MOOC
Objectif IPv6 !
vers l'internet nouvelle génération

Document Compagnon[1]

Séquence 2

Le protocole IPv6

1 Le contenu de ce document d'accompagnement du MOOC IPv6 est publié sous Licence Creative Commons **CC BY-SA 4.0 International**.

Licence Creative Nommons CC BY-SA 4.0 International

Attribution - Partage dans les Mêmes Conditions 4.0 International (CC BY-SA 4.0)

Avertissement Ce résumé n'indique que certaines des dispositions clé de la licence. Ce n'est pas une licence, il n'a pas de valeur juridique. Vous devez lire attentivement tous les termes et conditions de la licence avant d'utiliser le matériel licencié.

Creative Commons n'est pas un cabinet d'avocat et n'est pas un service de conseil juridique. Distribuer, afficher et faire un lien vers le résumé ou la licence ne constitue pas une relation client-avocat ou tout autre type de relation entre vous et Creative Commons.

Clause C'est un résumé (et non pas un substitut) de la licence.

http://creativecommons.org/licenses/by-sa/4.0/legalcode

Vous êtes autorisé à :

- **Partager** — copier, distribuer et communiquer le matériel par tous moyens et sous tous formats
- **Adapter** — remixer, transformer et créer à partir du matériel
- pour toute utilisation, y compris commerciale.

L'Offrant ne peut retirer les autorisations concédées par la licence tant que vous appliquez les termes de cette licence.

Selon les conditions suivantes :

Attribution — You must give **appropriate credit**, provide a link to the license, and **indicate if changes were made**. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

Partage dans les Mêmes Conditions — Dans le cas où vous effectuez un remix, que vous transformez, ou créez à partir du matériel composant l'Oeuvre originale, vous devez diffuser l'Oeuvre modifiée dans les même conditions, c'est à dire avec **la même licence** avec laquelle l'Oeuvre originale a été diffusée.

No additional restrictions — Vous n'êtes pas autorisé à appliquer des conditions légales ou des **mesures techniques** qui restreindraient légalement autrui à utiliser l'Oeuvre dans les conditions décrites par la licence.

Notes: Vous n'êtes pas dans l'obligation de respecter la licence pour les éléments ou matériel appartenant au domaine public ou dans le cas où l'utilisation que vous souhaitez faire est couverte par une **exception.**

Aucune garantie n'est donnée. Il se peut que la licence ne vous donne pas toutes les permissions nécessaires pour votre utilisation. Par exemple, certains droits comme l**es droits moraux, le droit des données personnelles et le droit à l'image** sont susceptibles de limiter votre utilisation.

Les informations détaillées sont disponibles aux URL suivantes :

- http://creativecommons.org/licenses/by-sa/4.0/deed.fr
- http://fr.wikipedia.org/wiki/Creative_Commons

Les auteurs

Bruno Stévant

Bruno STEVANT est enseignant chercheur à l'IMT Atlantique. Il intervient dans l'enseignement et sur les projets de recherche autour d'IPv6 depuis plus de 10 ans. Il est secrétaire et responsable des activités de formation de l'association G6, association pour la promotion et le déploiement d'IPv6 en France.

Jacques Landru

Enseignant chercheur au département Informatique et Réseaux à l'IMT Lille Douai, Jacques est responsable de l'UV de spécialisation ARES (Architecture des RESeaux) à la fois dans le mode traditionnel présentiel que dans sa forme à distance dans le cadre du cursus diplômant TutTelNet.

Jean-Pierre Rioual

Ingénieur Conseil Réseaux – EURÊKOM. Fort de 30 années d'expérience dans le domaine des réseaux, il intervient auprès des entreprises pour des missions d'expertise sur leurs réseaux de transmission de données (intégration, mesures, optimisation, administration), conçoit et anime des actions de formation "réseaux".

Pascal Anelli

Pascal ANELLI est enseignant-chercheur à l'Université de la Réunion. Il enseigne les réseaux depuis plus 20 ans. Il est membre du G6 depuis sa création. A ce titre, il est un des contributeurs du livre IPv6. En 1996, il a participé au développement d'une version de la pile IPv6 pour Linux.

Joël Grouffaud

Joël GROUFFAUD est professeur agrégé de mathématiques. Il est chef du département Réseaux et Télécommunications de l'IUT de la Réunion, une composante de l'université de La Réunion. Au sein du département, il enseigne les réseaux et IPv6. Il anime l'académie Cisco (formations CCNA) de La Réunion.

Pierre Ugo TOURNOUX

Pierre Ugo TOURNOUX est enseignant chercheur à l'Université de la Réunion. Il est responsable des enseignements d'administration réseau, de routage et des réseaux sans fil dans lesquels il intègre IPv6 depuis de nombreuses années.

Remerciements à :

- Vincent Lerouvillois, pour son travail de relecture attentive ;
- Bruno Di Gennaro (Association G6) ;
- Bruno Joachim (Association G6) pour sa contribution à l'activité « Contrôler la configuration réseau par DHCPv6 » ;
- Richard Lorion (Université de la Réunion) pour sa contribution à l'activité « Etablir la connectivité IPv6 tunnels pour IPv6 ».

Table des activités

Activité 21: Le format de l'en-tête IPv6

Introduction

Comme rappelé en introduction de la séquence, la fonction principale du protocole IP est d'acheminer un paquet d'un point à un autre de l'Internet. Le modèle datagramme impose que chaque paquet soit traité indépendamment, sans avoir besoin d'informations issues d'autres paquets déjà transmis. L'en-tête IP doit donc comporter toutes les informations pour réaliser cet acheminement. C'est la raison pour laquelle chaque paquet doit contenir l'adresse de l'émetteur ainsi que du destinataire du paquet. Ces paquets auto-descriptifs sont appelés des datagrammes.

L'adresse du destinataire est fondamentale pour les noeuds intermédiaires réalisant la fonction de routage. C'est en effet à partir de cette adresse que le noeud décide vers quelle interface le paquet sera remis. La lecture de l'adresse du destinataire dans l'en-tête IP est donc une étape cruciale pour la performance globale de l'acheminement du paquet. Afin d'optimiser cette étape, deux principes ont été appliqués dans la spécification de l'en-tête IP:

- Une adresse IP est de taille fixe, permettant ainsi d'être récupérée dans l'en-tête directement en lisant un nombre de bits prédéterminé, sans avoir à lire ni interpréter au préalable un champ de longueur d'adresse;
- L'adresse IP destination est à un emplacement fixe dans l'en-tête, emplacement aligné en mémoire, facilitant ainsi son extraction de l'en-tête par un simple décalage en mémoire, qui est une opération optimisée au niveau matériel.

Format de l'en-tête du datagramme IPv6

Le format d'en-tête du datagramme IPv6 est spécifié par le RFC 8200 (page 5). Cet en-tête, avec les champs le composant, est représenté par la figure 1. L'en-tête IPv6 est de taille fixe et se compose de 5 mots de 64 bits (contre 5 mots de 32 bits pour IPv4). La taille de l'en-tête IPv6 est donc de 40 octets.

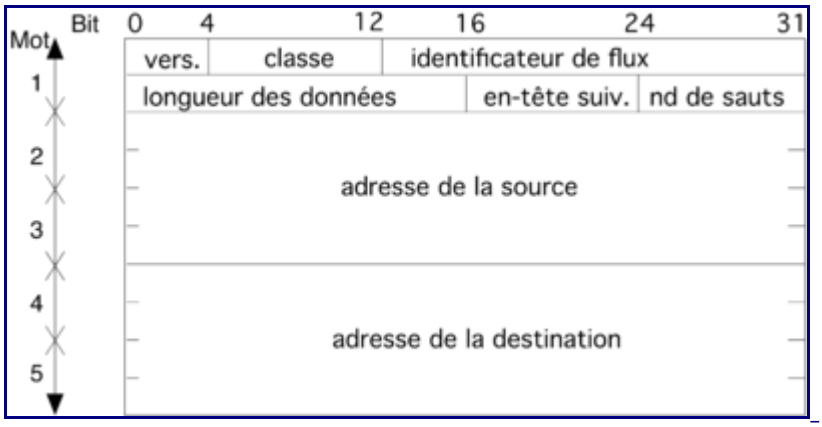

Figure 1: Format de l'en-tête d'un paquet IPv6.

Valeurs des champs de l'en-tête

Version

Le champ `Version` est au même emplacement et de même longueur, quelle que soit la version du protocole IP (IPv4 ou IPv6). Cette similarité permet à la pile réseau d'aiguiller correctement le paquet en se basant sur les 4 premiers bits de l'en-tête de niveau 3. Dans le cas d'IPv6, sa valeur est de 6; elle est de 4 pour IPv4. Le numéro de version 5 avait déjà été attribué au protocole Stream qui finalement n'a pas eu le succès attendu (RFC 1190 , RFC 1819).

Classe de trafic (*Traffic Class*)

Dans la version standardisée par le RFC 8200 , un champ `Classe de trafic` , sur 8 bits, permet la différenciation de services, conformément aux spécifications du RFC 2474 .

Le champ `Classe de trafic` est défini de façon similaire au champ `Differentiated Services` (DS, ou *DiffServ*) de l'en-tête IPv4, qui a lui-même pris la place de l'octet `Type of Service` de la spécification initiale d'IPv4. Ce champ est découpé en deux parties (cf. figure 2). Le sous-champ DSCP (*Differentiated Services Code Point*) contient les valeurs identifiant les différents traitements que les routeurs appliquent. La valeur par défaut 000000 correspond au service *au mieux* (ou *best effort*). Les deux derniers bits du champ, notés ECN (*Explicit congestion Notification*), servent aux routeurs à reporter un risque de congestion en combinaison avec l'algorithme RED (*Random Early Detection*). Le codage des 2 bits ECN est décrit à la page 6 du RFC 6040 .

Figure 2: Format de l'octet `classe de trafic` .

Plus de détails sur l'utilisation de ce champ sont donnés en Annexe 1 .

Identificateur de flux (*Flow Label*)

Ce champ, introduit dans le RFC 8200 puis spécifié en détails dans le RFC 6437 , contient un numéro unique, choisi par la source, pour identifier le flux de données d'une application (l'ensemble des paquets voix d'une application de voix sur IP, par exemple).

