IPv6

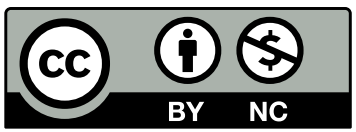
Pierre David
pda@unistra.fr

Université de Strasbourg – Master CSMI

2023 – 2024

Plan

Licence d'utilisation



Plan

Pourquoi changer ?

Années 1990 : Internet devient « victime » de son succès

- environ 2 millions de machines en 1993
- 23494 réseaux alloués à la fin janvier 1994
- croissance exponentielle (doublement tous les 12 mois)

Problèmes :

- saturation de l'espace d'adressage
  (plus de numéros de réseaux)
  Exemple : d'après étude théorique en 1990, épuisement des classes B en mars 1994.
- explosion des tables de routage

Pourquoi changer ?

Les adresses IPv4 (32 bits) sont trop courtes

$\Rightarrow$ utiliser des adresses plus larges

$\Rightarrow$ changement de protocole (incompatible avec IPv4)

Pourquoi changer ?

Évolution des besoins en réseaux :

- historiquement : marché restreint à l'informatique
  (grands ordinateurs, puis stations de travail, puis PC, etc.)
- puis ouverture à de nouveaux marchés
  - terminaux mobiles
    (grande mobilité, liaisons radio ou GSM)
  - grand-public
    (vidéo interactive, domotique, etc. : Internet jusque dans les téléviseurs ou les cafetières)
  - internet des objets
    (explosion du nombre d'objets connectés)

$\Rightarrow$ explosion du nombre de machines connectées

$\Rightarrow$ nouveaux besoins

Pourquoi changer ?

IP doit prendre en compte :

- le problème des adresses
  $\Rightarrow$ interopérabilité avec IPv4
- les nouveaux besoins
  $\Rightarrow$ mise en œuvre rapide pour éviter l'apparition de protocoles spécifiques

Pourquoi changer ?

Constat :

- de toutes manières, il faut changer le protocole
  $\Rightarrow$ cela prendra du temps
- ça ne résoud pas le problème des tables de routage
  $\Rightarrow$ le problème de saturation est immédiat

Novembre 1991 : approche en deux temps

- à court terme : reculer l'échéance de la saturation de l'espace d'adressage
  $\Rightarrow$ pour gagner du temps
- à moyen/long terme : IPng ($\rightarrow$ IPv6)
  véritable refonte du protocole IP
  $\Rightarrow$ pour les 10 ou 20 prochaines années

Approche à court terme

Reculer l'échéance :

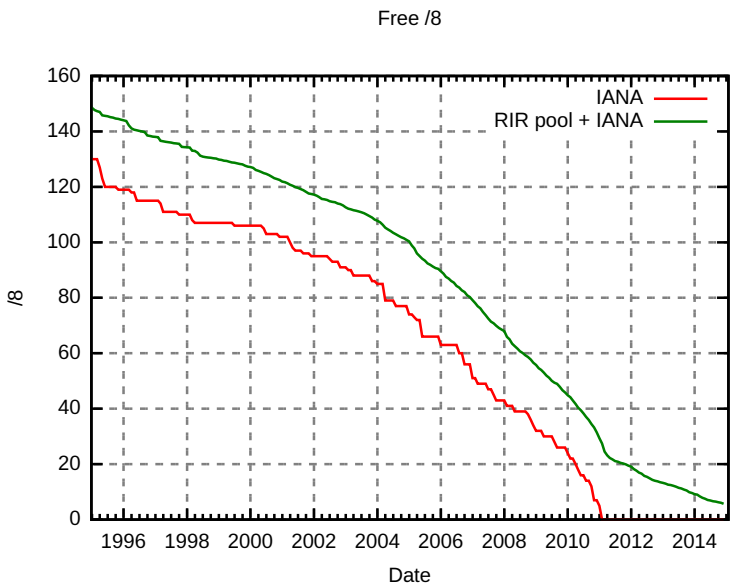
- durcir la politique d'allocation des adresses IP
  $\Rightarrow$ plus difficile d'obtenir des adresses
- utiliser CIDR
  $\Rightarrow$ allocations plus fines que /8, /16 ou /24
- utiliser des adresses non routables RFC 1918
  $\Rightarrow$ utilisation grandissante de NAT
- rendre les adresses non utilisées
  $\Rightarrow$ Stanford a rendu 36.0.0.0/8 à l'IANA en 2000

Approche à court terme

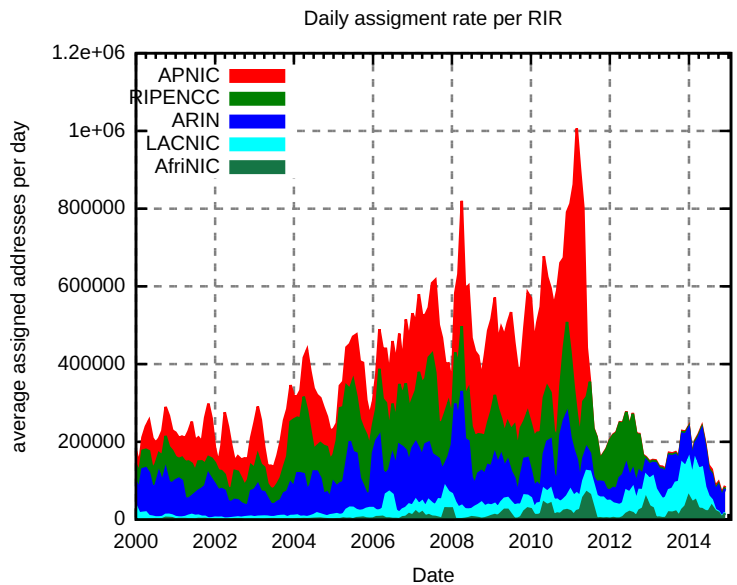
L'échéance est maintenant dépassée... presque partout !

RIR	Zone	Dernier bloc alloué
APNIC	Asie-Pacifique	avril 2011
RIPE	Europe	septembre 2012
LACNIC	Amérique du sud	juin 2014
ARIN	Amérique du nord	septembre 2015
AFRINIC	Afrique	avril 2017

- RIR = *Regional Internet Registry* (organisme de gestion des adresses)
- Les RIR continuent à allouer des adresses (récupération d'adresses inutilisées)



Blocs d'adresses IPv4 /8 disponibles – Mro Wikimedia Commons



Nb d'adresses allouées par RIR – Mro Wikimedia Commons

Approche à court terme

L'approche à court terme continue...

Drafts IETF en cours (2022) pour utiliser des blocs réservés jusqu'ici :

- `0/8` : réseau 0
- `240/4` : ancienne classe E « expérimentale »
- `127/8` (sauf `127.0/16`) : localhost

Tout ce qui est rare est cher :

- adresses allouées par les RIR : « prêtées » et non « vendues »
  - les opérateurs peuvent facturer la « location » d'adresses à leurs clients
- développement d'un marché noir des adresses IPv4
  - des *brokers* se sont lancés dans le commerce d'adresses
  - coût (2021) de blocs d'adresses IPv4 : environ $ 40 par adresse

Approche à long terme – Historique

- Depuis février 1992 : développement de propositions de nouveaux protocoles.
- Décembre 1993 : appel aux *White-Papers*
  (« cahier des charges » de IPng par les utilisateurs)
  21 *White-Papers* reçus
- Juillet 1994 : choix entre les propositions par l'IETF
- Novembre 1994 : publication par l'IETF du document de référence : « Recommendations for the IP Next Generation Protocol » (RFC 1752)
- Mars 1995 : premiers paquets échangés entre implémentations différentes
- Décembre 1995 : premières RFC de spécification d'IPv6
- Juillet 1996 : création du 6bone

Approche à long terme – Historique

Principales propositions reçues par l'IETF :

- TUBA (TCP and UDP with Bigger Addresses)
- SIPP (Simple Internet Protocol Plus)
- CATNIP (Common Architecture for the Internet)

Choix par l'IETF en juillet 1994 :

- SIPP avec adresses sur 128 bits
- numéro de version : 6

Plan

Caractéristiques

Ce qui ne change pas par rapport à IPv4 :

- principe du datagramme (absence de fiabilité, routage entre adresses IP)
  $\Rightarrow$ simplicité
- indépendance du support physique
- TCP et UDP
- multicast
- protocole de contrôle similaire à ICMP
  nouveau nom : ICMPv6, avec plus de possibilités

Caractéristiques

Ce qui change par rapport à IPv4 :

- simplification de l'en-tête
  $\Rightarrow$ ce qui n'est pas utilisé est supprimé (options, etc.)
  $\Rightarrow$ adaptation aux réseaux à très haut débit
- adresses sur 128 bits (16 octets)
  $\Rightarrow$ 10^9 réseaux, routage « géographique »
  $\Rightarrow$ autoconfiguration
- sécurité
  $\Rightarrow$ authentification, chiffrement et intégrité
  $\Rightarrow$ intégrée dès les couches les plus basses
- mécanismes d'autoconfiguration
  $\Rightarrow$ support des mobiles, changements d'adresses
- tunneling
  $\Rightarrow$ liberté de topologie

Caractéristiques

Ce qui change par rapport à IPv4 (suite et fin) :

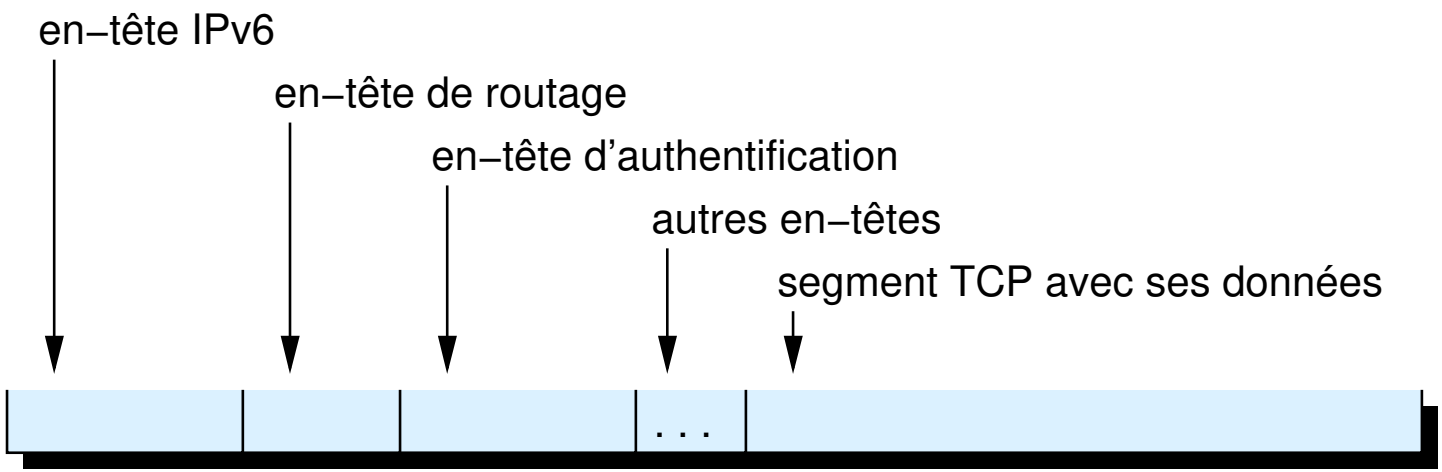
- classes de service
  $\Rightarrow$ trafic temps-réel, trafic non urgent, etc.
- pas de fragmentation par les nœuds intermédiaires
  $\Rightarrow$ performance
- pas de somme de contrôle au niveau IP
  $\Rightarrow$ pas suffisante pour garantir l'intégrité
- options extensibles
  $\Rightarrow$ possibilité d'étendre IPv6 sans casse (théoriquement)
- disparition de ARP
  $\Rightarrow$ partie intégrante de ICMPv6

