat & TUN 122
Tun & Tap : la programmation 126
Un VPN «Light» avec SSH 129
QoS : «*packet switching*» avec MPLS 131

Remarques sur le routage

- La table de routage : comment en diminuer leur taille ?
  - le routage est effectué de **saut en saut**, «*next hop*», depuis la source jusqu'à la destination ;
  - le routage ne prend en compte que l'**adresse de destination** du datagramme.

- À chaque saut :
  - le routeur prend une **décision autonome** pour la sélection de la route empruntée par le datagramme : la meilleur décision en fonction de sa table de routage (*Best effort*) ;
  - un routeur n'a qu'une **connaissance partielle** du routage ;
  - la notion de «*route par défaut*» permet de réaliser un routage sans connaitre toutes les destinations possibles.

- Trouver une nouvelle route en cas de panne d'un routeur ?
  - les tables de routage doivent être **cohérentes** tout le temps ;
  - les routeurs voisins de celui qui est en panne doivent **modifier leur table de routage** pour acheminer les datagrammes suivants par une nouvelle route ;
  - le routage IP peut être **dynamique**.

Routage statique et dynamique

- **Constantes** :
  - routage en fonction de l'adresse destination uniquement ;
  - routage de proche en proche : chaque routeur prend la décision qui lui paraît la meilleure (en fonction de sa table de routage).

- **Routage statique**
  - Utilisation de la commande «ip address» pour la configuration d'une interface ;
  - Utilisation de la commande «ip route» qui permet d'indiquer un chemin vers : un réseau (net), un équipement (host) ou une route par défaut (default).

- **Routage dynamique** : utilisation d'un «*Routing protocol*», algorithme de routage :
  - Il sert à l'échange d'informations de routage pour la construction automatique de tables de routage.

Statique vs Dynamique

- **Routage statique**
  - réalisation des tables de routage à la main : difficultés des mises à jour ;
  - convient pour des réseaux de taille réduite ou pour un réseau où la *l'aspect sécurité est important* ;
  - en cas de panne, le routage est modifié uniquement après la découverte de cette panne.

- **Routage dynamique**
  - adaptation aux conditions du réseau ;
  - adaptation à des réseaux de grande taille ;
  - découverte automatique de la modification de la topologie.

Les protocoles de routage

Le but d'un protocole de routage est de **maintenir les tables de routage** de manière cohérente (le but n'est pas de router).

Le protocole de routage travaille :

- en fonction d'une **métrique** ou d'un coût associé à chaque lien de communication :
  Le coût d'un lien varie de façon dynamique (mesure de la congestion par exemple) ou non (mesure faite par rapport au débit).

- en exploitant une **connaissance de la topologie** du réseau :
  - **globale**
    - chaque routeur connait toute la topologie du réseau d'interconnexion : tous les routeurs et toutes les liens entre ces routeurs ;
  - **locale**
    - chaque routeur ne connait que les routeurs adjacents (auquel il est directement connecté) et les liens qui le relient à eux (voisinage) ;

- suivant un **fonctionnement** :
  - centralisé ou distribué ;
  - itératif (local) ou direct (global).

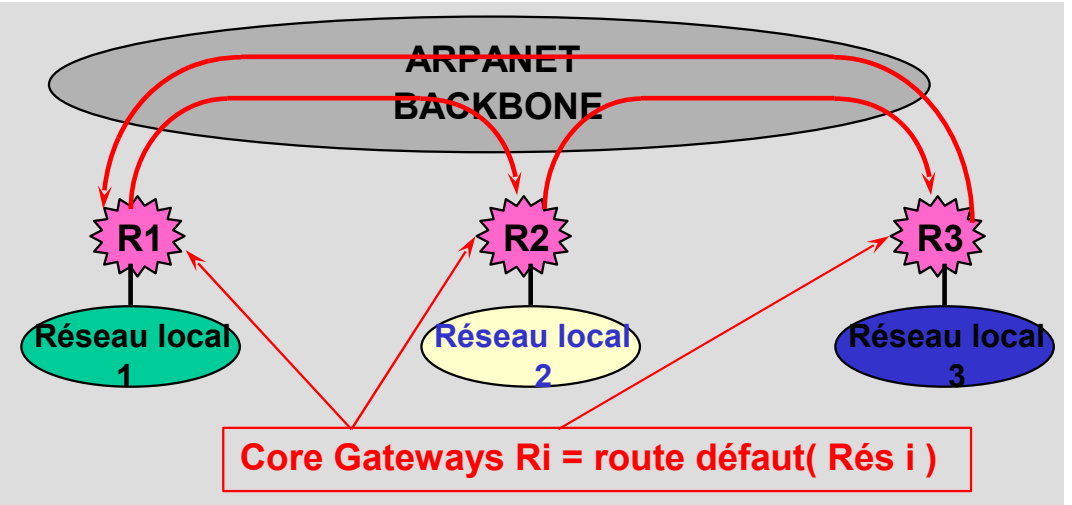
Deux classes de «routing protocol» :

- routage par **états de liens**, «*link state protocol*» ;

- routage par **vecteur de distance**.

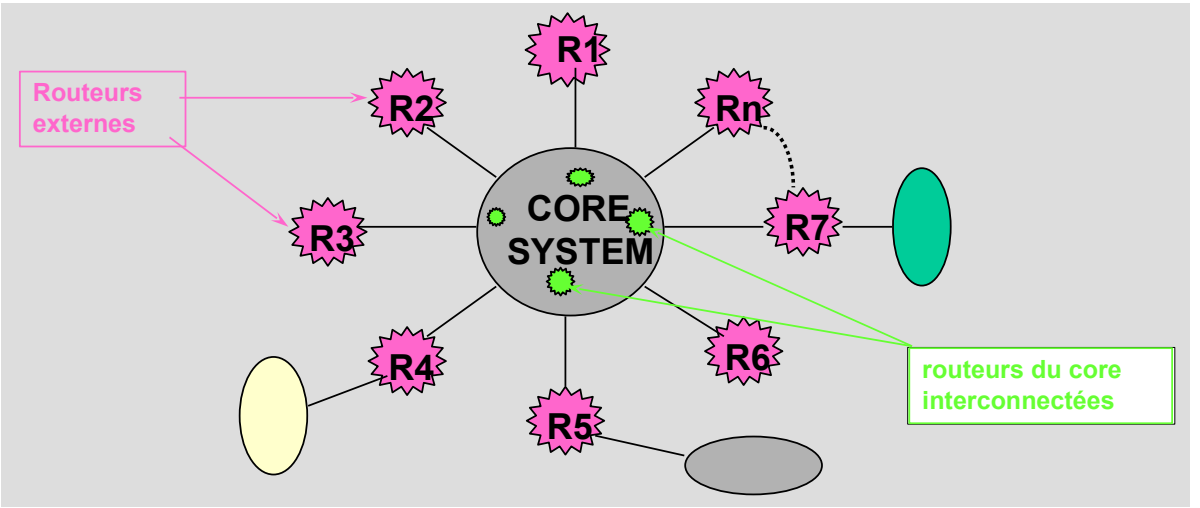
Le modèle d'ArpaNet

Utilisation de routeur particulier, le «core gateway» :



- Interconnexion de réseaux locaux ;
- routage circulaire utilisant la notion de route «par défaut» :
  pour aller du Réseau 2 au Réseau 1, on passe par les routeurs R2 puis R3 et enfin R1.
- **routage peu efficace !**

Technique du Core System



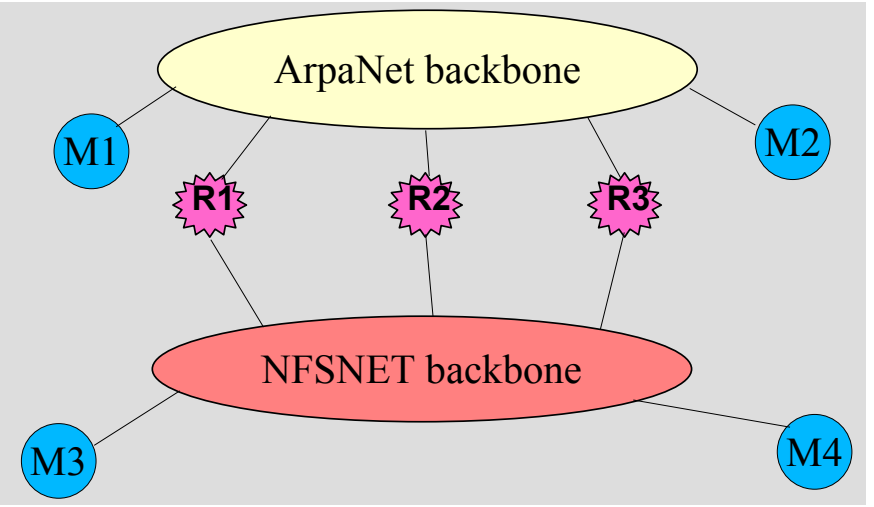
Organisation **centralisée** évitant le routage par défaut :

- ▷ les passerelles internes au «*core system*» connaissent la route pour atteindre n'importe quelle station (pas de route par défaut) ;
- ▷ les passerelles externes routent par défaut vers le core.

Impossible à mettre en œuvre quand le système est de grande taille !

- ⋆ impossibilité de connecter un nombre arbitraire de réseaux ;
- ⋆ le core ne connait qu'un seul réseau par routeur externe ;
- ⋆ les tables de routage deviennent énormes.

Autres limites du modèle



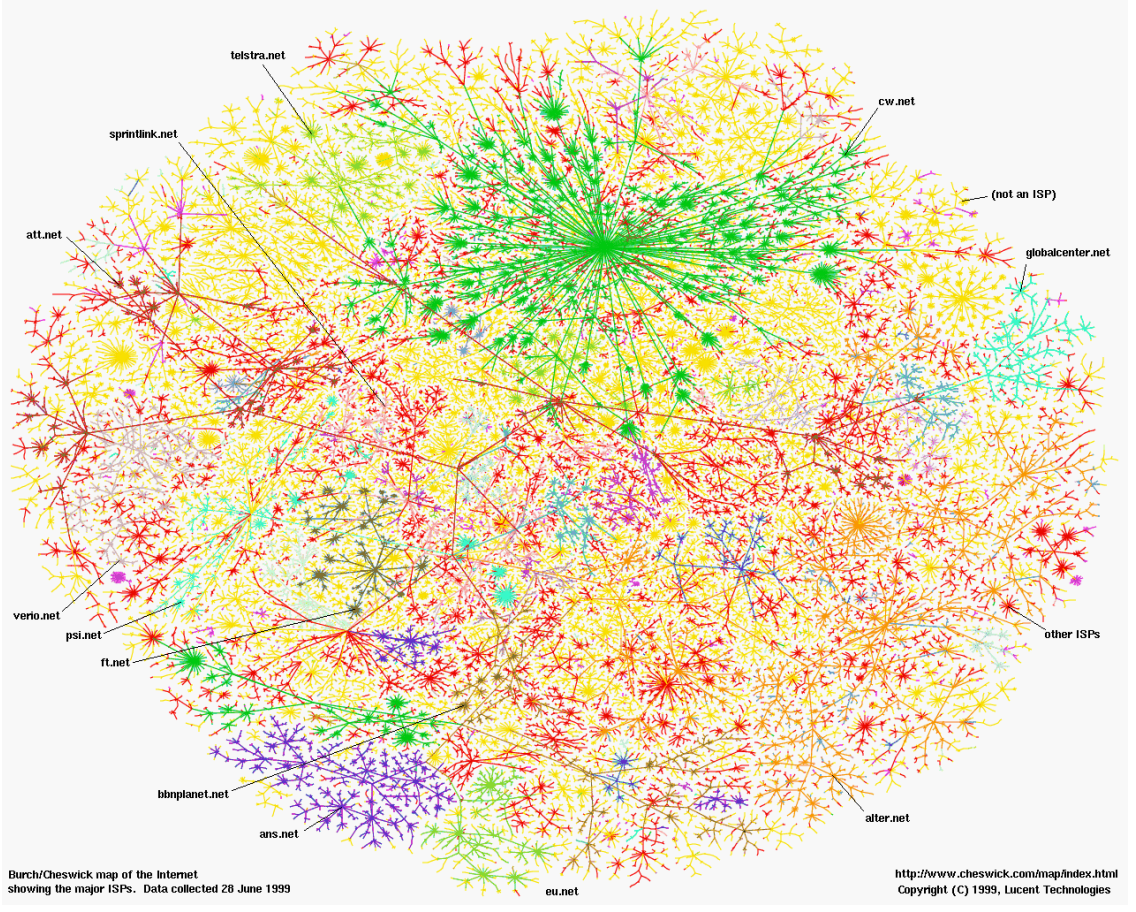
L'utilisation de « core » ne convient pas à l'interconnexion de backbones :

- il existe plusieurs choix de routes possibles entre deux stations :
  exemple : M1 ⟶ R1 ⟶ M4 ou M1 ⟶ R1 ⟶ R2 ⟶ R3 ⟶ M4

- il faut maintenir la cohérence entre toutes les machines des deux backbones ;

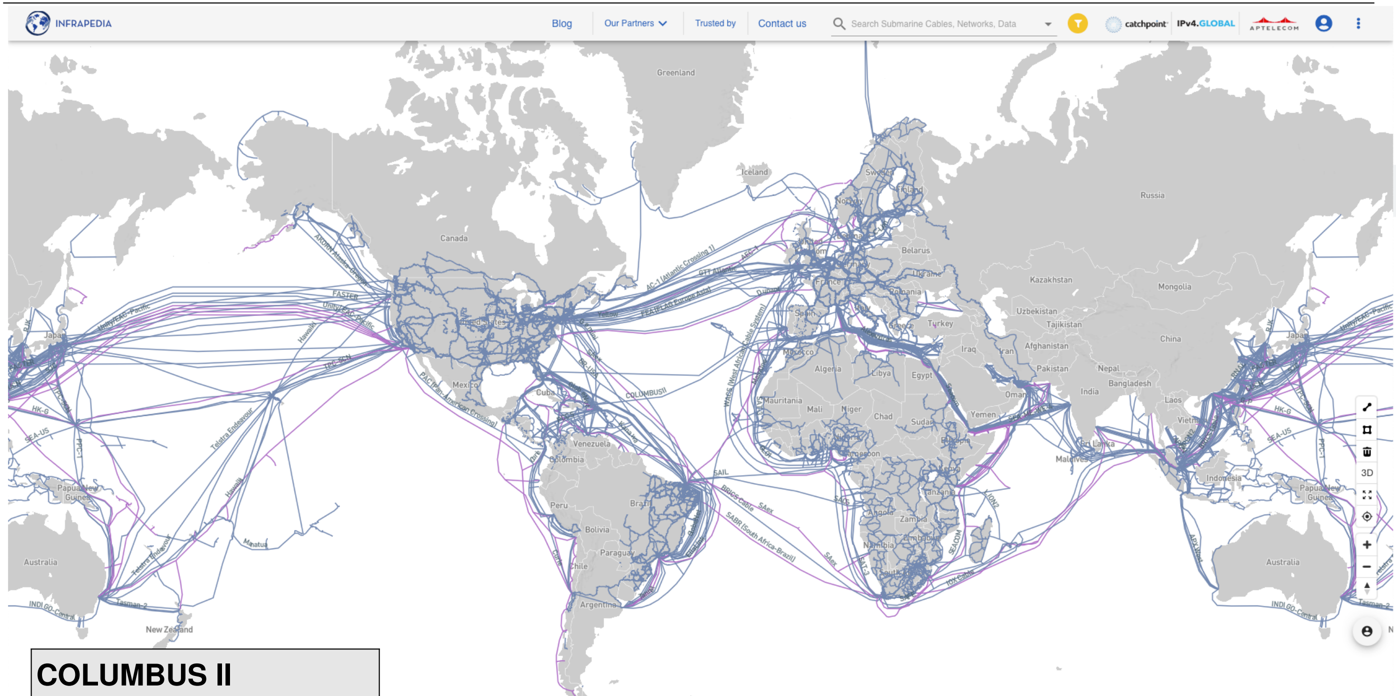
- l'utilisation de route par défaut peut introduire des boucles.

Internet en 1999



Et aujourd’hui ?

Les câbles sous-marin déployés dans le monde



COLUMBUS II	
System Status	On
System Length	11914 km
Latency	59.57 ms
Activation Year	1994

Comment faire ? Utiliser une approche hiérarchique et un partionnement

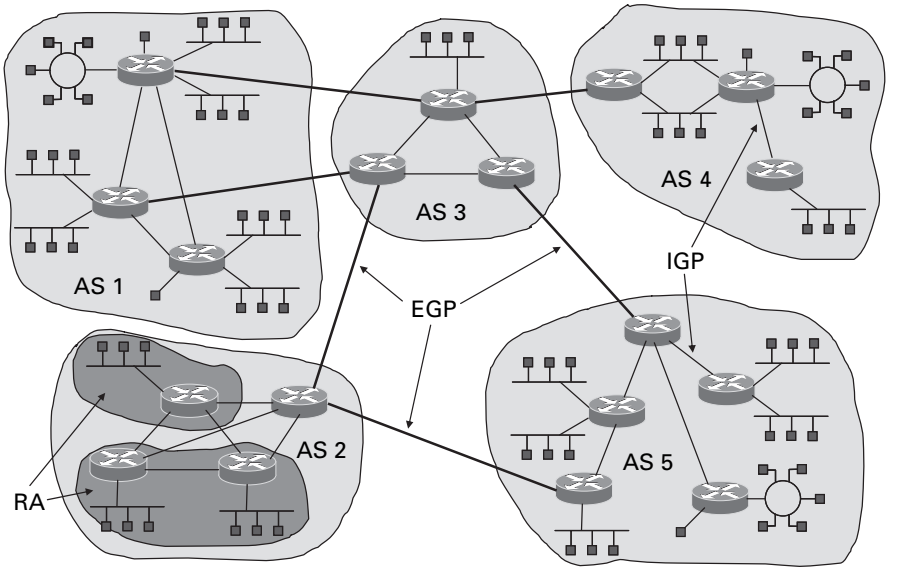
- Internet est une **collection de réseaux** et de routeurs interconnectés utilisant le protocole IP ;
- le réseau entier est **partitionné en différentes régions**, «*areas*» :
  - ◇ les routeurs à l'intérieur d'une région sont responsables du routage des datagrammes entre les réseaux internes à la même région ;
  - ◇ les tables de routage de ces routeurs internes est diminuée de manière drastique ;
- lorsque la destination ne se situe pas à l'intérieur d'une région, alors le routeur interne doit router le datagramme à **l'extérieur de la région**, vers des routeurs spéciaux :
  - ◇ placés à la frontière, «*border*», de la région ;
  - ◇ connaissants la topologie externe du réseau ;
  - ◇ disposants de grandes tables de routage incluant tous les préfixes de réseau possibles ;
  - ◇ chargés de router les datagrammes entre les différentes régions.
- le **partitionnement** en différentes régions peut être fait :
  - ◇ de manière itérative ;
  - ◇ en créant plusieurs niveaux dans la hiérarchie de routage, ce qui limite le nombre de «*border router*» qui ont besoin de maintenir une table de routage complète.
- l'avantages de ce partitionnement en régions est que le processus de routage à l'intérieur d'une région est **indépendant** :
  - ◇ de celui réalisé à l'intérieur des autres régions ;
  - ◇ de celui réalisé entre les régions.

Conclusion :

- ◇ Différentes régions peuvent choisir des algorithmes de routage «internes» différents ;
- ◇ Le même protocole de routage «externe» doit être utilisé entre les différentes régions.

L'AS ou «Autonomous System»

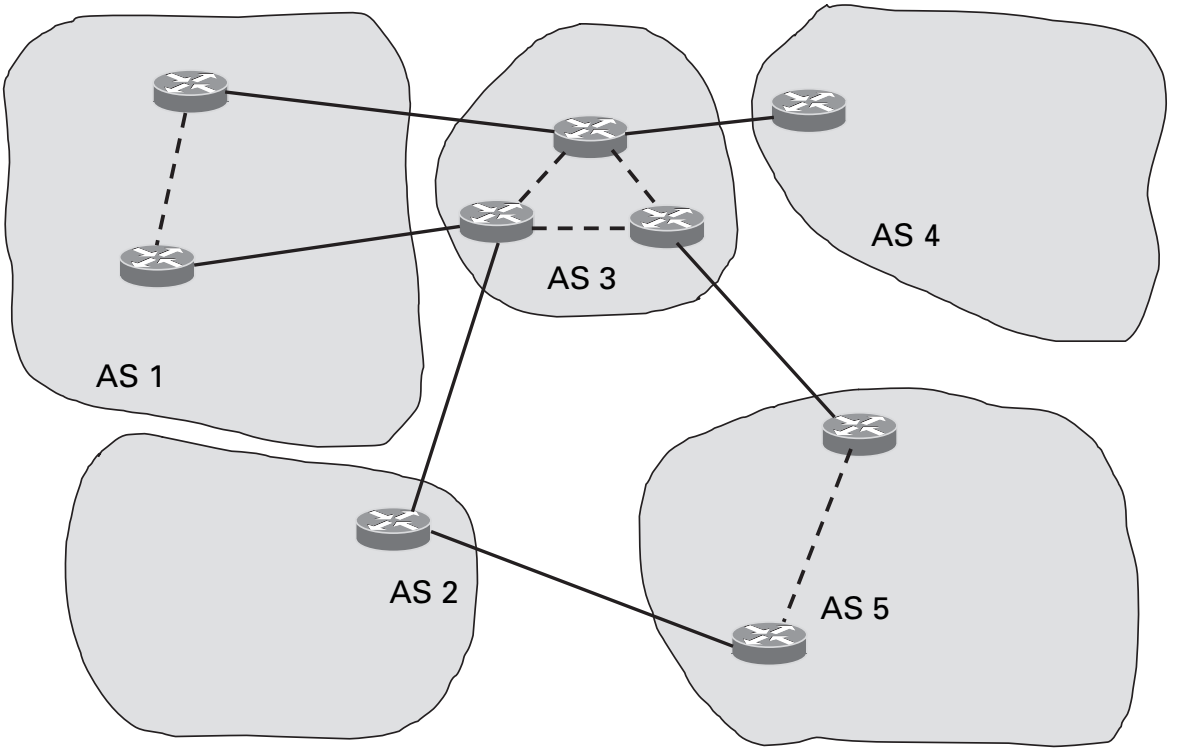
- il correspond au niveau hiérarchique le plus haut du partionnement d'Internet ;
- il représente un ensemble de réseaux et de routeurs soumis à la même entité administrative ;



Quelques définitions :

- les protocoles de routage internes à une AS sont appelés IGP, «Interior Gateway Protocol» ;
- ceux utilisés entre les AS sont appelés EGP, «Exterior Gateway Protocol» ;
- une AS peut être partitionnée en RA, «Routing Area» suivant l'IGP utilisé.

La vue du réseau du point de l'EGP



Seul les routeurs et leurs informations de routage sont visibles.

Définition de domaines de routage

Cela correspond à associer des réseaux et des routeurs sous la responsabilité d'une autorité unique.
L'architecture de routage est indépendante entre les systèmes autonomes.

Règle de découpage de l'Internet :

- ▷ deux réseaux locaux d'une même institution nécessitant un autre AS pour communiquer ne peuvent constituer un AS unique ;
- ▷ une AS peut être plus ou moins grande : un FAI peut avoir une AS aux USA, une en Europe et une autre pour le reste du monde.

Chaque système autonome est identifié :

- ◦ par un numéro unique attribué par le NIC : ASN, numéro sur 16bits ou 32 bits ;
- ◦ les numéros 64512 à 65535 sont pour des AS privées.

Exemple : Renater Limousin AS1935, Renater AS1717

À l'origine, les AS étaient connectés sur le noyau ARPANET (également considéré comme un AS).
Aujourd'hui, il existe seulement des AS interconnectés.

xterm

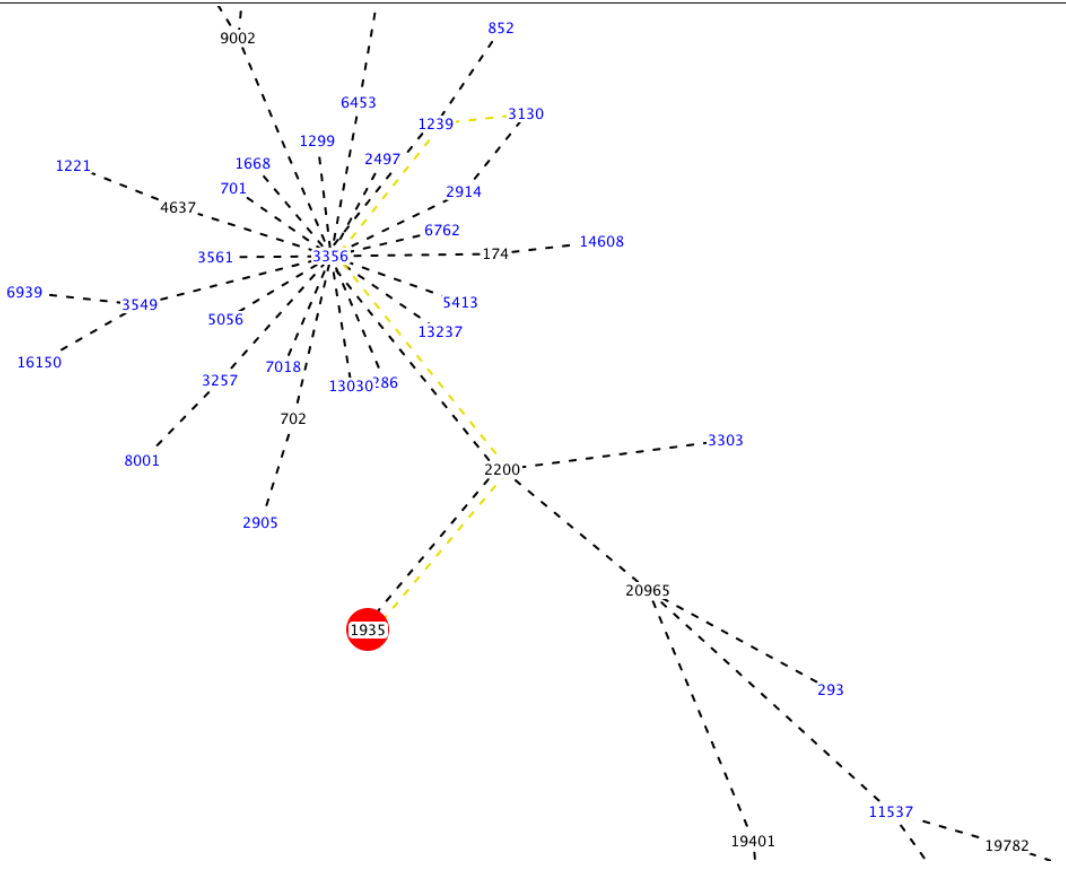
```
whois -h whois.ripe.net 164.81.0.0/16
% Information related to '164.81.0.0/16AS2200'

route:        164.81.0.0/16
descr:        FR-U-LIMOGES
origin:       AS2200
mnt-by:       RENATER-MNT
source:       RIPE # Filtered

whois -h whois.ripe.net AS1935
% Information related to 'AS1935'

aut-num:      AS1935
as-name:      FR-RENATER-LIMOUSIN
descr:        Reseau Regional Limousin
descr:        FR
import:       from AS2200 action pref=100; accept ANY
export:       to AS2200 announce AS1935
default:      to AS2200 action pref=10; networks ANY
admin-c:      GR1378-RIPE
tech-c:       GR1378-RIPE
mnt-by:       RENATER-MNT
source:       RIPE # Filtered
```

Un exemple d'AS

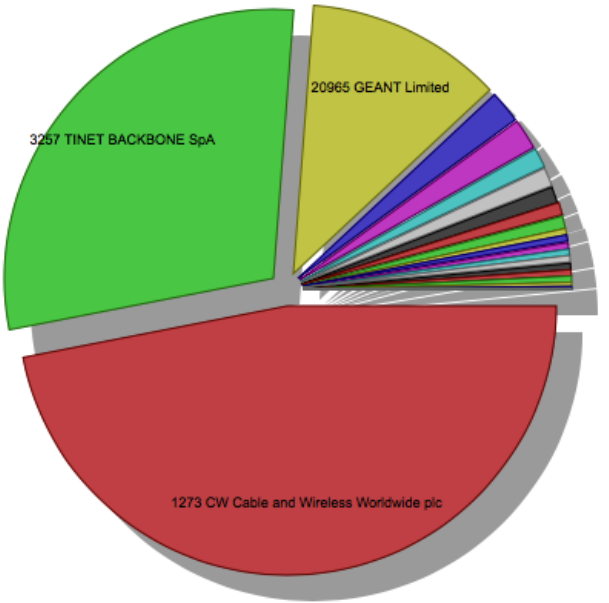


Les dernières infos sur : `http://www.robtex.com/as/as2200.html`

Les liens avec l'AS2200 : en 2016

AS-INFO

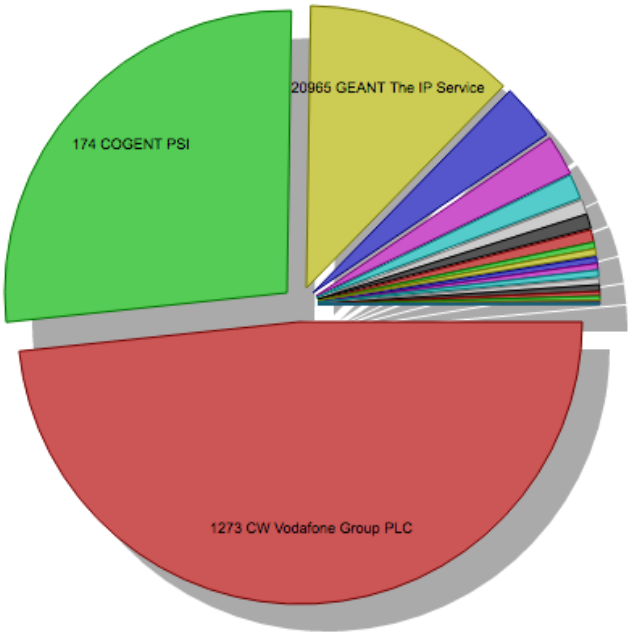
Source	Information	Date
	This report generated	Jan 20, 2016 10:30:56 AM
RADB	Routes	Jan 16, 2016 2:53:32 PM
	AS Info	Jan 15, 2016 5:01:37 PM
peeringdb.com	Peering information	Jan 17, 2016 2:25:43 PM
Internet	BGP	Jan 20, 2016 4:34:32 AM



Les liens avec l'AS2200 : en 2017

AS-INFO

Source	Information	Date
	This report generated	Jan 11, 2017 9:00:47 AM
RADB	Routes	Jan 11, 2017 9:00:49 AM
	AS Info	Jan 11, 2017 9:00:48 AM
peeringdb.com	Peering information	Jan 5, 2017 10:16:33 AM
Internet	BGP	Jan 9, 2017 12:21:51 PM