Cette valeur a pour but de faciliter le travail des routeurs dans le traitement différencié de paquets et donc la mise en oeuvre des fonctions de qualité de service, comme RSVP (RFC 2205). En identifiant par cette valeur les datagrammes provenant d'un même flux, le routeur peut alors réaliser un traitement particulier: choix d'une route, traitement en "temps réel" de l'information.

Habituellement, les routeurs se basent sur les valeurs de cinq champs pour construire un contexte propre à un même flux de données: adresses de la source et de la destination,

numéros de port de la source et de la destination, et protocole. Ce contexte sert à router plus rapidement les paquets puisqu'il évite de fouiller dans les en-têtes successives (niveau 3 pour les adresses, niveau 4 pour les info de protocoles et de port source et destination). Ce contexte est détruit après une période d'inactivité.

Avec IPv6, cette technique est officialisée. Le champ `Identificateur de flux` peut être rempli avec une valeur aléatoire qui servira à référencer le contexte. La source gardera cette valeur pour tous les paquets qu'elle émettra pour cette application et cette destination. Le traitement est optimisé puisque le routeur n'a plus à consulter cinq champs pour déterminer l'appartenance d'un paquet, mais un seul. De plus, si une extension de confidentialité est utilisée, les informations concernant les numéros de port sont masquées aux routeurs intermédiaires.

Passage à l'échelle

Le passage à l'échelle désigne la capacité d'un produit ou d'un mécanisme à s'adapter à un changement d'ordre de grandeur de la demande (montée en charge), en particulier sa capacité à maintenir ses fonctionnalités et ses performances en cas de forte demande [1] . Il faut donc comprendre qu'une difficulté à passer à l'échelle signifie que la propriété d'extensibilité n'est pas acquise.

À la date de rédaction de ce document, l'utilisation de l'étiquette de flux reste encore floue. Les micro-flux, c'est-à-dire de flux applicatifs, ne sont pas analysés dans le coeur du réseau pour des raisons de passage à l'échelle. De plus, MPLS a repris la notion de "routage spécifique en fonction d'une étiquette". Pour l'instant, ce champ peut être vu comme réservé. Son utilisation pourra être mieux spécifiée dans le futur.

Longueur des données utiles (*Payload Length*)

Ce champ indique la dimension, en octets, du reste du datagramme (en-tête exclus, donc), c'est-à-dire les données à la suite de l'en-tête. Les éventuelles extensions à l'en-tête IPv6 sont incluses dans ces données. Ce champ, sur 16 bits, peut donc indiquer une taille des données utiles, allant jusqu'à 64 kibioctets (64 * 1024 octets). Pour des paquets dont la taille serait supérieure, ce champ vaut 0 et l'émetteur ajoute une extension d'en-tête de "proche en proche" avec l'option *jumbogramme* définie par le RFC 2675 . Cette extension est essentiellement prévue pour la transmission à grand débit entre deux noeuds.

En-tête suivant (*Next Header*)

Ce champ identifie le prochain en-tête se trouvant à la suite de l'en-tête IPv6. Il peut s'agir d'un protocole de niveau supérieur (ICMP, UDP, TCP...) ou de la désignation d'une extension (cf. tableau *Valeurs du champ en-tête suivant pour IPv6*).

Valeurs du champ en-tête suivant pour IPv6

valeur	Hexa	Protocole ou *Extension*
0	0x00	*Proche-en-proche*
4	0x04	IPv4
6	0x06	TCP

17	0x11	UDP
41	0x29	IPv6
43	0x2b	*Routage*
44	0x2c	*Fragmentation*
50	0x32	*Confidentialité*
51	0x33	*Authentification*
58	0x3a	ICMPv6
59	0x3b	Fin des en-têtes
60	0x3c	*Destination*
132	0x84	SCTP
135	0x87	*Mobilité*
136	0x88	UDP-lite
140	0x8c	*Shim6*

Nombre maximal de sauts (*Hop Limit*)

Ce champ représente le nombre maximal de routeurs que le paquet peut traverser. Il est initialisé par la source du paquet, et ensuite, décrémenté à chaque routeur traversé. Un datagramme retransmis par un routeur est rejeté avec l'émission d'un message d'erreur ICMPv6 vers la source si la valeur, après décrémentation, atteint 0. Ce mécanisme permet d'éliminer des paquets persistant trop longtemps dans le réseau pour cause de problèmes de configuration, comme les boucles de routage. Il est aussi utilisé par des outils d'ingénierie des réseaux comme `traceroute` pour signaler à la source l'ensemble des routeurs traversés jusqu'à une destination.

La valeur initiale de ce champ, à l'émission du paquet, devrait être donnée dans un document annexe de l'IANA (http://www.iana.org/); ce qui permettrait de la modifier en fonction de l'évolution de la topologie du réseau. La valeur n'est pas encore officiellement attribuée mais certaines implantations prennent actuellement la valeur conseillée pour IPv4: 64.

La valeur par défaut peut être dynamiquement attribuée aux hôtes émetteurs par les annonces des routeurs en configuration automatique. Une modification de ce paramètre sera donc relativement simple quand la limite actuelle sera atteinte. On peut noter une limitation puisque ce champ, codé sur 8 bits, n'autorise la traversée que de 255 routeurs. En réalité, dans l'Internet actuel, le nombre maximal de routeurs traversés est d'une quarantaine, ce qui laisse une bonne marge pour l'évolution du réseau.

Adresses source et destination

Ces adresses sont renseignées par l'émetteur du paquet pour désigner l'émetteur et le destinataire du paquet. Pour renseigner l'adresse source, l'émetteur choisit de préférence une adresse IPv6 parmi celles configurées sur l'interface utilisée pour transmettre le paquet sur le réseau. Cette adresse est choisie pour avoir une portée compatible avec l'adresse de destination et, ainsi, permettre au destinataire de répondre. Par exemple, il serait problématique d'essayer de joindre une machine de l'Internet en donnant comme adresse source une adresse *lien-local* . Le mécanisme de choix de l'adresse source est spécifié dans le RFC 6724 .

L'adresse destination, elle aussi, peut être choisie dans la liste des adresses valables pour le destinataire; liste pouvant provenir de la résolution d'un nom en adresses par le DNS. L'adresse choisie doit être compatible avec la portée des adresses disponibles au niveau de l'émetteur. Par exemple, si l'émetteur ne possède que des adresses de type ULA, et que le destinataire est connu avec des adresses globales et ULA, ces dernières adresses seront à préférer. Le mécanisme du RFC 6724 précise le processus de choix de l'adresse destination.

Extensions à l'en-tête IPv6

L'ajout de nouvelles fonctionnalités est problématique quand l'en-tête est de taille fixe comme c'est le cas d'IPv6. La solution proposée consiste à ajouter des en-têtes spécifiques pour chaque fonctionnalité. Par exemple, si un paquet doit être fragmenté, une extension de fragmentation sera ajoutée par l'émetteur afin que le récepteur puisse rassembler les morceaux correctement. Les extensions sont ajoutées par l'émetteur du paquet pour signaler un traitement spécifique à réaliser, soit par les routeurs intermédiaires, soit par le destinataire du paquet. Il existe plusieurs types d'extensions selon le traitement demandé. Elles se placent après l'en-tête IPv6 et avant la charge utile du paquet (voir la figure 3). La présence d'une extension est signalée par le champ `En-tête suivant` (*Next Header*) de l'en-tête IPv6 qui possède alors la valeur correspondant à cette extension. Ainsi, elle est traitée par les routeurs intermédiaires comme un protocole de niveau supérieur à IP. L'utilisation des différents types d'extensions d'en-tête IPv6 sera abordé dans l'activité 24.

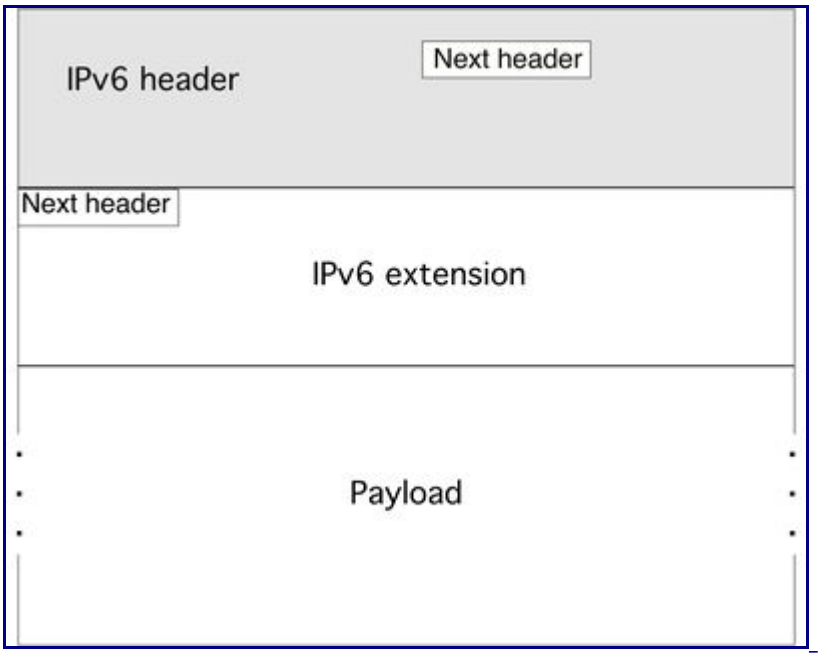

Figure 3: Format d'un paquet IPv6.

Evolution de l'en-tête depuis IPv4

Hormis la modification de la taille des adresses, ce qui conduit à une taille d'en-tête de 40 octets (le double d'un en-tête IPv4 sans options), le protocole IP a subi un toilettage [2] , reprenant l'expérience acquise au fil d'une trentaine d'années avec IPv4, défini en 1981 (!) par le RFC 791 . Le format des en-têtes IPv6 est ainsi simplifié et permet aux routeurs de meilleures performances dans leurs traitements. L'idée est de retirer du cœur de réseau les traitements compliqués. Les routeurs ne font que retransmettre les paquets vers la destination, les autres traitements sont faits par l'émetteur et le destinataire du paquet.