Plan

Principes

- adresses
- l'en-tête IPv6 est constitué de 40 octets
- alignement sur des mots de 64 bits (8 octets)
- en-têtes d'extension optionnels

Exemple :



Principes

Ordre des extensions :

1. IPv6 (obligatoire)
2. Options examinées à chaque nœud intermédiaire
3. Options examinées à la destination (à la première, i.e. à chaque routeur lors d'un routage explicite)
4. Routage explicite
5. Fragmentation
6. Authentification
7. Sécurité
8. Options examinées à la destination (destinataire ultime du message)
9. Protocole encapsulé (TCP, UDP, ICMP... ou même IPv6)

Principes

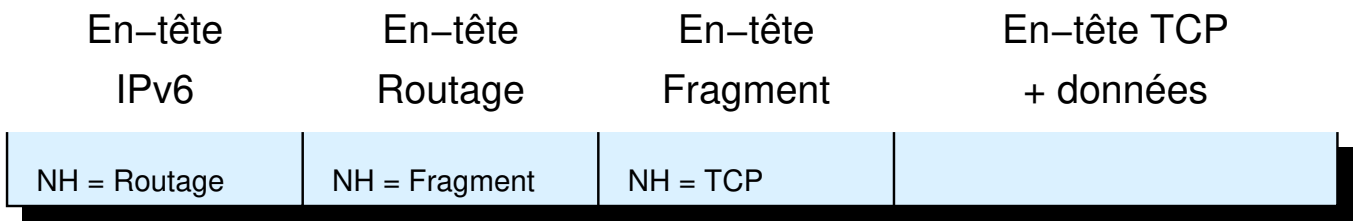
Notes sur l'ordre des extensions :

- une extension n'apparaît au plus qu'une seule fois (sauf pour 3 et 8)
- d'autres extensions peuvent être définies ultérieurement
- les extensions 2 et 3/8 sont différentes

Intérêt : la plupart des extensions ne sont pas examinées par les nœuds intermédiaires $\Rightarrow$ performance

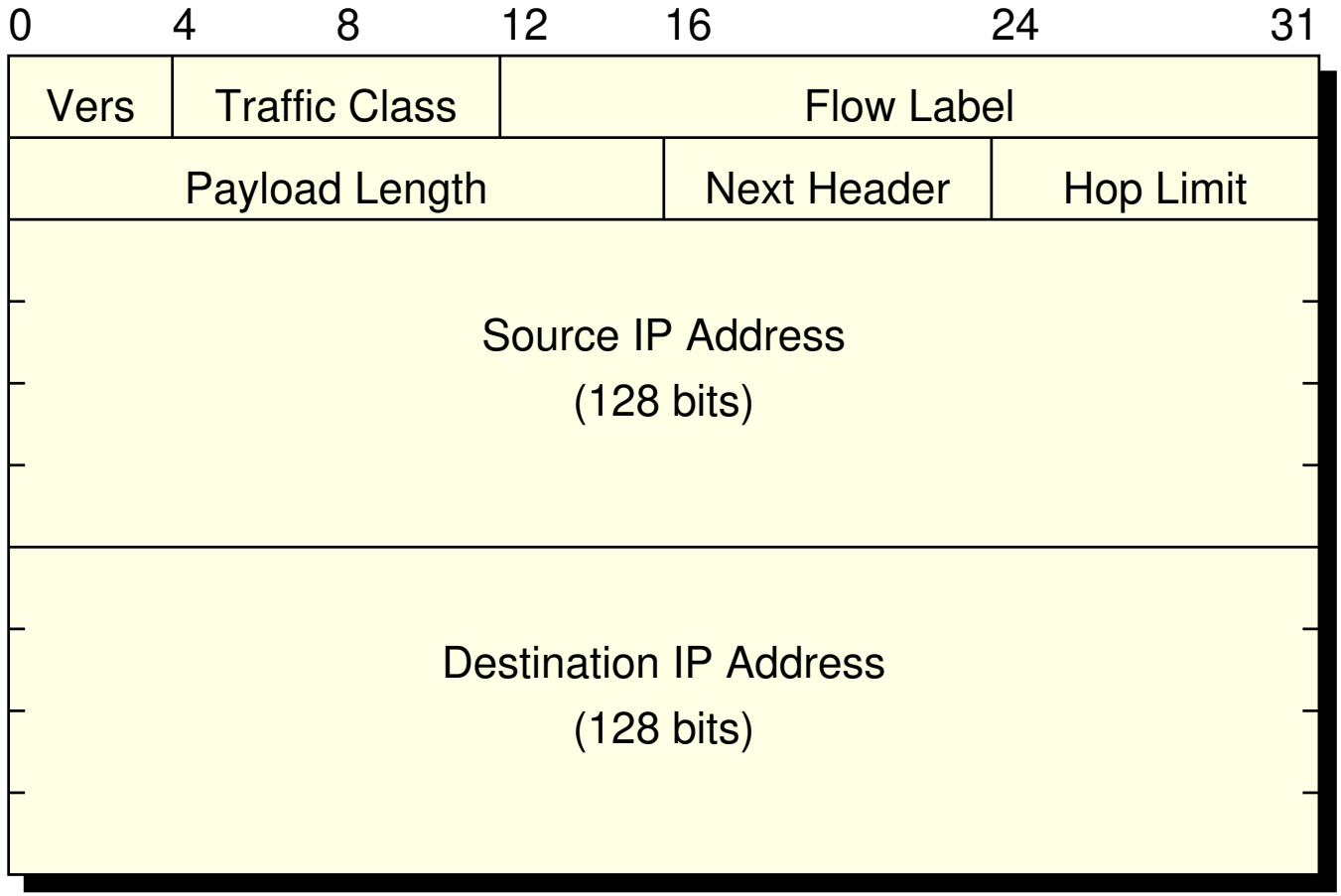
Principes

Chaînage des extensions : chaque en-tête contient le type de l'extension suivante (champ NH : *Next Header*).



Principes

En-tête IPv6 :



Longueur totale : 40 octets (fixe)

Principes

En-tête IPv6 :

- *Vers* : numéro de la version IP (= 6)
- *Traffic Class* : classe de trafic
- *Flow Label* : identificateur de flux
- *Payload Length* : longueur en octets du reste du paquet (non compris l'en-tête IPv6)
- *Next Header* : type de l'en-tête suivant
- *Hop Limit* : nombre maximum de nœuds intermédiaires ($\Rightarrow$ champ TTL dans IPv4)

Principes – Options

Traitement des extensions d'options

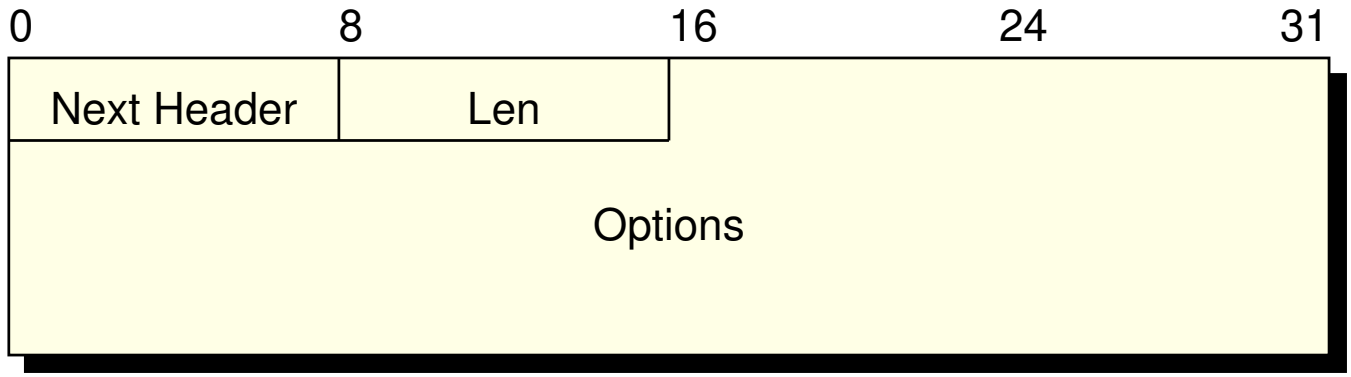
Deux types :

- en-tête *hop-by-hop*
  $\Rightarrow$ options examinées à chaque nœud
  Options définies : alignement, *jumbo payload length*
- en-tête *end-to-end*
  $\Rightarrow$ options examinées uniquement par la destination (intermédiaire ou ultime)
  Options définies : alignement

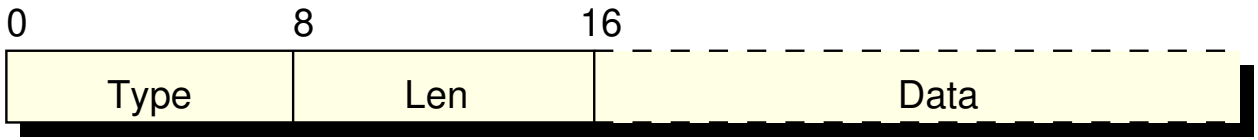
mais format d'en-tête identique

Principes – Options

Format des extensions d'options :



- *Next Header* : type de l'en-tête suivant
- *Len* : longueur (en blocs de 8 octets) de l'en-tête non compris les 8 premiers
- *Options* : suite de spécifications d'options



Principes – Options

Les deux bits de poids fort du type de chaque option codent l'action à effectuer par le nœud si l'option n'est pas reconnue :

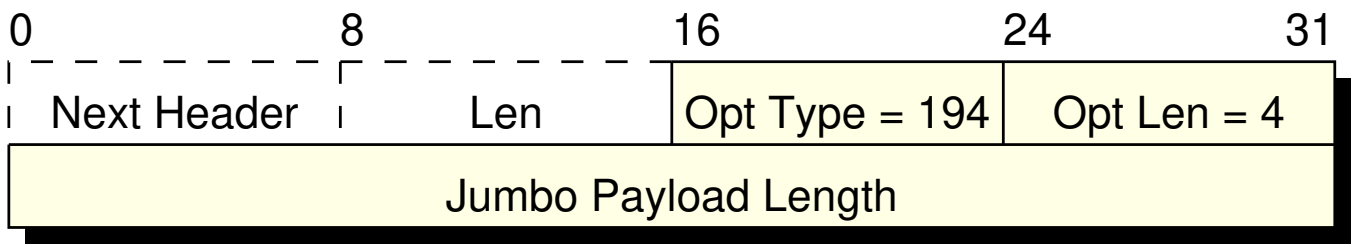
bits	action
00	ignorer l'option
01	ignorer le paquet
10	ignorer le paquet et renvoyer un message ICMP
11	ignorer le paquet et renvoyer un message ICMP seulement si l'adresse de destination n'est pas une adresse multicast

Principes – Options

Option *jumbo payload length*

- paquets de moins de 2^{16}-1 octets :
  $\Rightarrow$ la longueur dans l'en-tête IPv6 suffit
- paquets de plus de 2^{16} octets :
  $\Rightarrow$ ajout d'une option (*jumbo payload length*) examinée à chaque nœud
  $\Rightarrow$ longueur dans l'en-tête IPv6 = 0

Format de l'option :



Principes – Fragmentation

Performances $\Rightarrow$ fragmentation déconseillée en IPv6

Comment la diminuer ?