AS2200 et BGP

Peers

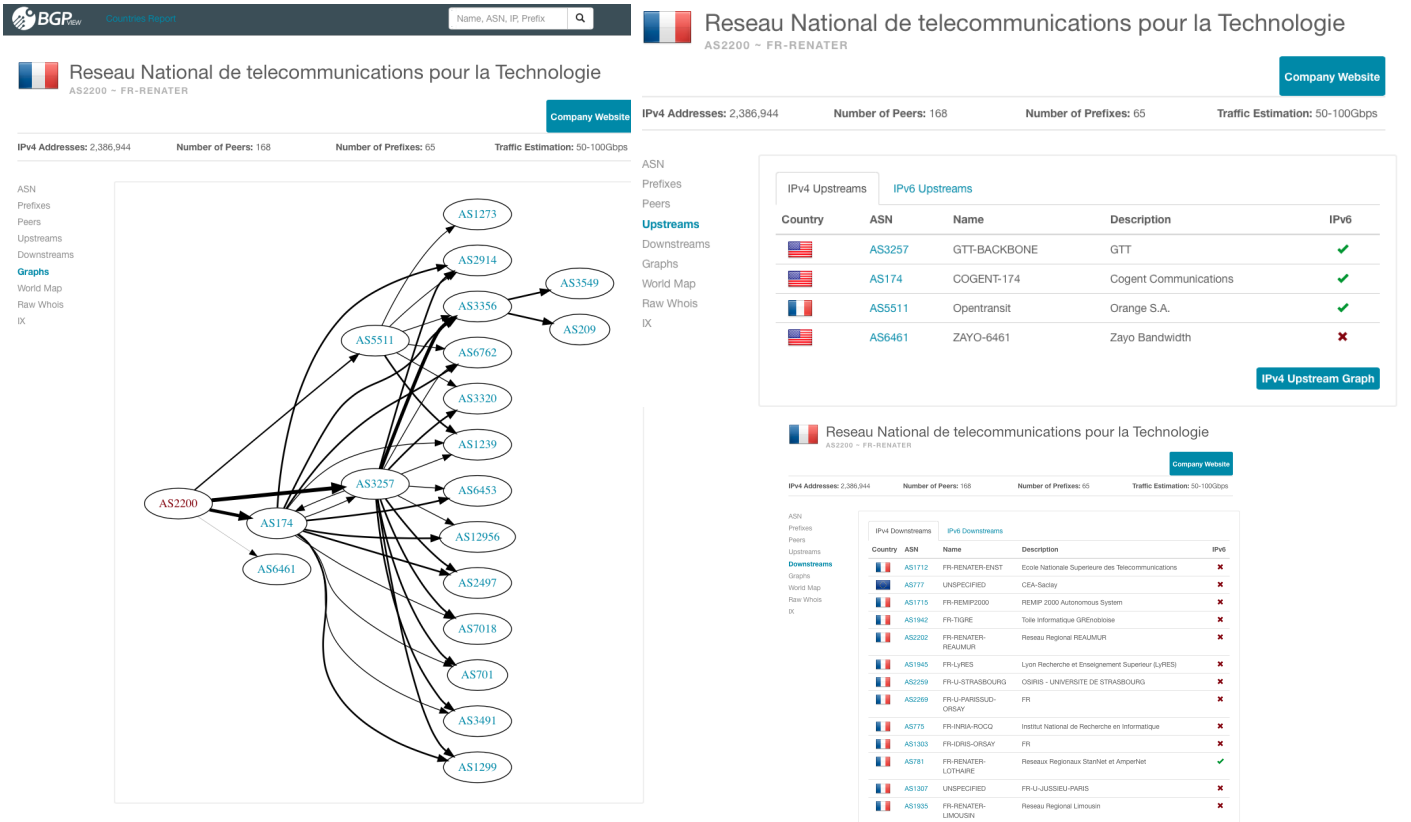
peer-AS↓	import/export/bgp	Macro	This IP	Other IP
174 COGENT PSI	b			
	e	any		
	i	any		
		any		
	e	as-renater		
261 FR CINES MONTPELLIER		any		
	i	as261		
286 KPN	e	as-renater		
	i	as-kpn		
513 CERN	b			
	e	any		
	i	<^$\|^as-cernext$>		
702 AS702 Verizon Business EMEA Commercial	e	as-renater	194.68.129.103	194.68.129.236
	i	as702:rs-fr as702:rs-fr-custom	194.68.129.103	194.68.129.236
776 FR INRIA SOPHIA Antipolis	e	any		
	i	as776		
777 UNSPECIFIED CEA Saclay	e	any		
	i	as777		

BGP

BGP	Reg	Network↓	Description	Announced by	Registered by
✓	✓	81.194.0.0/16	RENATER	2200	2200
✓	✓	91.236.25.0/24	FR-EPPG-LAVILLETTE		
✓	✓	128.93.0.0/16	FR-INRIA-NET | FR-INRIA-NET		775 2200
✓	✓	129.20.0.0/16	FR-VERDUR		2200
✗	✓	129.88.0.0/16	FR-MI2S | FR-MI2S	~~2200~~ 1942	1942 2200
✓	✓	129.102.0.0/16	FR-IRCAM	2200	2200
✓	✓	129.104.0.0/16	FR-EP-PALAISEAU | FR-EP-PALAISEAU		1948 2200
✗	✓	129.175.0.0/16	RENATER | FR-U-PARISSUD-ORSAY	~~2200~~ 2269	2200 2269
✓	✓	129.199.0.0/16	FR-ENS-NET | FR-ENS-NET	2200	2200 2422
✓	✓	130.66.0.0/16	FR-ECN-NANTES		2200
✗	✓	130.84.0.0/16	FR-CIRCE | RENATER	~~2200~~ 1303	1303 2200
✗	✓	130.120.0.0/16	FR-UPS-TOULOUSE | FR-UPS-TOULOUSE	~~2200~~ 1715	1715 2200
✗	✓	130.190.0.0/16	FR-SIMSU | FR-SIMSU	~~2200~~ 1942	1942 2200
✗	✓	131.254.0.0/16	FR-IRISA-RENNES | FR-IRISA-RENNES	~~2200~~ 1938	1938 2200

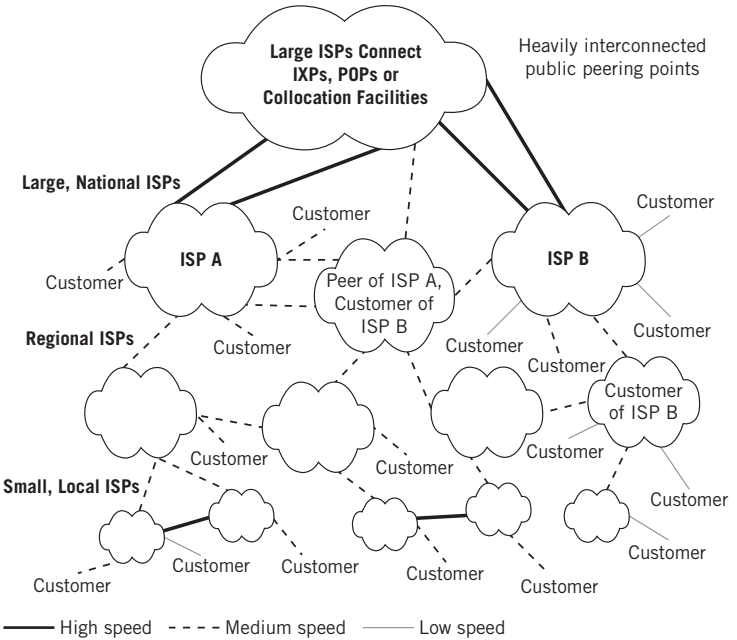
https://www.robtex.com/as/as2200.html

les liens avec l'AS2200 en 2021



InterNet : une collection d'AS organisées en «*transit*», «*Peer*» ou «*Customer*»

Internet se présente comme un maillage de FAI, ou ISP, «*Internet Service Provider*», auxquels sont connectés des entités privées, gouvernementales et éducatives :



- **IX** ou IXP : *Internet eXchange Points* : pour définir des points d'interconnexion de niveau 2 entre différents réseaux (*LANs ethernet*).
  Appelés aussi «Public Peering».
  Peu de suivi sur le trafic échangé entre les différents réseaux interconnectés.
  `https://en.wikipedia.org/wiki/Peering`
- **POP** : *Points of Presence*, où sont placés les IXPs de manière logique ;
- *collocation facilities* : des lieux physiques d'interconnexion avec des liens redondants, des alimentations ;
- **ISP** ou FAI : ils s'interconnectent entre eux pour donner à leur client accès à Internet.
  Les liens entre ces FAIs sont des «peering agreement».
- **PNI**, «*Private Network Interconnect*» : connexion directe entre deux réseaux avec de la fibre ou au travers d'un réseau de TelCo ;
  Suivi précis du trafic et de sa métrologie.
  exemple d'OVH : `https://peering.ovh.net`

- les peers, «pairs», sont égaux entre eux : il sont en accord pour acheminer gratuitement leur trafic réciproque, mais font payer leur clients ;
- les plus petits FAIs sont seulement clients d'un plus gros FAI ;
- les débits varient de 100Mbps (Ethernet) à 10 Gbps (fibre optique).

Les IX de l'AS2200 en 2021

Reseau National de telecommunications pour la Technologie

AS2200 ~ FR-RENATER

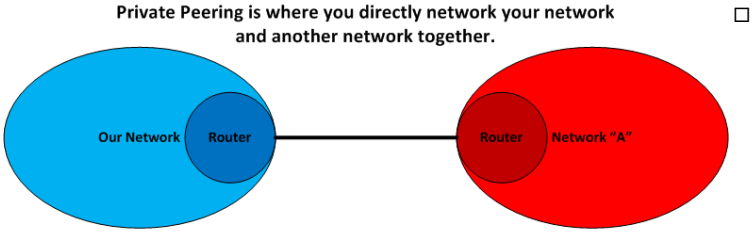
Company Website

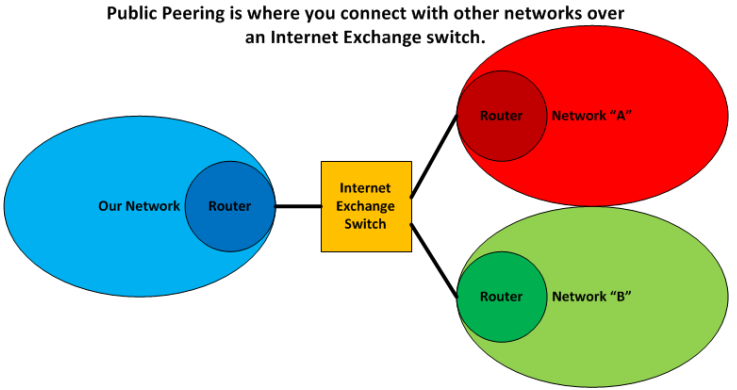
IPv4 Addresses: 2,386,944 **Number of Peers:** 168 **Number of Prefixes:** 65 **Traffic Estimation:** 50-100Gbps

ASN
Prefixes
Peers
Upstreams
Downstreams
Graphs
World Map
Raw Whois
IX

Country	IX	Name	IPv4	IPv6	Port Speed
	SFINX	Service for French INternet eXchange	194.68.129.102	2001:7f8:4e:2::102	10 Gbps
	SFINX	Service for French INternet eXchange	194.68.129.103	2001:7f8:4e:2::103	10 Gbps
	LyonIX	Lyonix, the Lyon IX	77.95.71.17	2001:7f8:47:47::11	10 Gbps
	Equinix Paris	Equinix Internet Exchange Paris	195.42.145.38	2001:7f8:43::2200:1	100 Gbps
	France-IX Paris	FranceIX Paris	37.49.236.19	2001:7f8:54::19	10 Gbps

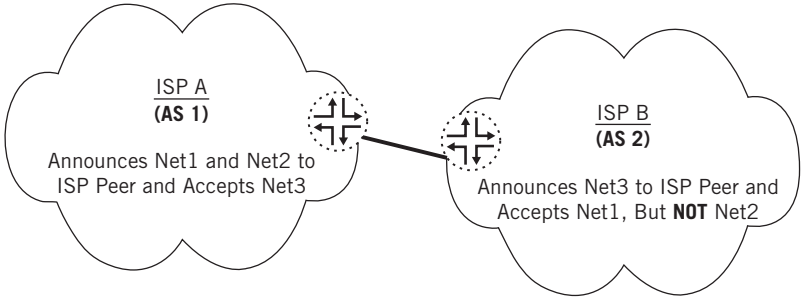
Public vs Private Peering





- «*Peering*» : connexion entre deux AS qui sont **d'accords pour échanger** l'une avec l'autre leurs informations de routage ainsi que celles de leur clients (customers) ;

- deux formes de connexion :
  - connexion directe par PNI :
    - lien «**point à point**» 10 ou 100Gb/s entre les routeurs de deux AS ;
    - **suivi précis** du trafic dans les deux sens : en entrée, «*inbound*», et en sortie, «*outbound*».
  - connexion au travers d'un IX :
    - utilisation d'un switch Ethernet permettant la connectivité de **un vers plusieurs** ;
    - **peu de suivi** du trafic entre les réseaux interconnectés.



Les préfixes des réseaux d'un ISP doivent être :

- diffusés auprès des routeurs de frontières, sous forme éventuellement réduite par agrégation ;
- **acceptés** suivant la «*routing policy*».

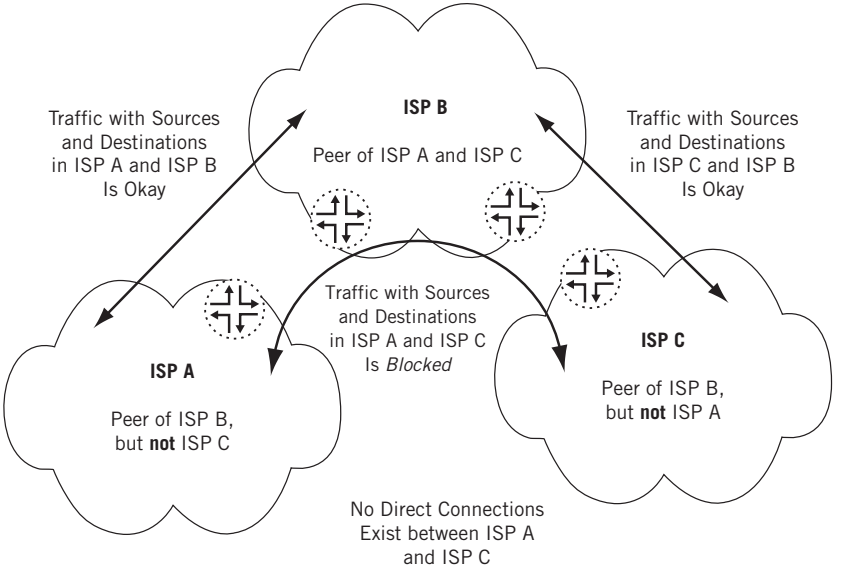
Ces informations de routage sont échangées par BGP.

Suivant la «*routing policy*» :

- elles peuvent ne pas être transmises ;
- elles peuvent être ignorées.

Dans le cas où cette information n'est pas transmise ou refusée, les machines d'une AS concernées par cette information ne pourront pas atteindre les autres machines.

Ici, Net2 n'est pas accepté et ne peut envoyer de trafic au travers de l'AS2.

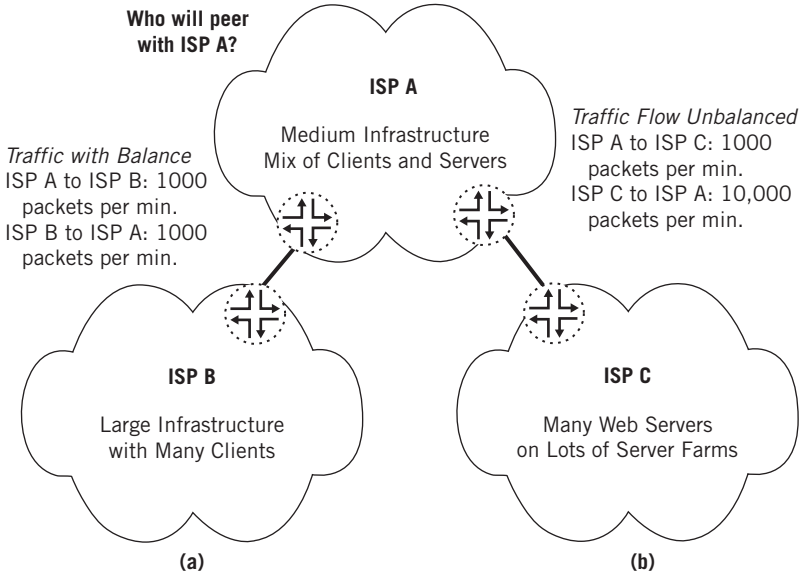


Les «routing policies» : le peering

Le peering correspond pour une AS à rechercher une AS paire, afin d'échanger du trafic réseau.

La démarche est «un peu» similaire à celle des opérateurs de téléphonie, mais avec des difficultés :

– les «appels» sont des des flux de datagrammes ;
– un FAI peut être seulement un relais entre deux FAIs ;
– la règle du «*sender keeps all*» : c'est le premier FAI qui reçoit l'argent des clients et le garde.
  Cela fonctionne si le trafic dans un sens est **équilibré** avec celui dans l'autre sens.



Ici, le trafic de ISP A à ISP C ne peut pas passer par ISP B.

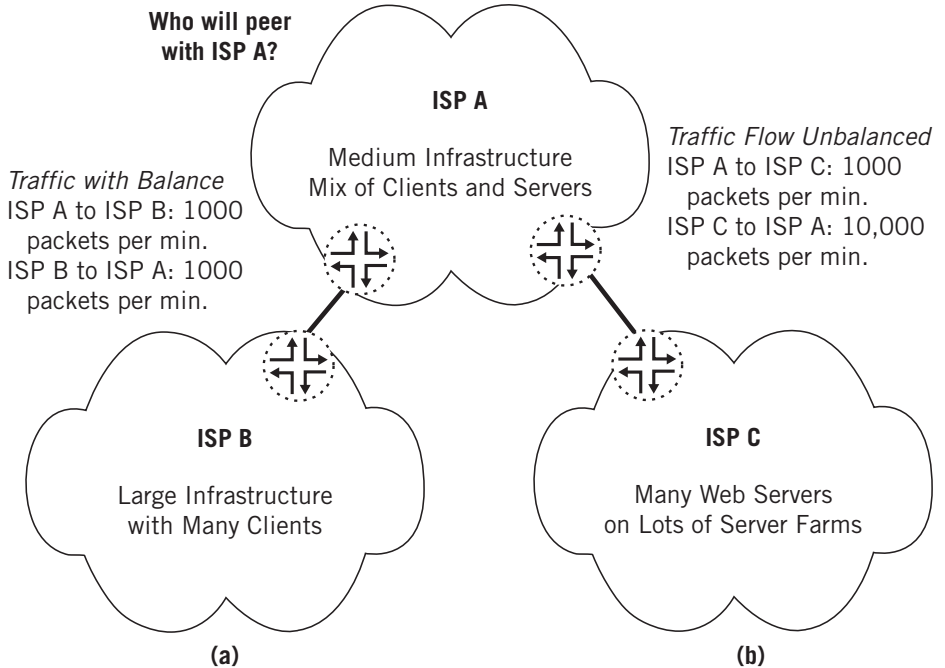
Les difficultés sont :

- il peut y avoir plusieurs FAIs entre un client et un serveur ;
- un client peut effectuer de nombreuses connexions successives et rapides (consultation Web) et il est dur de suivre ces connexions (origine, chemin) ;

La solution : le «***peering***» :

- un accord, un «*settlement*» est défini entre FAI pour ne pas faire payer entre pairs ;
- **mais** le trafic ne peut pas transiter par l'intermédiaire d'un pair commun sans devenir client d'un des ISP.

Le choix du pair

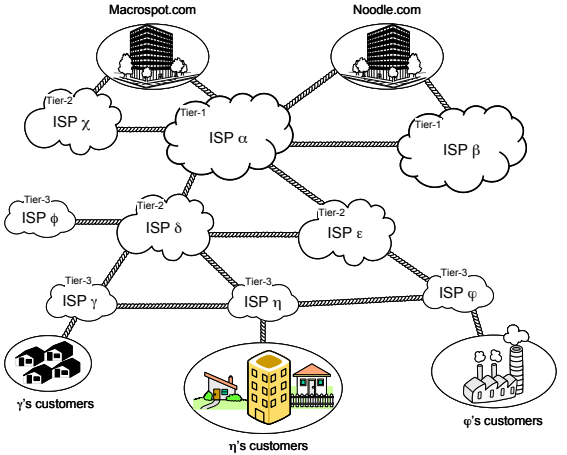


Ici, le FAI A a le choix entre le FAI B et le FAI C : il pourra choisir B et prendre C en client.

Autre solution :

- utiliser un IXP ou un WAN pour s'interconnecter directement (lien public) ;
- la tendance : liens entre FAIs (lien privé).

Exemple de réseaux d'ISP, «*Internet Service Provider*»



Sur le schéma :

- ⋆ «Tier-1», ISP α ou β : il possède une vision globale du réseau, et ses tables de routage ne contiennent pas de route par défaut, «default-free» (il en existe une quinzaine dans le monde) ;
- ⋆ «Tier-2», ISP δ ou ϵ : niveau régional ou pays ;
- ⋆ «Tier-3», ISP γ ou η : niveau local.

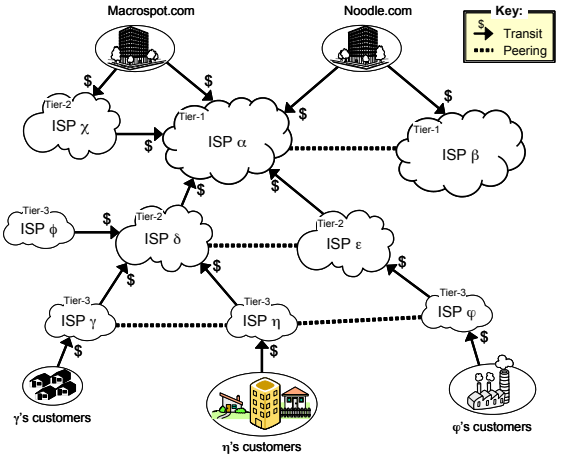
Les accords de peering se font entre «Tiers» de même niveau, donc de même dimension, avec un volume de trafic équivalent.

Chaque ISP peut appliquer la règle de transit suivante :

- ▷ vendre du transit, ou de l'accès internet, à une autre AS : un «transit provider» vend du transport à un «transit customer» ;
- ▷ échanger du trafic, peering, avec une autre AS ;
- ▷ acheter du transit depuis une autre AS : ce transit peut alors être vendu à d'autres AS clientes.

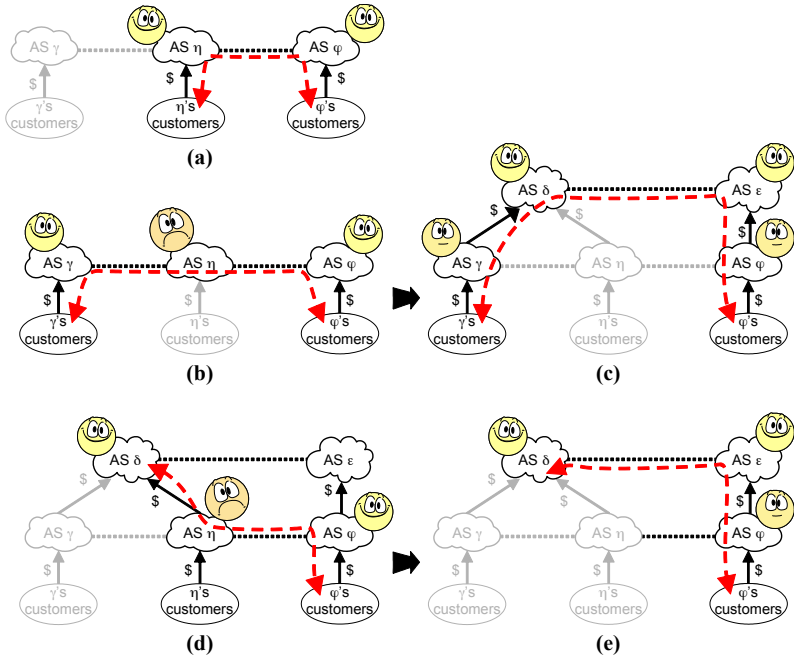
Sur le schéma :

- ○ *«multihomed AS» : les AS des deux sociétés,* noodle *et* macrospot*, ne relaient pas de trafic et sont connectées à deux AS ;*
- ○ *«Stub AS» : celles ne possédant qu'une connexion à une AS.*



Les liens entre AS : l'organisation des échanges entre AS

- ▷ les relations de transit génèrent des revenus alors que celles de peering n'en génère pas (ou indirectement) ;
- ▷ le peering :
  - ◇ peut réduire les coûts de transit et permet d'économiser pour l'ensemble des pairs ;
  - ◇ améliore la redondance en diminuant la dépendance envers un «*transit provider*» ;
  - ◇ améliore la performance en augmentant le nombre de chemins disponibles (éviter les «*bottlenecks*») ;
  - ◇ augmente la capacité en répartissant le trafic entre différents réseaux ;
  - ◇ facilite l'obtention de capacité supplémentaire en cas d'urgence (auprès des pairs).



Les règles :

- ▷ Pour ses clients payants, une AS veut fournir un service de **transit illimité**.
- ▷ Pour ses fournisseurs et pairs, une AS veut fournir un service de **transit sélectif**.

a) les AS η et ϕ bénéficie de leur statut de peering au profit de leur clients respectifs ;

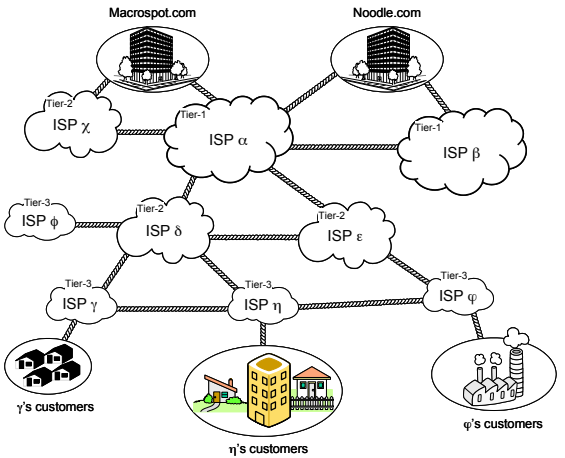
b) l'AS η pâtit du trafic entre γ et ϕ aux dépends de ses clients : η ne va pas relayer leur trafic et ils vont passer par les AS dont ils sont clients (δ et ϵ).

c) l'AS δ se sert de son client η pour atteindre ϕ. L'AS η pâtit de cette situation : elle paye δ pour le transit et elle ne veut pas fournir du transit gratuit en retour.
Le trafic entre l'AS δ passe alors par le pair ϵ pour atteindre l'AS ϕ.

Comment implémenter les règles de transit et d'échange de trafic ?

Pour mettre en œuvre les décisions économiques et éviter les situations défavorables, une AS définit des «*routing policies*» :

- ▷ si une AS veut éviter de fournir un transit entre deux AS dont elle est voisine, il suffit qu'elle n'annonce pas, «*advertising*», que l'une de ces AS peut être joignables de l'autre par son intermédiaire. Ces deux AS seront visibles l'une de l'autre par l'intermédiaire d'autres AS.
- ▷ pour ces clients pour le transit, une AS doit diffuser l'ensemble des réseaux qu'elle connaît (ils sont alors joignables, «*reachable*» ou visibles) ;
- ▷ pour ses pairs, une AS ne rend visible que ses propres clients pour le transit mais aucun de ses pairs ni de ses fournisseurs de transit pour éviter d'avoir à relayer du trafic sans contrepartie ;
- ▷ pour ses fournisseurs de transit, «*transit provider*», une AS doit rendre visible ses propres clients mais aucun de ses pairs ni de ses autres fournisseurs de transit pour éviter tout abus.

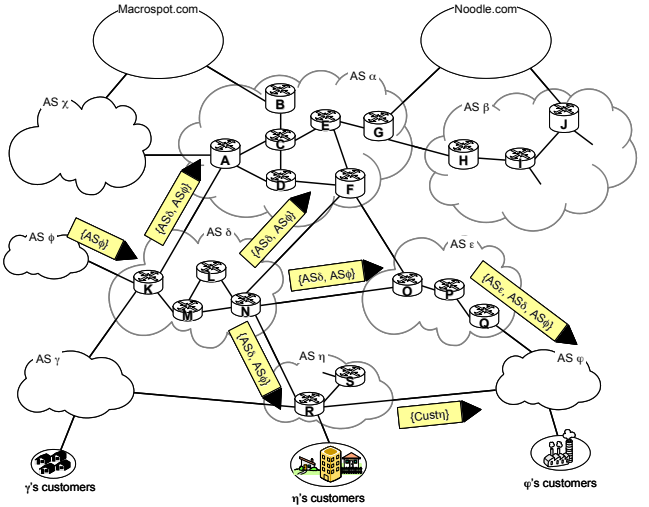


- ⋆ l'AS α et β voient tous leurs réseaux car ils sont pairs et tous les autres AS sont leur client ;
- ⋆ l'AS γ voit l'AS η et ses client directement mais par l'AS ϕ au travers de η ;
- ⋆ l'AS δ voit l'AS ϕ au travers de ϵ mais pas au travers de son son client l'AS η ;
- ⋆ le trafic de l'AS ϕ vers l'AS γ va passer par l'AS ϵ (et l'AS δ) mais pas par l'AS η.

Un protocole de routage : «*Routing Path Vector*»

Le routeur de l'AS ϕ veut diffuser le préfixe de destination `128.34.10.0/24` :

- ▷ il envoie un message contenant un «routing path vector» pour indiquer le chemin permettant d'atteindre la destination donnée :
  - ◇ ce message contient uniquement au départ le numéro de l'AS $\{AS_\phi\}$;
  - ◇ un routeur de frontière, «*border router*», K dans l'ASδ reçoit ce message et le diffuse vers tous les routeurs frontières dans l'ASδ :
    - ⋆ ces routeurs à l'intérieur de l'ASδ ajoute en entête leur propre numéro d'AS au message de «chemin de routage» qui devient $\{AS_\delta, AS_\phi\}$
    - ⋆ ces routeurs rediffusent ce nouveau message vers les AS adjacentes.



- ◇ l'ASη n'a pas d'intérêt économique à diffuser un chemin vers l'ASδ à ses pairs comme l'ASϕ, elle met à jour le message avec un «chemin de routage» indiquant uniquement ses clients :$\{Cust_\eta\}$
- ◇ l'ASϵ a un intérêt économique de diffuser les accès aux réseaux de l'ASδ à ses propres clients : elle diffuse un message modifié $\{AS_\epsilon, AS_\delta, AS_\phi\}$;
- ◇ les routeurs de l'ASϕ mettent à jour leur tables de routage en fonction des messages reçus :
  - ⋆ si un paquet à destination de `128.34.10.0/24` est envoyé depuis l'ASϕ il sera envoyé au «*next hop*», l'ASϵ.

Avantages

- adaptation à l'évolution du réseau ;
- ajout ou suppression de routeurs ;
- pertes ou ajouts de liaisons ;
- configuration simple de chaque routeur (configurer seulement les interfaces).

Contraintes

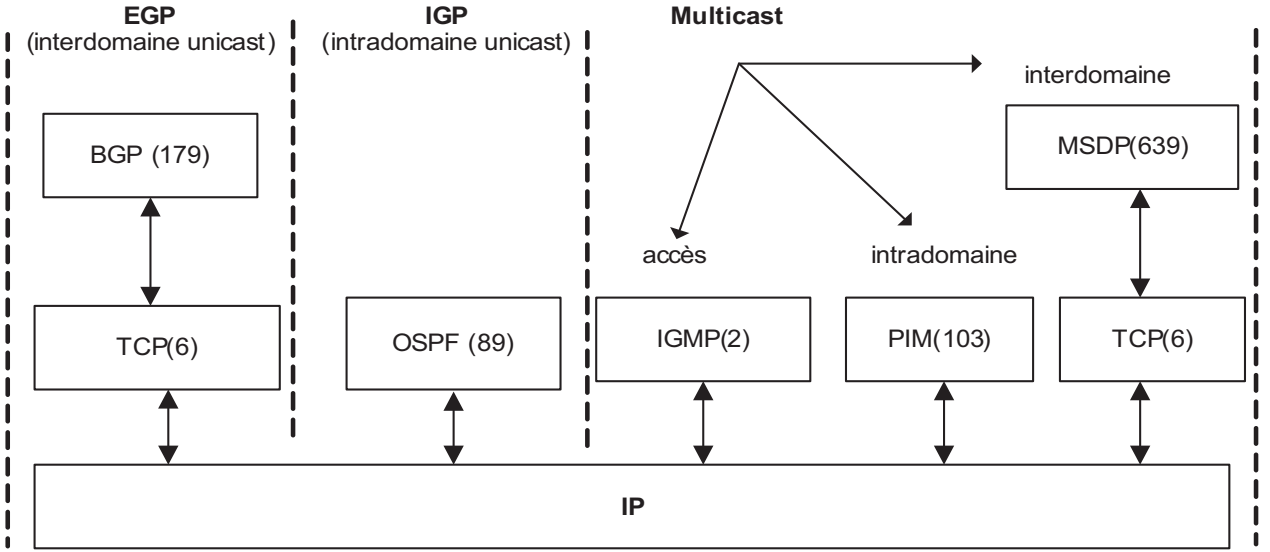
- Optimisation : sélection des meilleurs routes ;
- Cohérence : élimination des boucles de routage (routes circulaires) ;
- Efficacité : peu de consommation de bande passante et de temps CPU ;
- Stabilité : convergence (obtention du résultat global) et re-configurations rapides ;
- Simplicité : configuration simple de chaque élément.

Les protocoles intérieurs (IGP)

- à vecteur de distance : RIP, IGRP ;
- à états de liens : OSPF ;
- taille <100 routeurs, une autorité d'administration ;
- échange d'informations de routage entre routeur.

Les protocoles extérieurs (EGP)

- taille correspondant à celle d'Internet ;
- coopération entre entités indépendantes (AS) ;
- échange d'informations de routage entre AS ;
- BGP : protocole qui utilise TCP pour échanger les informations de routage :
  - il permet de faire du «*policy-based routing*», pour choisir entre différents chemins possibles ;
  - il est basé sur des «tables de chemin», «*path vector routing*», qui énumère les AS à emprunter jusqu'à la destination.