L'en-tête ne contient plus de contrôle d'erreur (*checksum*), qui devait être ajusté, pour IPv4, par chaque routeur en raison, entre autres, de la décrémentation du champ `durée de vie` . Par contre, pour éviter qu'un paquet, dont le contenu est erroné -- en particulier sur l'adresse de destination --, ne se glisse dans une autre communication, tous les protocoles de niveau supérieur doivent mettre en œuvre un mécanisme de contrôle d'erreur de bout en bout, incluant un pseudo-en-tête qui prend en compte les adresses source et destination. Le contrôle d'erreur d'UDP, facultatif pour IPv4, devient ainsi obligatoire. Pour ICMPv6, le contrôle d'erreur intègre le pseudo-en-tête alors que, pour ICMPv4, il ne portait que sur le message ICMP.

La fonction de fragmentation a aussi été retirée des routeurs. Les champs de l'en-tête IPv4 qui s'y reportent (`Identification, Drapeau, Place du fragment`) ont été supprimés. Normalement, les algorithmes de découverte du PMTU (*Path MTU*) évitent d'avoir recours à la fragmentation. Si celle-ci s'avère nécessaire, une extension est prévue et le découpage en fragments est réalisé uniquement au niveau de l'émetteur.

Une autre évolution majeure depuis l'en-tête IPv4 est la spécification des extensions d'en-tête pour remplacer les options. En effet, dans le cas d'IPv4, les options sont incluses dans l'en-tête. Celui-ci est donc de taille variable (taille indiquée dans le champ `Internet Header Length`

), ce qui peut compliquer le traitement dans les routeurs intermédiaires. Les extensions à l'en-tête IPv6 simplifient la mise en œuvre de ces fonctionnalités et permettent de garder la taille de l'en-tête IPv6 fixe à 40 octets.

Conclusion

Cette activité a présenté l'en-tête IPv6, définissant une nouvelle version du protocole IP. Cette version reprend les caractéristiques communes du protocole: des adresses de taille fixe, un en-tête définissant une position fixe pour l'adresse de destination, position de plus alignée en mémoire. L'objectif reste toujours le traitement optimal de l'en-tête dans les routeurs. IPv6 garde donc ces principes et va même plus loin que son prédécesseur IPv4 en reconsidérant des mécanismes jugés coûteux, comme la fragmentation, le *checksum* ou les options. Certains de ces mécanismes ont été éliminés dans la nouvelle version du protocole, d'autres remplacés par mécanismes plus performants. Enfin, nous pouvons signaler ce formulaire qui présente l'essentiel du paquet IPv6 en une seule page [3] . Ceci peut vous être utile pour la suite.

Références bibliographiques

1. ↑ Wikipedia. Définition de la scalabilité
2. ↑ Lee, D.C.; Lough, D.L.; Midkiff, S.F.et al. (1998). IEEE Network, Vol. 12, No. 1, January. The next generation of the Internet: aspects of the Internet protocol version 6.
3. ↑ Stretch, J. (2009) packetlife.net. Aide-mémoire tout en une page. IPv6

Pour aller plus loin

RFC et leur analyse par S. Bortzmeyer:

- RFC 791 Internet protocol (IPv4)
- RFC 1190 Experimental Internet Stream Protocol, Version 2 (ST-II)
- RFC 1819 Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+
- RFC 2205 Resource ReSerVation Protocol (RSVP)
- RFC 2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers Analyse
- RFC 2675 IPv6 Jumbograms
- RFC 6040 Tunnelling of Explicit Congestion Notification
- RFC 6437 IPv6 Flow Label Specification
- RFC 6724 Default Address Selection for Internet Protocol version 6 (IPv6) Analyse
- RFC 7098 : Using the IPv6 Flow Label for Load Balancing in Server Farms Analyse
- RFC 8200 Internet Protocol, Version 6 (IPv6) Specification Analyse

Annexe 1: la gestion de la qualité de service

L'Internet différencié permet aux fournisseurs d'accès de gérer différemment les congestions qui surviennent dans le réseau. Sans différenciation, les paquets ont la même probabilité de rejet. Avec la différenciation, plusieurs classes de trafic sont définies. Les paquets appartenant aux classes les plus élevées ont une probabilité de rejet plus faible. Bien entendu, pour que l'introduction de telles classes de trafic soit efficace, il faut introduire une gestion des ressources différente pour chacune des classes, et des mécanismes de contrôle pour vérifier que les flux des utilisateurs n'utilisent pas que les classes les plus élevées ou qu'ils ne dépassent pas leur contrat. Par exemple, un client peut établir un contrat de niveau de services appelé SLA (*Service Level Agreement*) avec son fournisseur d'accès.

L'intérêt principal de la différenciation de services est qu'elle ne casse pas le modèle initial de l'Internet (version 4 ou version 6). Les flux sont toujours traités en *Best Effort* même si certains sont plus *Best* que d'autres. Il n'y a aucune garantie qu'un trafic d'une classe haute arrive à destination, mais la probabilité est plus importante. L'autre intérêt des classes de trafic vient de la possibilité d'agrégation des flux. La classe d'appartenance est indiquée dans l'en-tête du paquet. Les applications peuvent marquer les paquets en fonction de paramètres locaux (flux multimédia, flux interactif, trafic priorisé...). Le fournisseur d'accès qui récupère le trafic n'a plus à se préoccuper des applicatifs. Il vérifie que le trafic d'une classe ne dépasse pas le contrat préalablement établi.

Dans le cœur du réseau, les routeurs prennent en compte les différentes classes. Le fournisseur d'accès devra également passer des accords avec les autres opérateurs pour pouvoir faire transiter les flux avec un traitement approprié. Cet aspect de dimensionnement de réseau et de négociation d'accords d'échange est au coeur du métier d'opérateur.

Pour signifier l'appartenance d'un paquet à une certaine classe de trafic, une valeur est renseigné au niveau de l'en-tête IP dans l'octet `Traffic Class` afin qu'elle puisse être analysée par tous les routeurs mettant en oeuvre le traitement différencié. Le format de ce champ est rappelé en Figure 2.

Le tableau 1 présente les différentes valeurs définies pour le champ DSCP (*Differential Service Code Point*). Les valeurs sont présentées en format binaire avec les 6 bits les plus significatifs de l'octet `Traffic Class` , puis leur conversion en décimal, leur nommage, la probabilité d'écartement, et l'équivalence avec les anciennes valeurs du champ TOS de l'IPv4:

DSCP Value	Decimal Value	Meaning	Drop Probability	Equivalent Precedence Value
101 110	46	EF Expedited Forwarding	N/A	101 – Critical
101 100	44	Voice Admit	N/A	101 – Critical (RFC 5865)
000 000	00	Best Effort / Default	N/A	000 – Routine
001 010	10	AF11 Assured Forwarding	Low	001 – Priority
001 100	12	AF12	Medium	001 – Priority
001 110	14	AF13	High	001 – Priority
010 010	18	AF21	Low	010 – Immediate
010 100	20	AF22	Medium	010 – Immediate
010 110	22	AF23	High	010 – Immediate
011 010	26	AF31	Low	011 – Flash
011 100	28	AF32	Medium	011 – Flash
011 110	30	AF33	High	011 – Flash
100 010	34	AF41	Low	100 – Flash Override
100 100	36	AF42	Medium	100 – Flash Override
100 110	38	AF43	High	100 – Flash Override
001 000	08	CS1 Class Selector		1 – Priority
010 000	16	CS2		2 – Immediate
011 000	24	CS3		3 – Flash
100 000	32	CS4		4 – Flash Override
101 000	40	CS5		5 – Critic /ECP
110 000	48	CS6		6 – Internetwork Control
111 000	56	CS7		7 – Network Control

Tableau 1: Format du champ DSCP.

Pour l'instant, deux types de comportement sont standardisés:

- Assured Forwarding [RFC 2597]: Ce comportement définit quatre classes de trafic et trois priorités, suivant que l'utilisateur respecte son contrat, le dépasse légèrement, ou est largement en dehors. Les classes sont donc choisies par l'utilisateur et restent les mêmes tout au long du trajet dans le réseau. La priorité, par contre, peut être modifiée dans le réseau par les opérateurs en fonction du respect ou non des contrats. Par exemple, pour la classe AF n°2, on dispose des 3 priorités suivantes: AF21, AF22 et AF23. Plus le chiffre est élevé, plus la priorité est faible. C'est-à-dire qu'en cas de saturation de cette classe de trafic, les paquets AF23 seront écartés avant AF22, puis AF21.
- Expedited Forwarding [RFC 2598]: Ce comportement est comparable à un circuit à débit constant réservé dans le réseau. Le trafic est mis en forme à l'entrée du réseau, en retardant l'émission des paquets qui sont hors contrat. En plus de ces comportements, l'octet DS a gardé, pour des raisons de compatibilité avec les équipements existants, les valeurs du bit ToS qui étaient le plus fréquemment utilisées. La valeur est 0xB8 (1011 1000 en binaire, et en tenant compte uniquement des 6 bits de poids forts: 46 en décimal).

- Voice Admit: Cette autre valeur a été par la suite proposée dans le RFC 5865 pour affiner le traitement de flux *temps réel* de différentes natures: voix, vidéo, signalisation *temps*

réel … (La valeur est 0xB0 soit 1011 0100 en binaire, et en tenant compte uniquement des 6 bits de poids forts: 44 en décimal).

- Network Control: Autre particularité: la valeur 0xE0 (1110 0000 en binaire, et en tenant compte uniquement des 6 bits de poids forts: 56 en décimal) correspond à CS7, la classe de contrôle du réseau (*Network Control*). Elle est utilisée dans des mises en oeuvre d'IPv6 pour l'émission de certains paquets ICMPv6. Cette valeur est dépréciée. Il est conseillé d'utiliser la valeur CS6 comme spécifié dans le RFC 4594 .