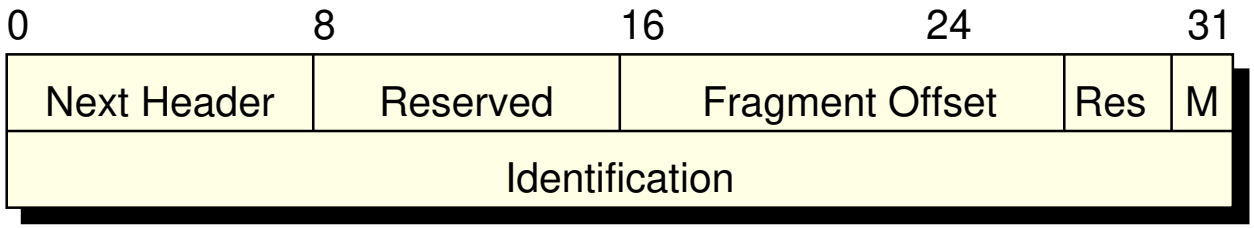
- MTU minimum garanti = 1280 (68 avec IPv4)
- technique du *path MTU discovery* pour trouver le MTU minimum sur un chemin
- ... ou alors n'envoyer que des paquets au MTU minimum garanti

Principes – Fragmentation

Comment fonctionne-t-elle ?

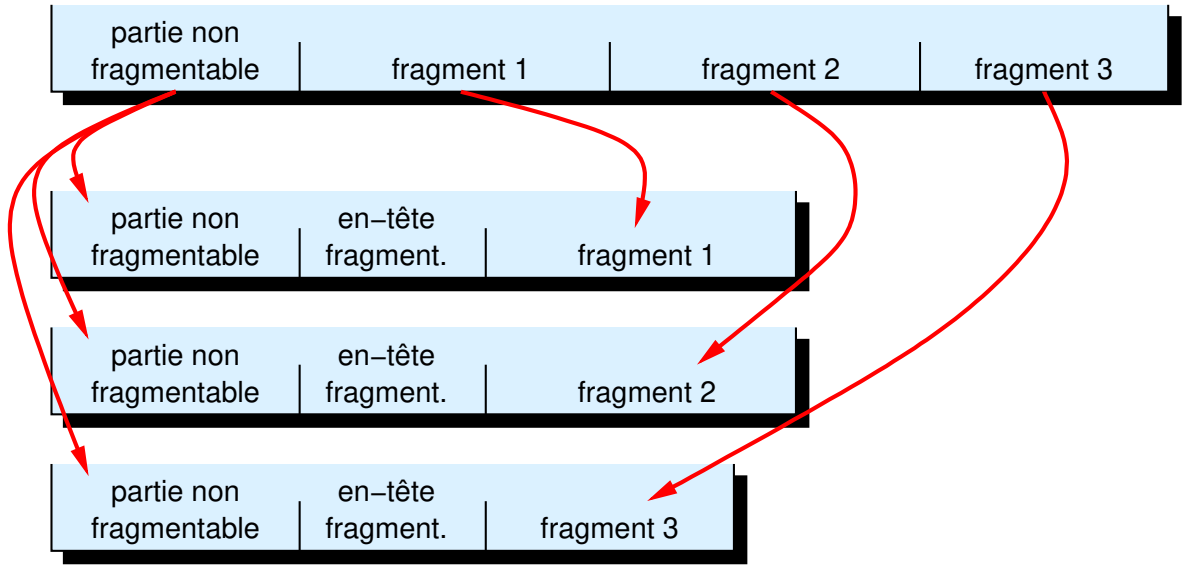
Principe similaire à IPv4, sauf :

- réalisée uniquement par la source
  $\Rightarrow$ pas de fragmentation par les nœuds intermédiaires
- nécessite une extension supplémentaire :



  - champ *Fragment Offset* et flag *M* : idem IPv4
  - champ *Identification* : identifie le paquet fragmenté

Principes – Fragmentation



- partie non fragmentable : en-tête IPv6 + en-têtes examinés par les nœuds intermédiaires
- en-tête fragment. = en-tête de fragmentation, avec :
  - *Next Header* : identification du premier en-tête suivant la partie non fragmentable
  - *Fragment offet*, Flag *M* et *Identification* : idem IPv4

Principes – Routage explicite

Problème : dans certaines situations, on désire spécifier un chemin pour les datagramme

Solution : routage explicite (ou *source routing*)
$\Rightarrow$ consiste à spécifier un ensemble d'adresses IP par lesquelles doit passer le datagramme

Note : routage explicite existait déjà dans les options IPv4, mais jamais mis en œuvre pratiquement

Principes – Routage explicite

Dans IPv4, deux catégories de routage explicite :

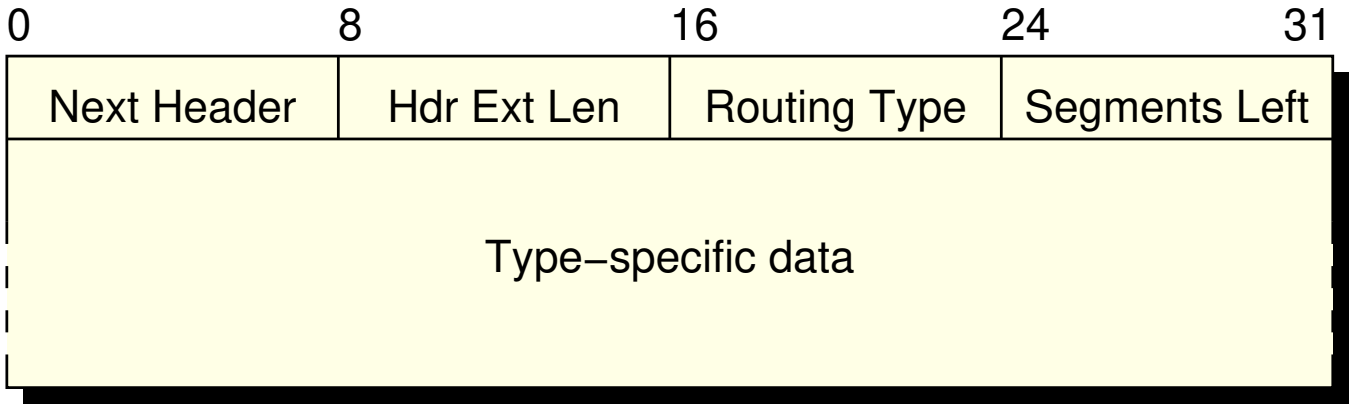
1. routage explicite strict (*strict source routing*)
   $\Rightarrow$ on spécifie les adresses IP de **tous** les nœuds intermédiaires (ils doivent être tous physiquement interconnectés)
2. routage explicite lâche (*loose source routing*)
   $\Rightarrow$ on spécifie **quelques** nœuds : chaque nœud utilise sa table de routage pour envoyer au suivant
   (ils ne sont pas forcément physiquement interconnectés)

Attention : le routage explicite ne fonctionne pas correctement dans toutes les implémentations d'IPv4

Dans IPv6, seul le routage explicite lâche est défini

Principes – Routage explicite

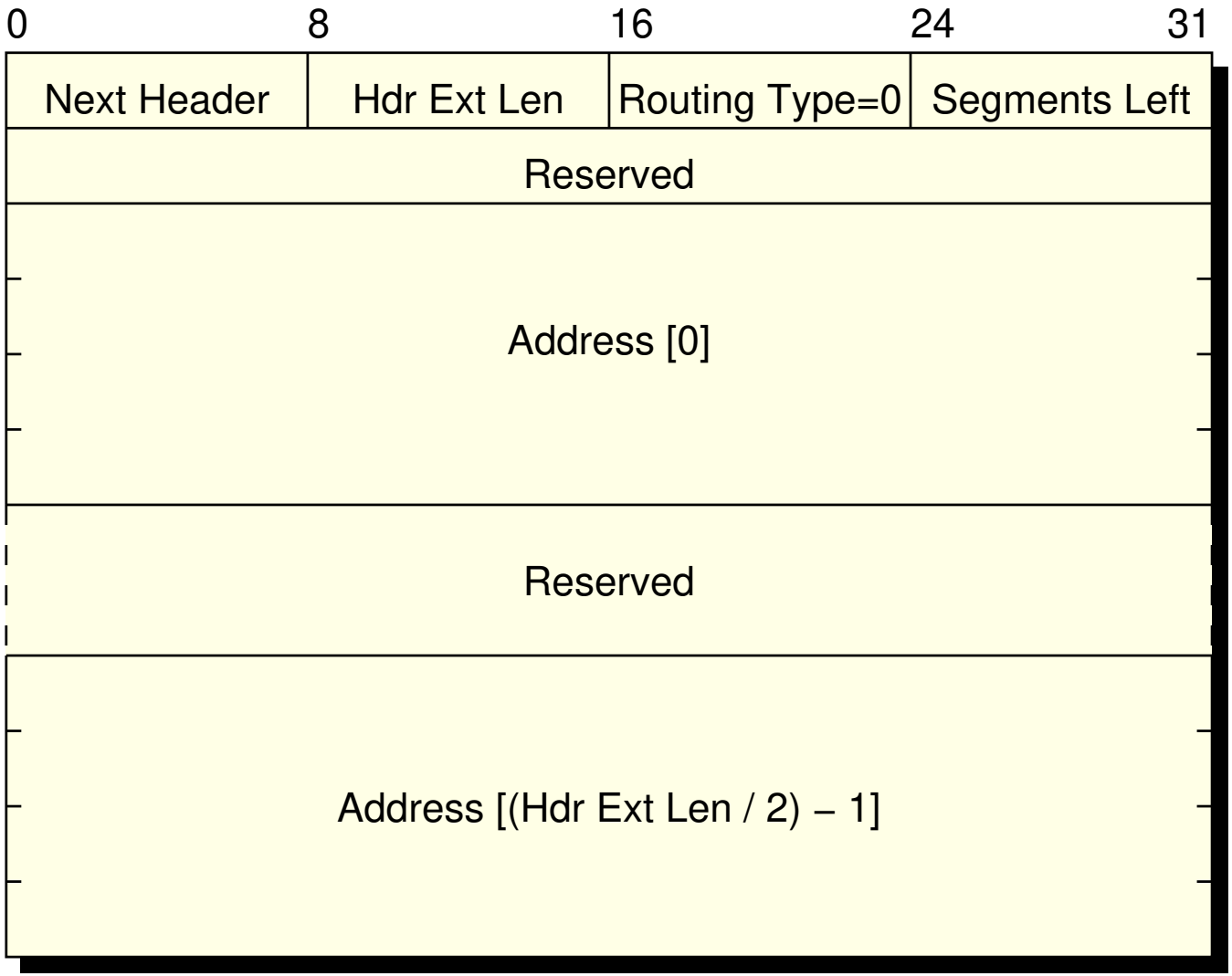
Format général de l'en-tête de routage :