- le protocole OSPF est intégré dans IP avec son propre numéro de protocole ;
- le protocole RIP est encapsulé dans UDP ;
- le protocole BGP est encapsulé dans TCP ;
- ceux concernant le multicast utilisent différentes encapsulations suivant le niveau de communication multicast :
  - accès local ;
  - accès dans un domaine ou AS ;
  - accès interdomaine ou entre AS.

EGP, «*Exterior Gateway Protocol*»

- BGP, «*Border Gateway Protocol*», RFC 1771 :
  - protocole basé chemin «***Path Vector Routing***», c-à-d où le chemin complet d'acheminement est connu ;
  - adapté à la prise de décision au niveau hiérarchique le plus haut, quant l'administration d'une AS doit décider :
    - quel type de trafic elle va autoriser à traverser ses réseaux ;
    - quelle relation de «*peering*» elle va établir avec d'autres AS ;
  - construit des chemins en accord avec des «*routing policies*» plutôt que des chemins optimaux ;
  - n'organise pas le routage intra-AS, *mais* l'utilise pour les échanges entre routeurs associés.

IGP, «*Interior Gateway Protocol*»

- RIP, «*Routing information protocol*», RFC 1058, 2453 :
  - protocole basé sur la distance, «***vector distance***» ;
  - adapté à des réseaux d'inter-connexion de petite taille déservant un nombre limité de réseaux : échelle d'une entreprise, d'une organisation partageant une même autorité administrative.
- OSPF, «*Open Shortest Path First*», RFC 2328 :
  - protocole basé sur l'état des liens entre routeurs, «***state-link***» ;
  - adapté à des réseaux d'inter-connexion de grande taille avec un nombre important de réseaux : échelle d'une région pouvant avoir des autorités administratives autonomes.

C'est un routage :

- «*inter-domain*», à l'intérieur d'un même domaine ;
- «*exterior routing*», entre différents AS.

Fonctionnement :

▷ chaque routeur de frontière d'une AS, appelé «*border*» ou «*edge router*», diffuse les destinations qu'il peut atteindre à ses routeurs voisins appartenant à d'autres AS ;

▷ Il est similaire au «*distance vector routing*», mais :

- ◇ les réseaux **ne sont pas diffusés** par une adresse de destination et la distance vers cette destination ;
- ◇ les réseaux **sont diffusés** par une liste d'adresses de destination et par des chemins menant à ces destinations, d'où le terme de «*path vector routing*» :
  - ⋆ une route est définie par une paire composée d'une destination et d'un chemin vers cette destination ;
  - ⋆ le chemin contient la liste complète des AS à traverser pour atteindre la destination ;
  - ⋆ le chemin contenant la liste la plus courte d'AS à traverser est privilégiée.

▷ chaque routeur diffuse ses informations à intervalles réguliers :

- ◇ sa propre adresse réseau ;
- ◇ une copie de son chemin à chacun des routeurs voisins (adjacents) ;

▷ lorsqu'un routeur reçoit les chemins de ses voisins, il réalise une **sélection de chemin** en concaténant l'information reçue avec celle existante dans son propre chemin.
Cette sélection de chemin se fait suivant une métrique comme dans le cas de l'algorithme par «vecteur de distance».

Sur ce réseau d'exemple, on va appliquer l'algorithme de «path vector routing» :

- au lieu de transmettre l'adresse de réseau, on transmettra le numéro de l'AS indiquée par une lettre grecque ;
- l'algorithme s'exécute depuis l'état initial jusqu'à atteindre un état d'équilibre.
- le fonctionnement est le suivant :
  - un chemin est noté $< d|\eta, \epsilon, \psi >$, où d indique la métrique et η, ϵ, ψ le chemin vers l'AS$_\psi$;
  - dans l'état initial de l'AS$_\alpha$: pas d'information concernant l'AS$_\delta$
  - l'AS$_\alpha$ reçoit des chemins de l'AS$_\beta$ et de l'AS$_\gamma$ à son premier échange :

Routing table at AS α: Initial

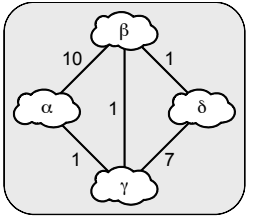
From	Path to α	Path to β	Path to γ
α	⟨0 | α⟩	⟨10 | β⟩	⟨1 | γ⟩
β	⟨∞ | ⟩	⟨∞ | ⟩	⟨∞ | ⟩
γ	⟨∞ | ⟩	⟨∞ | ⟩	⟨∞ | ⟩

+

Received Path Vectors (1st exchange)

From β	α	β	γ	δ
	⟨10 | α⟩	⟨0 | β⟩	⟨1 | γ⟩	⟨1 | δ⟩

From γ	α	β	γ	δ
	⟨1 | γ⟩	⟨1 | β⟩	⟨0 | γ⟩	⟨7 | δ⟩



Routing table at node α: After 1st exchange:

=

From	Path to α	Path to β	Path to γ	Path to δ
α	⟨0 | α⟩	⟨2 | γ, β⟩	⟨1 | γ⟩	⟨8 | γ, δ⟩
β	⟨10 | α⟩	⟨0 | β⟩	⟨1 | γ⟩	⟨1 | δ⟩
γ	⟨1 | α⟩	⟨1 | β⟩	⟨0 | γ⟩	⟨7 | δ⟩

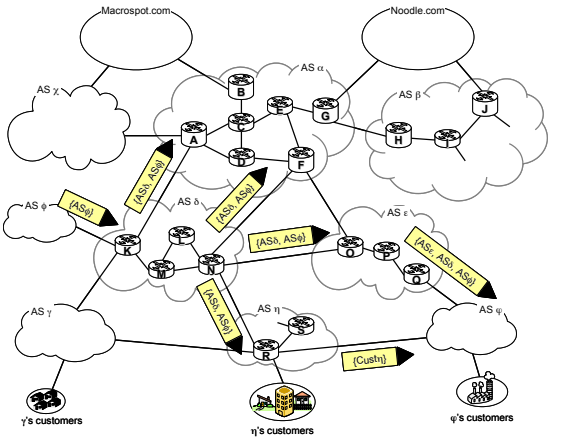
  - l'AS$_\alpha$ recalcule sa table de chemin :
    - $D_\alpha(\beta) = min\{c(\alpha,\beta) + D_\beta(\beta), c(\alpha,\gamma) + D_\gamma(\beta)\} = min\{10+0, 1+1\} = 2 \Rightarrow < 2|\gamma, \beta >$
    - $D_\alpha(\gamma) = min\{c(\alpha,\beta) + D_\beta(\gamma), c(\alpha,\gamma) + D_\gamma(\gamma)\} = min\{10+1, 1+0\} = 1 \Rightarrow < 1|\gamma >$
    - $D_\alpha(\delta) = min\{c(\alpha,\beta) + D_\beta(\delta), c(\alpha,\gamma) + D_\gamma(\delta)\} = min\{10+1, 1+7\} = 8 \Rightarrow < 8|\gamma, \delta >$

Deux objectifs différents de routage

- **à l'intérieur d'un AS** : «comment router d'une source vers une destination de la manière la plus efficace» ;
  - ◇ on utilise des protocoles IGP, «*Interior Gateway Protocols*» de type :
    - □ «par vecteur de distance» ;
    - □ «par état de liens» ;
- **entre AS** : «comment router de la manière la plus *profitable* ?» ;
  - ◇ on utilise des protocoles EGP, «Exterior Gateway Protocols» de type «par vecteur de chemin» ;
- un **routeur situé à la frontière** d'un AS, appelé «*gateway router*» ou «*speaker router*» :
  - ◇ doit maintenir deux tables différentes de routage :
    - □ une obtenue par un IGP ;
    - □ une obtenue par un EGP ;
  - ◇ organise son information de routage :
    - ⋆ à chaque réception d'une destination dans une autre AS, il l'intègre dans sa table de routage ;
    - ⋆ il échange ces informations avec tous les routeurs à l'intérieur de son AS afin d'unifier la **vision d'Internet** vu par tous les routeurs de l'AS, afin que :
      - ▷ **tous les routeurs** prennent la **même décision de routage** pour un datagramme ;
      - ▷ **tous les routeurs** partagent les tables de routage de **tous les routeurs de bordure** ;
    - ⋆ il échange ses informations de routage avec tous les routeurs à l'intérieur de son AS, c-à-d avec :
      - ▷ ceux qui sont eux-mêmes des routeurs de bordure situés en frontière de l'AS («*internal peering*») ;
      - ▷ les autres qui ne font fonctionner qu'un IGP avec une seule table de routage ;
    - ⋆ ces informations de routage contiennent :
      - ▷ des paires «(destination, port de sortie)» pour toutes les destinations possibles (le «port de sortie» correspond à l'@IP du routeur de prochain saut (sélection de la destination suivant le préfixe correspondant le plus long) ;
      - ▷ une entrée par défaut pour les adresses inconnues (seuls les AS «Tier-1» sont «*default-free*», car ils connaissent tous les préfixes réseaux d'Internet).

Si l'AS$_\phi$ diffuse le réseau `128.34.10.0/24` par un seul routeur «*speaker*» alors la table de routage est simple :

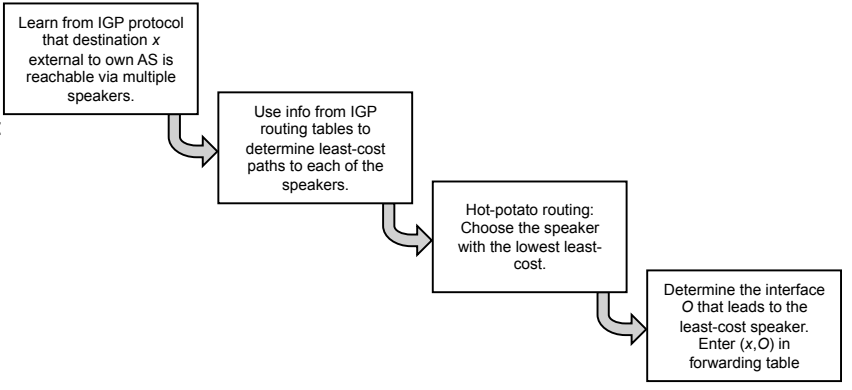
- Exemple, l'AS$_\eta$ possède un seul routeur «*speaker*» R qui le connecte aux autres AS ;
  - si le routeur S de l'AS$_\eta$ reçoit un paquet à destination de l'AS$_\phi$, ce paquet est routé suivant le plus court chemin (déterminé par un IGP à l'intérieur de AS$_\eta$) vers le routeur R qui le routera vers le routeur N dans l'AS$_\delta$.
- Exemple : le routeur B de l'AS$_\alpha$ reçoit un paquet destiné à l'AS$_\phi$;
  - B doit router ce paquet vers un autre «*speaker*» routeur, mais lequel ?



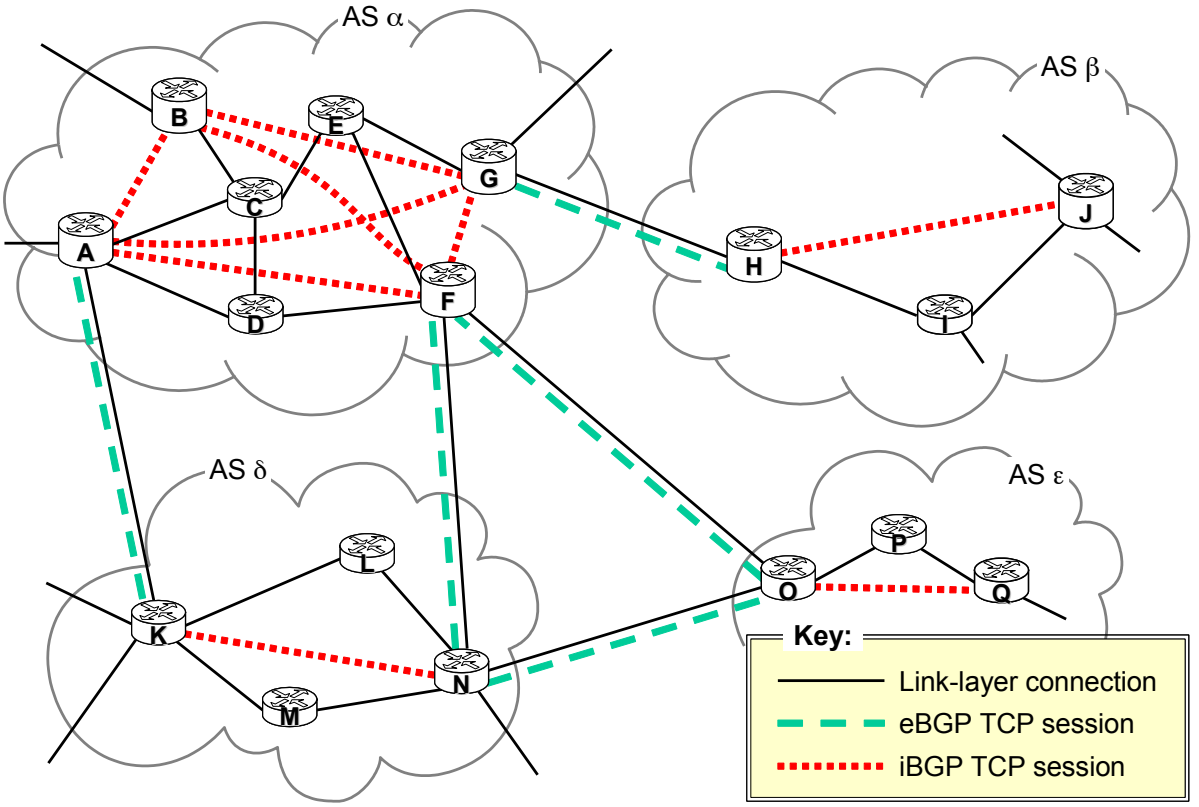
  - A et F possèdent le même chemin vers l'AS$_\phi$ car les routeurs «*speakers*» ont diffusé leurs informations à l'intérieur de l'AS à l'aide d'un IGP, ou d'une version adaptée de l'EGP (iBGP par exemple).

On utilise le routage «***hot potato***» :

- ▷ l'AS se «débarrasse» au plus vite et suivant la méthode la moins coûteuse :
  - le paquet est envoyé suivant le **plus court chemin à l'intérieur de l'AS**, vers le routeur «*speaker*» qui possède un chemin vers l'AS destination (pas de calcul du coût global).
    Sur l'exemple de B vers A et non vers F.

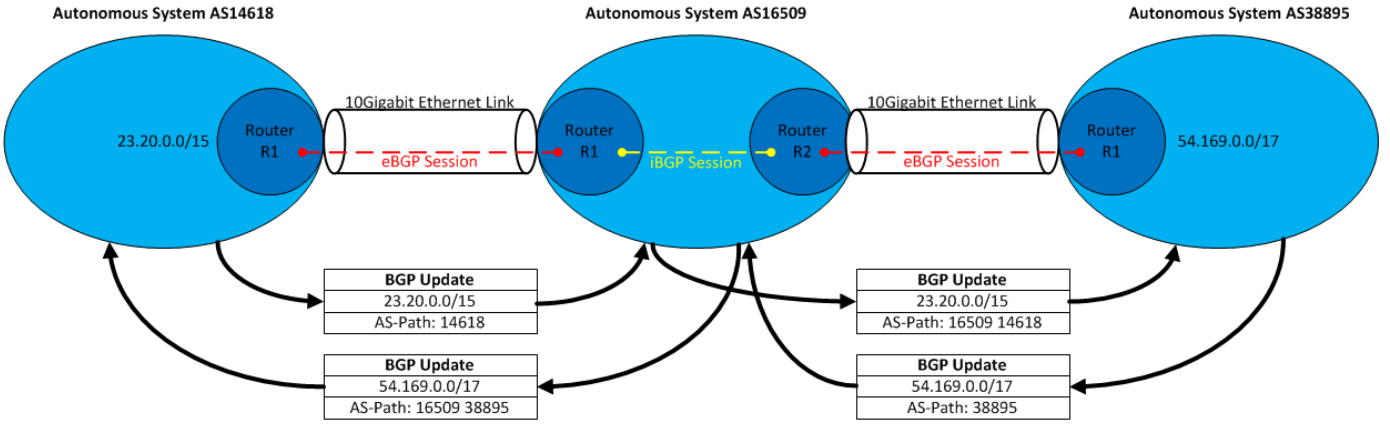


- c'est un protocole par «vecteur de chemin», «*path vector routing*» ;
- il utilise TCP et le port 179 ;
- il assure le routage entre AS, en répondant à ces questions :
  - *Quelle information de routage doit être diffusée aux autres AS ?*
  - *Comment traiter les informations reçues depuis les autres AS ?*
  - *Quelle information doit être rediffusée parmi les informations reçues ?*
  - *Comment une AS peut disposer d'une vue d'Internet partagée par tous ses routeurs de telle manière à ce qu'ils prennent la même décision de routage ?*
  - *Comment décider de diffuser ou non la possibilité de joindre d'autres AS voisines et de courir le risque de relayer du trafic sans obtenir de contrepartie ?*
- il est **complexe** :
  - dans l'implémentation dans les «*BGP speakers*», les routeurs de bordure, des contraintes «business»
  - pour la construction des chemins de routage ;
  - pour l'apprentissage des routes extérieures par les routeurs internes à l'AS.
- il met à jour de manière **incrémentale** les entrées de sa table de routage :
  - il n'y a pas d'échange périodique comme avec les IGPs tels que RIP ou OSPF ;
  - seules les modifications sont transmises ;
- une connexion TCP est **maintenue** avec chaque autre routeur BGP, «BGP session» :
  - une connexion reliant deux «speakers» routeurs appartenant à deux AS différentes ;
  - une connexion TCP entre deux «speakers» routeurs de la même AS pour réaliser du «*internal peering*» (on parlera de «BGP peers»).



- les sessions externes : «eBGP» ;
- les sessions internes : «iBGP» (besoin de $\frac{n*(n-1)}{2}$ connexions TCP).

BGP : échange des informations de routage



AS14618 Routing Table
Prefix: 23.20.0.0/15 AS-Path: Locally Originated
Prefix: 54.169.0.0/17 AS-Path: 16509 38895 Next-Hop: R1

AS16509 Routing Table
Prefix: 23.20.0.0/15 AS-Path: 14618 Next-Hop: R1
Prefix: 54.169.0.0/17 AS-Path: 38895 Next-Hop: R2

AS38895 Routing Table
Prefix: 23.20.0.0/15 AS-Path: 16509 14618 Next-Hop: R1
Prefix: 54.169.0.0/17 AS-Path: Locally Originated

BGP peering session

- correspond : session TCP établie entre deux routeurs d'AS différentes ;
- emprunte un lien : par exemple une liaison 10Gb/s ;
- échange des informations de routage : adresse IP de réseau et taille de préfixe.

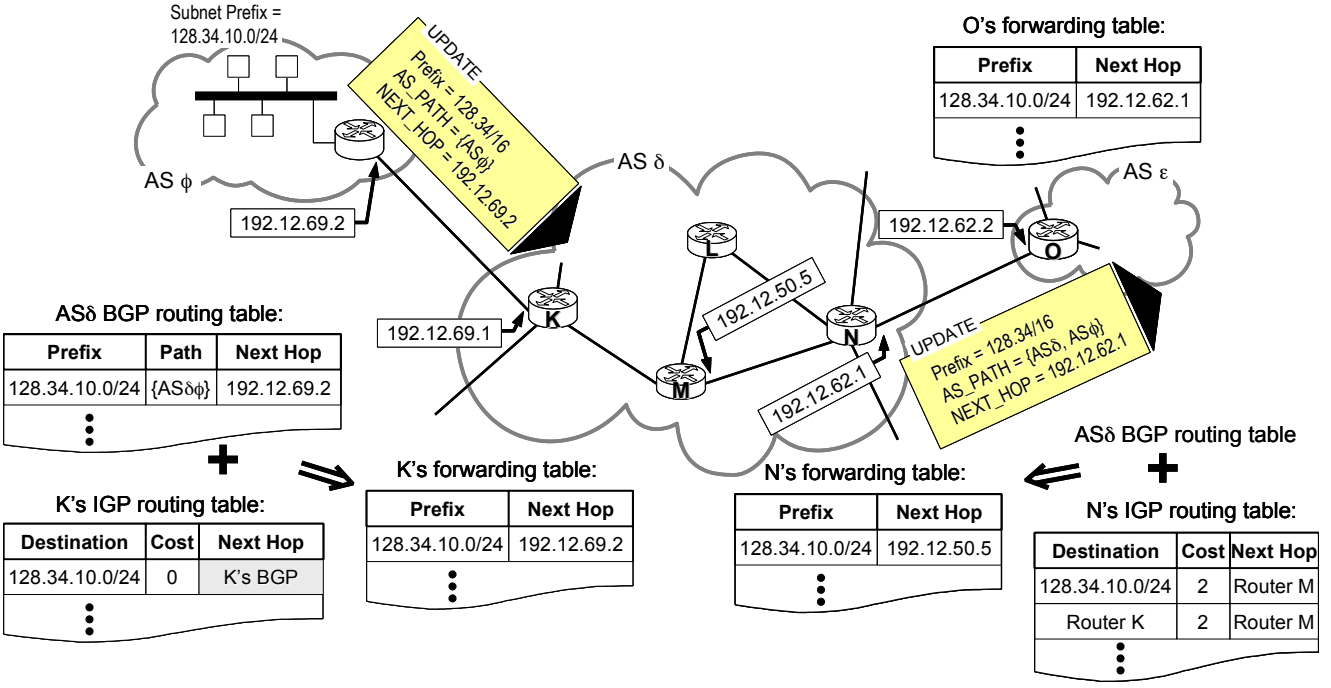
Les messages échangés :

- «UPDATE» :
  - «*announcement*» : informe des destinations atteignables, puis des destinations qui ont changé (mise à jour incrémentale) ;
  - «*withdrawal*» : supprime des destinations.
  - une route transmise au format CIDR est appelée NRLI, «Network Layer Reachability Information» :
    - préfixe de la destination ;
    - taille du préfixe ;
    - chemin des AS à traverser ;
    - adresse de prochain saut ;
    - des informations additionnelles pour décider de l'«*import policy*».
  - les informations de routage transmises ne sont pas la copie directe des informations disponibles mais celles filtrées par l'«*export policy*» ;
  - les informations de routage reçues sont filtrées suivant l'«*import policy*» qui définissent les règles de préférences.

Ces règles de préférences ne sont pas diffusées en dehors de l'AS (par eBGP) mais sont partagées à l'intérieur de l'AS (iBGP), elles sont définies à l'aide d'attributs définis dans la table ⟹

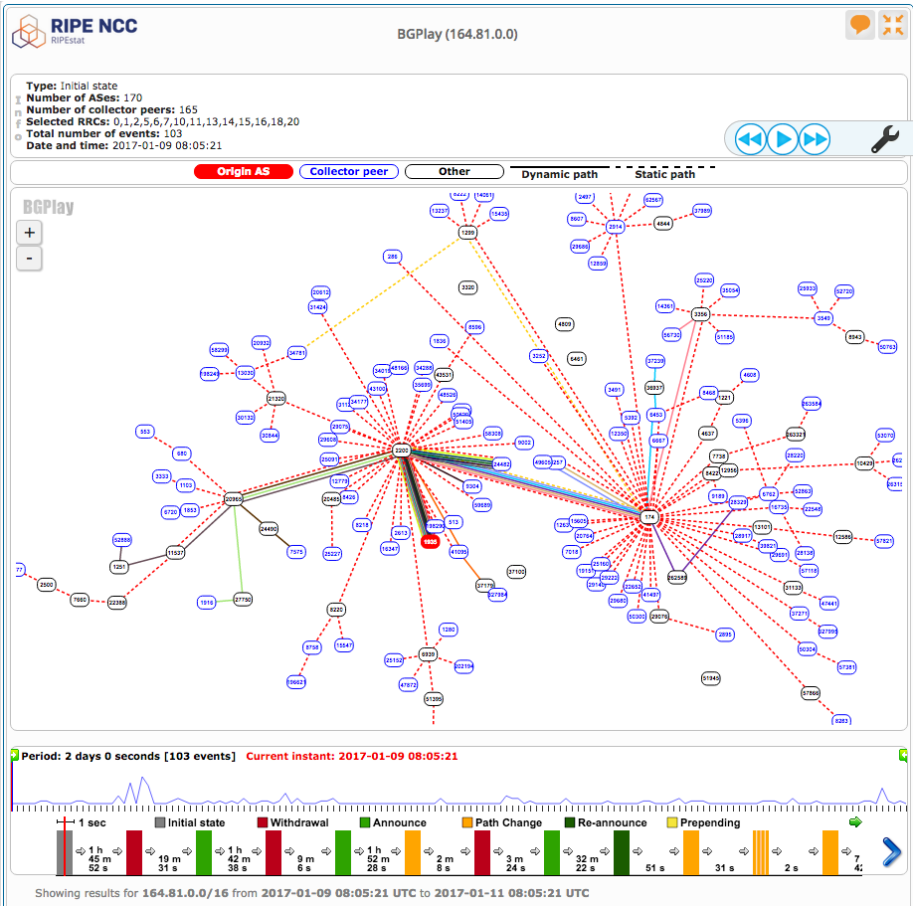
Priority	Rule	Comments
1	`LOCAL_PREF`	E.g., `LOCAL_PREF` specifies the order of preference as customer > peer > provider If more than one route remains after this step, go to the next step.
2	`AS_PATH`	Select shortest `AS_PATH` length (i.e., the list with the smallest number of ASNs, *not* smallest number of hops or lowest delay!)
3	`MED`	Select the route with the lowest `MULTI_EXIT_DISC` value, if there is financial incentive involved.
4	IGP path	Select the route for which the `NEXT_HOP` attribute, for which the cost in the IGP routing table is lowest, i.e., use hot-potato routing.
5	eBGP > iBGP	Select the route which is learned from eBGP over the one learned by iBGP (i.e., prefer the route learned first hand)
6	Router ID	Select the BGP router with the smallest IP address as the next hop.

Exemple de messages de type «UPDATE» échangés



- ⋆ tous les «*speakers*» routeurs de l'AS$_\delta$, K & N, possèdent la même table de routage BGP, appelée «ASδ BGP routing table» (cette table est échangée par iBGP) ;
- ⋆ chaque routeur dispose de sa table de routage propre, appelée «forwarding table» et construite à partir d'un IGP ;
- ⋆ chaque routeur associe ces deux tables pour décider de son routage inter et intra domaine.

BGP : l'évolution rapide des routes échangées



https://stat.ripe.net/widget/bgplay

HURRICANE ELECTRIC INTERNET SERVICES

Search

AS2200 Renater

Quick Links

- BGP Toolkit Home
- BGP Prefix Report
- BGP Peer Report
- Exchange Report
- Bogon Routes
- World Report
- Multi Origin Routes
- DNS Report
- Top Host Report
- Internet Statistics
- Looking Glass
- Network Tools App
- Free IPv6 Tunnel
- IPv6 Certification
- IPv6 Progress
- Going Native
- Contact Us

AS Info | Graph v4 | Graph v6 | Prefixes v4 | Prefixes v6 | Peers v4 | Peers v6 | Whois | IRR | IX

Company Website: http://www.renater.fr

Country of Origin: France

Internet Exchanges: 6

Prefixes Originated (all): 63
Prefixes Originated (v4): 62
Prefixes Originated (v6): 1

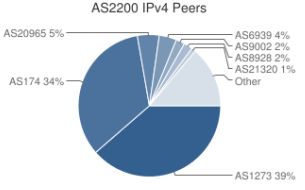
Prefixes Announced (all): 173
Prefixes Announced (v4): 169
Prefixes Announced (v6): 4

BGP Peers Observed (all): 84
BGP Peers Observed (v4): 84
BGP Peers Observed (v6): 35

IPs Originated (v4): 3,095,808
AS Paths Observed (v4): 721
AS Paths Observed (v6): 215

Average AS Path Length (all): 3.667
Average AS Path Length (v4): 3.825
Average AS Path Length (v6): 3.135

Like Be the first of your friends to like this.



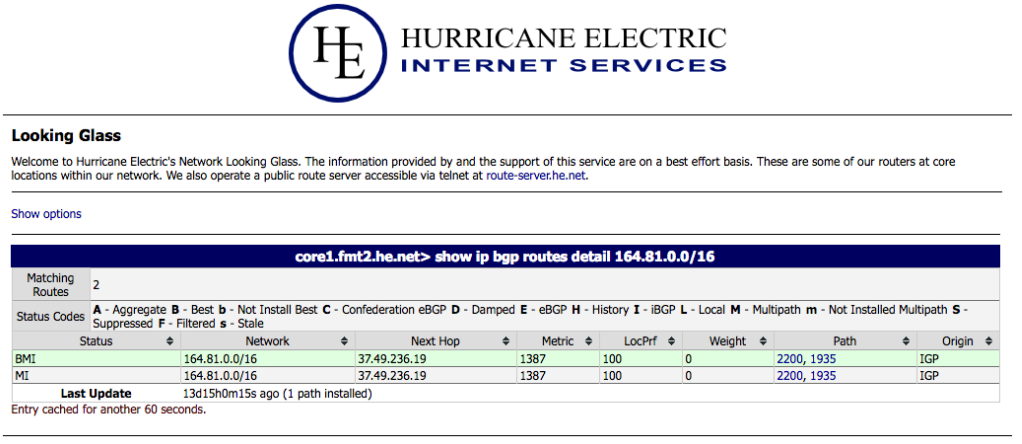
ASN	Name
AS1273	Cable and Wireless Worldwide plc
AS174	Cogent Communications
AS20965	GEANT Limited
AS6939	Hurricane Electric, Inc.
AS9002	RETN Limited
AS8928	Interoute Communications Limited
AS21320	GEANT Limited

bgp.he.net/AS2200

Un «*looking glass*» : le réseau de l'Université vu depuis Hurricane Electric

Depuis Frémont aux USA :

http://lg.he.net



HURRICANE ELECTRIC
INTERNET SERVICES

Looking Glass

Welcome to Hurricane Electric's Network Looking Glass. The information provided by and the support of this service are on a best effort basis. These are some of our routers at core locations within our network. We also operate a public route server accessible via telnet at route-server.he.net.

Show options

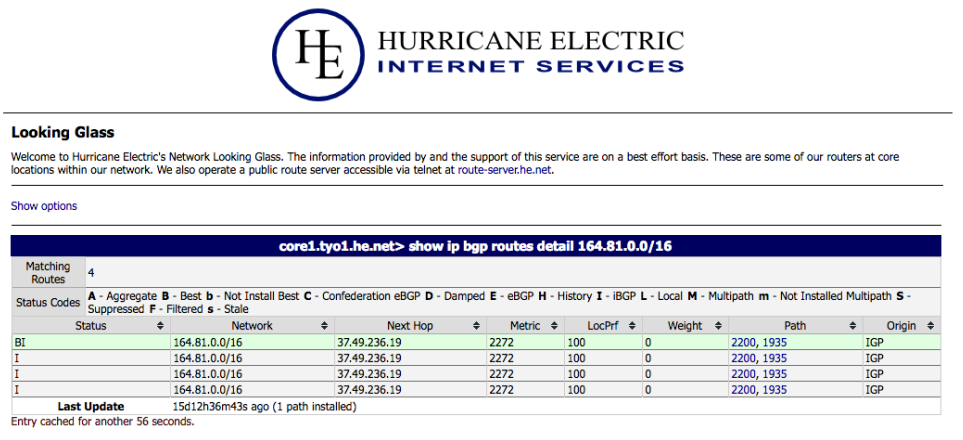
core1.fmt2.he.net> show ip bgp routes detail 164.81.0.0/16

Matching Routes	2
Status Codes	**A** - Aggregate **B** - Best **b** - Not Install Best **C** - Confederation eBGP **D** - Damped **E** - eBGP **H** - History **I** - iBGP **L** - Local **M** - Multipath **m** - Not Installed Multipath **S** - Suppressed **F** - Filtered **s** - Stale

Status	Network	Next Hop	Metric	LocPrf	Weight	Path	Origin
BMI	164.81.0.0/16	37.49.236.19	1387	100	0	2200, 1935	IGP
MI	164.81.0.0/16	37.49.236.19	1387	100	0	2200, 1935	IGP
Last Update	13d15h0m15s ago (1 path installed)						

Entry cached for another 60 seconds.

Depuis Tokyo au Japon :



HURRICANE ELECTRIC
INTERNET SERVICES

Looking Glass

Welcome to Hurricane Electric's Network Looking Glass. The information provided by and the support of this service are on a best effort basis. These are some of our routers at core locations within our network. We also operate a public route server accessible via telnet at route-server.he.net.