Références bibliographiques

- RFC 2597 Assured Forwarding PHB Group
- RFC 2598 An Expedited Forwarding PHB
- RFC 4594 Configuration Guidelines for DiffServ Service Classes
- RFC 5865 A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic

Activité 22: Les mécanismes d'encapsulation

Introduction

La notion d'encapsulation de protocoles repose sur le principe de l'empilement des couches représentatives des traitements nécessaires à effectuer dans les différents composants d'un réseau. Ces traitements affecteront toutes les couches dans les équipements d'extrémité, et certaines seulement pour les équipements réalisant le relais des échanges sur le réseau de communication. Dans cette activité, nous aborderons deux points importants en lien avec l'encapsulation, et impactés par le remplacement de IPv4 par IPv6 au sein de la couche de réseau: la longueur maximale des unités de transfert (*Maximum Transmission Unit* (MTU)), c'est à dire la longueur maximale des datagrammes encapsulés dans les trames de la couche de liaison, et la question de la détection d'erreur binaire par calcul de *checksum* , qui a disparu de l'en-tête IPv6.

Représentation de l'encapsulation

L'organisation internationale de normalisation ISO a défini le modèle OSI (*Open System Interconnection*) par une décomposition de l'architecture du réseau en 7 couches représentées du niveau Physique jusqu'au niveau Application comme représentée par la Figure 1. Le modèle TCP/IP (ou DOD *Department of Defense*) a eu une approche plus pragmatique en décomposant l'architecture de réseau en 4 couches. Les couches session, présentation et application sont agrégées en une seule couche applicative propre à chaque protocole.

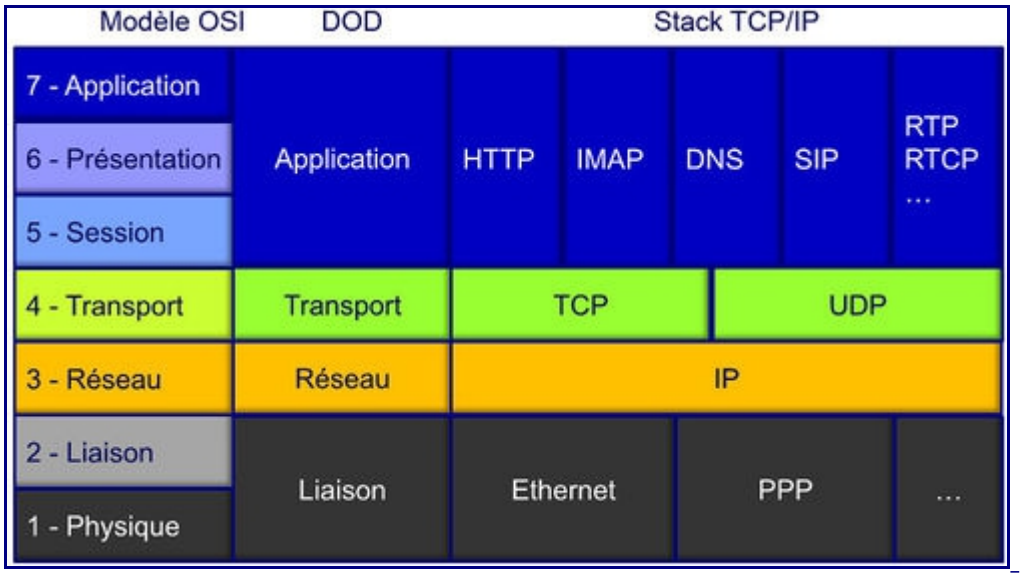

Figure 1: Comparaison modèle OSI - modèle TCP/IP.

Pour simplifier l'organisation, pour un noeud d'extrémité du réseau, nous pouvons considérer, que la carte réseau réalise les fonctions de niveau Physique et Liaison, que le traitement des couches Réseau et Transport est réalisé par les couches intermédiaires installées dans le système d'exploitation, et que le reste du système, avec les programmes applicatifs, gère les couches Session, Présentation et Application.

Traitement des couches basses

La méthode de transport d'un datagramme IPv6 entre deux machines directement reliées entre elles par un lien physique est le même que pour IPv4. Le datagramme est tout d'abord transmis vers l'interface d'émission qui l'encapsule dans une trame (PDU (*Protocol Data Unit*) de niveau 2 dans le modèle de référence OSI). Cette trame est transmise sur le lien avec l'adresse physique de la machine de destination (cette adresse sur un lien sera appelée adresse MAC dans la suite). La machine de destination reçoit la trame sur son interface, désencapsule le datagramme et le traite.

Les différences avec IPv4 sont les suivantes:

- Sur le support Ethernet, le RFC 2464 précise que le code protocole encapsulé de la trame est différent (champ `EtherType` d'une trame Ethernet). Par exemple, pour les réseaux à diffusion, le code est `0x86DD` alors que, pour IPv4, le code est `0x0800` . À l'origine, il était prévu de garder le même code et d'assurer l'aiguillage entre IPv4 et IPv6 en utilisant le champ `Version` du paquet. Mais certains équipements ne vérifient pas la valeur de ce champ et auraient eu un comportement incontrôlable en essayant de traiter un paquet IPv6 comme un paquet IPv4.
- La résolution de l'adresse MAC de destination à partir de l'adresse IP de destination du paquet change. Par exemple, sur un réseau à diffusion, cette résolution est faite en IPv4 par le protocole ARP alors qu'en IPv6 on utilise le protocole de découverte de voisins comme nous le verrons dans la séquence 3.
- La taille minimale d'une trame est passée à 1 280 octets; ceci peut forcer certains protocoles à utiliser plusieurs trames par datagramme IPv6.
- Enfin, certains protocoles ont des parties propres à IPv4. Ces parties doivent être modifiés. C'est le cas des protocoles de contrôle et de compression de PPP.

Couche physique

Commençons par la couche *physique* , qui est à la base de l'édifice de ce modèle. Les spécifications de cette couche dépendent du support lui-même. Nous devons gérer la transmission des informations binaires issues du codage des trames et des paquets sur un support cuivre, optique ou sans fil; d'où la nécessité d'adaptation aux caractéristiques des composants (câbles, connecteurs ou antennes) et d'une méthode appropriée de codage des données (représentation physique des données binaires).

La représentation binaire utilisée dépend du support. Sur du cuivre, on utilise des variations d'impulsions électriques; sur une fibre optique, ce sont des variations lumineuses sur une ou plusieurs longueurs d'ondes; en transmission sans fil, ce sont généralement des signaux radio, laser ou infrarouge. La couche *physique* coordonne le débit et la synchronisation de l'émetteur et du récepteur réseau, tout en tentant de garantir la transparence et l'intégrité d'un flux d'information binaire, sans notion d'interprétation du contenu.

Hélas, cette couche est fréquemment soumise à différentes perturbations issues de l'environnement extérieur au canal de transmission: rayonnements électromagnétiques, micro-

coupures ou altérations des signaux par différents facteurs. Les coupleurs intégrés dans les cartes réseau réalisent les fonctions nécessaires et utiles au niveau *physique* , et on dispose d'un indicateur de qualité de la transmission avec le calcul du CRC (*Cyclic Redondancy Check*).

Couche liaison

Le rôle de la couche *liaison* est, entre autres, de contrôler l'accès à la couche physique, en réalisant le multiplexage temporel sur le support de transmission, et de transformer la couche *physique* en une liaison à priori exempte d'erreurs de transmission pour la couche *réseau* . La couche *liaison* doit être capable d'écarter le trafic nécessaire à la synchronisation sur le lien physique, et de reconnaître les débuts et fins de trames. Cette couche écarte les trames en cas de réception erronée, comme en cas de non respect du format, ou bien en cas de problème sur la ligne de transmission. La vérification du champ CRC aide à faire ce tri. Cette couche intègre également une fonction de contrôle de flux pour éviter l'engorgement d'un récepteur incapable de suivre un rythme imposé.

L'unité de données de protocole de la couche *liaison de données* est la trame (*Link Protocol Data Unit* (LPDU)), qui est composée de plusieurs champs permettant d'identifier l'origine des échanges, le rôle de la trame, le contenu de l'enveloppe et, en fin de trame, le champ CRC, le tout étant encadré par une séquence particulière de codage de début et de fin de trame.

Si nous prenons l'exemple de la trame Ethernet (cf. Figure 2), un délai inter-trame minimum de 96 intervalles de temps est spécifié comme silence sur un support cuivre alors que, sur un support optique, tout silence est comblé par la transmission d'un ou plusieurs symboles particuliers «idle»; une parfaite synchronisation est alors maintenue entre les extrémités du lien optique.

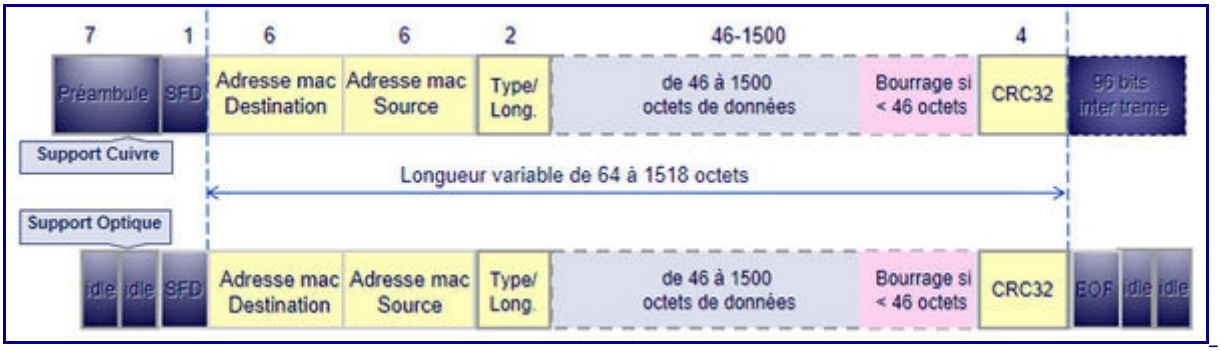

Figure 2: Format de la trame Ethernet.