- *Hdr Ext Len* : taille de l'en-tête de routage, en mots de 8 octets, non compris le premier mot
- *Routing Type* : type de routage (un seul défini)
- *Segments Left* : nombre d'éléments de routage restant à traiter

Principes – Routage explicite

Routage de type 0 :



Principes – Routage explicite

Fonctionnement du routage explicite :

- en-tête IPv6 contient l'adresse du prochain routeur
- le nombre *n* d'éléments de routage vaut *ExtHdrLen*/2
- l'indice du routeur suivant dans la liste d'éléments vaut *n* - *Segments Left*
- lorsqu'un routeur reçoit le paquet :
  - il permutte l'adresse figurant dans l'en-tête IPv6 (i.e. la sienne) avec l'adresse du prochain routeur
  - il décrémente le champ *Segments Left*

Problèmes de sécurité $\Rightarrow$ obsolète

Principes – Routage explicite

Exemple : A envoie un paquet à B en spécifiant les routeurs intermédiaires R_0, R_1, ..., R_n.
Note : le champ *Hdr Ext Len* vaut 2(n+1).

En-tête IPv6	En-tête de routage	
Dest. Addr.	Segments Left	0 1 ... n-1 n

Initialement, A envoie le paquet :

R_0	n+1	R_1 R_2 ... R_n B

Lorsque R_0 le reçoit, il transmet :

R_1	n	R_0 R_2 ... R_n B

Lorsque R_1 le reçoit, il transmet :

R_2	n-1	R_0 R_1 ... R_n B

etc. Puis R_{n-1} transmet :

R_n	1	R_0 R_1 ... R_{n-1} B

Lorsque R_n le reçoit, il transmet :

B	0	R_0 R_1 ... R_{n-1} R_n

Principes – Tunneling

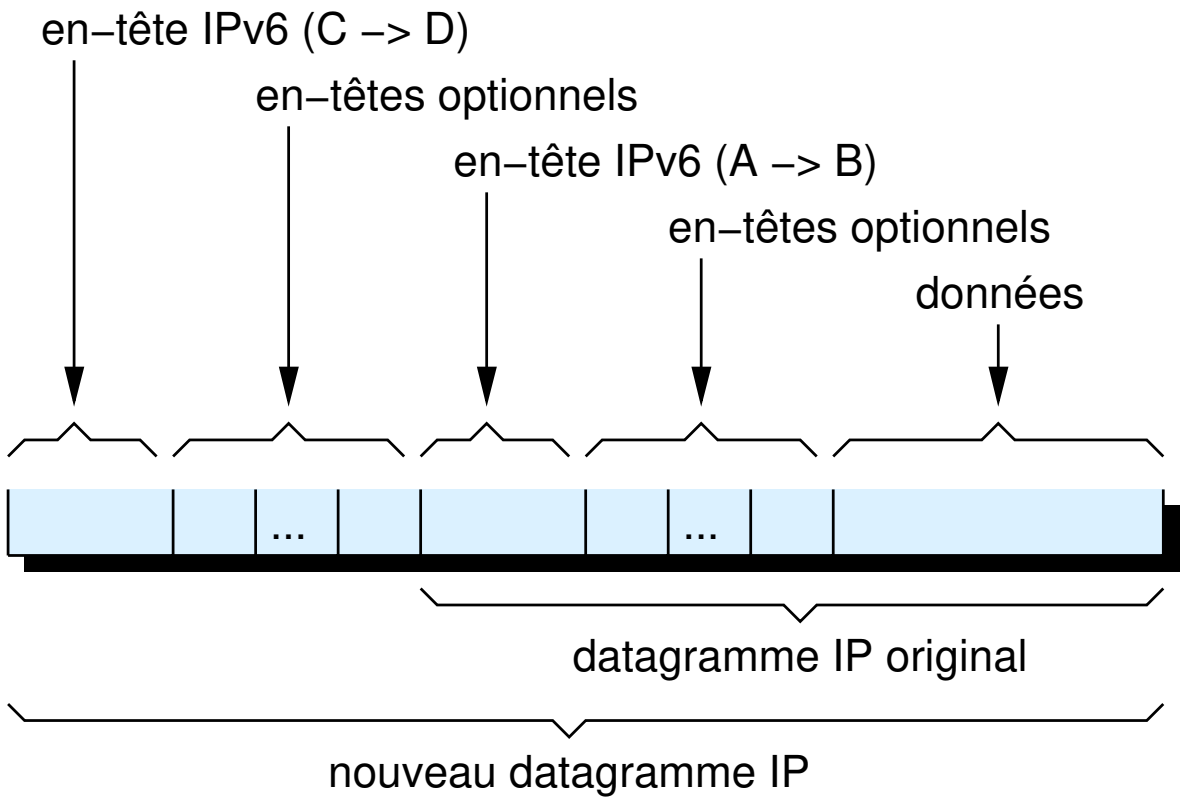
Tunneling : transport d'un protocole par un autre protocole de même niveau.

Idées :

- autre forme de routage explicite
- transit par des machines coupe-feu
- réseau « privé » par dessus un réseau « public »
- support des « mobiles »
- possibilité de routage fin
  $\Rightarrow$ ex: http par là, trafic ssh par ici

Principes – Tunneling

Avec IPv6 : encapsulation d'un paquet IPv6 ou IPv4 (de A vers B) dans un autre paquet IPv6 (de C vers D).
Le champ *Next Header* du dernier en-tête du nouveau paquet contient l'identification de *IPv6* ou *IPv4*.



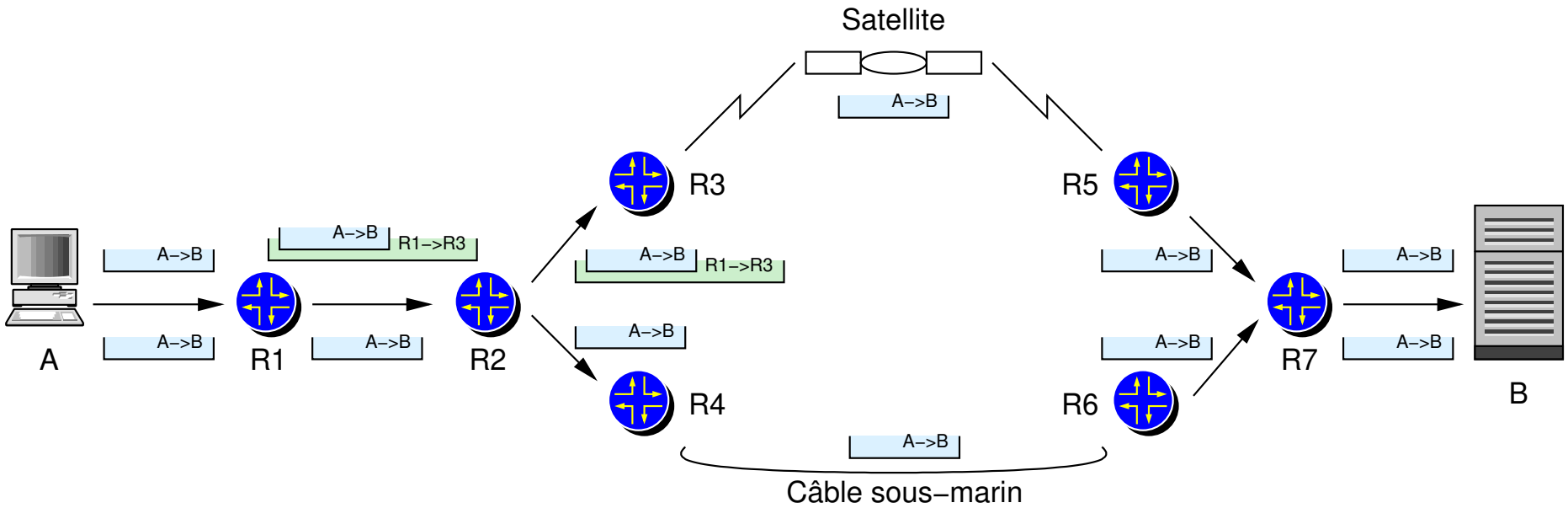
Principes – Tunneling

Possibilité de routage fin :

- pour différencier les classes de trafic
- effectué par un nœud intermédiaire
  (le routage explicite est demandé par la source)

Exemple :

- trafic interactif par câble sous-marin
- trafic *batch* par satellite
- R1 décide de la route à utiliser



Principes – Qualité de service

Qualité de service basée sur deux champs dans l'en-tête IPv6 :

- *Traffic Class* : classe de trafic (interactif, non interactif, résiduel, etc.) demandée par l'émetteur
- *Flow Label* : une application marque un flot de données
  $\Rightarrow$ en cas de choix, les routeurs tentent d'utiliser le même chemin pour ne pas désordonner les paquets

Plan

Adresses

Trois types d'adresses IPv6 sur 128 bits :

- adresses *unicast*
  identifie une interface
- adresses *anycast*
  identifie un ensemble d'interfaces (sur un ou plusieurs nœuds)
  un paquet est envoyé à l'interface « la plus proche »
- adresses *multicast*
  identifie un groupe d'interfaces
  un paquet est envoyé à toutes les interfaces

$\Rightarrow$ pas de *broadcast* avec IPv6

Adresses

Trois formats de représentation des adresses IPv6 :

- Format canonique :
  `FEDC:BA98:7654:3210:FEDC:BA98:7654:3210`
  `1080:0:0:0:8:800:200C:417A`
- Format compressé :
  `FF01::43` ($\Leftrightarrow$ `FF01:0:0:0:0:0:0:43`)
  `::1` ($\Leftrightarrow$ `0:0:0:0:0:0:0:1`)
- Format IPv4 translaté en v6 :
  `0:0:0:0:0:FFFF:198.51.100.1` ($\Leftrightarrow$ `::FFFF:198.51.100.1`)
- Format compatible IPv4 (obsolète) :
  `0:0:0:0:0:0:198.51.100.1` ($\Leftrightarrow$ `::198.51.100.1`)

Adresses

128 bits, est-ce assez ?