Show options

core1.tyo1.he.net> show ip bgp routes detail 164.81.0.0/16

Matching Routes	4
Status Codes	**A** - Aggregate **B** - Best **b** - Not Install Best **C** - Confederation eBGP **D** - Damped **E** - eBGP **H** - History **I** - iBGP **L** - Local **M** - Multipath **m** - Not Installed Multipath **S** - Suppressed **F** - Filtered **s** - Stale

Status	Network	Next Hop	Metric	LocPrf	Weight	Path	Origin
BI	164.81.0.0/16	37.49.236.19	2272	100	0	2200, 1935	IGP
I	164.81.0.0/16	37.49.236.19	2272	100	0	2200, 1935	IGP
I	164.81.0.0/16	37.49.236.19	2272	100	0	2200, 1935	IGP
I	164.81.0.0/16	37.49.236.19	2272	100	0	2200, 1935	IGP
Last Update	15d12h36m43s ago (1 path installed)						

Entry cached for another 56 seconds.

Algorithme de routage : vecteur de distance, «*vector-distance*»

Caractéristiques

- Itératif : fonctionnement tant qu'il y a des informations à échanger (jusqu'à la convergence de l'algorithme) ;
- Asynchrone : chaque nœud du réseau de routage est indépendant au niveau du temps ;
- Distribué : chaque nœud participe à la résolution de l'algorithme ;
- Local : aucun nœud n'a la vision complète du réseau.

Fonctionnement général

- ▷ Échange d'information entre routeurs adjacents
  - ◇ chaque routeur diffuse vers les autres nœuds adjacents leur table de routage ;
  - ◇ chaque table de routage est constituée de la liste des voisins et du coût de la liaison.
- ▷ Traitement à la réception d'une nouvelle table de routage :
  - ◇ si une entrée de la table n'est pas dans sa table, il la rajoute ;
  - ◇ si le coût de la route proposée par la table + le coût de la route pour aller jusqu'au routeur qui a transmis la table est inférieur au coût indiqué dans sa table, il modifie sa table pour prendre en compte cette nouvelle route ;
  - ◇ sinon il n'y a pas de changement.

La modification d'une entrée dans la table de routage d'un routeur engendre l'émission de la nouvelle table sur toutes les interfaces du routeur.

Les échanges entre les routeurs continuent jusqu'à ce que l'algorithme **converge** (plus de modification dans les tables des différents routeurs).

Utilisation d'une métrique simple

Il est possible de n'utiliser qu'une métrique simple pour le coût d'un chemin : le nombre de **sauts** pour atteindre la destination.

Inconvénients

- La taille des informations de routage est proportionnelle au nombre de routeurs du domaine.
- La métrique est difficilement utilisable : lenteur de convergence ;
- Bouclage éventuellement à l'infini ;
- Pas de chemins multiples ;
- Coûts des routes externes arbitraires.

Protocole RIP, «Routing Internet Protocol»

Il existe différentes versions :

– RIPv1 :
  - ◇ qui ne gère pas les préfixes dans les informations de routage transmises, d'où l'impossibilité d'utiliser différents préfixes ou de faire de l'aggrégation ;
  - ◇ qui ne supporte pas d'authentification : on peut facilement intercepter les paquets en les faisant router vers un routeur malveillant ;
– RIPv2, RFC 1388, 1723, 2453 : authentification (MD5+secret+compteur), préfixes transmis avec les routes, utilisation du multicast (224.0.0.9) ;
– RIPng, RFC 2080 : extension pour gérer IPv6.

L'équipement de routage diffuse (broadcast) toutes les 30s la liste des réseaux qu'il peut atteindre avec leur distance (nombre de sauts).

Avantages

– très connu, implanté sur tous les équipements de routage, peu gourmand en ressources CPU/Mémoire ;
– s'adapte automatiquement (panne, ajout de réseau ...)

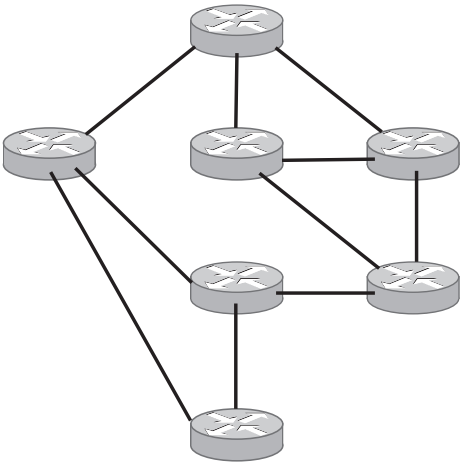
Désavantages

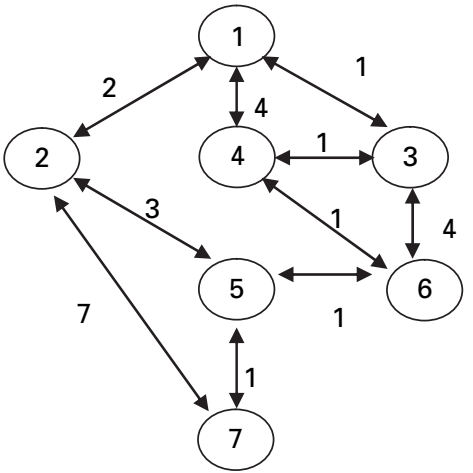
– la distance est une information réduite qui ne tient pas compte de la charge, du débit, du coût des lignes…
– distance maximale = 15 : d = 16 signifie réseau inaccessible (distance infinie) ;

Utiliser RIP sur un petit réseau que l'on contrôle et où l'on fait confiance aux administrateurs réseau.

Les algorithmes de constructions des chemins de routage

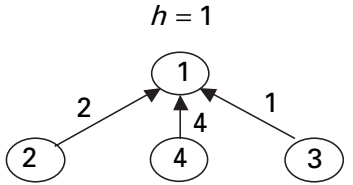
Depuis le schéma d'interconnexion des différents routeurs, on déduit un graphe dont les arêtes sont étiquetées par un poids :





Sur le graphe on ne prend en compte que les routeurs que l'on numérotera et pas les réseaux auxquels ils sont directement connectés pour simplifier la description du fonctionnement de l'algorithme.

Le déroulement de l'algorithme de Belmann-Ford



$h = 1$: Le nœud 1 apprend la distance qui le relie aux nœuds 2, 4 et 3 :

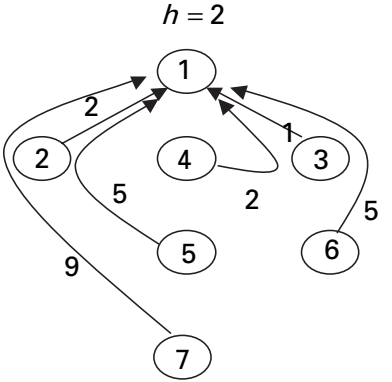
nœud	distance
2	2 par 2
4	4 par 4
3	1 par 3

$h = 2$: le nœud 1 reçoit les informations concernant les nœuds 5, 6 et 7 :

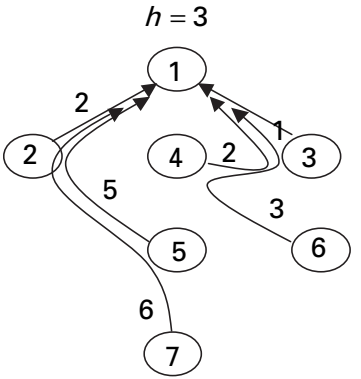
nœud	distance
5	5 (3+2) par 2
6	5 (4+1) par 3
7	9 (7+2) par 2

et mets à jour :

nœud	distance
4	2 (1+1) par 3



Le déroulement de l'algorithme de Belmann-Ford – Suite

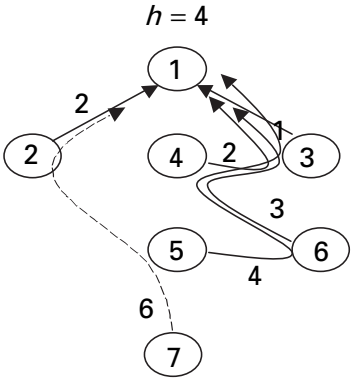


$h = 3$: le nœud 1 reçoit des informations concernant les nœuds 7 et 6 :

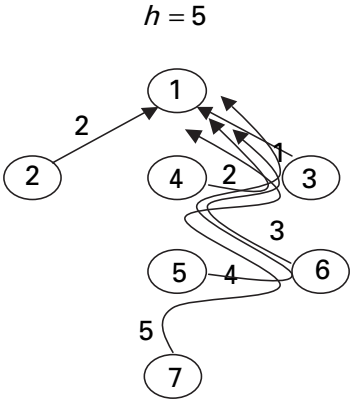
nœud	distance
6	3 (2+1) par 3
7	6 (5+1) par 2

$h = 4$: le nœud 1 reçoit des informations concernant le nœud 5, ce qui va modifier les informations du 7 :

nœud	distance
5	4 (3+1) par 3
7	6 (5+1) par 2 ⟶ à modifier...

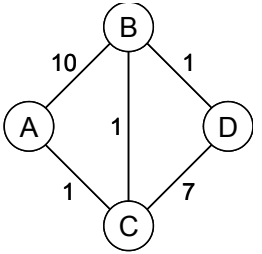


Le déroulement de l'algorithme de Belmann-Ford – Suite et fin



nœud	distance
2	2 par 2
3	1 par 3
4	2 par 3
5	4 par 3
6	3 par 3
7	5 (4+1) par 3

Autre exemple de réseau :



Pour le routeur A :

- il reçoit les «vecteurs de distance» des routeurs B et C (sur l'exemple, on considère que ces informations arrivent simultanément, ce qui est rarement le cas **mais ne modifie** pas le résultat de l'algorithme) :

Routing table at node *A*: Initial

Distance to

From	A	B	C
A	0	10	1
B	∞	∞	∞
C	∞	∞	∞

+

Received Distance Vectors

	A	B	C	D
From B	10	0	1	1
From C	1	1	0	7

=

Routing table at node *A*: After 1st exchange

Distance to

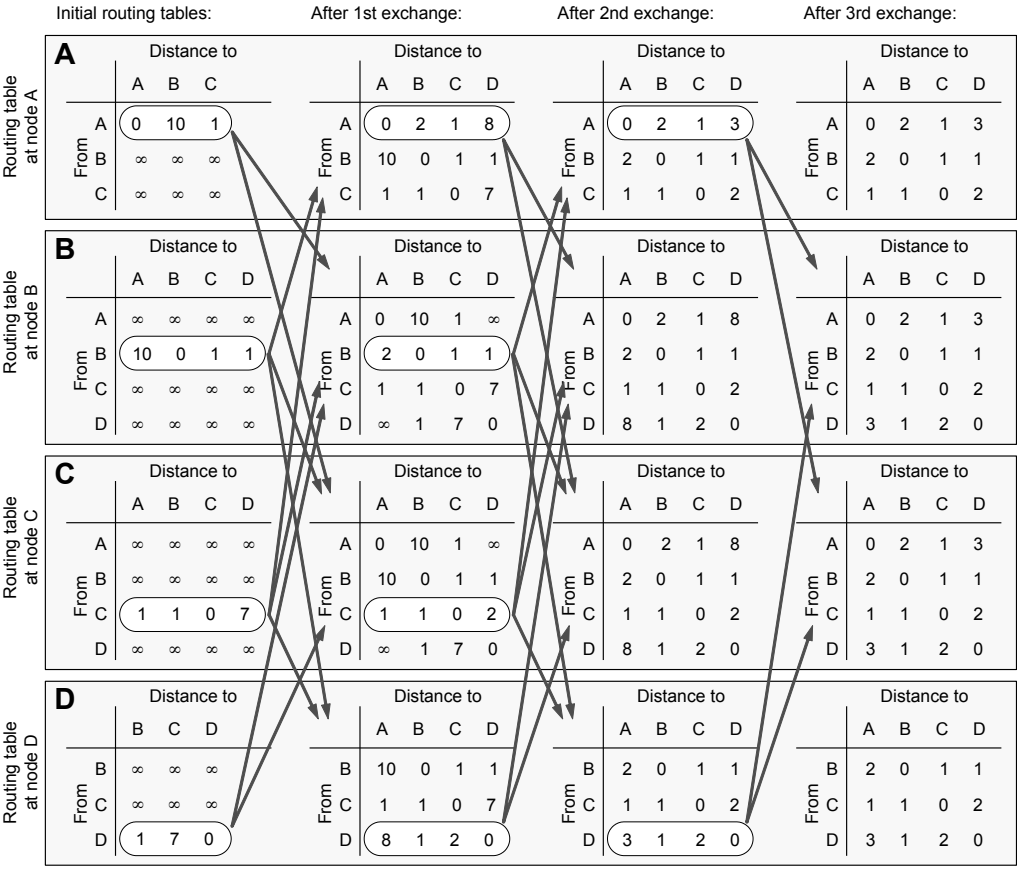
From	A	B	C	D
A	0	2	1	8
B	10	0	1	1
C	1	1	0	7

- le routeur A mets à jour sa table avec les nouvelles informations reçues :
  - $D_A(B) = min\{c(A,B) + D_B(B), c(A,C) + D_C(B)\} = min\{10+0, 1+1\} = 2$
  - $D_A(C) = min\{c(A,B) + D_B(C), c(A,C) + D_C(C)\} = min\{10+1, 1+0\} = 1$
  - $D_A(D) = min\{c(A,B) + D_B(D), c(A,C) + D_C(D)\} = min\{10+1, 1+7\} = 8$

Par vecteur de distance

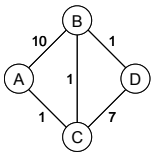
L'algorithme est **distribué** :

- chaque routeur échange ses informations ;
- mets à jour sa table ;
- si la table est modifiée, il la rediffuse ;
- il arrête lorsque la table est stable.

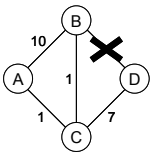


- **Gérer les plantages**
  - ◇ les routeurs doivent informer leurs voisins périodiquement de toute modification de la topologie du réseau :
    - ⋆ un routeur peut détecter les liens de connexion avec des paquets «*heartbeat*» ou «HELLO» ;
    - ⋆ mais si un **routeur plante** il ne peut prévenir ses voisins de ce changement ⟹ les routes doivent être associées à une durée de validité après laquelle est est retirée de la table de routage.
- **Éviter les boucles de routage**
  - ◇ l'algorithme fonctionne bien si les liens sont toujours «*up*» ;
  - ◇ lorsqu'un lien est coupé, un routeur recalcule son vecteur de distance et le distribue à ses voisins, mais **sans les informer** des raisons de ce recalcul :
    - ⋆ les routeurs voisins ne peuvent pas savoir si leur choix d'adresse de prochain saut va créer une boucle :
      - ▷ les informations diminuant le coût du routage circulent rapidement ;
      - ▷ les informations augmentant le coût du routage circulent lentement par petits incréments ;
      - ▷ ce problème est appelé «*counting to infinity problem*», exemple :

Original network



Scenario 2: Link BD outage



Ensuite, B diffuse sa nouvelle table et...le compteur démarre !

- ⋆ Le lien «BD» casse, B le détecte et mets à jour sa distance vers D à ∞ ;
- ⋆ si B envoie *immédiatement* sa nouvelle distance à C va détecter cette cassure car sa route vers D passe par B ;
- ⋆ Mais *imaginons* que C envoie sa table à B avant, il mets à jour

$$D_B(D) = min\{c(B,A) + D_A(D), c(B,C) + D_C(D)\} = min\{10+3, 1+2\} = 3$$

Routing table at node *B* **before** *BD* outage

From \ Distance to	A	B	C	D
A	0	2	1	3
B	2	0	1	1
C	1	1	0	2
D	3	1	2	0

1. *B* detects *BD* outage
2. *B* sets $c(B, D) = \infty$
3. *B* recomputes its distance vector
4. *B* obtains 3 as the shortest distance to *D*, via *C*

Routing table at node *B* **after** *BD* outage

From \ Distance to	A	B	C	D
A	0	2	1	3
B	2	0	1	**3**
C	1	1	0	2
D	3	1	2	0

Problème : les routeurs ont une vision limitée de la topologie du réseau, et C ne sait pas que sa route passe par B !

Lors d'une boucle de routage, le datagramme va rebondir d'un routeur à l'autre jusqu'à ce que son TTL devienne nul.
Les distances menant à la destination dans les tables de routages de ces deux routeurs ne cessent d'augmenter :«*counting-to-infinity*».

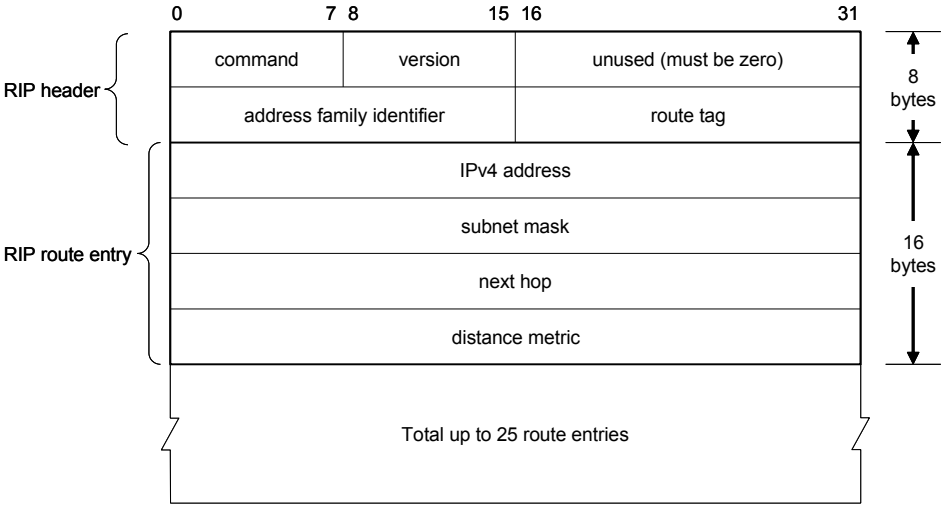
Solutions :

- introduire un temps d'attente de modification, «*hold-down*», supérieur au temps de convergence de l'algorithme :
  - ◇ lorsque le routeur détecte une rupture il déclenche sa mesure du temps d'attente ;
  - ◇ il diffuse l'information de cette rupture à ses voisins ;
  - ◇ les voisins font de même et déclenchent leur mesure du temps d'attente ;
  - ◇ il **ignore** les mises à jour réalisée par des routeurs qui ignorent la rupture ;
  - ◇ à la fin du temps d'attente, on reprends l'algorithme en mode normal.

- utiliser la solution du «*split-horizon*» : «*il est inutile d'envoyer une information concernant une destination vers un routeur qui est le prochain saut vers cette destination*»
  Dans l'exemple précèdent, C n'aurait pas diffuser de route vers D à B et n'aurait pas créer de boucle.

- utiliser la solution du «*split-horizon with poisoned reverse*» :
  - ◇ le routeur diffuse à tout ses voisins l'information de la rupture vers la destination en indiquant une distance ∞ ;
  - ◇ chaque routeur recevant cette information mets sa propre information à ∞ ce qui le force à chercher un nouveau chemin vers la destination ;
  - ◇ la valeur ∞ sert de «marquage» pour indiquer la rupture.
    Dans l'exemple précèdent, C diffuse une route vers D à B, mais avec la valeur ∞.

Le format du paquet RIP

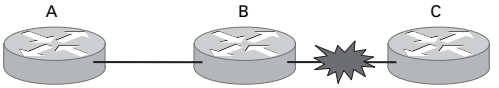


Il existe deux types de paquet RIP :

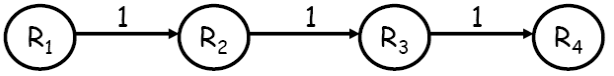
- ⋆ requêtes : qui demande la transmission des informations de routage aux routeurs (du groupe multicast) ;
- ⋆ réponses : qui contiennent les tables de routage (au plus 25 entrées de 16 octets chacune) :
  - ◇ le champs «next hop» sert lorsque le protocole RIP n'est pas exécuté par tous les routeurs du réseau : l'adresse doit correspondre à un «next hop» **directement accessible** depuis le sous-réseau dans lequel l'annonce est faite ;
  - ◇ une valeur `0.0.0.0` pour le «next-hop», indique que la route passe par l'émetteur du paquet ;
  - ◇ la métrique vaut au plus 15 (16 est considéré comme ∞) ;
  - ◇ les paquets sont diffusés toutes les 30 secondes et un routeur qui ne donne plus de nouvelles depuis 180 secondes (valeur du hold-down est considéré comme disparu.

Les mauvaises nouvelles se transmettent lentement !

Le problème du lien qui casse :



Détails



si le lien entre R3 et R4 se casse les routes se mettent à jour lentement pour atteindre un coût infini pour aller vers R4

Time	R_1	R_2	R_3
0	3,R_2	2,R_3	1, R_4
1	3,R_2	2,R_3	3,R_2
2	3,R_2	4,R_3	3,R_2
3	5,R_2	4,R_3	5,R_2
...	"Counting to infinity" ...		

Améliorations de l'algorithme

- définir l'infini comme un entier petit (16 par exemple) ;
- prévenir les routeurs en transmettant une route avec un coût infini, «reverse poisoning» & «split-horizon».

Algorithme de routage : par état de lien, «*state-link*»

Principe

Les routeurs :

- émettent des messages, LSP, «Link-State Packet», réalisant du LSA, «Link-State Advertisement», concernant l'état de chacun de ses liens et leur coût (d'où le nom de «Link State Routing») ;
- calculent tous les plus courts chemins de tous les noeuds vers lui-même ;
- envoient à tous les routeurs accessibles l'information au sujet de ses voisins ;
- récupèrent la topologie du réseau et le coût de chaque liaison.

Chaque routeur :

- possède une copie complète de la carte du réseau ;
- exécute le calcul des meilleurs routes localement en utilisant cette carte : plus de boucles !

Algorithme de routage OSPF, «*Open Shortest Path First*», RFC 1247

Il est destiné à remplacer les protocoles intérieurs propriétaires :

- les routeurs :
  - maintiennent une **carte complète du réseau** ;
  - calculent les meilleurs chemins **localement** en utilisant cette topologie ;
- les routeurs communiquent l'état de son voisinage :
  - le routeur teste périodiquement l'état des liens qui le relient à ses routeurs voisins, puis diffuse ces états (Link-State) à **tous les autres routeurs** du domaine, par inondation, «*flooding*» (un numéro de séquence permet de ne pas retransmettre un même message, et une durée de vie permet de les supprimer) ;
  - les messages diffusés contiennent l'état (up, down) et le coût d'un lien pour chacun des routeurs adjacents, ainsi que l'identifiant de l'émetteur ;
- lorsque un message parvient à un routeur, celui-ci :
  - met à jour sa carte de liens ;
  - recalcule localement pour chaque lien modifié, la nouvelle route selon l'algorithme de Dijkstra «shortest path algorithm» qui détermine le plus court chemin pour toutes les destinations à partir d'une même source.

L'algorithme du plus court chemin de Dijkstra

1. on commence avec le noeud source : il est étiqueté comme permanent et sa distance au noeud source est évidemment nulle.
   C'est le noeud actif ;

2. tous les noeuds adjacents au noeud actif sont examinés tour à tour ;

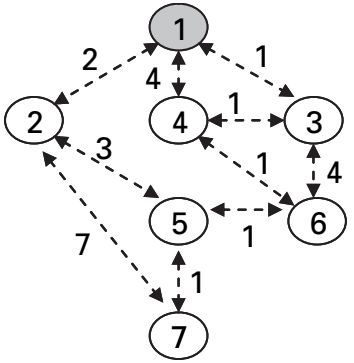
3. chaque noeud est étiqueté en indiquant le meilleur chemin connu au noeud source et la liaison à utiliser pour l'atteindre ;

4. à chaque tour, le «noeud actif» est celui qui, parmi tous les noeuds étiquetés du réseau, possède la valeur la plus faible vers le noeud source.
   Son étiquette devient permanente ;

5. on recommence au point 2 avec le nouveau noeud actif ;

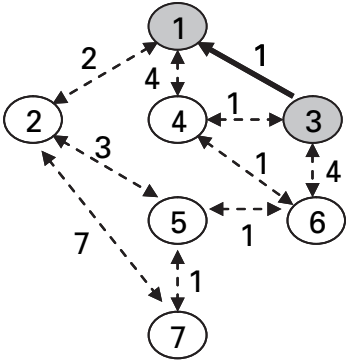
6. l'algorithme s'arrête quand l'étiquette de noeud destination est permanente.

Le déroulement de l'algorithme de Dijkstra

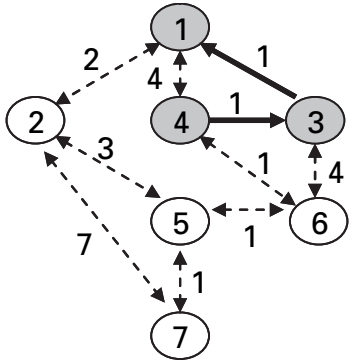


nœud	distance
1	0 par 1

nœud	distance
1	0 par 1
3	1 par 3

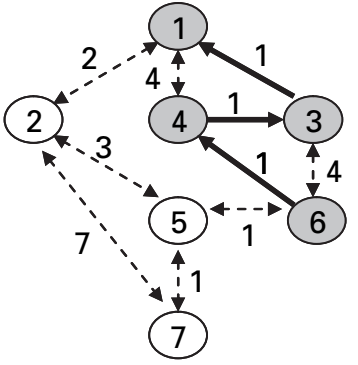


Le déroulement de l'algorithme de Dijkstra – Suite

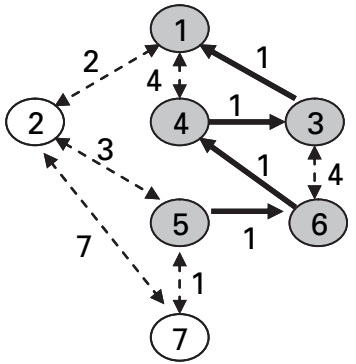


nœud	distance
3	1 par 3
4	2 par 3

nœud	distance
3	1 par 3
4	2 par 3
6	3 par 3

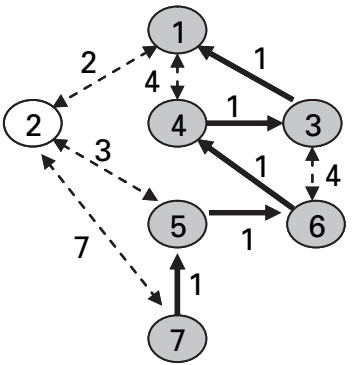


Le déroulement de l'algorithme de Dijkstra – Suite

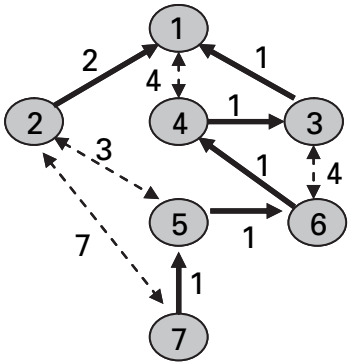


nœud	distance
3	1 par 3
4	2 par 3
5	4 par 3
6	3 par 3

nœud	distance
3	1 par 3
4	2 par 3
5	4 par 3
6	3 par 3
7	5 par 3



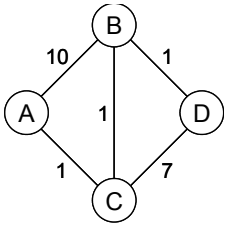
Le déroulement de l'algorithme de Dijkstra – Suite et fin



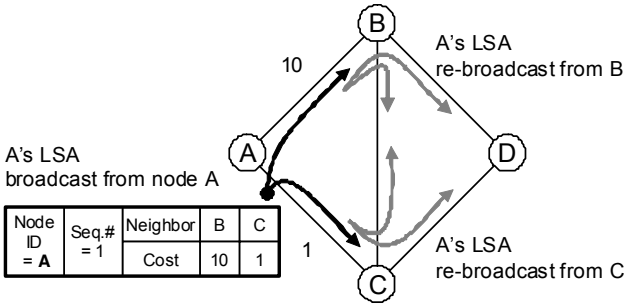
nœud	distance
2	2 par 2
3	1 par 3
4	2 par 3
5	4 par 3
6	3 par 3
7	5 par 3

Par état de lien : autre exemple

Soit le réseau suivant :



Le routeur A diffuse le LSA suivant :



Node ID = **A**	Seq.# = 1	Neighbor	B	C
		Cost	10	1

Tous les routeurs effectuent le même travail et diffusent leur LSA :

LSP from node B

Node ID = **B**	Seq.# = 1	Neighbor	A	C	D
		Cost	10	1	1

LSP from node D

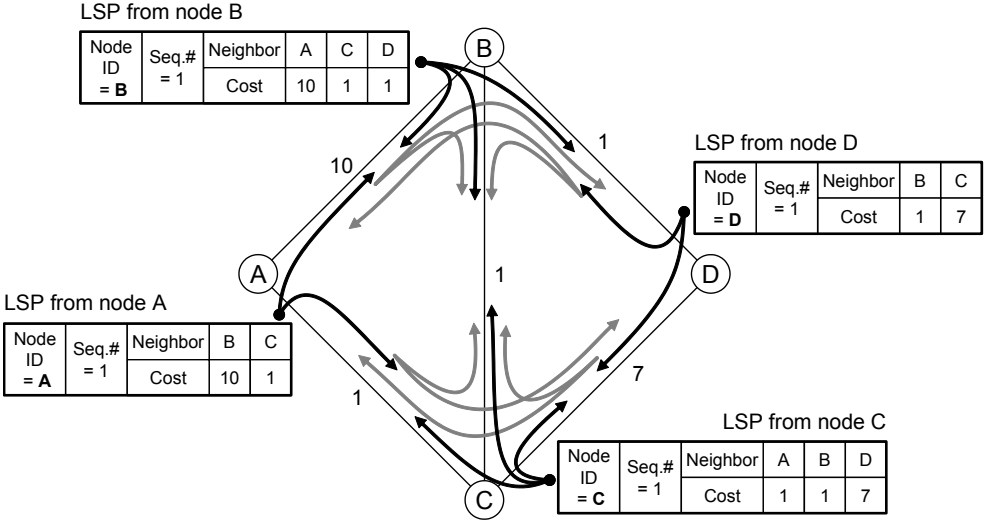
Node ID = **D**	Seq.# = 1	Neighbor	B	C
		Cost	1	7

LSP from node A

Node ID = **A**	Seq.# = 1	Neighbor	B	C
		Cost	10	1

LSP from node C

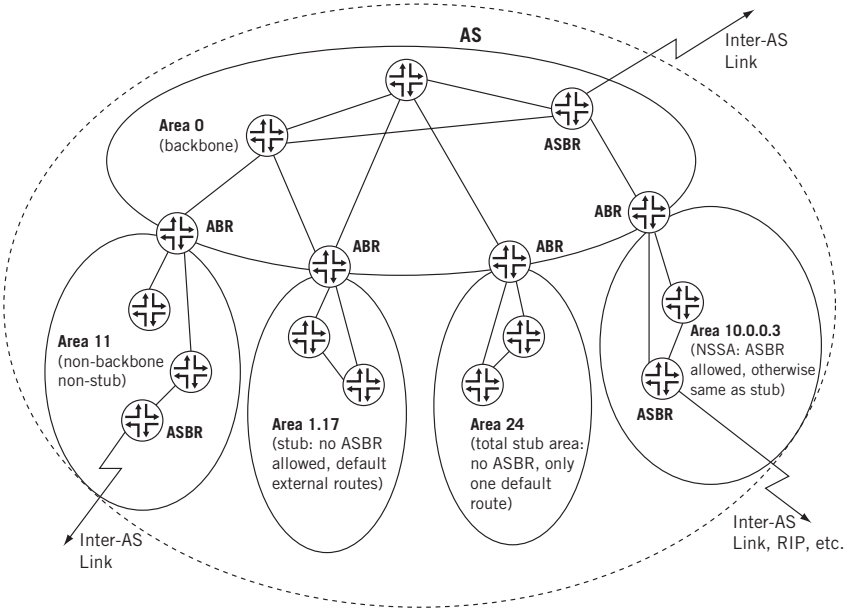
Node ID = **C**	Seq.# = 1	Neighbor	A	B	D
		Cost	1	1	7



«Par vecteur de distance» *vs* «par état de liens»

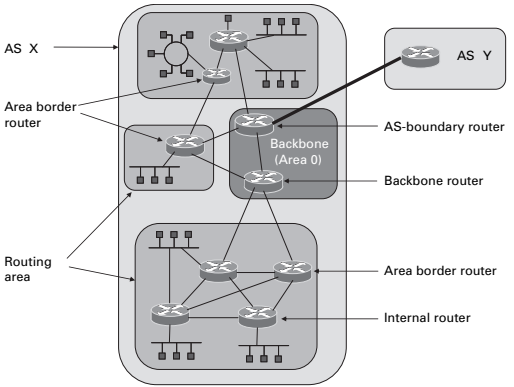
- le routage «par état de liens» :
  - consomme plus de ressource CPU/mémoire sur un routeur ;
  - crée beaucoup de trafic dans le réseau d'inter-connexion à cause du «flooding», inondation, des paquets LSA de chaque nœuds ;
  - converge plus rapidement lors de la rupture d'un lien ;
  - correspond à un travail local sur chaque routeur ;
  - les tables de routage ne contiennent **toujours** que l'adresse de prochain saut et non le chemin complet même s'il est connu ;
  - peut souffrir de «boucle de routage» si les routeurs ne travaillent pas tous sur la même carte du réseau (erreur de configuration, travail avec des LSA ne tenant pas compte d'une rupture de lien).