Une fois que l'arrivée d'une trame Ethernet est détectée par le coupleur, les premiers champs immédiatement accessibles correspondent aux adresses MAC `Destination` et `Source` puis, soit au champ `Longueur` dans le cas d'une encapsulation au standard 802.3, ou bien au champ `EtherType` dans le cas d'une encapsulation avec le standard Ethernet original. Ensuite, l'enveloppe de la trame transporte les données, qui correspondent aux paquets IPv6 dès lors que le champ `EtherType` vaut `0x86dd` . Vient ensuite le champ CRC codé sur 32 bits. Dans le cas d'une encapsulation au standard 802.1Q, d'autres champs permettent la reconnaissance du numéro de VLAN (*Virtual Local Access Network*) et du niveau de priorité

défini dans le standard 802.1p.

Un des éléments particulièrement importants est la capacité de transport de la trame. Dans l'exemple ci-dessus, nous voyons que la trame Ethernet traditionnelle dispose d'une enveloppe qui autorise le transport de 1500 octets maximum: MTU = 1500.

D'autres formats de trames permettent des échanges plus ou moins importants. Citons quelques MTU:

- PPPoE = 1492
- PPPoA = 1468
- MPLS = 1500 à 65535
- 802.15.4 (LowPAN) = 81
- Ethernet Jumboframe = 9000

Rappelons que la spécification du protocole IPv6 impose une taille minimale du paquet de 1280 octets. Pour les couches liaison imposant des tailles inférieures, il est donc obligatoire de mettre en place une *couche d'adaptation* comme 6LowPAN (RFC 4944) pour les réseaux 802.15.4. Cette couche située entre la couche *liaison* et la couche *réseau* IPv6 prend en charge le découpage des paquets IPv6 en fragments pouvant être transportés dans les trames et leur ré-assemblage au niveau du premier routeur de sortie.

Couches intermédiaires

Couche réseau

Étant donné que la taille minimum de l'en-tête IPv6 est de 40 octets, le MTU résiduel d'une trame Ethernet classique est de 1500 - 40 = 1460 octets; sachant que ces 1460 octets de données seront probablement encore amputées d'en-têtes de niveau transport, par exemple 20 octets minimum pour TCP et 8 octets pour UDP.

Parmi les différences existant entre les datagrammes IPv4 et IPv6, il y a la disparition de la somme de contrôle d'erreur (*checksum*) dans les en-têtes IPv6. Cette somme de contrôle était utilisée pour vérifier l'absence d'erreur binaire de l'en-tête du paquet traité. Une erreur binaire est le changement de valeur d'un bit effectué lors de la transmission. En IPv4, il est nécessaire de la vérifier et de l'ajuster lors de chaque retransmission par un routeur, ce qui entraîne une augmentation du temps de traitement du paquet. Cette somme ne vérifie que l'en-tête IPv4, pas le reste du paquet. Aujourd'hui, les supports physiques sont de meilleure qualité et savent détecter les erreurs (par exemple, Ethernet a toujours calculé sa propre somme de contrôle; PPP, qui a presque partout remplacé SLIP, possède un CRC). L'intérêt de la somme de contrôle au niveau réseau a diminué et ce champ a été supprimé de l'en-tête IPv6.

Le checksum sur l'en-tête IPv6 n'existant plus, il faut néanmoins se prémunir des erreurs de transmission. En particulier, une erreur sur l'adresse de destination va faire router un paquet dans une mauvaise direction. Le destinataire doit donc vérifier que les informations d'en-tête IP sont correctes avant d'accepter ces paquets. Dans les mises en oeuvre des piles de protocoles Internet, les entités de niveau transport remplissent certains champs du niveau réseau. Il a donc

été décidé que tous les protocoles au-dessus d'IPv6 devaient utiliser une somme de contrôle intégrant à la fois les données et les informations de l'en-tête IPv6. La notion de *pseudo-en-tête* dérive de cette conception. Pour un protocole comme TCP, qui possède une somme de contrôle, cela signifie qu'il faut modifier le calcul de cette somme. Pour un protocole comme UDP, qui possède une somme de contrôle facultative, cela signifie qu'il faut modifier le calcul de cette somme et le rendre obligatoire.

IPv6 a unifié la méthode de calcul des différentes sommes de contrôle. Le RFC 8200 définit, dans sa section 8.1, un *pseudo-en-tête* (cf. Figure 3), résultat de la concaténation d'une partie de l'en-tête IPv6 et du PDU du protocole concerné. L'algorithme de calcul du checksum est celui utilisé en IPv4. Il est très simple à mettre en œuvre et ne demande pas d'opérations complexes. Il s'agit de faire la somme en complément à 1 des mots de 16 bits du *pseudo-en-tête* , de l'en-tête du protocole de transport, et des données, puis de prendre le complément à 1 du résultat.

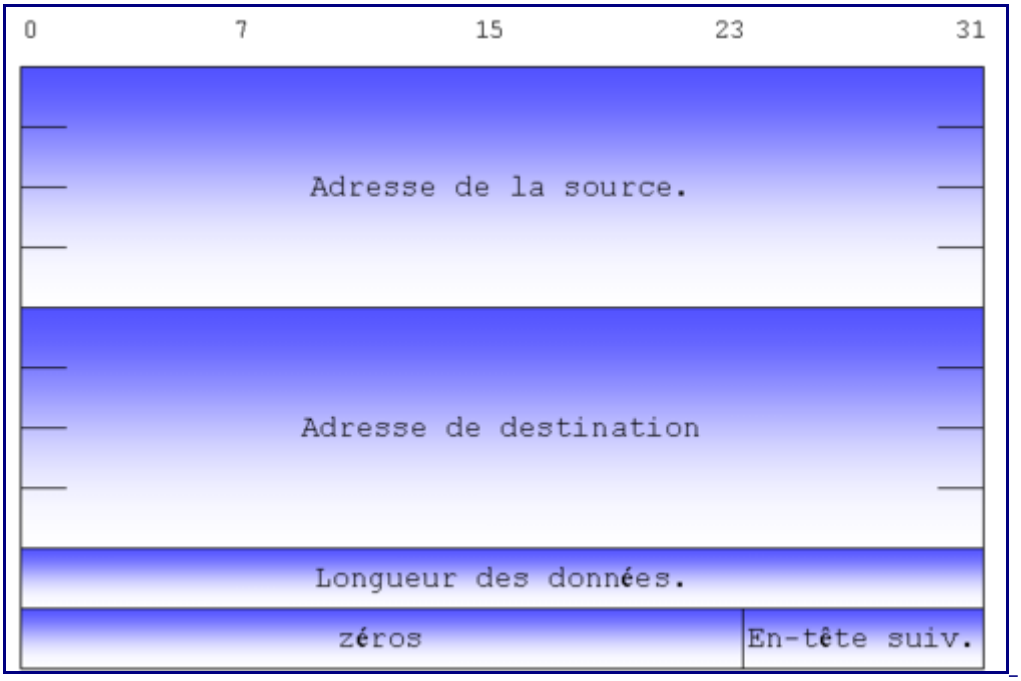

Figure 3: Champs du `pseudo-en-tête` .

Il faut noter que les informations contenues dans le *pseudo-en-tête* ne seront pas émises telles quelles sur le réseau. Le champ `en-tête suivant` du *pseudo-en-tête* ne reflète pas celui qui sera émis dans les paquets puisque les extensions ne sont pas prises en compte dans le calcul du checksum. Ainsi, si l'extension de routage est mise en œuvre, l'adresse de la destination est celle du dernier équipement. De même, le champ `longueur` est codé sur 32 bits pour contenir la valeur de l'option *jumbogramme* si celle-ci est présente.

Couches Transport

UDP et TCP

Les modifications apportées aux protocoles de niveau transport sont minimes. L'un des pré-requis à la mise en œuvre d'IPv6 était de laisser en l'état aussi bien TCP (*Transmission Control*

Protocol) qu'UDP (*User Datagram Protocol*). Ces protocoles de transport sont utilisés par la très grande majorité des applications réseau et l'absence de modification facilitera grandement le passage de IPv4 à IPv6.

La principale modification de ces protocoles concerne le calcul de la somme de contrôle (*checksum*) pour vérifier l'intégrité des données. Comme la couche *réseau* ne possède pas de champ de contrôle, c'est à la couche de transport que revient cette tâche. Le calcul de la somme de contrôle au niveau du transport a donc été adapté pour inclure des données de l'en-tête. De plus, la présence d'une somme de contrôle devient obligatoire pour tout protocole de niveau supérieur transporté au-dessus d'IPv6.

Nota: *Les RFC 6935 et RFC 6936 ont introduit une dispense à cette obligation, dans les cas des techniques de tunnel au dessus d'UDP. La nouvelle philosophie est résumée ainsi par S. Bortzmeyer: "par défaut, la somme de contrôle doit être calculée et mise dans le paquet. Mais on peut s'en dispenser dans le cas de tunnels (et uniquement celui-ci), sur le paquet extérieur. Le protocole à l'intérieur du paquet (qui n'est pas forcément de l'UDP et même pas forcément de l'IPv6) doit rester protégé par sa propre somme de contrôle. Cela ne doit s'appliquer que pour une liste explicite de ports (ceux utilisés par le tunnel) et pas pour tout le trafic UDP de la machine. Et le tout doit se faire en lisant le RFC 6936 qui précise les conditions d'application de cette nouvelle règle."*

Un autre changement au niveau des protocoles de niveau 4 concerne la prise en compte de l'option *jumbogramme* de l'extension *proche-en-proche* . Le RFC 2675 définit le comportement d'UDP et de TCP quand les jumbogrammes sont utilisés. En effet, les en-têtes de ces messages contiennent eux aussi un champ `longueur` codé sur 16 bits et, par conséquent, insuffisant pour coder la longueur du jumbogramme:

- Pour le protocole UDP, si la longueur des données excède 65 535 octets, le champ `longueur` est mis à 0. Le récepteur détermine la longueur des données par la connaissance de la taille dans l'option *jumbogramme* .
- Le protocole TCP pose plus de problèmes. En effet, bien que les messages TCP ne contiennent pas de champ `longueur` , plusieurs compteurs sont codés sur 16 bits.
- Le champ `longueur` de la fenêtre de réception ne pose pas de problème depuis que le RFC 1323 a défini l'option *TCP window scale* qui donne le facteur multiplicatif qui doit être appliqué à ce champ.
- À l'ouverture de connexion, la taille maximale des segments (MSS) est négociée. Le RFC 2675 précise que, si cette taille doit être supérieure à 65 535, la valeur 65 535 est envoyée et le récepteur prend en compte la longueur déterminée par l'algorithme de découverte du MTU.