Point de vue théorique :

- $2^{128} \approx 3 \times 10^{38}$ adresses
- 6×10^{22} adresses par mètre carré sur la Terre

Point de vue pratique (en tenant compte des hiérarchies imposées par le routage et par la fragmentation de l'espace d'adresses) :

- entre 10^{17} et 10^{33} adresses
- entre 1564 et 3×10^{18} adresses par mètre carré sur la Terre

Adresses

Répartition de l'espace d'adressage, selon les bits de poids fort :

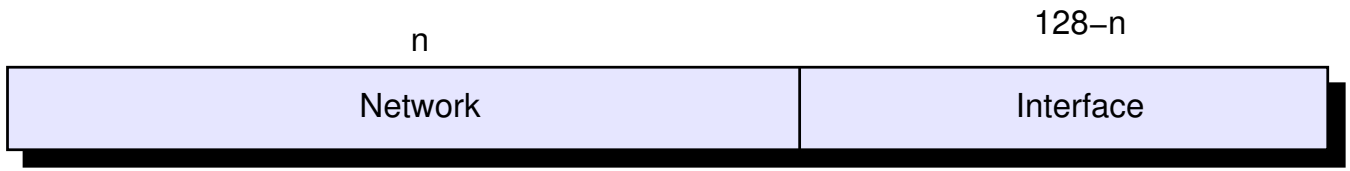
Préfixe	Notation	Type
0000 ... 0000	::0/128	Adresse non spécifiée
0000 ... 0001	::1/128	Adresse de loopback
1111 110	fc00::/7	Adresses unicast « privées »
1111 1111	ff00::/8	Adresses multicast
1111 1110 10	fe80::/10	Adresses unicast locales au lien
Tout le reste		Adresses unicast globales

Les adresses *anycast* font partie des adresses unicast

Adresses – Adresses unicast

Schéma d'allocation analogue aux adresses IPv4 pour les adresses unicast globales :

- préfixe de réseau sur *n* bits
- numéro d'interface sur 128 − *n* bits

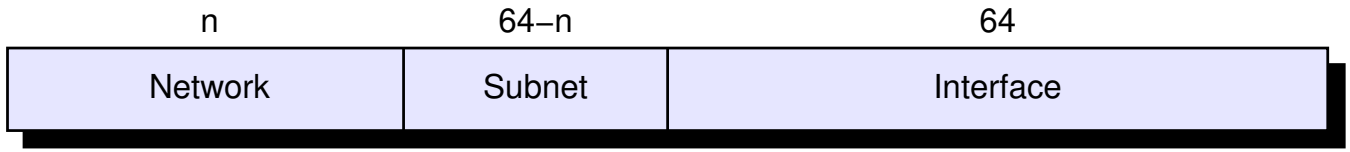


Exemples :

- `2001:0db8:1234:5678::0/127` et `...::1/127` pour les routeurs aux extrêmités d'une liaison spécialisée
- `2001:0db8:8765:4321::1/64` pour la première machine sur le réseau `2001:0db8:8765:4321::/64`

Adresses – Adresses unicast

Le plus souvent :



- Network : préfixe pour l'entité
- Subnet : numéro de sous-réseau défini par l'entité
- Interface : identificateur d'interface sur 64 bits

Adresses – Adresses unicast

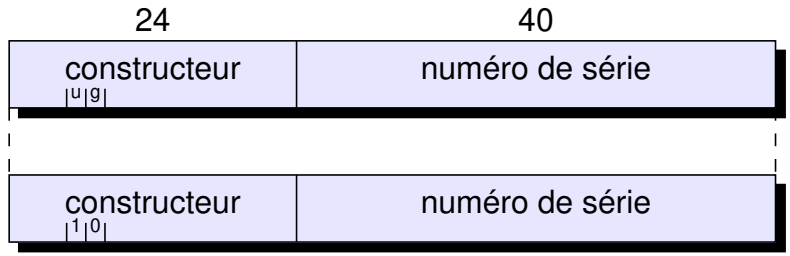
Exemple :

- `2001:660:4701::/48` alloué par Renater à l'Université de Strasbourg
- `2001:660:4701:f034::/64` alloué par l'Université de Strasbourg à l'équipe « Réseaux » du laboratoire ICube
- `2001:660:4701:f034:226:55ff:fe1f:66e8::/64` est l'adresse d'un serveur sur ce réseau

Adresses – Identifiant d'interface

Comment attribuer un identifiant d'interface ?

1. utiliser une valeur arbitraire : configuration manuelle
   Exemple : 1, 2, etc.
2. utiliser l'identifiant IEEE EUI64 s'il existe



Notes :

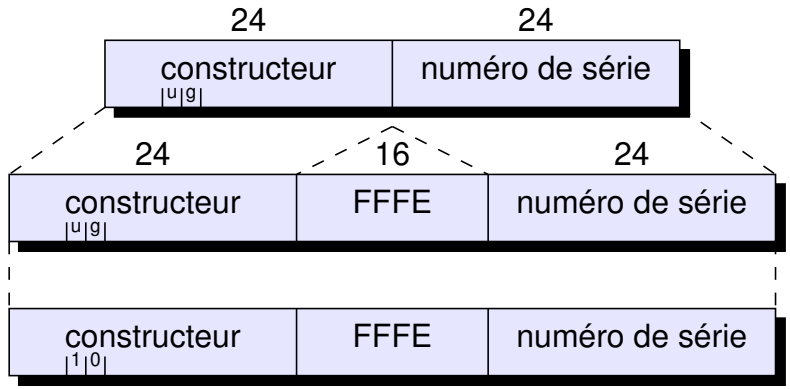
- u = universel/local (adresse MAC attribuée localement)
- g = global (interface unique ou adresse multicast)

$\Rightarrow$ peu utilisé en pratique

Adresses – Identifiant d'interface

Comment attribuer un identifiant d'interface ?

3. interface avec une adresse MAC (IEEE 802) sur 48 bits :



Note :

- Problème de vie privée : même identifiant d'interface quel que soit le réseau, on peut donc me suivre à la trace lors de mes déplacements
- $\Rightarrow$ méthode obsolète

Adresses – Identifiant d'interface

Comment attribuer un identifiant d'interface ?

4. choisir un nombre aléatoire (RFC 4941)
   ... et en changer souvent... (ex : toutes les 10 min)

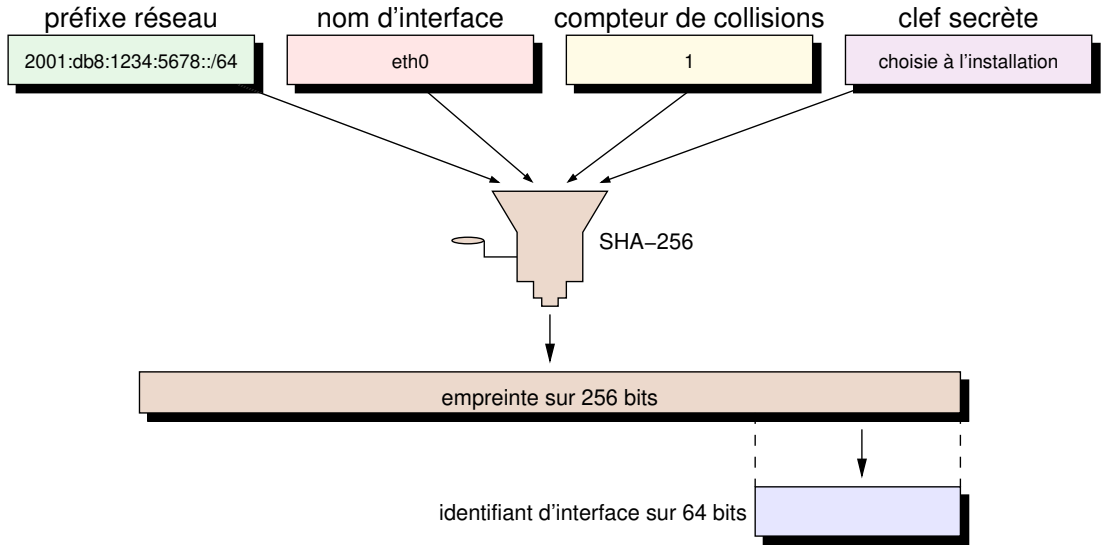
   Note :

   - difficile à tracer...
   - ... y compris pour l'administrateur réseau
     $\Rightarrow$ difficulté de débug
   - remplit les tables d'état des pare-feux
   - mode par défaut sur certains systèmes

Adresses – Identifiant d'interface

Comment attribuer un identifiant d'interface ?

5. déterminer un identifiant « stable » (RFC 7217)
   Par exemple :



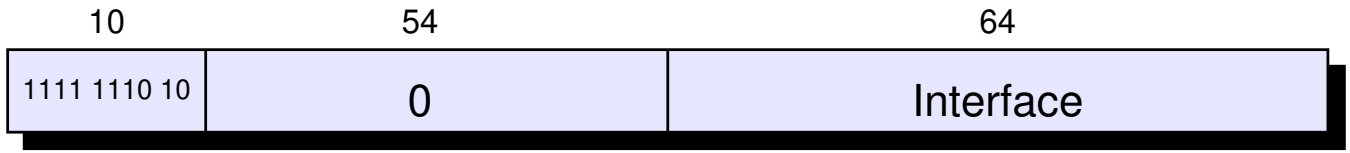
⇒ dépend du préfixe, mais ne varie pas dans un réseau

⇒ non traçable entre réseaux

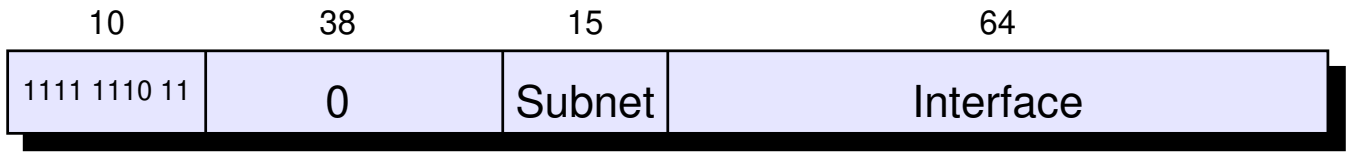
⇒ méthode désormais recommandée (RFC 8064)

Adresses – Adresses unicast

Adresses locales au lien : ne peuvent être utilisées que sur un seul câble (pas de routage)



Adresses locales au site (obsolètes) : ne peuvent être utilisées qu'à l'intérieur d'un site (pas de routage à l'extérieur)



Adresses – Adresses unicast

Adresses spéciales :

- adresse non spécifiée :
  `0:0:0:0:0:0:0:0`
  ⇒ lorsqu'un nœud ne connaît pas encore son adresse
- adresse de *loopback* :
  `0:0:0:0:0:0:0:1`
  ⇒ pour s'envoyer des paquets
- adresses IPv4 :
  `::FFFF:`*`a.b.c.d`*
  ⇒ transition IPv4 - IPv6

Adresses – Adresses anycast

Adresse *anycast* = adresse *unicast* utilisée pour identifier un ou plusieurs nœuds ou interfaces
$\Rightarrow$ le routeur choisit la plus « proche » interface ayant cette adresse.