- le routage «par vecteur de distance» :
  - moins gourmand en ressource ;
  - implémenté dans pratiquement tous les routeurs ;
  - facile à configurer ;
  - correspond à un travail distribué : le travail de chaque routeur contribue au résultat de l'algorithme ;
  - les tables de routage ne contiennent **toujours** que l'adresse de prochain saut ;
  - les messages échangées peuvent devenir gros (beaucoup de préfixe de réseaux à échanger).



Le routage est hiérarchisé pour simplifier le calcul des routes :

- le Système Autonome (AS) est découpé en AREAs :
  - ◇ un area est un ensemble de réseaux contigües identifié par un numéro sur 32bits exprimé en notation réseau ou en notation décimale : `0.0.1.5` ou Area 261 ;
  - ◇ chaque area se comporte comme un réseau indépendant ;
- deux niveaux de routage :
  - ◇ intra-area ;
  - ◇ inter-area.



- ⋆ il existe différents types de routeurs suivant leur position :
  - ◇ ABR, «*area border router*» appartiennent à plusieurs areas (backbone et area normale) et transmettent les informations récapitulatives des «*areas*» qu'ils relient ;
  - ◇ ASBR, «*autonomous system boundary router*» gèrent les liens avec l'extérieur de l'AS ;
  - ◇ intra-area et ceux situés dans le «*backbone*».

- ⋆ il existe 5 types d'areas :
  - ◇ l'area 0 ou `0.0.0.0`, «*backbone*» :
    - ⋆ si une AS se décompose en une seule area, alors elle sera de type 0 ;
    - ⋆ cette area est la seule à générer la carte de routage utilisée par les autres areas ;
    - ⋆ le routage inter-area passe par ce «backbone» ;
  - ◇ «*Stub Area*» : ne possède pas de lien vers l'extérieur, mais connait des informations de routage sur les autres areas ;
  - ◇ «*Non-backbone, Non-Stub Area*» : similaire en fonctionnement au backbone mais sans l'être ;
  - ◇ «*Total Stub Area*» : ne dispose que d'un lien vers le «backbone» en tant que **route par défaut** ;
  - ◇ «*Not-So-Stubby-Area*» : peut posséder un lien vers l'extérieur mais le gère en concertation avec les routeurs du backbone ;

Caractéristiques

- il est compatible CIDR et VLSM ;
- il permet de gérer plusieurs routes pour une même destination selon des critères différents (ex : délai court, débit important…) ;
- il permet d'équilibrer la charge, «load balancing», entre des routes de coûts équivalents ;
- il peut utiliser le Type of Service (ToS) présent dans les en-têtes des datagrammes suivant 5 classes (RFC 1349) pour contrôler le routage ;

L'échange entre les routeurs sont **authentifiés** et un calcul d'intégrité des messages est réalisé.

Avantages

⋆ plus de limitation sur la taille des réseaux (diamètre > 16).
⋆ amélioration du temps de convergence
⋆ métrique plus sophistiquée (prise en compte des débits)

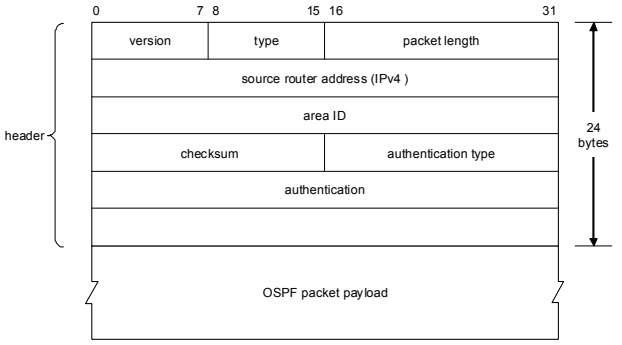
Inconvénients

▷ plus complexe : nécessite des routeurs plus puissants, sa configuration est également plus complexe.

Calcul des métriques en fonction du débit suivant les différentes technologie sur 16bits:

$$cout = \frac{10^9}{bande\_passante\_en\_bps}$$

Exemple : pour Ethernet à 100Mbit/s le coût est de 10.



Le paquet OSPF contient :

- un identifiant de routeur ;
- un identifiant d'area ;
- une authentification permettant d'éviter de prendre en compte des paquets émis de manière malveillante ;

Il existe 5 types différents de paquets :

1. Hello;
2. Database Description;
3. Link State Request;
4. Link State Update;
5. Link State Acknowledgment

- «*Hello*» : envoyés périodiquement (<30minutes).
  - le contrôle de l'état du lien ;
  - la découverte du voisinage ;
  - d'établir la connectivité et l'adjacence avec ce voisinage ;
  - d'élire un DR, «designated router» et un BDR, «backup designated router».
- LSA, «*Link State Advertisements*» : contient l'état des liens du routeur et envoyé en «flooding».
  Lorsqu'un routeur reçoit ce type de paquet, il met à jour sa LSD, «Link State Database».
  Entre deux routeurs adjacents, ils mettent à jour leur *database* pour vérifier qu'il n'y a pas de différence en échangeant des paquets «database description».
  Dans le cas où il y en a une il envoie un paquet «link state request».
- Quand un paquet LSA est envoyé le routeur attend un «link state acknowledgment», sinon il le renvoie.

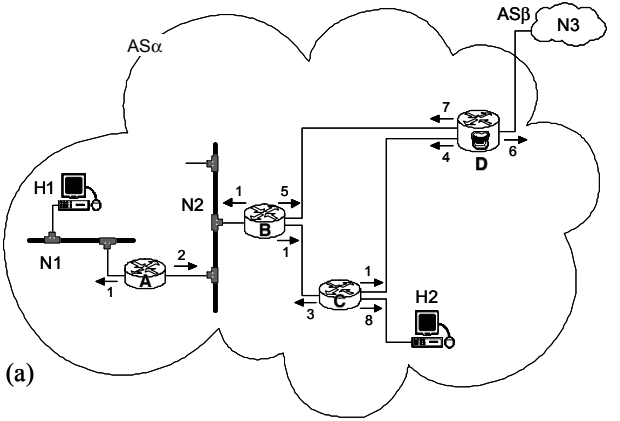
Le «DR» permet de diminuer l'envoi des états de liens :

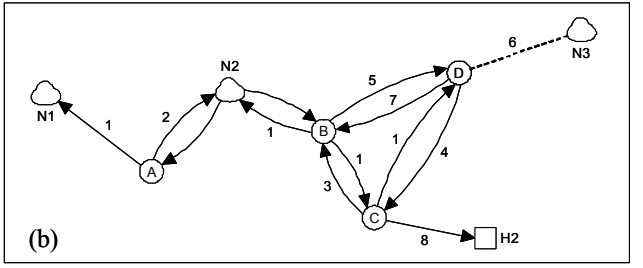
- au lieu que chaque routeur transmette à tous l'état de ses liens ;
- seul celui désigné par élection comme DR, pour un ensemble de routeurs, envoi l'état des liens de cet ensemble à tous les routeurs.

OSPF : exemple

Sur l'exemple l'AS$_\alpha$ utilise OSPF comme IGP :

- ▷ OSPF représente un réseau comme un graphe orienté dans le LSD :
  - ◇ les sommets correspondent à un routeur ou à un réseau ;
  - ◇ une arête du graphe :
    - ⋆ reliant deux routeurs correspond à :
      - ▷ un lien «point à point» physique entre deux routeurs ;
      - ▷ un accès à un réseau de transit en mode diffusion auquel ces routeurs sont tous deux connectés ;
    - ⋆ reliant un routeur à un réseau indique que le routeur dispose d'une interface connectée au réseau ;
  - ◇ un réseau peut être :
    - ⋆ de «transit» : indiqué par une flèche entrante et une flèche sortante ;
    - ⋆ un «stub» : indiqué par une seule flèche entrante.
  - ◇ un routeur connecté vers une autre AS est un «*speaker*» routeur.





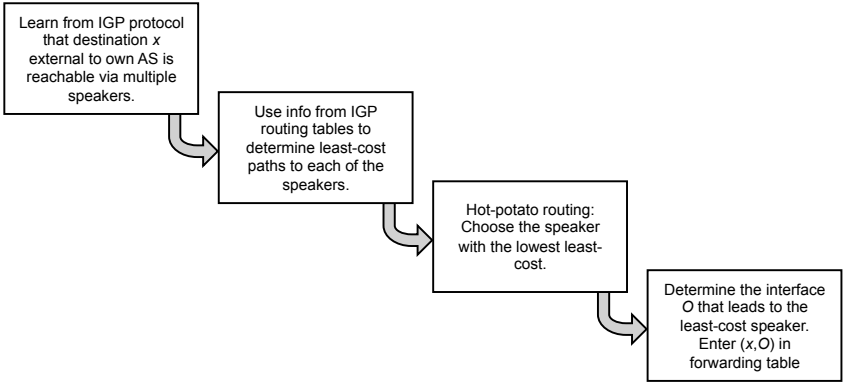
Différences entre les EGP, comme BGP, et les IGP, comme OSPF ?

La différence n'est pas technique mais elle est **administrative** :

- les IGP sont utilisés dans une organisation (entreprise, FAI, association, *etc.*) où les décisions (ajout ou suppression de ligne) peuvent être prises par une autorité unique ;
- les IGP cherchent à déterminer la route la plus efficace en faisant **confiance** aux autres routeurs ;
- les EGP sont utilisés entre organisations distinctes, et parfois même concurrentes :
  - il n'est pas possible de prendre une décision qui s'impose à tous (on peut ne pas être prévenu des modifications des « pairs » avec lesquels on utilise l'EGP) ;
- les EGP ne font pas confiance aux autres routeurs :
  - le but n'est pas de trouver la meilleure route, mais **d'empêcher les routeurs** de choisir une route dont on ne voudrait pas ;

Techniquement, il n'est pas possible dans un IGP comme OSPF d'exprimer que l'on ne veuille pas communiquer avec un routeur (on peut juste exprimer que l'on veut communiquer avec un routeur, il faut disposer de moyen de faire du filtrage.

Intégration IGP/EGP :

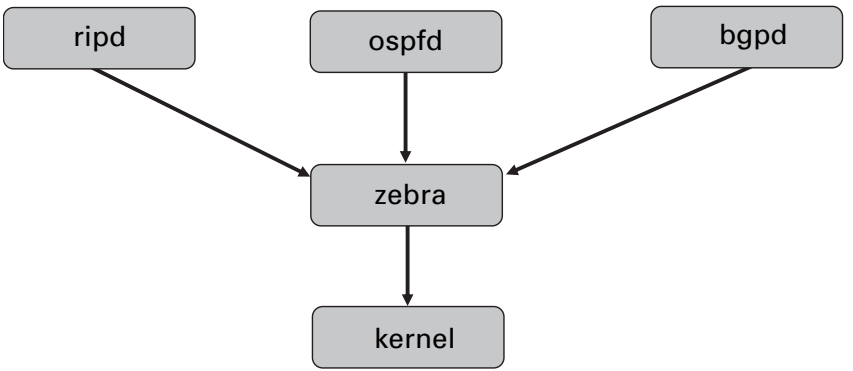


GNU/Linux :

– son noyau permet de router et de «commuter», «*forwarding*», les datagrammes entre différentes interfaces ;
– peut se tansformer en routeur complet, en utilisant des implémentations open-source des protocoles de routage, comme celles fournies par XORP ou Quagga.

Utilisation de Quagga

Quagga implémente les protocoles RIPv1, RIPv2, RIPng, OSPF, BGP, OSPF6 et IS-IS au travers de différents démons spécialisés dans la réalisation de chacun de ces protocoles.
Afin de coordonner les informations de routage utilisées et obtenues au travers des différents protocoles, un démon appelé «zebra» est utilisé :



Zebra sert également d'interface pour la configuration des différents protocoles, en particulier pour définir les routes statiques.

Pour utiliser quagga, il est nécessaire de l'installer :

xterm
```
$ sudo apt-get install quagga
```

Pour l'activer, il faut éditer le fichier «/etc/quagga/daemons» :

xterm
```
$ sudo vi /etc/quagga/daemons
```

Et modifier les lignes pour activer les démons «zebra» et «ripd» :

```
1 zebra=yes
2 bgpd=no
3 ospfd=no
4 ospf6d=no
5 ripd=yes
6 ripngd=no
7 isisd=no
```

Copier les fichiers de configuration :

xterm
```
# cp /usr/share/doc/quagga/examples/zebra.conf.sample /etc/quagga/zebra.conf
# cp /usr/share/doc/quagga/examples/ripd.conf.sample /etc/quagga/ripd.conf
```

Pour ensuite relancer Quagga :

xterm
```
$ sudo /etc/init.d/quagga restart
```

Dans le répertoire /var/run/quagga sont créés des fichiers temporaires indiquant les PIDs des différents démons.

Pour configurer le routeur, vous pouvez utiliser la commande `vtysh` (on peut également utiliser la commande «`telnet`» vers les ports 2601 ou 2602) :

xterm

```
pef@pef-desktop: $ sudo vtysh

Hello, this is Quagga (version 0.99.17).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

pef-desktop# configure  terminal
pef-desktop(config)# router rip
pef-desktop(config-router)# network 10.0.0.0/8
pef-desktop(config-router)# exit
pef-desktop(config)# exit
pef-desktop# show ip rip
```

La commande `network` permet d'indiquer sur quelle réseau le routeur va utiliser son protocole de routage et diffuser ses requêtes/réponses, ici `10.0.0.0/8`.

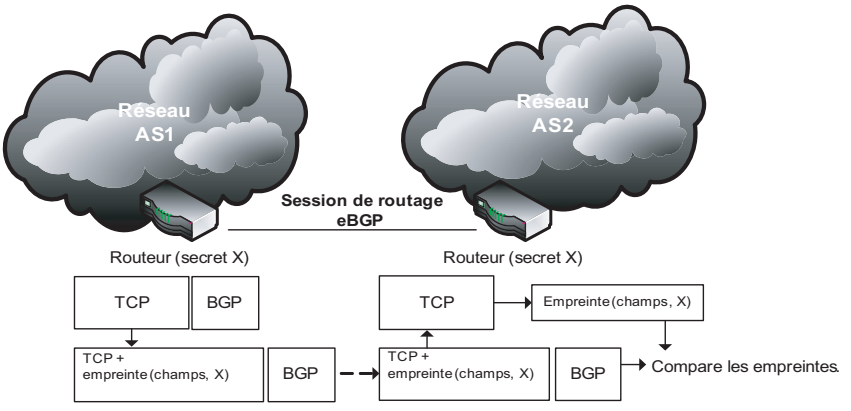
Sécurité et BGP : Sécuriser les sessions extérieures de BGP : eBGP

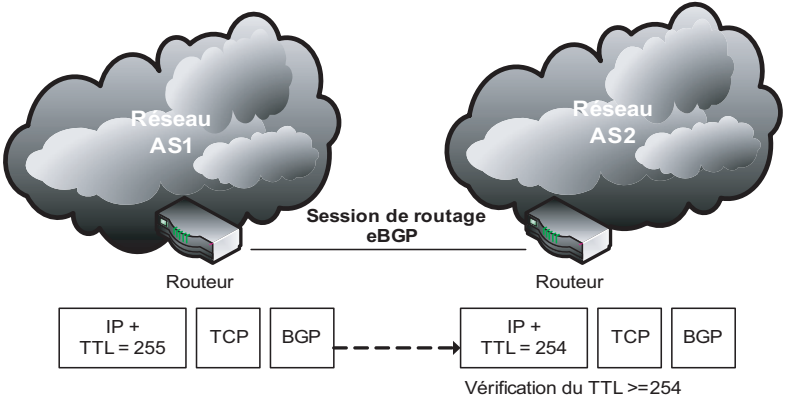
1ère méthode :

a. les deux routeurs partagent un secret X ;
b. dans la communication BGP en TCP entre les deux routeurs :
   - ◇ un **hash** est calculé sur les données de routage échangées combiné avec le secret partagé X (MD5, SHA-x *etc.*) ;
   - ◇ ce hash est transmis avec les données à l'autre routeur ;
   - ◇ le routeur qui reçoit les données peut **recalculer** l'empreinte et vérifier l'utilisation du secret partagé.

2ème méthode :

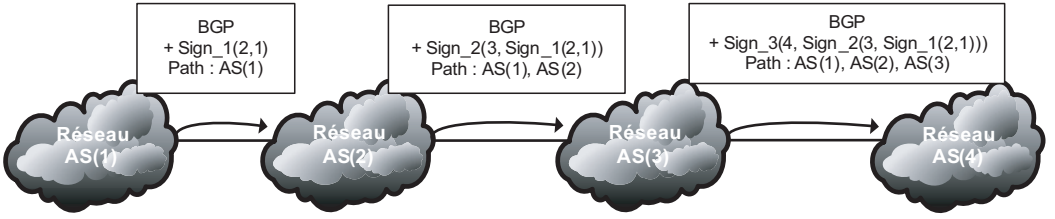
a. on mets le TTL des paquets à 255 (valeur max) ;
b. on vérifie à la réception qu'ils sont arrivés **directement** du routeur et non d'une autre origine : si le paquet venait d'un autre routeur alors il aurait un TTL inférieur à 254 (passage par un routeur supplémentaire).





Sécurité et BGP : contrôle de l'authentification des routes échangées

⋆ **Avec des certificats et des biclés asymétriques** :



◇ Chaque routeur **signe** le chemin auquel il s'ajoute lors de la traversée des différentes AS ;

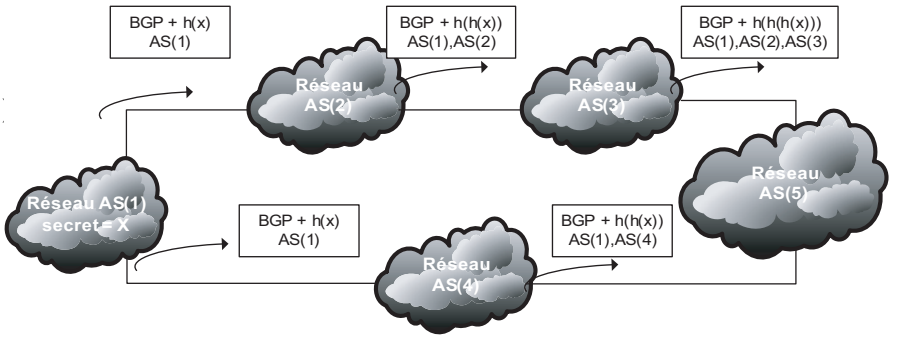
◇ Le routeur final peut vérifier que le chemin proposé passe bien par des routeurs authentifiés.

⋆ **Avec des chaîne de hachés** :

a. L'AS_1 et l'AS_5 partage un secret X ;

b. une empreinte $e_0 = h(X)$ est calculée par AS_1 ;

c. une nouvelle empreinte $e_n = h(e_{n-1}$ est calculée par chaque routeur de rang n sur l'ancienne empreinte reçue ;



d. AS_5 peut vérifier la longueur k des deux chemins reçus en calculant $h^k(X)$ et en comparant à l'empreinte reçue pour chacun de ces chemins (la chaîne de haché est de longueur k et X ne peut être retrouvé).

Le «*Routing*», ou le routage suivant la destination

Le routage est un processus piloté suivant la destination.

Ce processus se décompose de la façon suivante :

a. Chaque paquet qui entre dans un routeur est inspecté pour déterminer l'**adresse IP de destination** ;

b. À partir de cette adresse destination, le routeur consulte sa **table de routage** pour savoir où envoyer le paquet ;

c. Le seul élément important pour le routeur est l'**adresse de destination** ;

d. Si elle n'est pas présente dans la table de routage, le routeur utilise la **route par défaut** ;

e. S'il n'existe pas de « route par défaut», le paquet est **détruit** et un message **ICMP d'erreur est renvoyé** à l'émetteur du paquet.

L'outil pour manipuler les adresses et routes : la commande «ip»

xterm

```
pef@cerberus:~$ ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:a7:08:97 brd ff:ff:ff:ff:ff:ff
    inet 192.168.127.131/24 brd 192.168.127.255 scope global eth0
    inet6 fe80::20c:29ff:fea7:897/64 scope link
       valid_lft forever preferred_lft forever

pef@cerberus:~$ ip route list dev eth0
default via 192.168.127.2
192.168.127.0/24  proto kernel  scope link  src 192.168.127.131

pef@cerberus:~$ ip route get 1.1.1.1
1.1.1.1 via 192.168.127.2 dev eth0 src 192.168.127.131 uid 1000
    cache
```

permet d'interroger le routage par rapport à une destination

La métaphore du routage

Imaginons que vous quittiez votre maison pour prendre la route : vous avez le choix entre aller ***à gauche*** *ou* ***à droite*** *suivant la destination à laquelle vous voulez vous rendre. Peut-être, devez vous d'abord rejoindre l'autoroute avant de décider, ce qui s'assimile à utiliser la «route par défaut».*

Sur l'autoroute, vous avez le choix de prendre une ***voie normale*** *ou bien celle pour les* ***véhicules lents****. Suivant la nature de votre véhicule vous êtes amené à faire un choix supplémentaire par rapport à celui de votre destination :*

- ▷ *en camion, je prendrais la voie pour véhicule lent ;*
- ▷ *en voiture, la voie normale…*

Ce choix supplémentaire est fait suivant la nature de la source.

En **réseau**, on parle de règle de routage, «*Policy Routing*», où l'on prend en compte :

- ☐ l'adresse source,
- ☐ le protocole IP,
- ☐ le TOS,
- ☐ le protocole de transport utilisé,
- ☐ ou même le contenu du paquet…

La notion de «*Policy Based Routing*»

Une «*policy*» est un ensemble de règles de prescriptions ou d'interdictions, ainsi que d'actions qui servent à mettre en place un «but» défini comme «souhaitable».

Comme but « souhaitable», on peut citer :

- la **QoS**, c-à-d router suivant la nature du trafic en l'associant à des protocoles : TOS (Type of Service), ou Differentiated Service (DiffServ) ;
- la **répartition** et la **séparation** de trafic à des buts d'**équilibrage de charge** ou de **sécurité** ;
- la **décision** de router un datagramme suivant son **adresse d'origine** ;
- le **blocage** de trafic.

«Policy based Routing» ≠ «Routing Policy»

- le «*Routing policy*» s'applique sur les mécanismes réseau qui réalisent le routage :
  - consultation de la **table de routage**, «*RIB*», «*routing information base*» ;
  - détermination de l'interface de sortie en «*matchant*» la destination du trafic avec une destination dans la table ;
- le «*Policy based Routing*» :
  - est **administratif** ;
  - impose des **contraintes** sur la gestion du trafic : intégration de la sécurité, QoS, aspects financiers, *etc.* ;
  - est consulté **avant** de consulter la RIB ;
- les deux sont **imbriqués** : d'abord la PBR, puis le Routage, mais doivent être gérés **séparéments** ;
- le «*policy based routing*» est une **extension** du routage, avec **plus de contrôle** où l'on peut manipuler l'information de routage (comme le fait BGP par exemple en bloquant l'annonce de préfixes réseaux).

Les 3 éléments fondamentaux

- ▷ **Adresse** : définit la localisation d'un service ;
- ▷ **Route** : définit la localisation de l'adresse ;
- ▷ **Règle** : définit la localisation de la route.

L'adresse

Exemple : un serveur Web en IPv4 :

- pour se connecter dessus : on entre son **URL** dans un navigateur ;
- une **résolution de nom** est réalisé au travers d'un serveur DNS pour connaître l'adresse IPv4 associé ;
- on demande alors le service http accessible à cette adresse ;
- *Quel rapport entre cette adresse et l'adresse physique de la machine hébergeant le serveur Web ?* Aucun !
- l'adresse sert de « pointeur» pour le navigateur pour lui permettre de trouver le contenu voulu.
- Du point de vue de la «*Policy based Routing*», l'adresse d'origine et aussi importante que celle de destination, ainsi que l'intégralité du paquet : cela définiera la route à emprunter, empêchera l'utilisation d'une adresse IP non autorisée (IP spoofing), *etc*.

Notion de portée ou «scope» d'une adresse :

- ⋆ le préfixe CIDR de l'adresse ≠ du préfixe CIDR du réseau ;
- ⋆ si un réseau est défini par `192.168.1.0/24`, et que l'on utilise l'adresse `192.168.1.1/25` : pas de problème ;
- ⋆ l'adresse `192.168.1.1/16` peut également être utilisée : le réseau ne **s'occupe pas** de savoir quelle est la portée de l'adresse utilisée **du moment** que la machine obéit aux règles de routage du réseau et utilise la bonne adresse de diffusion associée au réseau ;
- ⋆ l'«adresse de diffusion», *broadcast*, peut être **différente** de la portée de l'adresse !
- ⋆ la portée ou «scope» sert à **associer** l'adresse à un **regroupement**, qui, à son tour, définit la **route** à utiliser.

Les 3 éléments fondamentaux

- ▷ **Adresse** : définit la localisation d'un service ;
- ▷ **Route** : définit la localisation de l'adresse ;
- ▷ **Règle** : définit la localisation de la route.

La route

La route définit la méthode de «*forwarding*» ou relayage, pour aller vers l'adresse de destination.
La «*Policy routing*» permet :

- ◦ de sélectionner une route différemment mais une fois la route obtenue, elle est utilisée de la même façon qu'avant ;
- ◦ de définir d'autres destinations, en plus de celle traditionnelle de l'adresse d'un routeur :
  - ◇ spécifier une option de «rejet», *reject* ;
    Cette option est similaire à un « lookup failure» et retourne un message d'erreur ICMP de type 3 et code 0 «network unreachable».
  - ◇ spécifier une option de d'«interdit», *denied* ;
    Il est possible de renvoyer un message ICMP d'erreur ou bien de «jeter», drop*, le paquet.*
  - ◇ réaliser du NAT en version «one-to-one».
    Cette version est plus rapide que celle proposée par NetFilter, elle est appelée «FastNat» mais elle n'utilise pas de «Connexion Tracking» et se réduit à du NAT «one-to-one».

La règle ou *rule*

La règle permet de mettre en œuvre une sorte d'« ACL», *Access Control List*, pour les routes.
Une règle permet de :

- ▷ définir les filtres pour l'appariement de paquet, *packet matching*,
- ▷ la route à utiliser lorsqu'il y a correspondance entre un paquet et un filtre de sélection.

La règle permet, entre autre, de sélectionner une route suivant l'origine du paquet.

Pour pouvoir **tirer parti au maximum** des possibilités des règles, il faut disposer de **plusieurs tables de routage**.

La «*Routing Policy Database*» ou «RPDB»

Les entrées de la table de routage :

- **indiquent** des chemins vers les autres réseaux ;
- sont, chacunes, constituées de :
  - un préfixe, c-à-d une adresse réseau et la taille de l'identifiant réseau (notation CIDR `xx.xx.xx.xx/yy`) ;
  - le TOS associé (information optionnelle) ;
  - une valeur de préférence ;
  - une interface de sortie ;
  - l'adresse du routeur de prochain saut, «next hop».

La correspondance, «*matching*», entre un paquet et une entrée de la table de routage :

- l'adresse de destination du paquet **correspond** au préfixe réseau indiqué dans l'entrée ;
- le TOS de la route est zéro ou égal à celui du paquet.

Lorsque **plusieurs routes correspondent aux paquets**, la sélection, «*lookup*», se fait de la manière suivante :

1. les routes de plus long préfixe correspondant sont sélectionnées, les autres sont ignorées ;
2. parmi les routes restantes :
   - les routes ne possédant pas le même TOS sont ignorées ;
   - si aucune(s) route(s) possédantt le même TOS n'ont été trouvés, et qu'il existe des routes avec un TOS$= 0$ alors ces routes sont choisies et les autres ignorées ;
   - sinon $\longrightarrow$ échec du «*lookup*»
3. S'il reste plusieurs routes après le filtrage précédent, on choisit la route avec la **valeur de préférence** la plus élevée (il ne devrait rester plus qu'une route par définition).

Les différents types de routes

- ⋆ `unicast` : destination directe désignée par un préfixe ;
- ⋆ `unreachable` : destinations inacessibles, les paquets détruits et un paquet ICMP «*host unreachable*» est renvoyé ;
- ⋆ `blackhole` : destinations inacessibles et les paquets sont détruits sans avertissement ;
- ⋆ `prohibit` : destinations inacessibles, les paquets détruits et un paquet ICMP «*communication administratively prohibited*» est renvoyé ;
- ⋆ `local` : destinations associés à l'hôte lui-même : les paquets seront en «loop back» et remis en local ;
- ⋆ `broadcast` : destinations correspondant à des adresses de diffusion ;
- ⋆ `throw` : le «lookup» échoue, signifiant qu'il n'existe pas de route. Sans règle supplémentaire un message ICMP «*network unreachable*» est renvoyé.
- ⋆ `nat` : route spéciale capable de réaliser du NAT avec l'attribut «*via*» ;
- ⋆ `anycast` : similaire à `local` avec une restriction : l'@IP source du paquet ne peut appartenir à ces adresses ;
- ⋆ `multicast` : pour du routage multicast.

Les différentes tables

Les tables sont identifiée :

▷ par un numéro compris entre 1 et 255 ;

▷ par un nom dans le fichier `/etc/iproute2/rt_tables` ;

L'utilisation de plusieurs tables permet de faire du **«*Policy Routing*»**, c-à-d suivant la **nature de la source**.

Par défaut, toute les routes normales sont insérées dans la table «`main`», «`ID 254`».

Il existe aussi une table invisible*, la table «`local`», «`ID 255`», qui contient les routes pour les adresses locales et de broadcast associées. Elle est maintenue automatiquement par le noyau.*

La «*Routing Policy Database*»

Le noyau Linux permet de définir **plusieurs tables de routage** afin d'**adapter** le routage à différents besoins et usages :

▷ il existe différentes tables de routage prédéfinies, comme les tables `local` & `main` :

xterm
```
rezo@ishtar:~$ ip route list table local
broadcast 127.0.0.0 dev lo  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  proto kernel  scope link  src 127.0.0.1
broadcast 192.168.127.0 dev eth0  proto kernel  scope link  src 192.168.127.202
local 192.168.127.202 dev eth0  proto kernel  scope host  src 192.168.127.202
broadcast 192.168.127.255 dev eth0  proto kernel  scope link  src 192.168.127.202
```

xterm
```
rezo@ishtar:~$ ip route list table main
default via 192.168.127.2 dev eth0  metric 100
192.168.127.0/24 dev eth0  proto kernel  scope link  src 192.168.127.202
```

xterm
```
rezo@ishtar:~$ ip route
default via 192.168.127.2 dev eth0  metric 100
192.168.127.0/24 dev eth0  proto kernel  scope link  src 192.168.127.202
```

La table modifiée par la commande `ip route` est, par défaut, celles appelées `main`.

L'ajout de route avec la commande «`ip route add`»

Les arguments de cette commande :

* `to PREFIX`,`to TYPE PREFIX` : permet de définir la destination. Sans Indication de `TYPE`, le type «`unicast`» est utilisé ;
* `tos TOS` ou `dsfield TOS` : permet d'indiqué le TOS ou le dsfield ;
* `metric NUMBER` ou `preference NUMBER` : la valeur de préférence exprimée sur 32bits ;
* `table TABLEID` : la table à laquelle ajouter la route. Sans indication, c'est la table «`main`» qui est choisie sauf dans le cas d'un type de route `local`, `broadcast` et `nat` où la route est ajoutée dans la table `local` par défaut ;
* `dev NAME` : l'interface de sortie ;
* `via ADDRESS` : l'adresse de prochain saut. Suivant le type de route : pour le type `unicast`, c'est l'adresse du routeur de prochain saut, pour le type `nat`, c'est l'adresse de traduction d'adresse ;
* `src ADDRESS` : l'adresse source de préférence lors de l'envoi vers la destination ;
* `realm REALMID` : pour définir des groupes de destination (beaucoup de destinations de préfixes différents) ;
* `mtu MTU` ou `mtu lock MTU` : positionne la MTU de façon modificable ou non (le protocole Path MTU Discovery permet de le modifier) ;
* `window NUMBER` : permet de choisir la taille de fenêtre maximale pour TCP indiquée en octets ;
* `advmss NUMBER` : permet de définir le MSS de TCP (sinon il est dérivé de la MTU) ;
* `nethop NEXTHOP` : permet de définir des chemins multiples, «*multipath*».
  Chaque «`NEXTHOP`» :
  - `via ADDRESS` : le routeur de prochain saut ;
  - `dev NAME` : l'interface de sortie ;
  - `weight NUMBER` : indique la qualité ou le débit de la liaison par rapport aux autres «`NEXTHOP`» ;
* `equalize` : permet de rendre aléatoire la distribution des paquets sur un «mulipath». Sans cette option, le «multi-path» n'est réalisé que pour un «flux» donné, et non pour chaque paquet indépendemment ;
* `nat` : permet de faire du DNAT.