Pour l'envoi de données urgentes avec TCP, on utilise un bit spécifique de l'en-tête TCP (bit `URG`) ainsi que le champ `pointeur urgent` . Ce dernier sert à référencer la fin des données à traiter de manière particulière. Trois cas peuvent se présenter:

- Le premier, qui est identique à IPv4, est celui où le pointeur indique une position de moins de 65 535.

- Le second se produit lorsque le déplacement est supérieur à 65 535 et supérieur ou égal à la taille des données TCP envoyées. Cette fois-ci, on place la valeur 65 535 dans le champ `pointeur urgent` et on continue le traitement normal des paquets TCP.
- Le dernier cas intervient quand le pointeur indique un déplacement de plus de 65 535 qui est inférieur à la taille des données TCP. Un premier paquet est alors envoyé, dans lequel on met la valeur 65 535 dans le champ `pointeur urgent` . L'important est de choisir une taille de paquet de manière à ce que le déplacement dans le second paquet, pour indiquer la fin des données urgentes, soit inférieur à 65 535.

Il existe d'autres propositions pour faire évoluer TCP. Il faut remarquer que le travail n'est pas de même ampleur que pour IP. En effet, TCP est un protocole de bout en bout. La transition vers une nouvelle génération du protocole peut se faire par négociation entre les deux extrémités. Pour IP, tous les routeurs intermédiaires doivent prendre en compte les modifications.

UDP-lite

UDP-lite permet de remonter aux couches supérieures des données erronées pendant leur transport. Si, dans un environnement informatique, une erreur peut avoir des conséquences relativement graves quant à l'intégrité des données, il est normal de rejeter ces paquets. Dans le domaine du multimédia, cette exigence peut être relâchée. En effet, la plupart des décodeurs de flux audio ou vidéo sont capables de supporter un certain nombre d'erreurs binaires dans un flux de données. Pour améliorer la qualité perçue par l'utilisateur, il est donc préférable d'accepter des paquets erronés plutôt que de rejeter un bloc complet d'informations qui se traduirait par une coupure perceptible du flux.

En IPv4, l'utilisation du *checksum UDP* étant optionnelle (la valeur 0 indique que le checksum n'est pas calculé), UDP peut être utilisé pour transporter des flux multimédias. Avec IPv6, l'utilisation du checksum a été rendue obligatoire puisque le niveau 3 n'en possède pas. Pour éviter qu'un paquet comportant des erreurs ne puisse pas être remonté aux couches supérieures, le protocole UDP-lite a été défini par le RFC 3828 . Les modifications sont minimes par rapport à UDP. Le format de la trame reste le même; seule la sémantique du champ `longueur` est changée. Avec UDP, ce champ est inutile puisqu'il est facilement déduit du champ `longueur` de l'en-tête IP. UDP-lite le transforme en champ `couverture du checksum` . Si la longueur est 0, UDP-lite considère que le checksum couvre tout le paquet. La valeur 8 indique que seul l'en-tête UDP est protégé par le checksum (ainsi qu'une partie de l'en-tête IP grâce au pseudo-header). Les valeurs comprises entre 1 et 7 sont interdites car le checksum UDP-lite doit toujours couvrir l'en-tête. Une valeur supérieure à 8 indique qu'une partie des données sont protégées. Si la couverture est égale à la longueur du message, on se retrouve dans un cas compatible avec UDP.

SCTP

Le protocole SCTP (*Stream Control Transmission Protocol*) RFC 4960 est fortement lié au protocole IPv6. SCTP est un protocole de niveau 4 initialement conçu pour transporter des informations de signalisation [1] . La fiabilité est donc un prérequis important et la gestion de la multi-domiciliation est prise en compte. L'idée est de permettre aux deux équipements

terminaux d'échanger, à l'initialisation de la connexion (appelée, dans le standard, *association*), l'ensemble de leurs adresses IPv4 et IPv6. Chaque équipement choisit une adresse privilégiée pour émettre les données vers l'autre extrémité et surveille périodiquement l'accessibilité des autres adresses. Si l'équipement n'est plus accessible par l'adresse principale, une adresse secondaire est choisie.

SCTP permet une transition douce d'IPv4 vers IPv6 puisque l'application n'a plus à se préoccuper de la gestion des adresses. Si les deux entités possèdent une adresse IPv6, celle-ci sera privilégiée. De plus, SCTP peut servir de brique de base à la gestion de la multi-domiciliation IPv6. En effet, avec TCP, une connexion est identifiée par ses adresses. Si une adresse n'est plus accessible, le fait d'en changer peut conduire à la coupure de la connexion. Il faut avoir recours à des subterfuges, comme la mobilité IP, pour maintenir la connexion. SCTP brise ce lien entre la localisation de l'équipement et l'identification des associations.

Conclusion

Cette activité a fait un rappel des différentes couches de la pile réseau d'un système. La couche réseau, ou niveau 3, est le langage commun de tous les équipements connectés à l'Internet. L'introduction d'IPv6 a donc un impact sur les différentes couches, notamment la couche sous-jacente, couche liaison, et les couches supérieures, notamment la couche transport. Même si ces couches sont censées être indépendantes dans le modèle théorique OSI, force est de remarquer que dans la pratique, elles sont interdépendantes. Pour preuve le champ de contrôle d'erreur n'a pas été retenu au niveau de la couche IP car il y a déjà un contrôle fait au niveau *liaison* . Et un autre contrôle d'erreur a été placé dans les couches supérieures afin de vérifier l'intégrité des données transportées. Cette vérification se fait uniquement par le destinataire. Le contrôle d'erreur est un exemple qui illustre l'interdépendance des couches.

Références bibliographiques

1. ↑ Fu, S. et Atiquzzaman, M. (2004). IEEE Communications Magazine, Vol. 42, No. 4, April. SCTP: State of the Art in Research, Products, and Technical Challenges.

Pour aller plus loin

RFC et leur analyse par S. Bortzmeyer:

- RFC 1323 : TCP Extensions for High Performance Analyse
- RFC 2464 : Transmission of IPv6 Packets over Ethernet Networks
- RFC 2675 : IPv6 Jumbograms
- RFC 3828 : The Lightweight User Datagram Protocol (UDP-Lite) Analyse
- RFC 4944 : Transmission of IPv6 Packets over IEEE 802.15.4 Networks
- RFC 4960 Stream Control Transmission Protocol Analyse
- RFC 5692 : Transmission of IP over Ethernet over IEEE 802.16 networks Analyse
- RFC 6691 : TCP Options and MSS Analyse
- RFC 6935 : IPv6 and UDP Checksums for Tunneled Packets Analyse

- RFC 6936 : Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums Analyse
- RFC 6951 : UDP Encapsulation of SCTP Packets for End-Host to End-Host Communication Analyse
- RFC 8200 : Internet Protocol, Version 6 (IPv6) Specification Analyse

Activité 23: Les principes du routage en IPv6

Introduction: Qu'est ce que le routage?

Le routage est la fonction permettant au réseau d'acheminer un paquet vers sa destination [1] . C'est donc une fonction cruciale pour le bon fonctionnement du réseau. Le routage s'effectue au niveau IP, indépendamment des couches physique et liaison sous-jacentes. Grâce au routage, un même paquet IP pourra être relayé entre des réseaux utilisant des couches basses différentes, d'un réseau LTE vers un réseau local Ethernet par exemple. L'acheminement d'un paquet au sein d'un réseau utilisant les mêmes couches basses (un même réseau local Ethernet par exemple) s'effectue à partir des informations présentes dans les en-têtes de l'unité protocolaire de la couche liaison. On parle alors de commutation et non de routage.

La fonction de routage est distribuée sur les différents noeuds actifs au niveau réseau, c'est-à-dire comportant une pile IP. Lorsqu'un paquet IP arrive sur un noeud, celui-ci décide si ce paquet lui est destiné ou s'il doit le retransmettre. Dans ce dernier cas, la fonction de routage doit décider vers quel réseau faire suivre le paquet afin qu'il atteigne sa destination. Cette décision s'appuie d'une part sur les informations contenues dans l'en-tête IP du paquet, principalement **l'adresse destination** . D'autre part, la décision de routage dépendra des informations sur la position relative de la destination par rapport au routeur qui doit relayer le paquet. Ces informations, représentées dans la **table de routage** , constituent la connaissance locale à un noeud de la topologie du réseau. Grâce à ces informations, un noeud déterminera vers quel réseau faire suivre le paquet, qui arrivera alors sur un nouveau noeud. Ainsi, de proche en proche, le paquet sera relayé depuis l'émetteur jusqu'à sa destination.

Topologie

La topologie de réseau correspond à l'arrangement (physique ou logique) de ses équipements et de ses liaisons.

La connaissance de la topologie du réseau peut être communiquée à chaque routeur de plusieurs façons. L'administrateur peut configurer manuellement la table de routage au niveau des différents routeurs. Mais ce mode de configuration est peu adapté lorsque le réseau évolue (lorsqu'une nouvelle liaison apparait par exemple). On parle alors de **routage statique** . Une autre méthode consiste, pour chaque routeur, à propager sa connaissance locale du réseau et à intégrer les informations fournies par d'autres routeurs. Ces échanges s'effectuent grâce à des **protocoles de routage** . Ce mécanisme permet d'envisager une prise en compte automatique des évolutions du réseau par les routeurs. On parle alors de **routage dynamique** .

Cette activité présente les différents éléments de configuration du routage IPv6 sur un noeud. Le fonctionnement du mécanisme de routage se base sur ces configurations ainsi que sur les protocoles de routage disponibles en IPv6. Les algorithmes de routage permettant de calculer une représentation de la topologie du réseau ne seront pas détaillés dans ce MOOC.

Routage d'un paquet au niveau d'un routeur

La fonction de routage traite de la décision prise par un routeur pour relayer un paquet vers sa destination. Un paquet est à relayer lorsqu'il arrive sur un routeur et que l'adresse destination de ce paquet ne concerne aucune interface de ce routeur.