Utilisations possibles :

- router un paquet via une organisation sans spécifier l'adresse précise d'un routeur
- atteindre l'un des routeurs à la frontière d'un domaine (sous-réseau, site, fournisseur, etc.)
- accéder au serveur de messagerie local
- équilibrer la charge entre serveurs WWW
- etc.

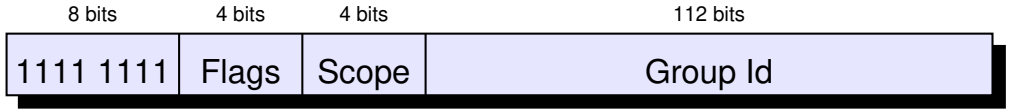
Adresses – Adresses anycast

Exemple d'adresse *anycast* : tous les routeurs connectés sur le même lien

$\Rightarrow$ tous les bits du numéro d'interface à 0

Adresses – Adresses multicast

Une adresse multicast identifie un groupe d'interfaces



- *Flags* : un seul défini
  - T = 0 : adresse permanente (officielle)
  - T = 1 : adresse temporaire
- *Scope* : domaine de visibilité de l'adresse

Scope	Visibilité
1	nœud
2	lien
5	site
8	organisation
E	global

- *Group Id* : numéro du groupe multicast

Adresses – Adresses multicast

Adresses multicast prédéfinies :

Adresse	Signification
FF01::1	Toutes les interfaces sur un même nœud
FF02::1	Toutes les machines sur un même lien (broadcast IPv4)
FF01::2	Toutes les interfaces sur un même routeur
FF02::2	Tous les routeurs sur un même lien
FF05::2	Tous les routeurs dans un même site

Adresse spéciale : *adresse de sollicitation*

FF02:0:0:0:0:1:FF*xx*:*xxxx*

où *xxxxxx* correspond aux 24 bits de poids faible de l'adresse IPv6 cherchée. Voir ICMPv6

Plan

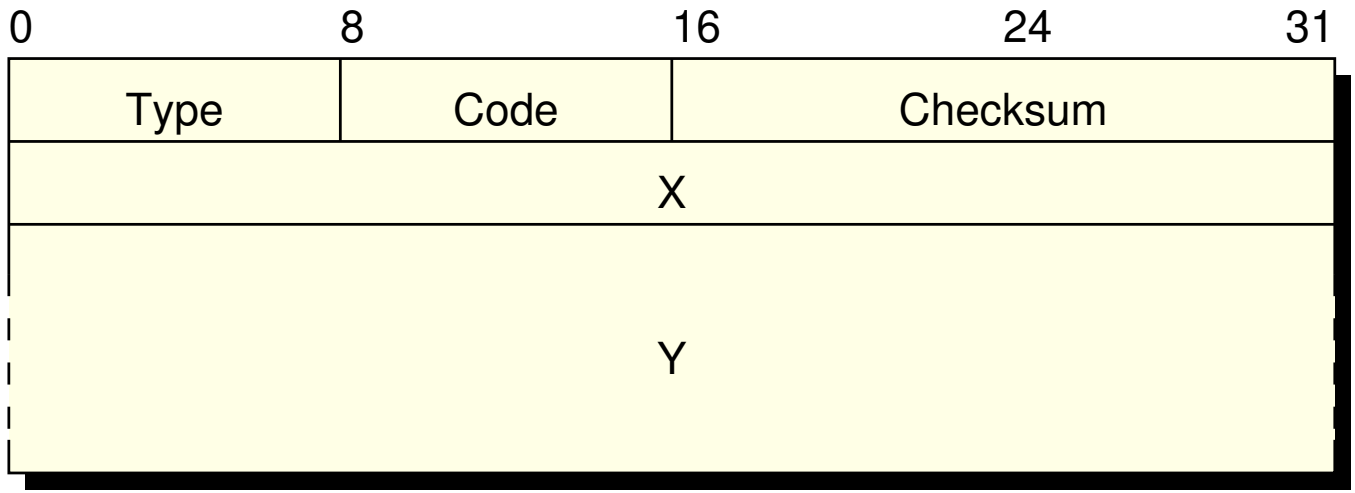
Protocole de contrôle – ICMPv6

Extension de ICMP pour IPv6 :

- simple adaptation pour les messages « classiques »
  - Echo Request / Echo Reply
  - Destination Unreachable
  - Time Exceeded
  - Packet Too Big
  - Parameter Problem
  - Redirect
- nouvelles fonctionnalités : découverte des voisins
  $\Rightarrow$ découverte d'informations sur l'environnement (MTU du lien, routeurs, préfixe de l'adresse, etc.)

Protocole de contrôle – ICMPv6

Simple adaptation pour les messages « classiques » :



Avec :

Type	**X**	**Y**
Dest. Unreach.	Inutilisé	Début paquet original
Packet Too Big	MTU	Début paquet original
Time Exceeded	Inutilisé	Début paquet original
Param. Problem	Pointeur	Début paquet original
Echo	Id + Num. séq.	Données
Redirect	Inutilisé	Adr. dest. et routeur

Protocole de contrôle – ICMPv6

Nouvelles fonctionnalités : « découverte des voisins »

- découverte des routeurs
- découverte du préfixe
- découverte des paramètres du lien (MTU, Hop limit)
- mode d'auto-configuration des adresses
- résolution d'adresses (ARP)
- découverte du routeur pour une destination
- détection des voisins inatteignables
- détection de duplication d'adresse

Protocole de contrôle – ICMPv6

Principes :

- chaque routeur diffuse périodiquement un message *Router Advertisement* qui peut contenir :
  - le préfixe du lien
  - la valeur du *Hop Limit*
  - la valeur du MTU
  - la manière de faire l'autoconfiguration
  - etc.
- un nœud peut solliciter un message *Router Advertisement* en envoyant un message *Router Solicitation* à l'adresse `FF02::2`

D'où :

- découverte des routeurs et du routeur par défaut
- découverte du préfixe
- découverte des paramètres du lien (MTU, Hop limit)
- mode d'auto-configuration des adresses

Protocole de contrôle – ICMPv6

Principes (suite) :

- un nœud qui veut connaître l'adresse MAC d'un autre envoie un message *Neighbor Solicitation* à l'adresse de *sollicitation* (adresses multicast spéciales)
- le nœud destinataire répond par un message *Neighbor Advertisement* à l'adresse du nœud émetteur.

D'où :

- résolution d'adresses (plus besoin de ARP)
- détection de duplication d'adresse (message envoyé, par l'adresse source `0::0`, à l'adresse spéciale de sollicitation pour l'adresse à tester)
- annonce de changement d'adresse MAC

Plan

Autoconfiguration

Idée : *plug'n play*
$\Rightarrow$ aucune intervention manuelle
Indispensable :

- utilisation massive d'IP par des néophytes
  $\Rightarrow$ ex: postes de télévision, cafetières, automobiles
- grands volumes de machines à configurer
  $\Rightarrow$ réduire le travail manuel
- renumérotation des adresses IP d'un site
  $\Rightarrow$ ex : changement de fournisseur IP

Autoconfiguration

Deux modes d'autoconfiguration :

- mode « sans état » (*stateless*), ou « autonome »
  Comment : adresse MAC + écoute de paquets
- mode « avec état » (*stateful*)
  Comment : demander à un serveur DHCPv6

Choix entre les deux modes :

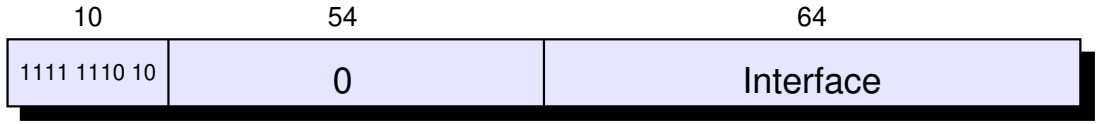
- sur décision de l'administrateur du nœud
- sur réception d'un message ICMP *Router Advertisement*
  $\Rightarrow$ choix par l'administrateur du routeur
- par défaut : de manière autonome
  $\Rightarrow$ choix implicite

Les deux modes ne sont pas forcément exclusifs : par exemple, une autoconfiguration avec état peut commencer par une autoconfiguration sans état

Autoconfiguration – sans état

Principe :

1. fabrication d'une adresse locale au lien :



2. vérification de l'unicité de l'adresse
   - 2.1 envoi d'un message *Neighbor Sollicitation* :
     - source = `0::0`
     - destination = adresse de sollicitation
     - adresse à tester = adresse fabriquée
   - 2.2 si retour de *Neighbor Advertisement*
     $\Rightarrow$ au secours ! configuration manuelle...
   - 2.3 si pas retour de *Neighbor Advertisement* (1 s)
     $\Rightarrow$ l'adresse n'est pas utilisée
3. découverte du préfixe de site
   $\Rightarrow$ réception d'un message *Router Advertisement*
   Si pas de *Router Advertisement*, ou si *Router Advertisement* spécifie une configuration « avec état » $\Rightarrow$ tenter le mode « avec état »

Autoconfiguration – sans état

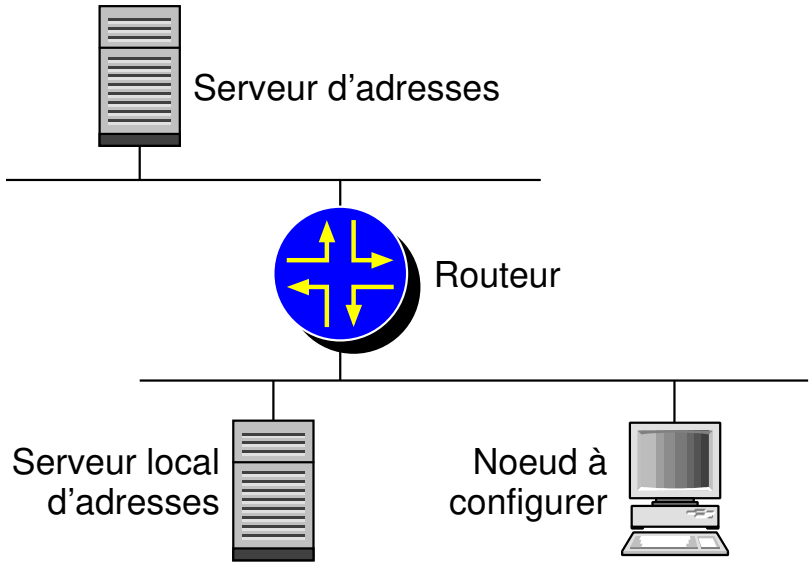
Durée de vie d'une adresse :

- la durée de vie est spécifiée dans le message *Router Advertisement*
- lorsque l'adresse devient « dépréciée », elle peut continuer à être utilisée dans les communications existantes...
- ... mais une nouvelle adresse doit être obtenue, qui deviendra l'adresse « de préférence »
- lorsque l'adresse « dépréciée » devient « invalide », elle ne peut plus être utilisée

$\Rightarrow$ facilite la renumérotation des sites

Autoconfiguration – avec état

Utilisation de DHCPv6 :



- serveur local d'adresses :
  - au moins un par câble
  - retourne une adresse globale ou transmet la requête à un autre serveur
  - probablement couplé avec le routeur
- serveur d'adresses : tâches plus complexes (ajout dans le DNS, allocation des numéros, etc.)