Des exemples d'usage de la commande «`ip route add`»

- ajouter une route vers le réseau `10.0.0.0/24` par la passerelle `193.233.7.65` :

```
xterm
# ip route add 10.0.0.0/24 via 193.233.7.65
```

- ajouter un chemin multiple pour répartir la charge entre l'interface `eth0` et `eth1` :

```
xterm
# ip route add default netxhop dev eth0 via 193.233.7.65 nexthop dev eth1 via 210.45.67.18
```

- traduire l'adresse de source `192.168.80.144` par l'adresse `193.233.7.83`

```
xterm
# ip route add nat 192.168.80.144 via 193.233.7.83
```

Attention, il n'y a pas de suivi de trafic comme dans le cas de l'utilisation de NetFilter.

*Il **faudra** une autre opération pour faire l'opération de SNAT :*

```
xterm
# ip rule add from 193.233.7.83 nat 192.168.80.144
```

L'indication de la table dans laquelle est fait le «lookup» est facultatif : il sera réalisé par défaut dans la table «main».

L'utilisation de règles pour sélectionner les tables de routage

Le choix de la table de routage à utiliser se fait suivant des règles, «*rules*» :

```
xterm
rezo@ishtar:~$ ip rule list
0:   from all lookup local
32766:   from all lookup main
32767:   from all lookup default
```

*Chaque règle est préfixée par un numéro qui définit sa priorité, permettant de les **classer** .*

1. `Priorité 0` : sélectionner la table locale «ID 255» assurant le routage pour les adresses locales ou de multicast.

   Cette règle ne peut être effacée ou remplacée ;

2. `Priorité 32766` : sélectionner la table main «ID 254» qui est la table de routage normale, sans route utilisant du «policy».

   Cette règle peut être effacée ou redéfinie.

3. `Priorité 32767` : sélectionner la table default «ID 253». La table «*default*» est vide et réservée pour du «post-traitement» lorsque les règles précédentes n'ont pas été sélectionnées.

   Cette règle peut être effacée.

- `type TYPE` : le type de la règle
- `from PREFIX` : sélectionner le préfixe source ;
- `to PREFIX` : sélectionner le préfixe de destination ;
- `iif` : sélectionner l'interface d'entrée. Si l'interface d'entrée est la «loopback» alors la règle ne peut correspondre qu'à des paquets provenant de l'hôte local. Ainsi, il est possible de définir des tables de routages différentes pour les paquets «relayer», «*forwarded*», et ceux «locaux» et de séparer ces trafics ;
- `tos TOS` ou «dsfield TOS» : sélectionner le TOS ;
- `fwmark MARK` : sélectionner le marquage de paquet. Nécessaire pour faire le lien avec NetFilter ;
- `priority PREFERENCE` : la priorité de la règle. Cette valeur est unique pour chaque règle ;
- `table TABLEID` : la table de routage à consulter, «*lookup*» si la règle à été sélectionnée ;
- `realms FROM` : le «royaume» à sélectionner si la règle a été sélectionnée et si le «lookup» à réussi ;
- `realms TO` : le «royaume» à sélectionner si la route choisie n'en a pas sélectionné ;
- `nat ADDRESS` : réalise la traduction d'adresse SNAT.

Attention

Après toute modification de la RPDB avec ces commandes, il est **conseillé** de lancer la commande :

xterm
```
# ip route flush cache
```

Exemples

xterm
```
# ip rule add from 192.203.80.0/24 table ma_table prio 220
```

xterm
```
# ip rule add from 193.233.7.83 nat 192.203.80.144 table 1 prio 320
```

Il est possible d'ajouter de nouvelles tables en éditant le contenu du fichier `/etc/iproute2/rt_tables` :

```
$ more /etc/iproute2/rt_tables
#
# reserved values
#
255   local
254   main
253   default
0   unspec
#
# local
#
#1   inr.ruhep
100 ma_table_a_moi
```

On peut alors ajouter une règle permettant d'utiliser la nouvelle table en définissant le **sélecteur** :

xterm
```
# ip rule add ❶from 10.0.0.10 ❷lookup ma_table_a_moi
```

❶⇒le sélecteur : ici, sélectionner les datagrammes en provenance de l'adresse `10.0.0.10` ;

❷⇒définit la table à consulter :la nouvelle table `ma_table_a_moi`.

xterm
```
$ ip rule
0:   from all lookup local
32765:   from 10.0.0.10 lookup ma_table_a_moi
32766:   from all lookup main
32767:   from all lookup default
```

Une fois la table ajoutée, on peut la renseigner :

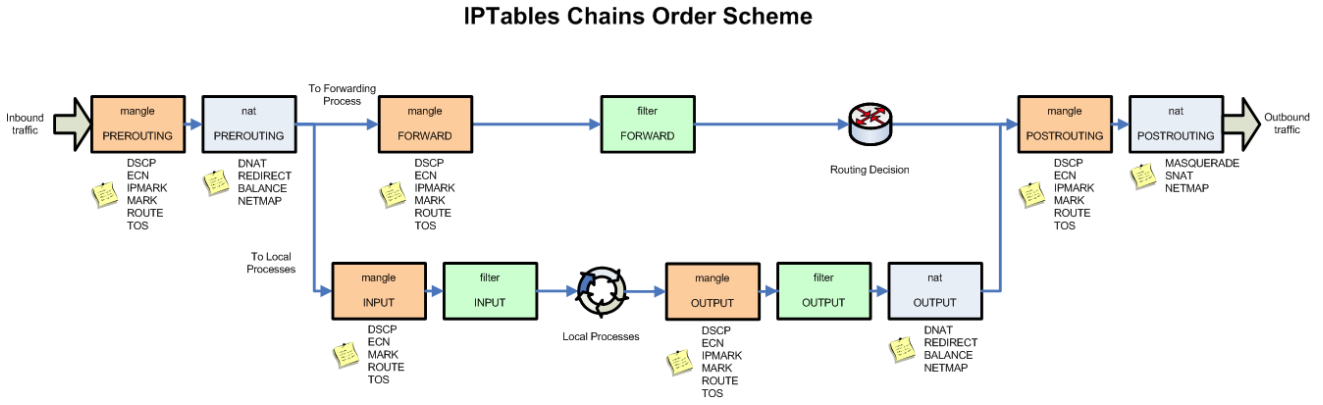
xterm
```
# ip route add default via 192.168.0.3 dev eth1 table ma_table_a_moi
# ip route flush cache
```

il ne faut pas oublier de vider le cache avec la commande `ip route flush cache`.

Explication de la configuration :

- pour des datagrammes provenant de l'@IP `10.0.0.10`, il faut sélectionner la table `ma_table_a_moi` ;
- la table `ma_table_a_moi` rédéfinit la route par défaut et l'interface d'accès ;
- le datagramme est routé suivant une **table différente** suivant sa provenance ⟹ On fait du routage suivant la source, ce qui s'appelle du «*routing policy*».

Le parcours du datagramme et le routage



Intégration avec le «Routing Policy»

- routage suivant le TOS/dsfield : manipulation de ce TOS grâce à NetFilter ;
- utilisation de la table mangle et du marquage de paquet : le choix de la table grâce à «`fwmark`» ;

Attention

▷ Pour l'utilisation de NetFilter dans le cadre du «*Routing Policy*», on utilisera la chaîne «`PREROUTING`».

▷ Rapport entre NetFilter et le «*Policy Routing*» :

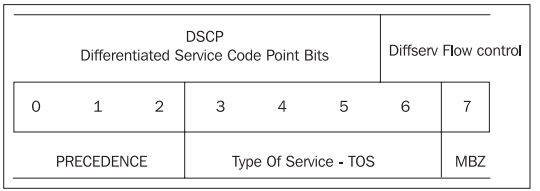
- ◇ NetFilter est **selectif** dans la notion d'interface de sortie : il peut sélectionner un paquet suivant son interface de sortie, mais il **ne peut pas** forcer un paquet vers une interface de sortie ;
- ◇ Seul le **routage** décide de l'interface de sortie d'un paquet ;
- ◇ La chaîne PREROUTING permet à NetFilter d'**influencer** le routage **mais pas** «POSTROUTING».

Netfilter : modifier les paquets, la table `mangle`

L'utilisation de cette table permet de :

- **modifier** le champs TOS, «*Type Of Service*»/DSCP, «*Differentiated Services field*» : modifier la priorité du datagramme IP en fonction de son origine, de son contenu *etc.*
  Cette modification permet de faire de la QoS entre routeurs capables de gérer ces priorités (RFC 2474, 2475).

Precedence Level	Description
7	Stays the same (link layer and routing protocol keep alive)
6	Stays the same (used for IP routing protocols)
5	Express Forwarding (EF)
4	Class 4
3	Class 3
2	Class 2
1	Class 1
0	Best effort



xterm
```
iptables -t mangle -A FORWARD -p tcp --dport 80 -j DSCP --set-dscp 1
iptables -t mangle -A FORWARD -p tcp --dport 80 -j DSCP --set-dscp-class EF
```

- **marquer** le paquet dans sa gestion au sein du noyeau pour faire du «*Routing Policy*» :

xterm
```
iptables -t mangle -A PREROUTING -i eth0 -p tcp --dport 80 -j MARK --set-mark 1
```

Attention

La marque n'existe que dans le noyau : elle ne sort pas de celui-ci et ne peut être communiquée par réseau.

Netfilter : sélection d'un paquet possédant une marque

Le module «*mark*» :

* `--mark` : permet de sélectionner le paquet suivant une marque

Exemple d'utilisation

Utiliser une marque pour «*mémoriser*» l'interface d'entrée du paquet afin de réaliser du SNAT lors de sa sortie :

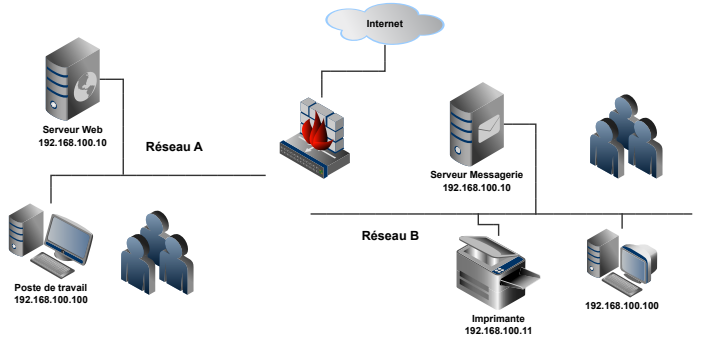
▷ Positionner une marque sur les paquets en entrée sur l'interface `eth1` :

xterm
```
sudo iptables -t mangle -A PREROUTING -i eth1 -j MARK --set-mark 1
```

▷ Sélectionner la marque dans une règle de SNAT :

xterm
```
sudo iptables -t nat POSTROUTING -m mark --mark 1 -o eth0 -j MASQUERADE
```

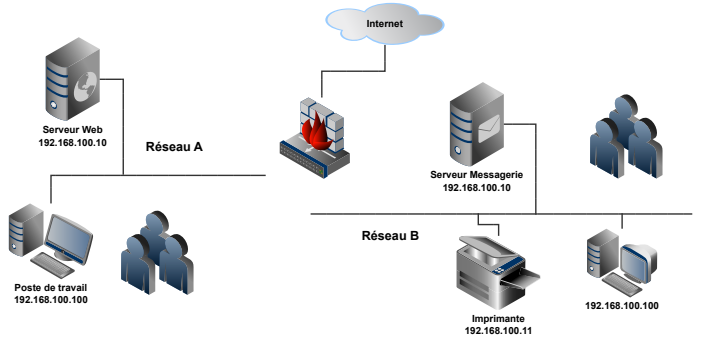
Un exemple : fusionner deux réseaux identiques



Il faut fusionner deux réseaux identiques en `192.168.100.0/24` et permettre l'accès d'un client d'un réseau vers le serveur de l'autre réseau…

Comment faire ?

Un exemple : fusionner deux réseaux identiques

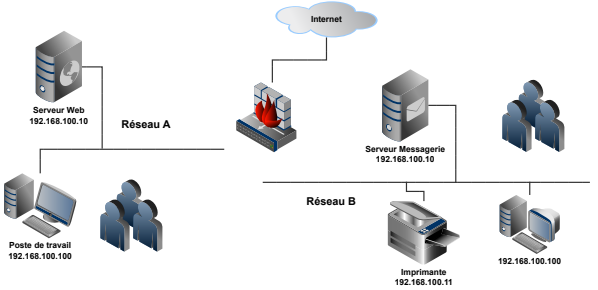


Il faut fusionner deux réseaux identiques en `192.168.100.0/24` et permettre l'accès d'un client du réseau B vers un serveur du réseau A…

Comment faire ?

- ▷ Donner aux interfaces du routeur deux adresses différentes, une pour chaque réseau ;
- ▷ Utiliser le firewall/routeur comme destination du service offert dans A pour le réseau B :
  le port 8090 du routeur sera redirigé vers le port 8080 du serveur
- ▷ Créer une «vue» différente du réseau `192.168.100.0/24` pour le routeur suivant l'interface d'entrée du datagramme :
  ⟹ «*Policy Routing*» !
- ▷ permettre aux paquets allant de A vers B d'atteindre B :
  ⟹ Utiliser le marquage avec NetFilter et le «*policy routing*»
- ▷ permettre aux paquets allant de B vers A d'atteindre A :
  ⟹ Utiliser le marquage avec NetFilter et le «*policy routing*»

Un exemple : fusionner deux réseaux identiques



Il faut fusionner deux réseaux identiques en `192.168.100.0/24` et permettre l'accès d'un client d'un réseau vers le serveur de l'autre réseau…

- le routeur disposera d'une adresse différente dans le réseau A, «`eth0`», et B, «`eth1`» :

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP ql
en 1000
    link/ether 00:0c:29:3e:eb:1c brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.253/24 brd 192.168.100.255 scope global eth0
    inet6 fe80::20c:29ff:fe3e:eb1c/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNO
WN qlen 1000
    link/ether 00:50:56:3b:95:6d brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.254/24 brd 192.168.100.255 scope global eth1
    inet6 fe80::250:56ff:fe3b:956d/64 scope link
       valid_lft forever preferred_lft forever
```

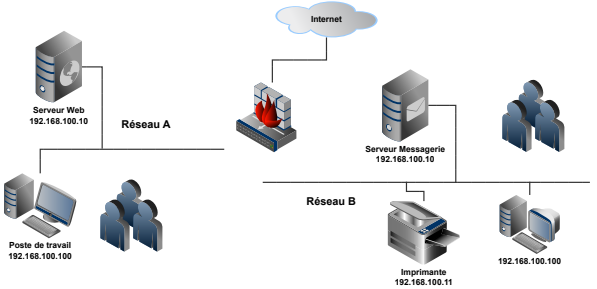
- on va créer deux tables de routages :

```
root@routeur:~# more /etc/iproute2/rt_tables
#
# reserved values
#
255     local
254     main
253     default
0       unspec
#
# local
#
#1      inr.ruhep
100     vers_a
200     vers_b
```

Chaque table de routage ne connaitra qu'une seule interface :

```
root@routeur:~# ip route show table vers_a
192.168.100.0/24 dev eth0  scope link
root@routeur:~# ip route show table vers_b
192.168.100.0/24 dev eth1  scope link
```

Un exemple : fusionner deux réseaux identiques



Il faut fusionner deux réseaux identiques en `192.168.100.0/24` et permettre l'accès d'un client d'un réseau vers le serveur de l'autre réseau…

- on ajoute les règles de NetFilter suivantes pour le fonctionnement de TCP du client vers le serveur donné :
  - pour aller de A ⟶ B :

```
xterm
iptables -t mangle -A PREROUTING -i eth0 -s 192.168.100.10 -p tcp --sport 8080
   -j MARK --set-mark 1
```

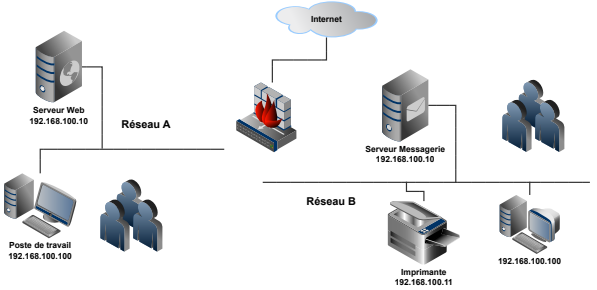
  - pour aller de B ⟶ A :

```
xterm
iptables -t mangle -A PREROUTING -i eth1 -d 192.168.100.254 -p tcp --dport 8090
   -j MARK --set-mark 10
```

- on définit les règles suivantes :

```
root@routeur:~# ip rule
0:      from all lookup local
20:     from 192.168.100.253 lookup vers_a
25:     from 192.168.100.254 lookup vers_b
30:     from all fwmark 0x1 lookup vers_b
49:     from all fwmark 0xa lookup vers_a
32766:  from all lookup main
32767:  from all lookup default
```

Un exemple : fusionner deux réseaux identiques



Il faut fusionner deux réseaux identiques en `192.168.100.0/24` et permettre l'accès d'un client d'un réseau vers le serveur de l'autre réseau…

▷ on vérifie l'activation des règles de NetFilter de la table «mangle» :

```
root@routeur:~# iptables -t mangle -nvL
Chain PREROUTING (policy ACCEPT 221K packets, 19M bytes)
 pkts bytes target     prot opt in     out     source               destination

   89  5219 MARK       tcp  --  eth1   *       0.0.0.0/0            192.168.100.
254      tcp dpt:8090 MARK set 0xa
   18   978 MARK       tcp  --  eth0   *       192.168.100.10       0.0.0.0/0
            tcp spt:8080 MARK set 0x1
```

▷ on ajoute des règle de SNAT et DNAT :

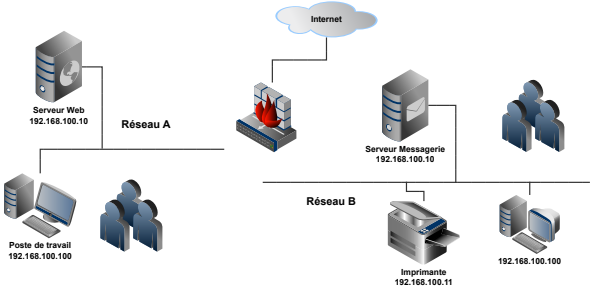
xterm
```
iptables -t nat -A PREROUTING -m mark --mark 10 -p tcp -j DNAT --to 192.168.100.10:8080
```

Cette règle permet au paquet marqué d'atteindre le serveur.

xterm
```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 192.168.100.253
```

Cette seconde règle permet au serveur de répondre vers le routeur.

Un exemple : fusionner deux réseaux identiques



Il faut fusionner deux réseaux identiques en `192.168.100.0/24` et permettre l'accès d'un client d'un réseau vers le serveur de l'autre réseau…

▷ on vérifie l'activation des règles de NetFilter pour la table «nat» :

```
root@routeur:~# iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 205K packets, 15M bytes)
 pkts bytes target     prot opt in     out     source               destination

   32  1913 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
        mark match 0xa to:192.168.100.10:8080

Chain INPUT (policy ACCEPT 178K packets, 10M bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 467 packets, 78138 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain POSTROUTING (policy ACCEPT 231 packets, 38736 bytes)
 pkts bytes target     prot opt in     out     source               destination

  247 40069 SNAT       all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
        to:192.168.100.253
```

Il ne faudra pas oublier de changer la configuration du noyau :