Plusieurs cas sont alors possibles:

- La destination est sur un des réseaux sur lequel le routeur est directement connecté. Le paquet doit alors être remis à la destination.
- La destination n'est sur aucun des réseaux directement connectés, mais sur un réseau connecté à un autre routeur. Le paquet doit alors être relayé vers cet autre routeur qui prendra en charge le routage du paquet.
- La destination est inconnue. Le routeur ne peut décider vers où le paquet doit être relayé. Le paquet doit donc être éliminé et un message d'erreur ICMP (ICMPv4 ou ICMPv6 selon la version du protocole IP utilisé) est émis vers la source du paquet pour lui indiquer le problème de routage.

La détermination du cas approprié se fait à partir des informations connues par le routeur contenues dans sa **table de routage** .

La table de routage

La table de routage d'un noeud contient la liste des réseaux accessibles depuis le noeud. À chacun de ces réseaux est associé le prochain saut (*Next Hop*) pour atteindre ce réseau depuis le noeud; information qui va servir à la retransmission du paquet. Le prochain saut de la table de routage est un routeur qui est local au noeud. Ils partagent tous les deux le même préfixe réseau.

Parmi les réseaux connus dans la table de routage, on retrouve les réseaux directement connectés au noeud; c'est-à-dire que le noeud possède une interface connectée sur l'un de ces réseaux. Lorsque l'interface du noeud est configurée sur un réseau, elle obtient une adresse IPv6 à laquelle s'ajoute la longueur du préfixe; c'est-à-dire le nombre de bits communs aux adresses de toutes les interfaces connectées au même réseau. À la table de routage IPv6 s'ajoute alors automatiquement le préfixe du réseau connecté, défini par les bits communs de l'adresse. Le prochain saut pour ce réseau est alors défini par l'identifiant de l'interface connectée à ce réseau. Cela signifie au noeud que les paquets destinés à ce réseau doivent être envoyés sur cette interface.

Voici un exemple de configuration d'une interface réseau et l'entrée correspondante dans la table de routage sur un système Linux. Notez bien la correspondance entre le préfixe de l'adresse de l'interface `eth0` et l'entrée correspondante dans la table de routage.

```
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:18:73:68:21:20
          inet6 addr: 2001:db8:1:1:218:73ff:fe68:2120/64 Scope:Global
          inet6 addr: fe80::218:73ff:fe68:2120/64 Scope:Link
(...)
```

```
$ netstat -rn -A inet6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use
If
2001:db8:1:1::/64              ::                         UAe  256 0345733
eth0
(...)

$ ip -6 route
2001:db8:1:1::/64  dev eth0  proto kernel  metric 256  expires 2592155sec
mtu 1500 advmss 1440 hoplimit 0
(...)
```

La table de routage peut aussi comporter des préfixes de réseaux auxquels le noeud n'est pas directement connecté. Ces préfixes peuvent être statiquement configurés par l'administrateur réseau ou alors, appris dynamiquement grâce à des protocoles de routage. Ces préfixes peuvent être spécifiques à un réseau local (généralement de longueur 64 bits) mais peuvent être plus larges pour désigner un ensemble de réseaux. Le prochain saut est alors configuré avec l'adresse d'un routeur qui va prendre en charge la suite du routage du paquet.

L'exemple suivant montre une table de routage d'un routeur VyOS comportant un préfixe plus large que celui connecté sur son interface. Notez que l'adresse du prochain saut est une adresse **lien-local** , ce qui signifie que le noeud vers lequel transmettre le paquet est sur le réseau connecté à l'interface `eth0` .

```
vyos(config)# do show ipv6 route
C*> 2001:db8:1:1::/64 is directly connected, eth0
S*> 2001:db8:1::/48 [110/1] via fe80::290:bff:fe1e:c4fe, eth0, 1d09h16m
```

Un dernier type d'entrée de la table de routage permet à un noeud de retransmettre les paquets pour tous les réseaux qu'il ne connait pas, évitant ainsi de les éliminer parce qu'il n'a pas une connaissance suffisante du réseau. Cette entrée s'appelle la **route par défaut** . Le préfixe utilisé pour désigner ainsi tous les réseaux ne doit comporter aucun bit spécifié. En IPv6, ce préfixe se note `::/0` ; la longueur du préfixe à 0 signifiant qu'aucun bit n'est spécifié comme commun. La route par défaut possède comme prochain saut l'adresse du routeur qui prendra en charge le routage des paquets vers les réseaux non connus localement. Ce routeur est communément appelé **routeur par défaut** , ou **passerelle par défaut** . Dans un réseau local domestique par exemple, le routeur par défaut des stations, comme un ordinateur portable, est généralement le boitier de l'opérateur, car c'est lui qui sait comment joindre les différents réseaux de l'Internet.

L'exemple suivant montre l'entrée correspondant à la route par défaut d'un noeud sous Windows 7 avec l'outil en ligne de commande `netsh` .

```
netsh> interface ipv6
netsh interface ipv6 show routes
Recherche du statut actif...

Type      Mét  Préfixe                  Idx  Nom passerelle/interface
--------  ---  -----------------------  ---  ------------------------
Auto        8  2001:db8:1:1::/64          4  Connexion au réseau local 4
Auto      256  ::/0                        4  fe80::290:bff:fe1e:c4fe
```

Le test d'adjacence

Le test d'adjacence effectué par un noeud du réseau consiste à vérifier si le destinataire est directement accessible en passant par une des interfaces connectées de ce noeud:

- Pour cela, le noeud va comparer le préfixe de la destination avec les préfixes des réseaux directement connectés. En cas de correspondance, le noeud peut réaliser une remise directe. Le mécanisme ICMPv6 de **découverte des voisins** va permettre aux noeuds connectés sur le même réseau de se découvrir les uns les autres et de déterminer l'adresse physique d'un noeud à partir de son adresse IPv6. Cette fonction sera développée dans la séquence 3.
- Dans le cas présenté Figure 1, les deux stations A et B peuvent directement communiquer car ils sont connectés sur le même réseau local Ethernet à l'aide d'un commutateur qui relaie de manière transparente les trames au niveau 2. Le préfixe IPv6 `2001:db8:0001::/64` est paramétré sur chaque machine. Ainsi, les échanges sont possibles directement, sans l'intervention d'un routeur.

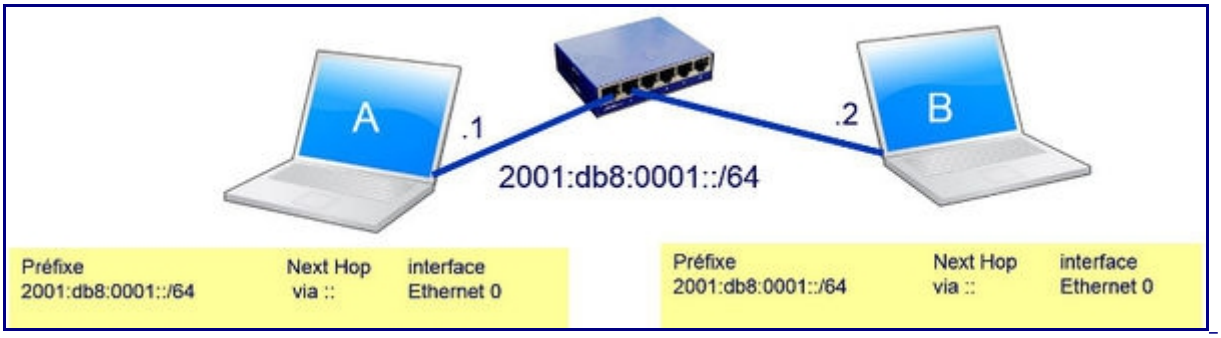

Figure 1: Routage statique direct.

Dans le cas contraire, un acheminement indirect s'impose. Le noeud doit confier les paquets vers cette destination à un autre noeud qui s'occupera de leur transfert. C'est le principe du routage indirect. Dans le cas présenté Figure 2, la station A peut atteindre les deux stations B et C. Par contre, B et C ne peuvent pas directement communiquer car elles sont connectées sur des réseaux avec des préfixes IPv6 différents:

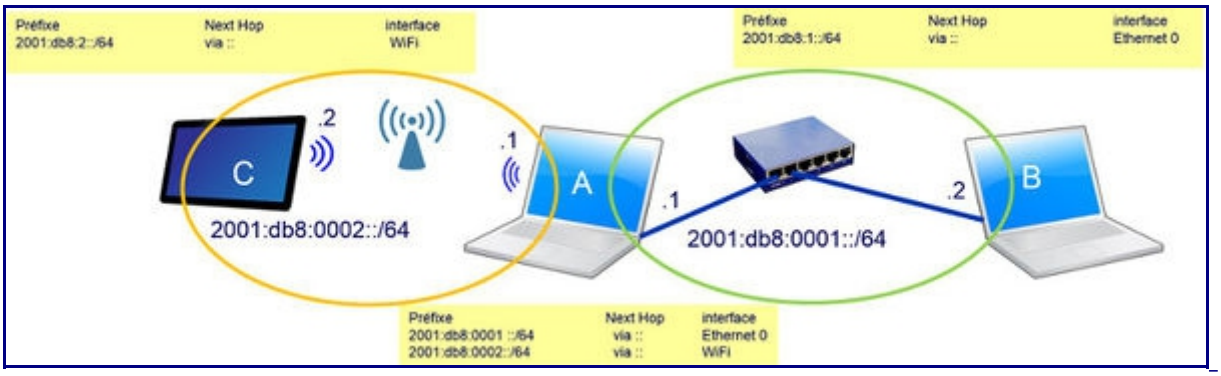

Figure 2: Routage statique indirect.

- Ainsi, dans la table de routage de B, il faudra introduire une entrée vers le préfixe distant

`2001:db8:0002::/64` en précisant l'adresse de A, `2001:db8:0001::1/64`, comme routeur par défaut, qui, lui, est directement accessible par B (cf. Figure 3). Les paquets émis depuis B vers C seront dès lors relayé.