Autoconfiguration – avec état

DHCPv6 = simplification de DHCPv4

- le client dispose d'une adresse valide
  $\Rightarrow$ le serveur n'a pas à faire de *broadcast*
- pas besoin de compatibilité avec BOOTP
  $\Rightarrow$ simplification de quelques champs
- support de plusieurs adresses par interface
  $\Rightarrow$ dépréciation des adresses
- utilisation de la « découverte des voisins » en ICMPv6
  $\Rightarrow$ simplification d'options
- etc.

Plan

Mobilité

1. le mobile a une adresse A (*home address*) dans son organisation
2. lorsque le mobile se déplace, il se procure une adresse L (*care-of address*) dans le contexte local
   $\Rightarrow$ configuration automatique ou manuelle
3. le mobile enregistre L auprès d'un routeur (*home agent*) de son organisation
4. le routeur intercepte les paquets pour A, et les encapsule (avec un tunel) dans un paquet pour L
5. lorsque le mobile reçoit un paquet encapsulé, il communique à l'émetteur son adresse L
6. lorsque l'émetteur reçoit l'adresse L, il met cette adresse dans un cache local et commence à utiliser L à la place de A

Mobilité

Implémentation : 4 nouvelles options (examinées uniquement par le destinataire)

- *Binding Update*
  Émise par le mobile pour informer le *home agent*, ou le correspondant, de l'association (A, L). Permet de spécifier la durée de vie de l'association.
  Émise également par le mobile pour annuler une association (si la durée de vie = 0).
- *Binding Acknowledgement*
  Émise suite à la réception d'un *Binding Update*
- *Binding Request*
  Émise par le correspondant pour demander un *Binding Update* pour rafraîchir son cache d'associations.
- *Home Address*
  Émise par le mobile, lors des nouvelles connexions, pour indiquer au correspondant qu'il doit utiliser l'adresse A et non L (source du paquet).

Plan

DNS

Le DNS doit intégrer les requêtes :

- nom symbolique $\rightarrow$ adresse IPv6
  Problème : une adresse IPv6 ne peut être incluse dans un enregistrement `A` $\Rightarrow$ limités à 32 bits
- adresse IPv6 $\rightarrow$ nom symbolique
  Problème : une adresse IPv6 est représentée par une suite de mots de 16 bits en hexadécimal

DNS – Conversion

IPv4 : enregistrement de type `A`
IPv6 : définition d'un nouveau type d'enregistrement :

```
vagabond IN AAAA 2001:db8:1234:5678:b80f:8fd8:22f:c16
```

$\Rightarrow$ pas mutuellement exclusif avec l'ancien type `A`

Attention : les requêtes `NS` et `MX` qui cherchent en plus les enregistrements `A` doivent aussi chercher maintenant les enregistrements `AAAA`

DNS – Conversion inverse

IPv4 : utilisation du domaine `in-addr.arpa`
IPv6 :

- utilisation du domaine `ip6.arpa`
- décomposition en quartets ($\Rightarrow$ facilite la délégation)

Exemple : pour chercher le nom associé à l'adresse `123::4567:89AB`,
il faut chercher l'enregistrement de type `PTR` associé à :
`B.A.9.8.7.6.5.4.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.2.1.0.IP6.ARPA.`

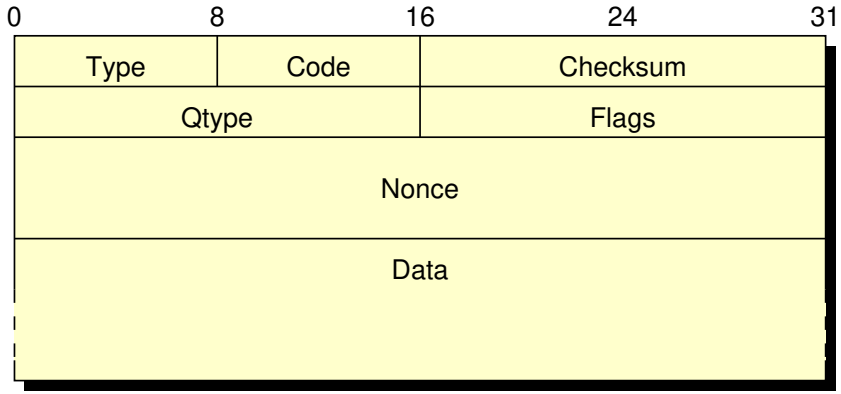
DNS – Conversion inverse – RFC 4620

Alternative : demander directement à l'intéressé...
Intérêts :

- pas de maintenance de `ip6.arpa`
  $\Rightarrow$ pas d'erreurs de maintenance de `ip6.arpa`
  $\Rightarrow$ pas d'administrateur
- pas de saturation des serveurs de noms de `ip6.arpa`
- pas de frontière de délégation
  (8 bits avec IPv4, 4 avec IPv6)

DNS – Conversion inverse – RFC 4620

Méthode : nouveau message ICMPv6



Avec :

- *Qtype* : type de requête

1	inutilisé
2	obtenir le nom
3	obtenir tout ou partie des adresses IPv6
4	obtenir tout ou partie des adresses IPv4

- *Nonce* : valeur aléatoire choisie par le requêteur
- *Data* : les valeurs demandées, avec un TTL éventuel

$\Rightarrow$ statut expérimental $\Rightarrow$ pas ou peu utilisé

Plan

Transition

Le premier janvier 201x, on bascule tout l'Internet en IPv6
$\Rightarrow$ « au fou ! »

Problème : impossible en une période limitée

Solution : schéma de transition pour faire coexister les nœuds IPv4 avec des nœuds IPv6/IPv4.

- déploiement immédiat de nœuds sous IPv6
- passage progressif de sites sous IPv6
- test facile
- expérience (apprentissage) progressive
- pas d'interdépendances entre les phases

Méthode : les nouveaux nœuds comprennent IPv6 et IPv4.

Transition

Plusieurs phases :

- tout l'Internet est IPv4
- nœuds IPv4 et nœuds IPv6/IPv4
- nœuds IPv4, nœuds IPv6/IPv4 et nœuds IPv6
- tout l'Internet est IPv6

Il y aura encore longtemps des sites IPv4 !

Transition – Adresses

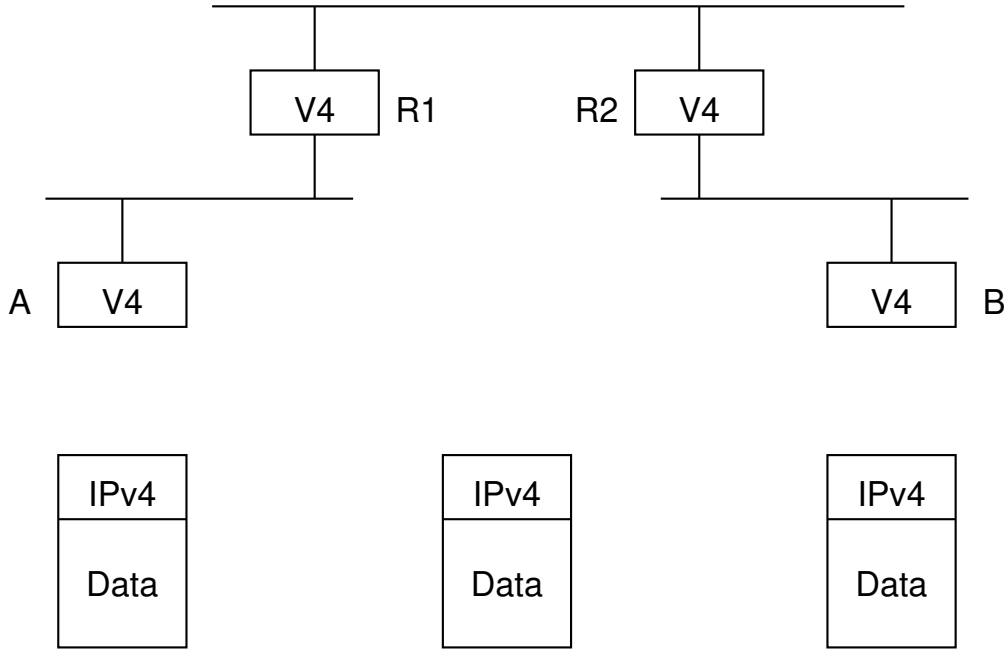
Adresse annoncée par le DNS ⇒ reflète la volonté de communiquer avec des nœuds IPv4

- `0:0:0:0:0:0000:a.b.c.d`
  Adresses IPv4 sous forme IPv6
  ⇒ DNS : enregistrement de type `A` seulement
- `0:0:0:0:0:FFFF:a.b.c.d`
  Adresses IPv4 de nœuds IPv6/IPv4 ou IPv6
  ⇒ DNS : enregistrement de type `A`
  ⇒ DNS : enregistrement de type `AAAA`
  ⇒ communication directe si nœud IPv6/IPv4
  ⇒ communication via un routeur traducteur si nœud IPv6
- autres adresses : adresses IPv6 seulement
  ⇒ DNS : enregistrement de type `AAAA` seulement
  ⇒ pas de communication possible avec IPv4

Important : un nœud peut avoir plusieurs adresses sur une interface
⇒ adresse compatible IPv4 en particulier

Transition – Phases

Utilisation d'IPv4 de bout en bout :