```
xterm
sysctl -w net.ipv4.conf.all.rp_filter=0
```

On désactive le «Reverse Path Filtering», qui s'appuie sur la notion de «Reverse Path Forwarding» RFC 3704, RFC 1827.

Les VPNs, «*Virtual Private Network*»

L'utilisation d'un VPN permet, 4 usages :

- la **connexion entre un Site et un autre Site** :
  - entre deux composantes d'une même entreprise séparées géographiquement ;
  - utilisant Internet pour faire circuler les données ;
  - en assurant les propriétés de sécurités de confidentialité et d'authentification ;

  Sans VPN, on utilise des réseaux WAN privés pour interconnecter ces différentes composantes :
  - *circuits «point-à-point» dédiés comme ATM (réseau opérateur téléphonique), ou MPLS (réseau de type informatique), ou une combinaison SDSL, «Symmetrical Digital Subscriber Line» + MPLS;* `http://www.mympls.fr/`
  - *assurant une bonne latence, une meilleure disponibilité, mais coûteux*
- un **accès distant**, «remote access» :
  - permettre la connexion d'un utilisateur depuis n'importe où sur Internet ;
  - indispensable pour les «*roadwarriors*», c-à-d les employés d'une société qui doivent voyager souvent pour leur travail ;
  - permet la connexion de «télé-travailleurs», de sous-traitants qui doivent disposer d'un accès temporaire au réseau de l'entreprise.
- une **protection pour les réseaux sans-fil**, «wireless» ;
- un **relais sécurisé** : lorsque l'on ne fait **pas confiance** au réseau local où l'on est connecté et que l'on se sert du VPN pour aller «directement» sur Internet (protection contre l'ARP Spoofing).

Qu'est-ce qu'un VPN ?

«une connexion privée entre deux éléments terminaux», c-à-d une liaison «point-à-point» entre ces deux terminaux dont le contenu n'est accessible qu'à ces deux terminaux.

Plus concrètement ?

- ⋆ il faut pouvoir **encapsuler** des communications quelconques dans cette liaison point-à-point, c-à-d faire passer les datagrammes IP de ces communications dans la liaison ;
- ⋆ cette liaison emprunte InterNet et doit pouvoir être routé : elle utilise des datagrammes IP.

*Ainsi, la liaison est considérée comme un **tunnel** qui peut être emprunter de manière transparente par ces communications.*

Quelles différences avec une liaison physique «point-à-point» ?

La liaison physique point-à-point est *physique* (câble de liaison entre deux routeurs) alors que la liaison VPN est *virtuelle* : elle est simulée au travers d'une communication **isolée des autres éléments** du réseau :

- ⋆ seuls deux bouts communiquent au travers de la liaison (après authentification mutuelle) ;
- ⋆ les données échangées peuvent être rendues «*inaccessibles*» à l'observation d'un tiers :
  - ◇ par l'utilisation de la **cryptographie** : confidentialité assurée par chiffrement ;
  - ◇ par **routage** : passer dans un réseau d'interconnexion contrôlé avec MPLS.

Comment mettre en place un VPN ?

- ▷ Il faut transporter des datagrammes IP :
  - ◦ avec un protocole de niveau 2, c-à-d en faisant circuler des trames avec *éventuellement* des étiquettes (MPLS et VLANs) ;
  - ◦ avec un protocole de niveau 3, c-à-d en faisant circuler **directement** des datagrammes (encapsulation) ;
- ▷ Il faut isoler ces échanges du reste du trafic :
  - ◇ en le rendant **confidentiel** à l'aide de chiffrement (IPSec) ;
  - ◇ en **contrôlant son acheminement** dans le réseau d'interconnexion (MPLS).

Les VPNs de niveau 2

Leur travail consiste à :

- ▷ « encapsuler» des datagrammes IP : dans ce cas, le protocole est considéré comme de niveau 2 car il se substitue à la couche 2 du modèle OSI ;
- ▷ transporter des trames : dans cet autre cas, il véhicule du «niveau 2».

Un VPN considéré comme de «niveau 2»

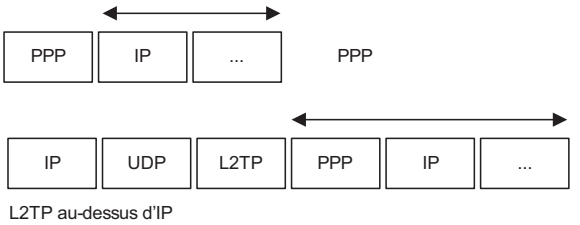
Historiquement, on commence par PPP, «*Point-to-Point Protocol*», RFC 1661, 1547 :

- ◦ considéré comme un protocole de niveau 2 : il encapsule les datagrammes dans une **liaison série** ;
- ◦ crée pour permettre des communications sur des lignes téléphoniques par **modem** (avant l'ADSL avec des débits de 56Kbits/seconde) ;
- ◦ permet une **authentification mutuelle** avec :
  - ◇ PAP, «*Password Authentication Protocol*» (transmission de mot de passe en clair) ;
  - ◇ CHAP, «*Challenge-Handshake Authentication Protocol*» (secret partagé échangé haché) ;
  - ◇ EAP, «*Extensible Authentication Protocol*» (protocole utilisant différentes méthodes avancés jusqu'à l'utilisation de certificat) ;
- ◦ permet de faire de la **compression** (réduction des transmissions), du **contrôle d'erreur**, du **chiffrement** avec ECP, «*Encryption Control Protocol*» ;
- ◦ peut être utilisé au travers d'**autres protocoles** : PPPoE, «*PPP over Ethernet*», PPPoA, «*PPP over ATM*» pour ses capacités à établir des sessions dans le cadre de l'accès ADSL.
- ◦ peut être **réutilisé** : en fractionnant du PPP sur de l'UDP avec L2TP, RFC 2661 ;
  On reviendra sur le protocole L2TP qui a évolué au-delà de PPP dans sa version 3 (RFC 3931).

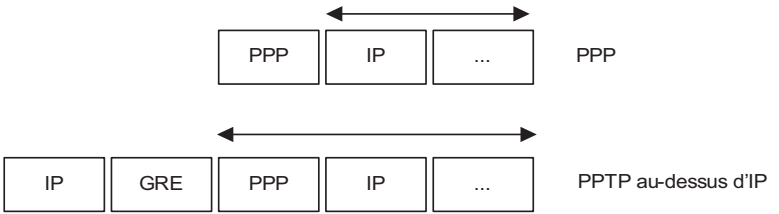
Quelques protocoles de niveau 2

- PPTP, *Point to Point Tunneling Protocol*, RFC 2637 : capable mais limité à un seul tunnel à la fois entre deux interlocuteurs, développé par Microsoft. Il peut néanmoins encapsulé d'autres protocoles qu'IP, comme IPX ;
- L2F, *Layer 2 Forwarding*, RFC 2341 : développé en même temps que PPTP par d'autres sociétés dont CISCO et permet de gérer, entre autre, plusieurs tunnels simultanés ;
- L2TP, *Layer 2 Tunneling Protocol*, RFC 2661 : standard qui combine les avantages de PPTP et de L2F, en ajoutant des possibilités améliorées de sécurité comme ceux d'IPSec ;
- L2Sec, *Layer 2 Security Protocol*, RFC 2716 : développé pour corriger des problèmes de sécurité d'IPSec lorsque celui-ci était encore en phase de développement, mais plus coûteux en traitement, mieux sécurisés et basé sur SSL/TLS.
- *etc.*

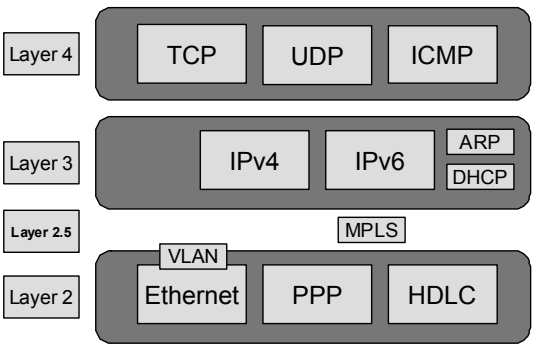
- le protocole L2TP :



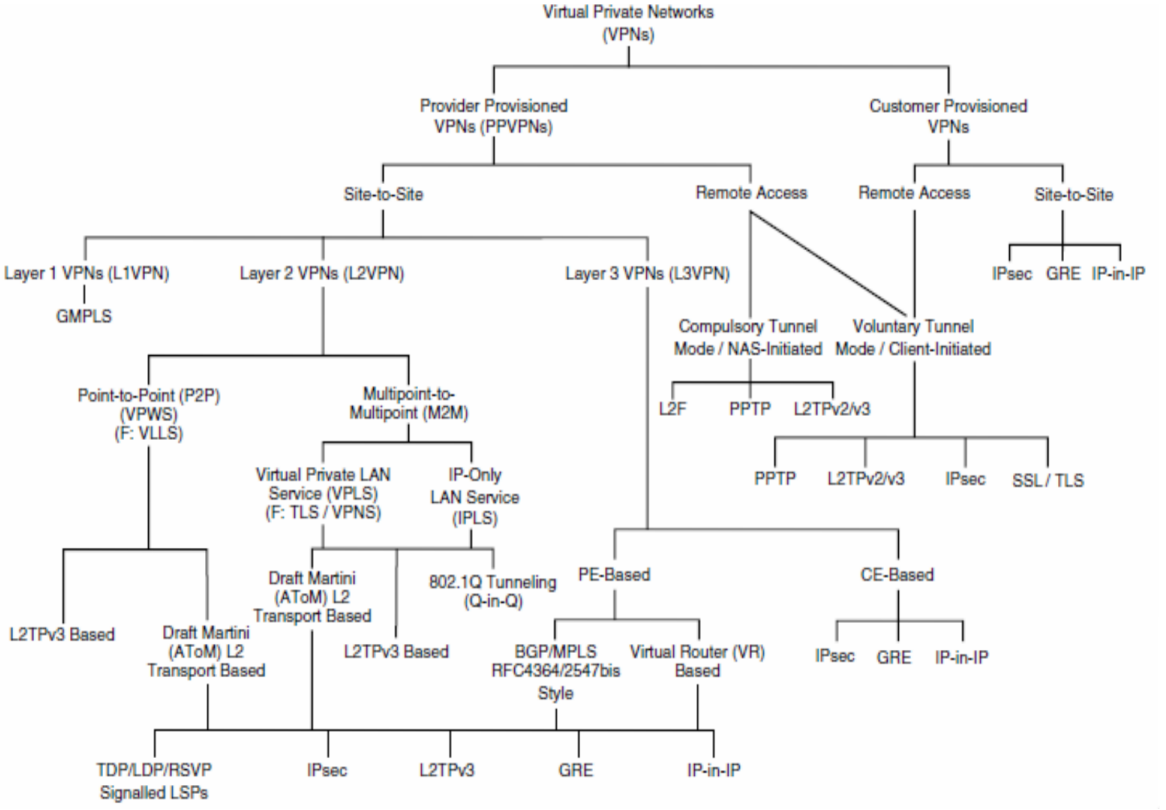
- le protocole PPTP :



- les protocoles PPP, VLANs, MPLS dans le modèle OSI :



Tunnels de niveau 2 : une comparaison



Tiré d'une présentation `http://prezi.com/orwqgsm9mdgg/untitled-prezi/`

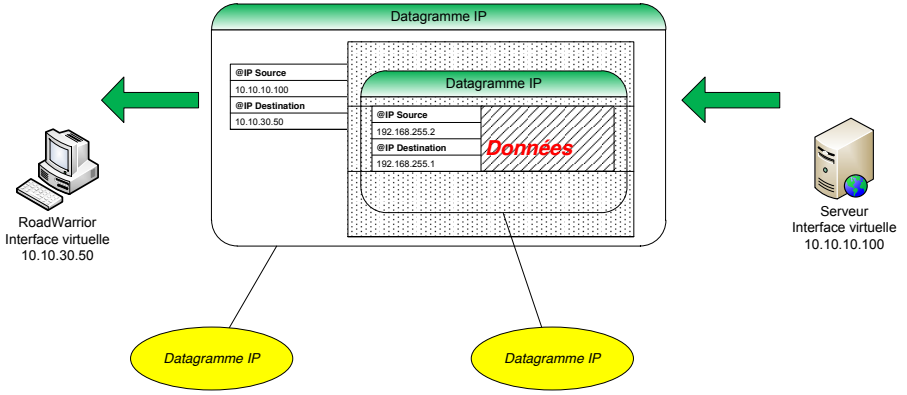
Les VPNs de niveau 3

- avec une **encapsulation IP dans IP** comme GRE, «*Generic Routing Encapsulation*» , RFC 2784 :
  - intégré dans la couche de niveau 3 : numéro de protocole 47 ;
  - permet de transporter des datagrammes IP :
    - pour tous les protocoles encapsulables (ICMP, UDP, TCP *etc.*) ;
    - pour des @IP source et destination quelconques (applications à une liaison entre routeurs).
- avec un **protocole d'isolation comme IPSec**, «*Encryption Control Protocol*», RFC 1825, 1829 :
  - disponible dans IPv4, intégré dans IPv6 ;
  - négocie les éléments cryptographiques entre les deux extrémités (algorithmes, clés, construction de clés de session *etc.*) ;
  - travaille au niveau du datagramme IP, c-à-d en mode «connectionless» :
    - permet d'authentifier les deux extrémités de la liaison dans l'en-tête du datagramme IP ;
    - permet de chiffrer le contenu du datagramme IP ;
    - fonctionne en **mode tunnel** (tout le contenu du paquet est protégé y compris les adresses nécessaires au routage) ou **transport** (les adresses nécessaires au routage sont visibles).
- en combinant du PPP et du GRE et en ajoutant du chiffrement et de la compression : le protocole **PPTP**, «*Point-to-Point Tunneling Protocol*», RFC 2637 (une connexion TCP vers le port 1723 permet d'établir le tunnel GRE) ;
- en fractionnant du PPP sur de l'UDP avec **L2TP**, RFC 2661, 3931 ;
- en combinant du GRE et de l'IPSec en mode transport ;
- en combinant du L2TP et de l'IPSec (avec L2TPv3 on aurait du niveau 2) ;
- en combinant **MPLS** et **BGP** : RFC 2547 ;
- en utilisant des dérivés **d'openSSL** :
  - en combinant du SSH (liaison TCP sécurisée) et du PPP ;
  - avec SSH et son mode VPN en niveau 3 (interface TUN) ou 2 (interface TAP) ;
  - avec openVPN en mode UDP ou TCP.

Le protocole GRE

Avec ce protocole :

- on établit le tunnel entre deux routeurs : c'est le **routage** qui décidera de l'encapsulation du trafic ;
- le tunnel est «*stateless*» : il n'y a pas de configuration associée au tunnel qui doit être mémorisée sur chaque extrémité, chaque datagramme IP empruntant le tunnel est encapsulé dans un nouveau datagramme IP et envoyé sans contrôle d'erreur ;
- chaque datagramme empruntant le tunnel (suivant son routage) est traité **séparément** :



- le trafic entre la machine RoadWarrior et la machine Serveur empruntera ce tunnel grâce à des **règles de routage spécifiques** aux extrémités.

Établissement d'un tunnel GRE sous Linux

Soient les routeurs :

- «Routeur1» : une interface externe en `192.168.100.254/24`, un réseau interne en `10.10.10.0/24` ;
- «Routeur2» : une interface externe en `172.16.1.253/24`, un réseau interne en `10.10.20.0/24` ;

Pour établir le tunnel GRE :

⋆ Sur routeur1 :

xterm
```
root@Routeur1:~# ip tunnel add mon_tunnel mode gre local 192.168.100.254 remote 172.16.1.253
root@Routeur1:~# ip link set mon_tunnel up
root@Routeur1:~# ip addr add dev mon_tunnel 10.0.0.1/24
root@Routeur1:~# ip addr
32: mon_tunnel: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue state UNKNOWN
    link/gre 192.168.100.254 peer 172.16.1.253
    inet 10.0.0.1/24 scope global mon_tunnel
root@Routeur2:~# ip route add 10.10.20.0/24 via 10.0.0.2
```

⋆ Sur routeur 2 :

xterm
```
root@Routeur2:~# ip tunnel add mon_tunnel mode gre local 172.16.1.253 remote 192.168.100.254
root@Routeur2:~# ip link set mon_tunnel up
root@Routeur2:~# ip link
31: mon_tunnel: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue state UNKNOWN
    link/gre 172.16.1.253 peer 192.168.100.254
root@Routeur2:~# ip addr add dev mon_tunnel 10.0.0.2/24
root@Routeur2:~# ip route add 10.10.10.0/24 via 10.0.0.1
```

On remarque que les deux extrémités du réseau appartiennent à un même réseau `10.0.0.0/24` indépendant de tous les autres (sinon des problèmes de routage peuvent survenir).

- ▷ IPSec permet la création de VPN, «*Virtual Private Network*», en utilisant des solutions cryptographiques.
- ▷ IPSec est intégré dans IPv6 et peut être utilisé dans IPv4.
- ▷ IPSec propose :
  - ◇ **deux protocoles** :
    - ⋆ AH, «*Authentication Header*» pour l'**authentification** ;
    - ⋆ ESP, «*Encapsulating Security Payload*», pour le **chiffrement** et l'authentification pour le paquet encapsulé en mode tunnel.

    On peut utiliser séparément l'un ou l'autre et, plus souvent, les deux ensembles.
    Le protocole AH assure l'intégrité et l'authentification de l'origine pour l'ensemble des champs de l'entête du datagramme IP, à l'exception de ceux qui peuvent changer lors du transfert du datagramme, c-à-d les champs «TTL» et «checksum».
  - ◇ **choix entre différents algorithmes cryptographiques** : «MD5», «SHA-1», «DES», «3DES», «AES», *etc.* : la mise en œuvre d'une connexion IPSec impose de faire des choix, mais chaque connexion ne fait appel qu'à deux, voire trois, algorithmes à la fois.
    - ⋆ L'authentification calcule un ICV, «*Integrity Check Value*», sur le contenu du paquet, ce qui est réalisé au travers d'une fonction de hachage comme MD5 ou SHA-1.
      Il incorpore une clé secrète connue des deux interlocuteurs, ce qui permet au destinataire de calculer l'ICV de la même façon.
      Ainsi, si le destinataire reçoit la même valeur, alors l'émetteur s'est authentifié avec succès (cela repose sur le fait qu'une fonction de hachage ne peut être inversée).
      AH fournit toujours de l'authentification alors qu'ESP peut la fournir en option.
    - ⋆ Le chiffrement utilise un clé secrète pour chiffrer les données avant leur transmission et cela permet de «cacher» le contenu du paquet et de le protéger d'éventuelles écoutes.
      Il est possible de choisir parmi différents algorithmes de chiffrement et en particulier, entre DES, 3DES, Blowfish et AES.

- IPSec propose :
  - deux modes de fonctionnement :
    - le mode «**Transport**» :
      - permet d'établir une liaison sécurisée directement entre deux matériels ;
      - encapsule le chargement du datagramme IP : les @IP source et destination reste celles de ces matériels ;
    - le mode «**Tunnel**»
      - permet d'établir une liaison sécurisée entre deux routeurs ;
      - permet d'encapsuler la totalité du datagramme IP passant par ces routeurs ce qui permet d'offrir un «*secure hop*», c-à-d le passage sécurisé entre deux routeurs (les datagrammes IP ne contiennent que les @IP source et destination des routeurs, mais pas celles des machines empruntant ce tunnel ;
      - permet d'établir des VPNs entre deux sites au travers d'Internet.

Dans le cas du mode Tunnel, on utilise rarement le protocole AH, dans la mesure où il interdit la modification de l'entête du datagramme IP, ce qui rend impossible l'utilisation de NAT ce qui peut être bloquant. On préférera alors l'utilisation d'ESP avec une forme simplifiée d'authentification : elle utilise les mêmes algorithmes que ceux utilisés par AH, mais cette authentification ne porte que sur l'entête et les données du contenu ESP, et pas sur l'entête du datagramme IP qui le contient.

L'utilisation du mode Tunnel est transparente, puisqu'elle s'applique uniquement entre deux routeurs, et que deux interlocuteurs utilisant ces routeurs n'ont rien à faire pour bénéficier de cette protection.

L'utilisation du mode Tunnel permet de «masquer» d'un observateur extérieur quels sont les deux interlocuteurs qui communiquent dans chacun des réseaux reliés par le VPN : on parle alors de «privacy», ou « respect de la vie privée».

- IPSec utilise un protocole de négociation des éléments de sécurité :
  - IKE, «*Internet Key Exchange*» *vs* «Clés fournies manuellement» : les deux extrémités de la communication doivent connaître les valeurs secrètes utilisées pour la fonction de hachage et le chiffrement, ce qui pose le problème de les échanger.
    La «fourniture manuelle» des clés requiert d'entrer manuellement les clés sur les deux extrémités probablement sans se servir du réseau pour le faire.
    IKE est un moyen sophistiqué pour le faire de manière «*online*».
    Sous Linux, le service «racoon» réalise le protocole IKE.
  - Mode «principal» ou «agressif» : le choix entre ces deux modes représentent un compromis entre efficacité et sécurité pour le protocole IKE d'échange de clés.
    Le mode principal requiert l'échange de 6 paquets dans un sens et dans l'autre, alors que le mode agressif en requiert la moitié, tout en transmettant des informations en clair ce qui diminue la sécurité.
- Intégration d'IPSec dans IPV4 :
  Le champ «protocole» du datagramme IP indique la nature du contenu :
  - 1 : ICMP ;
  - 6 : TCP ;
  - 17 : UDP ;
  - 47 : GRE ;
  - 50 : IPSec : ESP ;
  - 51 : IPSec : AH ;
- **La gestion des éléments cryptographiques :**
  Il est nécessaire de gérer des secrets sur les deux extrémités de la connexion sécurisée (les secrets permettant l'authentification et le chiffrement).
  Lorsqu'un paquet IPSec, AH ou ESP, arrive sur une interface réseau, comment cette interface peut savoir quel ensemble de paramètres (clé, algorithme et «politique de sécurité») utiliser ?
  Chacun de ces ensembles est spécifié au travers d'une SA, «Security Association», c-à-d une collection de paramètres spécifiques à une connexion, et chaque interlocuteur peut en posséder de nombreuses.

Afin de traiter un paquet IP à son arrivée, il faut :

- ◇ l'adresse IP de l'interlocuteur qui a envoyé le paquet ;
- ◇ la nature du protocole : ESP ou AH ;
- ◇ un SPI, «Security Parameters Index».

Une SA concerne «un seul sens» de communication, c-à-d qu'une communication bidirectionnelle en utilise deux.
Chaque protocole requiert sa propre SA pour chaque direction, ce qui fait que 4 SAs sont nécessaires pour un VPN utilisant AH+ESP.

Chaque interlocuteur dispose d'une SADB, une base de données des SAs qu'il possède.

Dans cette SADB, il y a :

- ◇ AH : l'algo. utilisé ;
- ◇ AH : le secret d'authentification ;
- ◇ ESP : l'algo de chiffrement ;
- ◇ ESP : la clé secrète de chiffrement ;
- ◇ ESP : la sélection d'une authentification ou pas ;
- ◇ des restrictions concernant le routage ;
- ◇ des politiques de sélection du contenu IP, «policy» ;
- ◇ des paramètres concernant l'échange des clés.

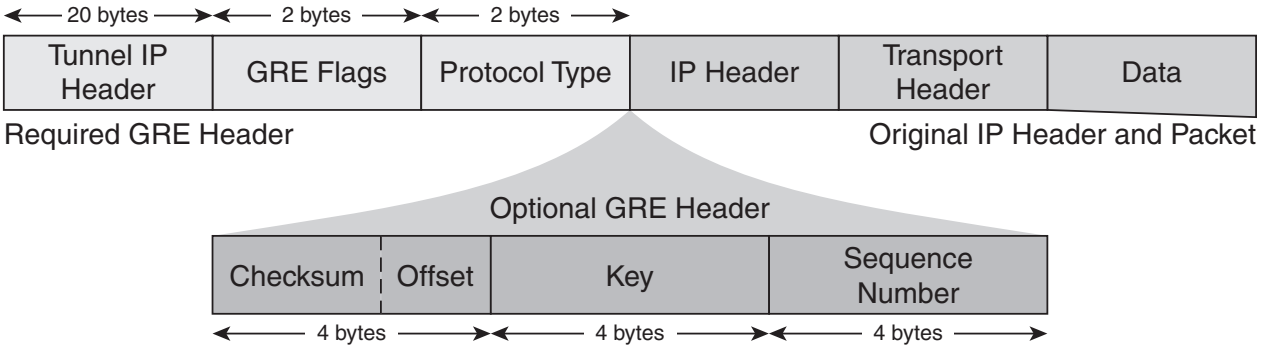
IPSec + GRE

- ▷ GRE permet d'encapsuler du trafic IPv6 ainsi que du trafic multicast, mais ne réalise pas de chiffrement ;
- ▷ IPSec ne s'applique que sur le datagramme IP ;

▷ on peut les combiner !

Application à l'encapsultation de trafic OSPF ou EIGRP envoyé en multicast au travers d'un tunnel « GRE over IPSec»

▷ le protocole GRE :



▷ le protocole GRE + IPSec :

Transport Mode

GRE IP Header	ESP Header	GRE	IP Header	TCP Header	Data	ESP Trailer

Un VPN doit posséder les qualités suivantes :

- l'**interopérabilité** : il doit être possible de mettre en place le VPN entre des matériels de différents constructeurs.
  - IPSec est à privilégier car disponible sur tous les matériels proposant du VPN ;
  - OpenVPN est moins répandu, surtout disponible dans les solutions OpenSource ;
  - PPTP ne permet pas les connexions de site-à-site ;
- les **méthodes d'authentification** :
  - seul PPTP permet l'authentification par «login/mdp» ;
  - IPSec et OpenVPN utilisent des **clés partagées**, «shared keys» ou des **certificats**.
- la **facilité de configuration** :
  - PPTP est très simple ;
  - IPSec possède des options qui pour les *non-initiés* peuvent être complexes ;
  - OpenVPN utilise des certificats qu'il faut savoir gérer.
- la disponibilité d'un **logiciel client** pour les «remote access» :
  - PPTP est intégré dans la plupart des systèmes d'exploitation ;
  - pour IPSec, des client existent pour Windows, Linux, BSD mais ne sont pas toujours intégrés. Pour Mac OS X, un client sans interface graphique est intégré ;
  - pour OpenVPN, des clients existent pour toutes les plateformes me sont pas intégrés.
- la possibilité de **gérer du «Multi-WAN»**, c-à-d utiliser plusieurs connexions simultanées à Internet :
  - PPTP utilise des tunnels GRE, *Generic Routing Encapsulation*, et ne sait pas gérer le «Multi-WAN» ;
  - OpenVPN et IPSec savent le gérer.

- le **passage à travers un Firewall** :
  - PPTP utilise une connexion de contrôle TCP sur le port 1723 et le protocole GRE non sécurisé qui est souvent bloqué par les firewalls ;
  - IPSec utilise le protocole UDP sur le port 500 et des paquets IP basés sur le protocole ESP, *Encapsulating Security Payloads*. L'utilisation d'ESP entraine le chiffrement du contenu du paquet IP et donc, l'impossibilité d'accéder au numéro de port source et destination, ce qui le rend difficile à gérer par un firewall pour effectuer du NAT.
    Il est possible d'utiliser du NAT-T, NAT Traversal, qui encapsule les paquets ESP dans UDP sur le port 4500.
  - OpenVPN peut utiliser UDP et TCP, ce qui le rend le plus apte à être utilisé au travers d'un firewall.
    Pour passer au travers d'un firewall, on peut utiliser les ports UDP 53 (DNS), TCP 80 (HTTP), TCP 443 (HTTPS).

- **sécurisé «cryptographiquement»** :
  - PPTP utilisant des «login/mdp» il est moins sécurisé que les autres solutions : le mdp peut être cracké par une méthode «brute-force» ;
  - l'utilisation de «pre-shared keys» dans IPSec peut le rendre vulnérable si cette clé n'est pas suffisamment robuste pour résister à une attaque *bruteforce*.
  - OpenVPN utilisant des certificats ou simplement des biclés (clé publique/clé privée) partagée, il faut s'assurer de la sécurité de la clé privée ou partagée.

VPN Type	Client included in most OSes	Widely interoperable	Multi-WAN	Crypto-graphically secure	Firewall friendly
IPsec	no	yes	yes	yes	no (without NAT-T)
OpenVPN	no	no	yes	yes	yes
PPTP	yes	n/a	no	no	most

- L'utilisation d'un VPN entraîne l'utilisation de chiffrement qui peut être coûteux car il s'applique sur toutes les données échangées :
  - IPSec utilise les algorithmes de chiffrement suivant : DES, 3DES, Blowfish, CAST128, AES et AES 256 ;
  - il est possible d'utiliser des «crypto-processeurs» pour réaliser le travail de chiffrement.

CPU	**Blowfish Throughput (Mbps)**
Pentium II 350	12.4 Mbps
ALIX (500 MHz)	16.5 Mbps
Pentium III 700	32.9 Mbps
Pentium 4 1.7 GHz	53.9 Mbps

- Le choix d'utiliser un VPN ou de louer une connexion WAN privée (du type ATM ou MPLS) :
  - différence de **latence** :
    - une connexion «point-à-point» de type Ethernet assure une latence de 3 à $5ms$;
    - une connexion, «*First Hop*», vers Internet fournie par un FAI est plus importante $> 20ms$;
    - une connexion par VPN augmente considérablement la latence : 30 à $60ms$.
      Il est possible de diminuer la latence en utilisant pour toutes les connexions entre site le même FAI.

  - **importance de la latence** sur les services utilisés :
    - le partage de fichier Microsoft, «SMB» : pour une latence $< 10ms$ tout marche bien. À partir d'une latence de $30ms$ ses performances s'effondrent, et à $50ms$ il devient insupportablement lent ;
    - l'utilisation de Microsoft Remote Desktop, RDP : une latence $< 20ms$ donne de bonnes performances.
      La latence de 50 et $> 60ms$ donnée par l'utilisation d'un VPN rende le travail d'un utilisateur distant difficile.

Ils sont nombreux

- la possibilité de faire des VPNs de niveau 2 & 3 : dans le cas du niveau 2, le VPN peut transporter des trames, et les protocoles Microsofts (qui sont des protocoles locaux, c-à-d non routables) ;
- la possibilité de bénéficier du firewall du serveur sur lequel on se connecte : un «*road warrior*» peut bénéficier des mêmes protections que les matériels connectés directement dans le réseau de l'entreprise ;
- un fonctionnement en mode client ou serveur, UDP ou TCP ;
- les connexions OpenVN peuvent **traverser** la plupart des firewalls et passer par des proxys : si on peut passer en «https» alors on pourra passer un tunnel OpenVPN en mode TCP ;
- un seul port à ouvrir sur le firewall pour le support d'OpenVPN avec la possibilité pour un serveur de gérer plusieurs clients avec ce seul port ;
- pas de problème avec le NAT ;
- la mise en oeuvre d'OpenVPN en tant qu'ajout de nouvelles interfaces virtuelles TUN/TAP sur le serveur et le client **autorise toutes les utilisation possibles du firewall et du routage !**
- très extensible avec la possibilité de **scripter** la mise en place du VPN et la configuration du client et du serveur ;
- le support transparent des clients utilisant des adresses IP dynamiques sans perte de connexion ;
- installation simple sur les différentes plateformes ;
- design modulaire : le découpage clair entre réseaux et sécurité permet d'envisager de nombreuses possibilités ;
- très actif au niveau de la communauté.

- Utilise des **protocoles cryptographiques modernes** :
  - Curve25519 : échange de clés ;
  - ChaCha20 : chiffrement ;
  - Poly1305 : authentification de données ;
  - BLAKE2 : hashage ;
  - SipHash24 : clés pour accès rapide dans une table ;
  - HKDF : dérivation de clé ;
- **Rapide** : basé **UDP**, primitives cryptographiques rapides, intégration dans le noyau Linux ;
- **Routage par clé cryptographique** :

Serveur

```
[Interface]
PrivateKey = yAnz5TF+lXXJte14t...rYUIgJBgB3fBmk=
ListenPort = 51820

[Peer]
PublicKey = gN65BkIKy1eCE9pP1...HLF2PfAqYdyYBz6EA=
AllowedIPs = 10.10.10.230/32

[Peer]
...
```

Client

```
[Interface]
PrivateKey = gI6EdUS...yiZxIhp3GInSWRfWGE=
ListenPort = 21841

[Peer]
PublicKey = HIgo9xNzKA...Z0U3wGLiUeJ1PKf8ykw=
Endpoint = 192.95.5.69:51820
AllowedIPs = 0.0.0.0/0
```

Si le serveur **reçoit un paquet** depuis gN65BkI et si, après déchiffrement/authentification il vient bien de `10.10.10.230` alors il est accepté $\Longrightarrow$ «*AllowedIPs*» : liste de contrôle d'accès.
Si le serveur envoie un paquet vers `10.10.10.230`, il le chiffre avec la clé publique gN65BkI et utilise le dernier TSAP connu de la machine associée $\Longrightarrow$ «*AllowedIps*» : sélection de la clé/routage.
Sur le client : HIgo9xN a le droit de lui envoyer toute origine de paquet `0.0.0.0/0` et tout paquet peut lui être envoyé comme une *route par défaut*.

- **Roaming** : le client comme le serveur peut **changer d'adresse** en cours de transferts par le tunnel :
  chaque paquet reçu informe de la nouvelle adresse de l'extrémité par le TSAPsource du paquet.
- Compatible **IPv6** et **Ipv4** : encapsulé de l'IPv6 dans IPv4 ou de l'IPv4 dans IPv6.
- Compatible **net namespace** : peut être utilisé comme interface de sortie pour un container.

Installation des commandes

xterm
```
$ sudo apt install wireguard-tools
```

Génération de clés asymétriques

xterm
```
$ wg genkey > privatekey
$ wg pubkey < privatekey > publickey
```

Exemple de configuration

xterm
```
# ip link add dev wg0 type wireguard
# ip address add dev wg0 192.168.2.1/24 # pour plusieurs peers
# ip address add dev wg0 192.168.2.1 peer 192.168.2.2 # pour seulement 2 peers
# wg set wg0 listen-port 51820 private-key /path/to/private-key peer ABCDEF... allowed-ips
192.168.2.0/24 endpoint 209.202.254.14:8172
# ip link set up dev wg0
```

Le **routage** est fait comme dans le cas des autres tunnels.

Firewall et Nat

Attention à autoriser l'entrée des paquets UDP du «*peer*» en communiquant d'abord avec le «*peer*».
Le firewall laissera entrer un paquet UDP avec les TSAP source et TSAP destination inversés par rapport à ceux du paquet UDP en sortie.
L'option de configuration `persistent-keepalive` permet d'envoyer des paquets UDP régulièrement pour conserver l'autorisation d'entrée du firewall.

WireGuard : démo

«*h1*» et «*h2*» veulent communiquer par tunnel Wireguard :

h1

```
$ sudo ip link add dev wg0 type wireguard
$ sudo ip address add dev wg0 172.16.0.1 peer 172.16.0.2
$ sudo wg set wg0 listen-port 51820 private-key privatekey_h1 peer $(cat publickey_h2) allowed-ips
172.16.0.2 endpoint 192.168.20.1:1234
$ sudo wg show
interface: wg0
  public key: 8pObJA8nPprq3/0p1xcsi2IgsCyRZZZEsDrsY1AsxwA=
  private key: (hidden)
  listening port: 51820
peer: VXZtEZYGypA4U9uffNFtqHLcyWvxaw5jf3Db/Kj4sQQ=
  endpoint: 192.168.20.1:1234
  allowed ips: 172.16.0.2/32
$ sudo ip link set wg0 up
```

h2

```
$ sudo ip link add dev wg0 type wireguard
$ sudo ip address add dev wg0 172.16.0.2 peer 172.16.0.1
$ sudo wg set wg0 listen-port 1234 private-key privatekey_h2 peer $(cat publickey_h1) allowed-ips
172.16.0.1 endpoint 192.168.10.1:51820
$ sudo ip link set wg0 up
$ ping -c 1 172.16.0.1
PING 172.16.0.1 (172.16.0.1) 56(84) bytes of data.
64 bytes from 172.16.0.1: icmp_seq=1 ttl=64 time=1.30 ms
--- 172.16.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.304/1.304/1.304/0.000 ms
$ sudo wg show
interface: wg0
  public key: VXZtEZYGypA4U9uffNFtqHLcyWvxaw5jf3Db/Kj4sQQ=
  private key: (hidden)
  listening port: 1234
peer: 8pObJA8nPprq3/0p1xcsi2IgsCyRZZZEsDrsY1AsxwA=
  endpoint: 192.168.10.1:51820
  allowed ips: 172.16.0.1/32
  latest handshake: 18 seconds ago
  transfer: 436 B received, 348 B sent
```

Les interfaces virtuelles : TUN & TAP

- ⋆ sont disponibles sous différents systèmes d'exploitation (même sous Windows) ;
- ⋆ correspondent à une interface réseau virtuelle :
  - ◇ TUN, «*network TUNnel*» : niveau 3 (IP), permet de faire du routage ;
  - ◇ TAP : niveau 2 (liaison de données), permet de faire un «*bridge*» ;
- ⋆ sont connectées à un *processus* simulant un réseau : elles permettent de faire passer les paquets dans le «*user-space*», c-à-d l'espace utilisateur au lieu de l'espace système ou noyau.

Utilisation avec Socat

- ⋆ Création d'une interface TUN sur chaque machine :
  - ◇ sur la machine `Serveur` :

```
xterm
$ sudo socat ❶-d -d ❷TCP-LISTEN:11443,reuseaddr ❸TUN:192.168.255.1/24,up
```

❶⇒permet d'avoir un affichage : «`Prints fatal, error, warning, and notice messages.`»

❷⇒définit un bout du tunnel : une socket TCP en attente de connexion sur le port 11443, sans attente du délai de libération de ce numéro de port ;

❸⇒définit l'autre bout du tunnel : une interface TUN associée à l'@IP `192.168.255.1/24` et activée.

- ⋆ sur la machine `Client` :

```
xterm
$ sudo socat ❹TCP:1.2.3.4:11443 ❺TUN:192.168.255.2/24,up
```

❹⇒définit un bout du tunnel par une connexion TCP vers la machine `Serveur`, d'@IP `1.2.3.4` et de port 11443 ;

❺⇒définit l'autre bout du tunnel : une interface TUN associée à l'@IP `192.168.255.1/24` et activée.

- ⋆ Un «tunnel» est mis en place entre le client et le serveur encapsulé dans une connexion TCP.

Les interfaces virtuelles TUN

▷ Sur la machine Serveur :

```
xterm
tun0      Link encap:UNSPEC  HWaddr
00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet adr:192.168.255.1  P-t-P:192.168.255.1
Masque:255.255.255.0
          UP POINTOPOINT  RUNNING NOARP MULTICAST  MTU:1500
Metric:1
```

```
xterm
pef@solaris:~$ ip route
default via 192.168.1.254 dev eth0  proto static
169.254.0.0/16 dev eth0  scope link  metric 1000
192.168.1.0/24 dev eth0  proto kernel  scope link  src
192.168.1.83  metric 1
192.168.255.0/24 dev tun0  proto kernel  scope link  src
192.168.255.1
```

L'interface est en mode :
«Point à Point»

D'après le routage, cette interface virtuelle permet de communiquer sur un réseau défini entre les deux interfaces virtuelles.

▷ Sur la machine Client :

```
xterm
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet adr:192.168.255.2  P-t-P:192.168.255.2  Masque:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
```

Pour le routage :

```
xterm
rezo@ishtar:~$ ip route
default via 192.168.1.254 dev eth0  metric 100
192.168.1.0/24 dev eth0  proto kernel  scope link  src 192.168.1.62
192.168.255.0/24 dev tun0  proto kernel  scope link  src 192.168.255.2
```

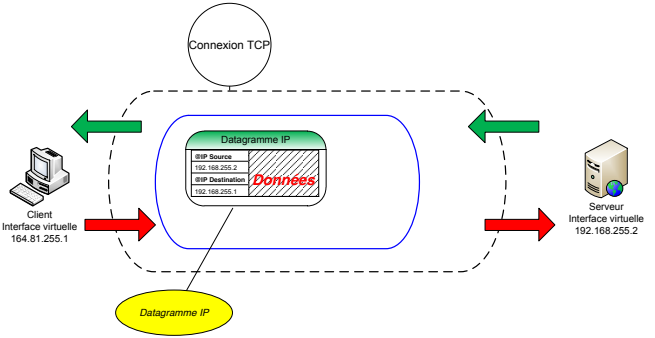
Test du tunnel et observation des échanges

xterm

```
rezo@ishtar:~$ ping -c 1 192.168.255.1
PING 192.168.255.1 (192.168.255.1) 56(84) bytes
of data.
64 bytes from 192.168.255.1: icmp_req=1 ttl=64
time=87.5 ms

--- 192.168.255.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet
loss, time 0ms
rtt min/avg/max/mdev =
87.538/87.538/87.538/0.000 ms
```

On exécute un «ping» de la machine client vers la machine serveur en passant par le tunnel.



Le tunnel est une connexion TCP bidirectionnelle dans laquelle les paquets IP sont encapsulés :

Vue depuis le client :

xterm

```
rezo@ishtar:~$ sudo tcpdump -lnvvX -i tun0 icmp
tcpdump: listening on tun0, link-type RAW (Raw IP), capture size 65535 bytes
23:43:55.011530 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1),
length 84)
    192.168.255.2 > 192.168.255.1: ICMP echo request, id 3503, seq 2, length 64
        0x0000:  4500 0054 0000 4000 4001 bb53 c0a8 ff02  E..T..@.@..S....
        0x0010:  c0a8 ff01 0800 3625 0daf 0002 bbaa 184f  ......6%.......O
        0x0020:  f52c 0000 0809 0a0b 0c0d 0e0f 1011 1213  .,..............
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
23:43:55.035389 IP (tos 0x0, ttl 64, id 24479, offset 0, flags [none], proto ICMP
(1), length 84)
    192.168.255.1 > 192.168.255.2: ICMP echo reply, id 3503, seq 2, length 64
        0x0000:  4500 0054 5f9f 0000 4001 9bb4 c0a8 ff01  E..T_...@.......
        0x0010:  c0a8 ff02 0000 3e25 0daf 0002 bbaa 184f  ......>%.......O
        0x0020:  f52c 0000 0809 0a0b 0c0d 0e0f 1011 1213  .,..............
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
```

La trace obtenue est «normale» :

- un datagramme IP contenant un «ICMP echo request» ;
- un autre en retour contenant l'«ICMP echo reply».

Vue extérieure du tunnel :

xterm

```
pef@solaris:~$ sudo tcpdump -nvvX -i eth0 tcp and port 11443
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
18:49:43.435486 IP (tos 0x0, ttl 64, id 47609, offset 0, flags [DF], proto TCP (6),
length 140)
    192.168.1.62.58791 > 192.168.1.83.11443: Flags [P.], cksum 0x9a80 (correct),
    seq 88:176, ack 89, win 1825, options [nop,nop,TS val 31589753 ecr 90574611],
length 88
   0x0000:  4500 008c b9f9 4000 4006 fc90 c0a8 013e  E.....@.@.....>
   0x0010:  c0a8 0153 e5a7 2cb3 7cde c770 9d76 38e5  ...S..,.|..p.v8.
   0x0020:  8018 0721 9a80 0000 0101 080a 01e2 0579  ...!...........y
   0x0030:  0566 0f13 0000 0800 4500 0054 0000 4000  .f......E..T..@.
   0x0040:  4001 bb53 c0a8 ff02 c0a8 ff01 0800 3625  @..S..........6%
   0x0050:  0daf 0002 bbaa 184f f52c 0000 0809 0a0b  .......O.,......
   0x0060:  0c0d 0e0f 1011 1213 1415 1617 1819 1a1b  ................
   0x0070:  1c1d 1e1f 2021 2223 2425 2627 2829 2a2b  ..... !"#$%&'()*+
   0x0080:  2c2d 2e2f 3031 3233 3435 3637            ,-./01234567
18:49:43.436464 IP (tos 0x0, ttl 64, id 1549, offset 0, flags [DF], proto TCP (6) ,
length 140)
    192.168.1.83.11443 > 192.168.1.62.58791: Flags [P.], cksum 0x9943 (correct),
    seq 89:177, ack 176, win 1810, options [nop,nop,TS val 90574855 ecr 31589753],
length 88
   0x0000:  4500 008c 060d 4000 4006 b07d c0a8 0153  E.....@.@..}...S
   0x0010:  c0a8 013e 2cb3 e5a7 9d76 38e5 7cde c7c8  ...>,....v8.|...
   0x0020:  8018 0712 9943 0000 0101 080a 0566 1007  .....C.......f..
   0x0030:  01e2 0579 0000 0800 4500 0054 5f9f 0000  ...y....E..T_...
   0x0040:  4001 9bb4 c0a8 ff01 c0a8 ff02 0000 3e25  @.............>%
   0x0050:  0daf 0002 bbaa 184f f52c 0000 0809 0a0b  .......O.,......
   0x0060:  0c0d 0e0f 1011 1213 1415 1617 1819 1a1b  ................
   0x0070:  1c1d 1e1f 2021 2223 2425 2627 2829 2a2b  ..... !"#$%&'()*+
   0x0080:  2c2d 2e2f 3031 3233 3435 3637            ,-./01234567
```

Les deux paquets ICMP sont encapsulés dans un segment TCP :

❶⇒le paquet «ICMP echo request» ;

❷⇒le paquet «ICMP echo reply».

Chaque paquet est envoyé dans le flux TCP en «`PUSH`» pour être immédiatement prise en compte de l'autre côté du tunnel (le tunnel ne fait circuler que des **datagrammes IP**).

Il est possible d'utiliser SSL, «*Secure Socket Layer*» pour chiffrer le contenu de la connexion TCP :

▷ Sur le serveur :

xterm