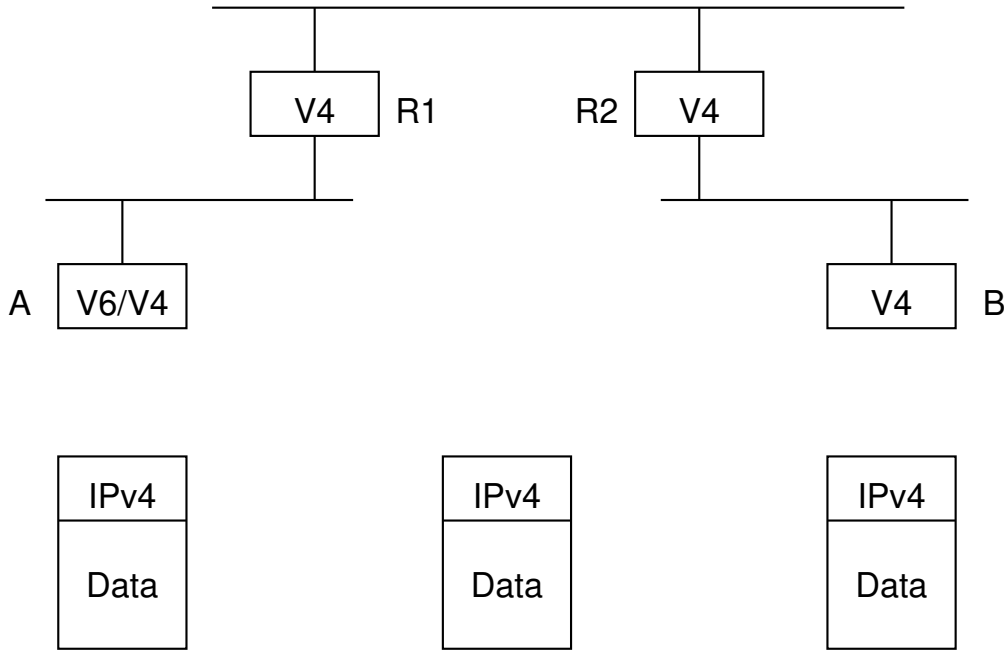


Pour envoyer un paquet de A vers B :

- A construit un paquet IPv4
- A envoie au routeur R1 un paquet IPv4
- routeur R1 envoie au routeur R2 un paquet IPv4
- routeur R2 envoie à B un paquet IPv4

Transition – Phases

Un nœud IPv4 change de version de système
$\Rightarrow$ il devient IPv6/IPv4

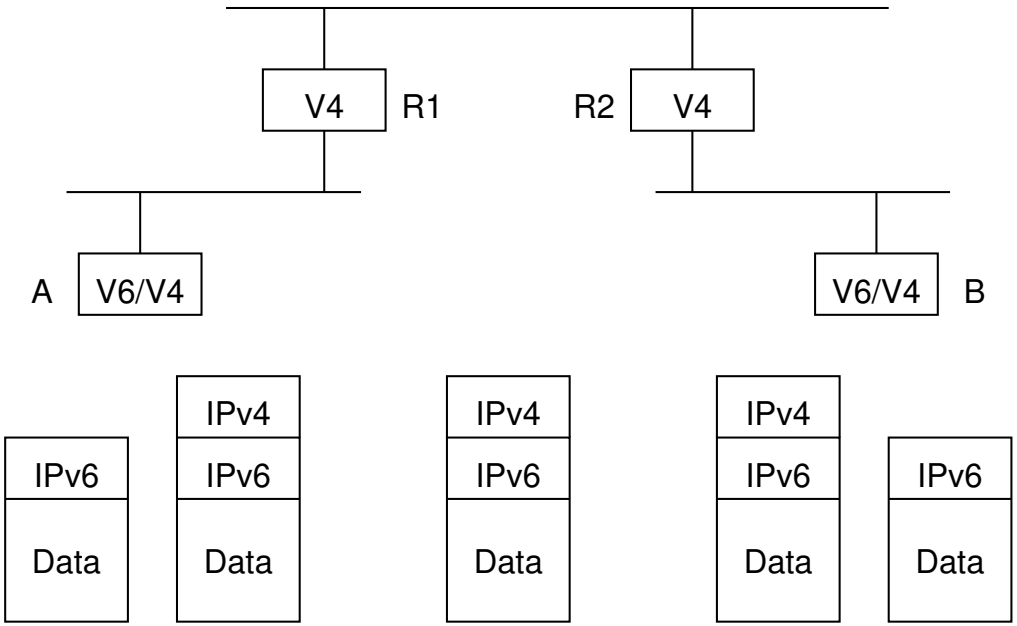


Pour envoyer un paquet de A vers B :

- B est IPv4 $\Rightarrow$ A construit un paquet IPv4
- le paquet IPv4 arrive jusqu'à B

Transition – Phases

Deux nœuds IPv6/IPv4 sont séparés par une zone IPv4 :

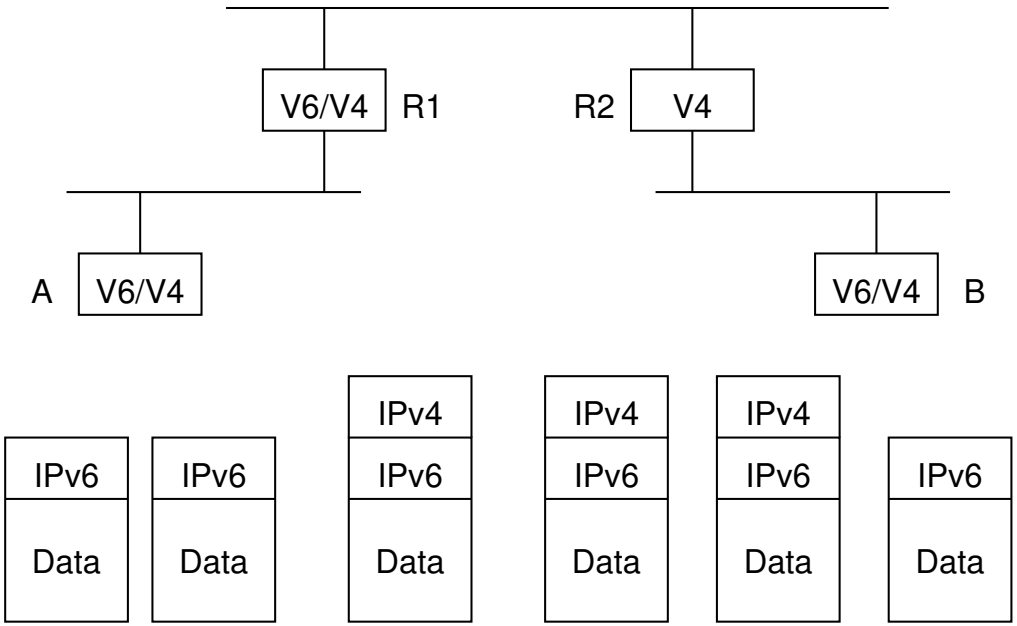


Pour envoyer un paquet de A vers B :

- B est IPv6 $\Rightarrow$ A construit un paquet IPv6
- R1 est IPv4 $\Rightarrow$ A encapsule dans IPv4
- le paquet IPv4 arrive jusqu'à B
- B décapsule le paquet IPv6

Transition – Phases

Il ne reste plus qu'un routeur IPv4 :

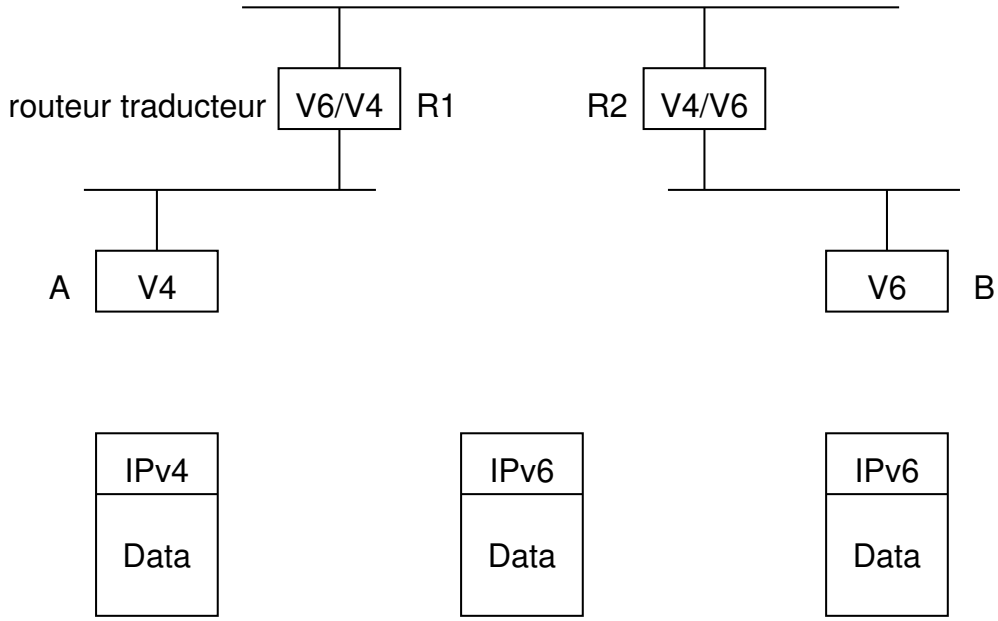


Pour envoyer un paquet de A vers B :

- B est IPv6 $\Rightarrow$ A construit un paquet IPv6
- le paquet IPv6 arrive jusqu'à R1
- R2 est IPv4 $\Rightarrow$ R1 encapsule dans IPv4
- B décapsule le paquet IPv6

Transition – Phases

Communication entre un nœud IPv4 et un nœud IPv6 seulement :



Pour envoyer un paquet de A vers B :

- A envoie un paquet IPv4
- R1 le traduit en paquet IPv6
- le paquet IPv6 arrive jusqu'à B

Schéma réciproque pour B vers A

Transition – Phases

Synthèse :

	IPv4 seule-ment	IPv6/IPv4 avec adr IPv4	IPv6/IPv4 sans adr IPv4	IPv6 avec adr IPv4	IPv6 sans adr IPv4
IPv4 seulement	D	D	N	T	N
IPv6/v4 avec adr. IPv4	D	D	D	D	D
IPv6/v4 sans adr. IPv4	N	D	D	D	D
IPv6 avec adr. IPv4	T	D	D	D	D
IPv6 sans adr. IPv4	N	D	D	D	D

D = interopérabilité directe

T = utilisation d'un routeur traducteur

N = pas d'interopérabilité possible

Conclusion

IPv6 : changement majeur

- nouvelles fonctionnalités très importantes...
- ... intégrées depuis à IPv4
- la plupart des systèmes et des routeurs sont prêts à migrer depuis longtemps...
- ... grande « résistance » d'IPv4

IPv6 doit durer quelques décennies...
Mais c'est lent à démarrer