```
$ socat openssl-listen:4433,reuseaddr,cert=server.pem,cafile=client.crt TUN:192.168.255.1/24,up
```

▷ Sur le client :

xterm

```
socat openssl-connect:1.2.3.4:4433,cert=client.pem,cafile=server.crt TUN:192.168.255.2/24,up
```

Des certificats sont utilisés pour authentifier les deux bouts du tunnel.

Création d'interface TUN

xterm

```
pef@solaris:~/$ sudo ip tuntap add dev mon_tun mode tun
pef@solaris:~/$ ifconfig -a
mon_tun   Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          POINTOPOINT NOARP MULTICAST  MTU:1500  Metric:1
          Packets reçus:0 erreurs:0 :0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:500
          Octets reçus:0 (0.0 B) Octets transmis:0 (0.0 B)
pef@solaris:~/$ ip link
15: mon_tun: <NO-CARRIER,POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 500
    link/none
```

L'interface TUN correspond à un lien «point-to-point» où seulement des datagrammes IP circulent.

Création d'interface TAP

xterm

```
pef@solaris:~/$ sudo ip tuntap add dev mon_tap mode tap
pef@solaris:~/$ ifconfig -a
mon_tap   Link encap:Ethernet  HWaddr 06:db:f7:1e:d1:9c
          BROADCAST MULTICAST  MTU:1500  Metric:1
          Packets reçus:0 erreurs:0 :0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:500
          Octets reçus:0 (0.0 B) Octets transmis:0 (0.0 B)
pef@solaris:~/$ ip link
16: mon_tap: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 500
    link/ether 06:4b:17:ff:7a:e2 brd ff:ff:ff:ff:ff:ff
```

L'interface TAP correspond à une interface de type Ethernet où peut circuler tout type de trame.

Attention

L'interface virtuelle est «DOWN» **tant qu'aucune application n'est accrochée à elle**, c-à-d :

- ▷ que tout datagramme ou trame envoyé dessus est détruit ;
- ▷ qu'il est impossible de faire de l'écoute ou de l'injection dessus.

Pour créer un **programme** «attaché» à une interface virtuelle :

- ▷ choisir le type d'interface : TUN (option `IFF_TUN`) ou TAP (option `IFF_TAP`) ;
- ▷ dans le cas de TAP : choisir de récupérer les trames avec l'«ethertype» en préfixe ou non (option `IFF_NO_PI` pour «*Packet Information*»).

Soit la version avec une interface TAP avec entête (2 octets + 2 octets de l'Ethertype) :

```
1  #!/usr/bin/python
2  import os, struct, fcntl
3  from scapy.all import *
4
5  nom_interface = "mon_tap"
6  TUNSETIFF = 0x400454CA
7  IFF_TUN = 0x0001
8  IFF_TAP = 0x0002
9  IFF_NO_PI = 0x1000
10
11 lien = os.open("/dev/net/tun", os.O_RDWR)
12 interface = fcntl.ioctl(lien, TUNSETIFF, struct.pack('16sH', nom_interface, IFF_TAP))
13 print "Interface %s"% interface[:16].strip('\x00')
14
15 # on laisse l'utilisateur configurer l'interface en dehors du programme
16 saisie = raw_input("Attente de configuration de l'interface")
17
18 paquet = os.read(lien,2048) # 2048 taille supérieure à la MTU + entête de la trame + CRC
19 print "Ethertype :",[hex(ord(x)) for x in paquet[2:4]]
20 trame = Ether(paquet[4:] # on donne la trame à Scapy en supprimant l'entête
21
22 # pour envoyer une trame précédée de l'entête 00000800 Ethertype d'IP
23 os.write(lien, '\x00\x00\x08\x00"+str(Ether()/IP()))
```

Lorsque le programme est lancé, l'interface est créée (avec une @MAC *différente* à chaque lancement) :

xterm

```
pef@solaris:~$ ip link
16: mon_tap: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 500
    link/ether 06:4b:17:ff:7a:e2 brd ff:ff:ff:ff:ff:ff
```

Soit la version avec une interface TAP et sans entête :

```
#!/usr/bin/python

import os, struct, fcntl
from scapy.all import *

nom_interface = "mon_tap"
TUNSETIFF = 0x400454CA
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

lien = os.open("/dev/net/tun", os.O_RDWR)
interface = fcntl.ioctl(lien, TUNSETIFF, struct.pack('16sH', nom_interface, IFF_TAP|IFF_NO_PI))
print "Interface %s"% interface[:16].strip('\x00')

❶# on laisse l'utilisateur configurer l'interface en dehors du programme
saisie = raw_input("Attente de configuration de l'interface")

# 2048 taille supérieure à la MTU + entête de la trame + CRC
paquet = os.read(lien,2048)
trame = Ether(paquet) # on donne la trame à Scapy
os.write(lien, str(Ether()/IP())) # pour envoyer une trame
```

En ligne ❶, l'utilisateur peut configurer le programme, par exemple en ajoutant l'interface créée «`mon_tap`» dans un *bridge* comprenant l'interface réseau de connexion vers l'extérieur :

Création du bridge et ajout de `eth0` pour l'accès extérieur :

```
sudo brctl addbr vers_eth0
sudo brctl setfd vers_eth0 0
sudo brctl addif vers_eth0 eth0
sudo ip link set vers_eth0 up
sudo ip addr flush dev eth0
sudo dhclient vers_eth0
```

Ajout de l'interface, une fois le programme lancé :

```
sudo ip link set mon_tap up
sudo brctl addif vers_eth0 mon_tap
```

L'interface n'existant que lors de l'exécution du programme, il est nécessaire de l'ajouter à chaque fois au bridge.

Un VPN au travers de SSH

- un VPN «à la demande» :
  - n'a pas besoin d'être configuré de manière statique ;
  - peut être mis en place et défait suivant les besoins de l'utilisateur : depuis n'importe où et à n'importe quel moment, le rêve du «road warrior» du dimanche ;
- crée des interfaces virtuelles TUN/TAP (nécessite une configuration avec des droits d'administration…) ;

Pour la configuration

- ⋆ il faut installer sur le client «tunctl» :

```
sudo apt-get install uml-utilities
sudo apt-get install openssh-server
```

- ⋆ il faut activer l'«IP forwarding» sur les deux :

```
sysctl -w net.ipv4.ip_forward=1
```

- ⋆ Pour la configuration du serveur SSH :
  - ◇ dans le fichier `/etc/ssh/sshd_config` :

```
[ ... ]
# Enable layer-3 tunneling. Change the value to 'ethernet' for layer-2 tunneling
PermitTunnel point-to-point
```

- ⋆ Et du client SSH :
  - ◇ dans le fichier `/etc/ssh/ssh_config` :

```
[ ... ]
# Enable layer-3 tunneling. Change the value to 'ethernet' for layer-2 tunneling
Tunnel point-to-point
```

Le déclenchement du VPN

- Pour pouvoir permettre le routage du trafic par l'intermédiaire du VPN, il est nécessaire de déclencher le fonctionnement en tant que routeur sur la machine :

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

- Ensuite, sur le poste client, vous exécutez la commande suivante :

```
$ sudo ssh -f -w any:any root@adresse_serveur true
```

Ici, l'option «-f» permet de mettre la commande «ssh» en tâche de fond.
Le «sudo» est nécessaire pour permettre la création d'une nouvelle interface sur la machine.
Le «any:any» permet de sélectionner des interfaces TUN local:distante libres.

- Une nouvelle interface est créee du côté client comme du côté serveur :

```
$ ifconfig -a
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          POINTOPOINT NOARP MULTICAST  MTU:1500  Metric:1
          Packets reçus:0 erreurs:0 :0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:500
          Octets reçus:0 (0.0 B) Octets transmis:0 (0.0 B)
```

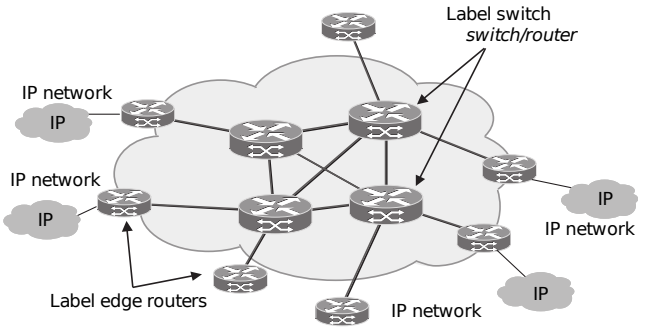
- Il est ensuite possible d'activer l'interface et de la configurer pour permettre le **routage** au travers du VPN, ainsi que le filtrage grâce au fait que le VPN ajoute une **interface virtuelle**.

Il est possible d'auto-configurer le VPN mis en place à l'aide d'une ligne de commande indiquée dans le fichier `~/.ssh/authorized_keys`.

Le protocole MPLS, «Multi-Protocol Label Switching», RFC 3031 & 3032

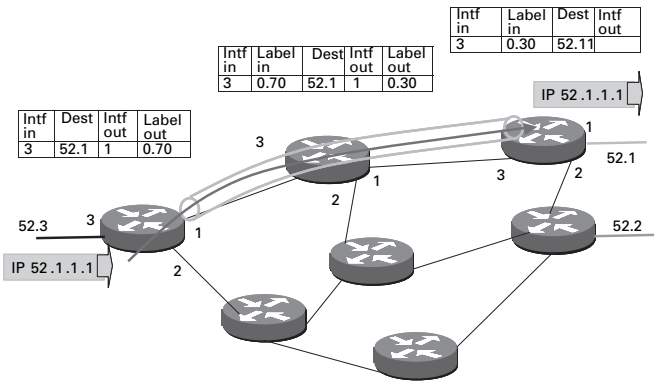
- il combine :
  - ◇ le «*packet switching*» en «circuit virtuel» (comme avec la technologie ATM ou «Frame Relay») ;
  - ◇ le routage IP ;
- le routeur :
  - ◇ **bascule** :
    - ⋆ du routage «hop by hop» ;
    - ⋆ au «switching» c-à-d où les datagrammes empruntent une «sorte» de circuit virtuel : un **chemin**, «path» ou **tunnel** ;
  - ◇ **évite** :
    - ⋆ d'analyser les en-têtes du datagrammes en mode «switching», ce qui diminue la latence ;
    - ⋆ de faire de la fragmentation ;
- cette approche est appelée «*tag switching*» ;
- aux bases de données de routage, «*Routing Information Base*», sont associées des «*Label Information Base*», contenant les étiquettes à affecter à certains flux de datagrammes :
  - ◇ une fois étiqueté, un datagramme passe directement en mode switching sans passer par le routage ;
  - ◇ les étiquettes doivent être gérées entre les différents routeurs à l'aide du «*label distribution protocol*» ou LDP ;
  - ◇ le réseau d'interconnexion est décomposé en :
    - ⋆ LER, «*Label Edge Routers*» ou *Edge LSR* : situés en bordure et chargés de classifier et d'étiquetter les flux ;
    - ⋆ LSR, «*Label Switching Routers*» : situés dans le noyau du réseau d'interconnexion et qui «switchent» les datagrammes en fonction de leur étiquettes ;
    - ⋆ le protocole LDP est utilisé entre ces différents routeurs pour maintenir la table des étiquettes et faire également la réservation d'un circuit virtuel, «*label-switched Path*», ou LSP, entre les routeurs : ils envoient des «LDP requests».
      Il est également possible d'employer le protocole RSVP-TE, «RSVP - Traffic Engineering» RFC 3209.
  - ◇ l'interconnexion des routeurs LSR est appelé un «domaine MPLS» ;
  - ◇ une étiquette MPLS définie un tunnel **unidirectionnel** : le LSP, «*Label Switch Path*» avec un routeur LSR d'entrée, «*ingress*» et de sortie «*egress*».

Le réseau d'interconnexion avec MPLS



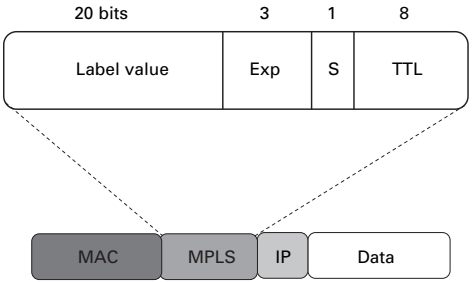
L'utilisation de MPLS est limité au réseau d'interconnexion contrôlé par une même organisation.

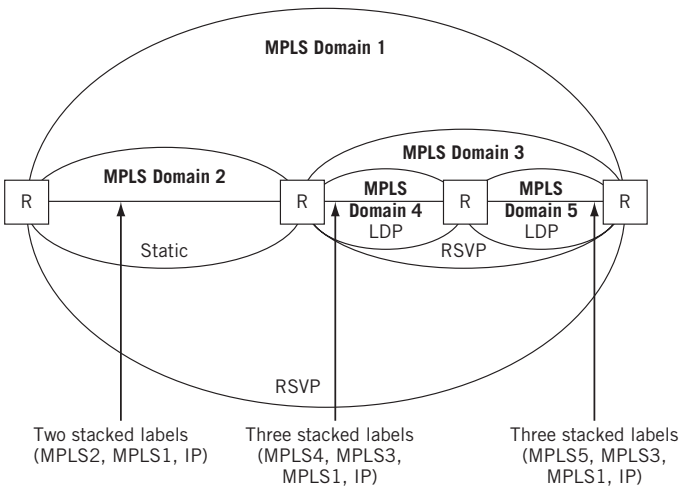
Un LSP, «label-switched path»



Les datagrammes empruntent le même chemin suivant l'étiquette MPLS qu'ils ont reçus : ils passent d'une interface d'entrée, «intf in»*, à une interface de sortie «intf out» en récupérant une étiquette lors de leur entrée ou en changeant lors de leur sortie.*

Integration dans TCP/IP : modification de la trame Ethernet 802.3





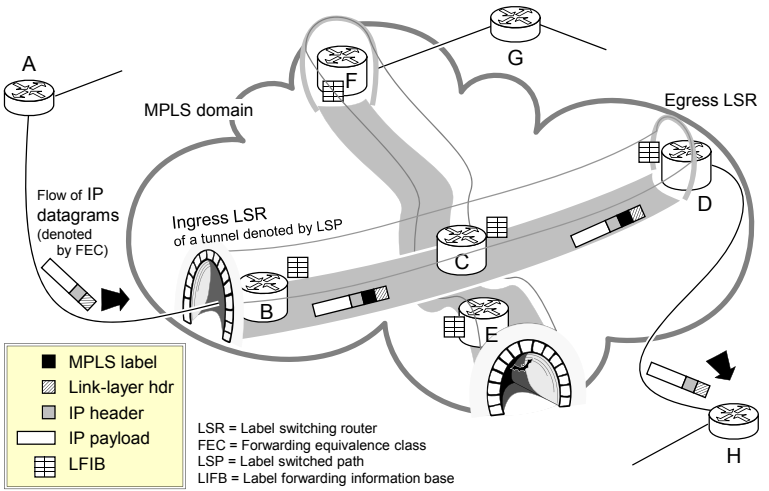
On insère une «étiquette MPLS» entre l'en-tête de la trame Ethernet 802.3 avec un type `0x8847`, et le datagramme IP :

- **Label** : 20bits identifie les paquets comme étant inclus dans un «tunnel» MPLS :
  - valeur choisie pour un lien et partagée seulement entre les deux routeurs LSR de ce lien ;
- **Exp** : CoS, «*Class of Service*» : 3bits utilisés pour classifier le flux parmi une des 8 catégories ;
  Similaire au TOS du datagramme IP.
- FEC, «*forwarding equivalence class*» :
  - exprime la QoS à l'intérieur d'un même tunnel ;
  - valeur combinée de |label|Exp| ;
- **S**, «*Stack bit*» : permet d'avertir le routeur qu'une autre étiquette est empilée après les 32bits de l'étiquette courante ;
- **TTL**, «*Time to Live*» : 8bits utilisés de la même manière que le TTL de l'entête du datagramme IP ;
  Cette valeur peut être copiée depuis ou vers celle du datagramme IP.

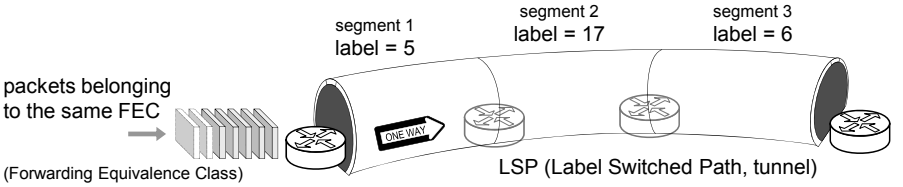
Les étiquettes peuvent être empilées ou «stackées», entre différents domaines, qui peuvent être combinés de différentes manières.

Remarque : les avancées en terme de puissance de routeur ne rend plus la solution MPLS incontournable, néanmoins elle permet de faire de la QoS, de manière très efficace : fournit des garanties sur la QoS.

Illustration

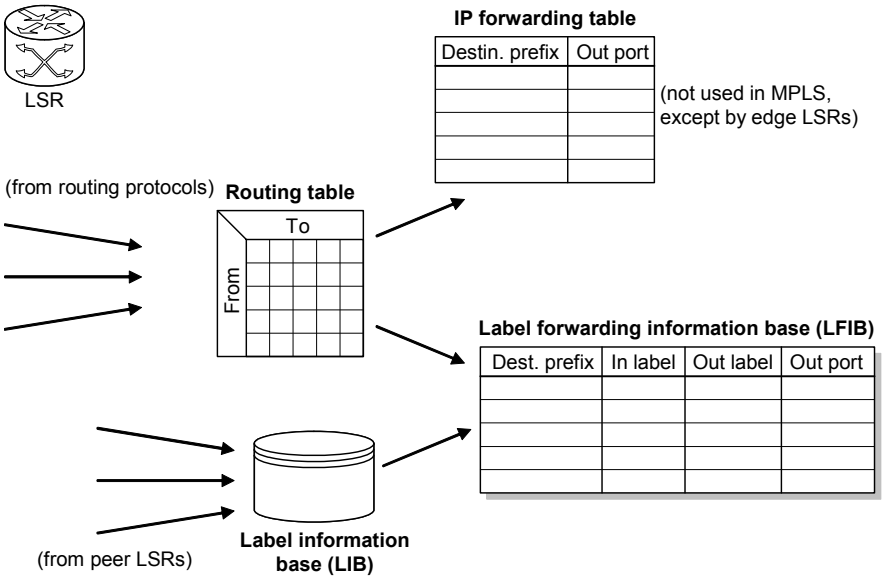


On peut considérer MPLS comme un mécanisme permettant de créer et d'utiliser des «chemins», path*, sortes de «tunnels» :*



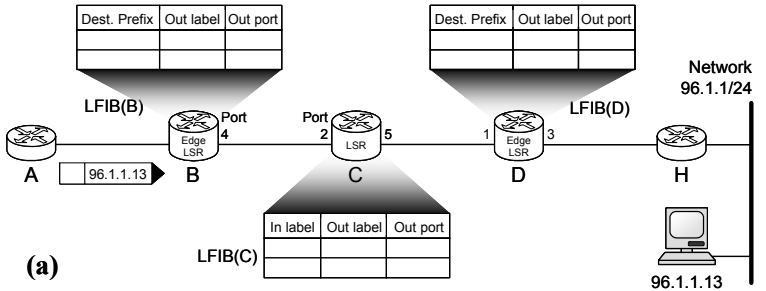
Attention : le «tunnel» MPLS est plus efficace qu'un «tunnel» VPN, mais se comporte de manière similaire (le «packet switching» est plus efficace que le «packet routing».

MPLS : choix entre routage et commutation



La LFIB, «Label Forwarding Information Base» est une structure de données qui sert à paramètrer le «forwarding» des paquets où les destinations et les étiquettes d'entrée sont associées avec les étiquettes et interfaces de sorties.

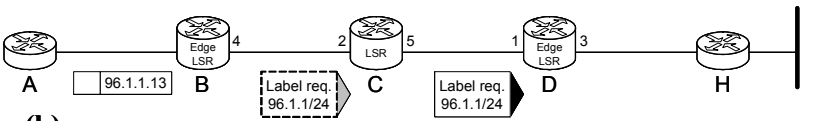
- *sur un router «LSR», seule la LFIB est utilisée ;*
- *sur un routeur «Edge LSR» :*
  - *en ingress : il faut consulter la «IP forwarding table» en plus de la LFIB ;*
  - *en egress : il faut consulter seulement la «IP forwarding table» si le paquet ne contient plus qu'une étiquette MPLS.*
- *suivant le Ethertytpe :* `0x0800` ⟶ *IPv4,* `0x8847` ⟶ *MPLS Unicast et* `0x8848` ⟶ *MPLS Multicast.*

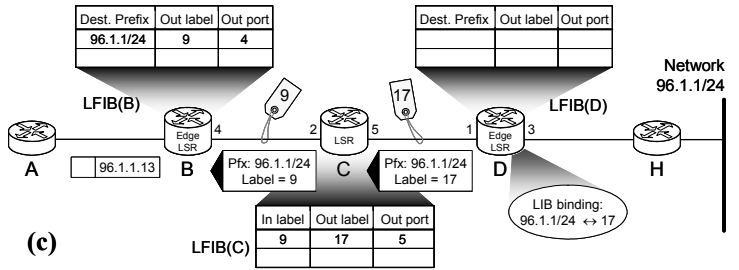


Ici, on a un tunnel B⟶C⟶D où

- *le «Edge LSR» B reçoit un paquet à destination «* `96.1.1.13` *»;*
- *B dispose dans sa table de routage de la destination «* `96.1.1.0/24` *» mais ne dispose pas de l'étiquette dans sa LFIB;*

- *le routeur B envoi une requête vers C et D suivant un LDP,* «Label Distribution Protocol»*, pour le réseau «* `96.1.1.0/24` *».*
- *le routeur D est le «egress LSR», c-à-d le routeur MPLS de sortie : il choisit une étiquette libre,* `17`*, ajoute une association* «`96.1.1.0/24`»↔*17 dans sa LIB, et répond vers C avec cette étiquette.*



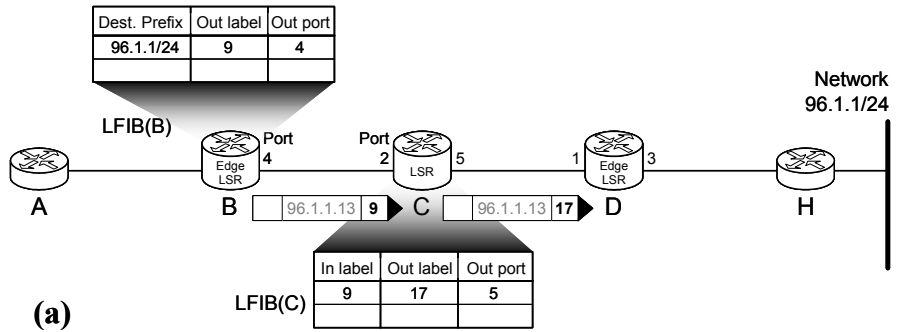


- le routeur C fait de même avec l'étiquette `9`.
- C est un LSR intermédiaire : il ajoute une information dans sa LFIB ;
- B est un «igress LSR», c-à-d un routeur LSR d'entrée, il ajoute l'association

«`96.1.1.0/24`»↔9

dans sa LFIB ;

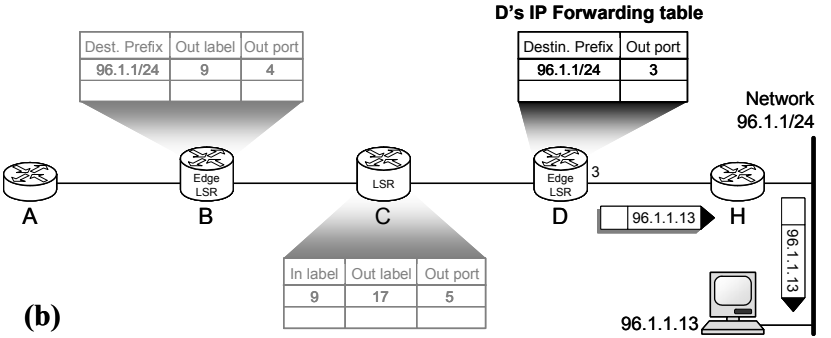
- lorsque C «*forward*» un paquet il fait du «*label swapping*», c-à-d de l'échange d'étiquette !



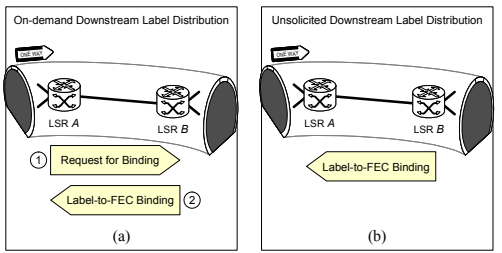
(a)

- *un paquet à destination de «* `96.1.1.13` *» est reçu par B qui utilise sa LFIB pour l'envoyer vers C ;*
- *C utilise sa LFIB pour envoyer le paquet vers D et «switch» d'étiquette entre 9 et 17 ;*

- *lorsque le routeur D reçoit le paquet, il regarde l'étiquette et constate qu'il est le «egress LSR» :*
  - *il retire l'étiquette ;*
  - *il utilise sa table de routage pour envoyer le paquet vers H.*



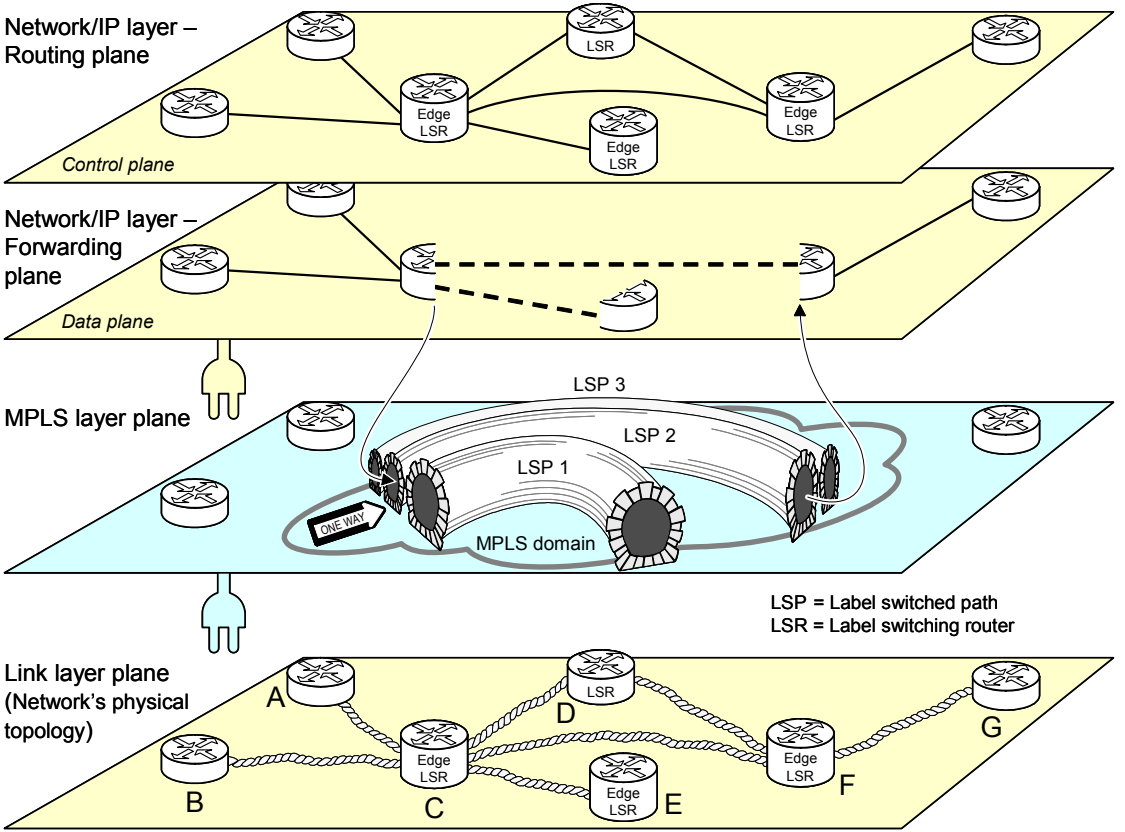
(b)

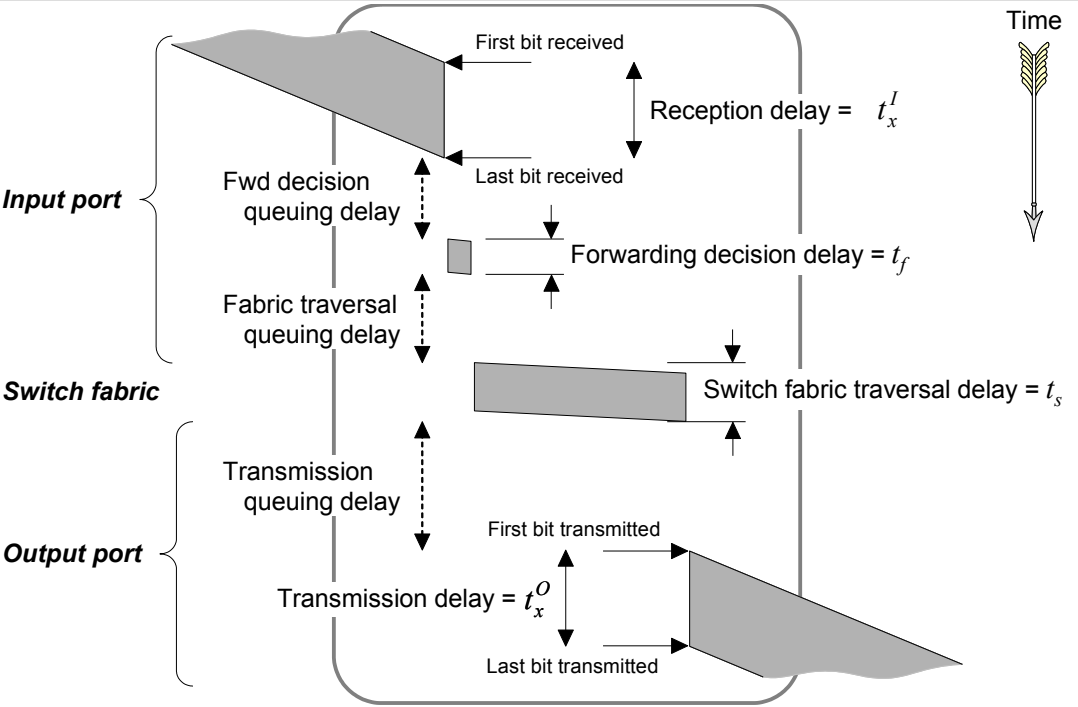


L'obtention d'une étiquette est faite à l'aide du protocole LDP :

- *des informations de liens, «binding», sont échangées pour associer une étiquette à un FEC ;*
- *soit à la demande explicite d'un routeur : «on-demand label distribution» ;*
- *soit de manière pro-active, par envoi d'un message contenant l'association.*

MPLS : les différentes couches





Pour le switching :

- ⋆ on consulte le «circuit-virtuel» auquel appartient le paquet en entrée ;
- ⋆ on supprime le temps de décision, «*Forwarding decision delay*» t_f ;
- ⋆ on réduit le temps de «switching», le «*Switch Fabric traversal delay*» t_s.

Avantages

- utilise la commutation, «switching», au lieu du routage : les étiquettes peuvent être utilisés comme index dans une table des interfaces de sortie de la table de commutation ;
- permet de définir des routes basées sur les contraintes de ressources plutôt que sur la distance la plus courte ;
- supporte la création de VPN, «*Virtual Private Networks*» : le contrôle de mécanismes de tunneling offrant des tunnels rapides entre domaines IPs (par exemple pour une entreprise nationale multi-site) :
  - avantage conservé même avec des routeurs plus puissants ;
  - **peut être transporté dans différents protocoles de niveau 2** : MPLS est appelé protocole de niveau 2,5 en OSI, comme Ethernet ou PPP ;
  - un tunnel MPLS peut avoir plusieurs FEC : un FEC peut être choisi par rapport à l'adresse IP destination ou l'adresse IP source le type de protocole, les ports UDP/TCP, le champs «Differentiated Services» *etc.* ;
- un tunnel MPLS **peut offrir un réseau de niveau 2** à la manière d'un switch VPLS,«*Virtual Private Lan Service*» :
  - support du domaine de diffusion : envoi en broadcast (par exemple de requête ARP), en multicast ;
  - limitation du domaine de collision : apprentissage des adresses MAC des machines connectées au tunnel ;

MPLS & Virtual Private Networks

Correspondance avec les VPNS :

routeurs MPLS	**routeurs VPN**
d'entrée et de sortie : «*Edge LSRs*»	PE «*Provider Edge Router*»
internes supportant MPLS :«*Core LSRs*»	internes P «*Provider Router*»
	CE «*Customer Edge Router*» en lien avec le routeur du réseau client