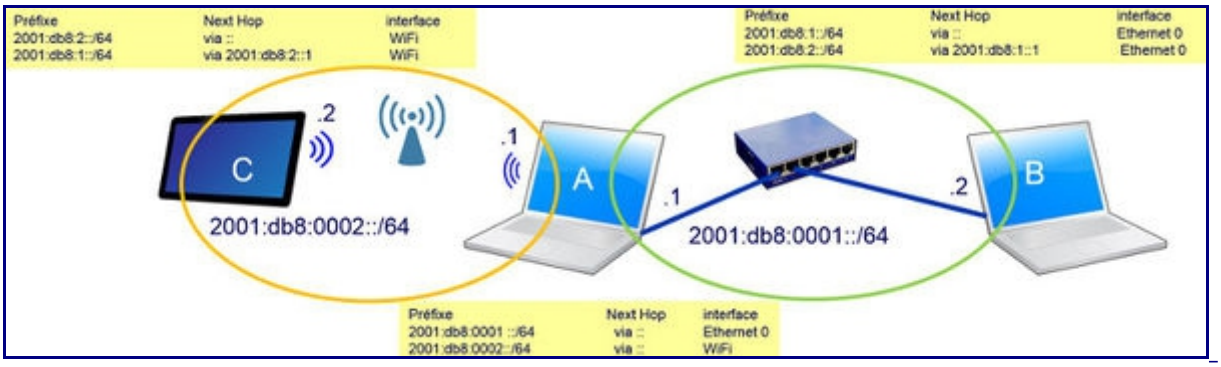

Figure 3: Routage statique indirect.

- Ensuite, il conviendra de ne pas omettre la même opération dans la table de routage de C; sans quoi, aucune réponse vers B ne sera possible. Il faudra introduire une entrée vers le préfixe distant `2001:db8:0001::/64` en précisant l'adresse de A, `2001:db8:0002::1/64`, comme routeur par défaut, qui, lui, est directement accessible par C (cf. Figure 3). Les paquets émis depuis C vers B seront dès lors relayé par A.

Routage vers Internet

La connection à Internet nécessite de pouvoir acheminer des paquets vers des réseaux qui ne sont pas connus des noeuds. La table de routage doit contenir une entrée pour indiquer vers quel routeur un noeud doit transmettre les paquets.

- Cas simple: un routeur par défaut est spécifié et tous les paquets qui visent des destinations inconnues lui seront remis. En quelque sorte, on fait confiance aux capacités et à la connectivité de ce routeur. Une route par défaut est présente dans la table de routage du routeur par défaut. L'exemple de la figure 4 montre la configuration avec une route par défaut sur le routeur IPv6:

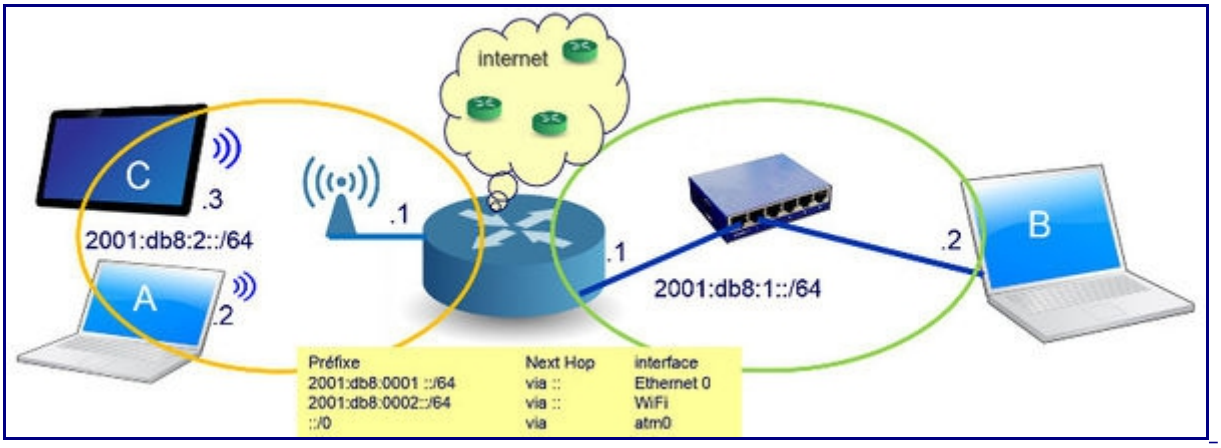

Figure 4: Routeur par défaut.

- Sur les stations, il est simple de confier tous les paquets à destination de réseaux

distants, au routeur par défaut représenté par le routeur connecté à un fournisseur d'accès à Internet. Une simple route par défaut est ajoutée à chaque station. La Figure 5 montre la table de routage des stations avec la route par défaut. Dans notre exemple, le routeur par défaut est le routeur local de la station A. Il n'y a pas de routeur intermédiaire entre la station A et le routeur par défaut.

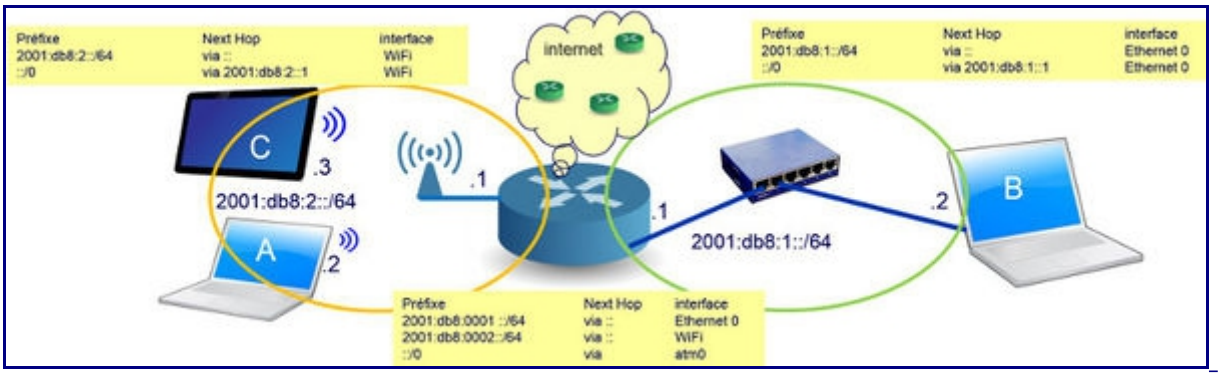

Figure 5: Route par défaut dans les stations.

- Des routes spécifiques peuvent être définies dès lors que l'on dispose d'une connectivité bien adaptée pour certains préfixes. Dans ce cas, une configuration manuelle de la table de routage est nécessaire.
- Les routes les plus spécifiques, c'est-à-dire celles avec un long préfixe, seront traitées en premier; puis, les routes moins spécifiques; et enfin, la route par défaut en dernier ressort.

Pour aller plus loin: Le routage dynamique

Comme pour IPv4, il faut faire la distinction entre deux grandes familles de protocoles de routage: les protocoles de routage interne (*Interior Gateway Protocols* , IGP) et les protocoles de routage externe (*Exterior Gateway Protocols* , EGP). La différence provient de la notion de **système autonome** (*Autonomous System* , AS), par la définition de la portée des échanges d'informations de routage des protocoles de routage. Ainsi, la propagation des préfixes réseaux internes à un AS se fait par un IGP, alors que les annonces de préfixes entre AS se fait par un EGP.

Pour connecter un site à l'Internet, il faut donc mettre oeuvre des protocoles de routage interne et des protocoles de routage externe. Ce chapitre traite des trois protocoles IGP suivants: RIPng (équivalent de RIPv2 pour IPv4), ISIS et OSPFv3 (équivalent d'OSPFv2 pour IPv4), ainsi que du protocole de routage externe BGP.

Les protocoles de routage interne visent à rendre automatique la configuration des tables de routage des routeurs à l'intérieur d'un même système autonome. Les routeurs déterminent le plus court chemin pour atteindre un réseau distant. Les protocoles de routage internes nécessitent une configuration minimale du routeur, notamment en ce qui concerne les annonces de routes initiées par ce routeur (exemple: réseaux directement accessibles par une interface du routeur, routes statiques...).

Deux types de protocoles de routage interne existent: les protocoles à vecteur de distance (

distance vector), et le protocoles à état de lien (*link state*). Les premiers génèrent des annonces de routeur, transmises aux routeurs voisins, qui contiennent des informations de direction (les réseaux accessibles par le routeur) et de distance associée aux destinations annoncées (la métrique, qui peut être le nombre de routeurs à traverser, pour RIP, mais qui peut être un coût lié au débit, comme pour EIGRP). Pour les protocoles à état de lien, les annonces de routeur ne contiennent plus d'informations issues de la table de routage, mais des informations sur les liens auxquels sont connectés les routeurs (adresses IP, nature du lien, coût calculé souvent à partir du débit, etc). Chaque routeur construit une base de données d'états de liens (*link state database*), qui permet de redessiner la topologie du réseau. Dans un second temps, un algorithme de recherche du plus court chemin permet à chaque routeur de construire sa table de routage, à partir de cette base de données.

RIPng ou RIP IPv6

RIPv2 (RFC 2453) est un algorithme à vecteur de distance, basé sur l'algorithme de Bellman-Ford et figure parmi les premiers algorithmes de routage interne utilisés dans l'Internet.

Les routeurs diffusent leurs tables de routage sur les liens auxquels ils sont connectés. Les autres routeurs modifient une route dans leur table si la **métrique** (le nombre de routeurs à traverser pour atteindre une destination) reçue est plus petite que celle déjà stockée dans la table. Si une annonce de route n'est pas présente dans la table, le routeur l'ajoute. Ces modifications sont à leur tour diffusées sur les autres réseaux auxquels sont connectés les routeurs. Elles se propagent donc sur l'ensemble du réseau à l'intérieur du système autonome. On montre que cet algorithme converge et, qu'en condition stable, aucune boucle n'est créée sur le réseau, c'est-à-dire qu'un paquet ne sera pas transmis indéfiniment de routeur en routeur sans jamais pouvoir atteindre sa destination.

Les tables sont émises périodiquement. Si un routeur tombe en panne, ou si le lien est coupé, les autres routeurs ne recevant plus l'information suppriment l'entrée correspondante de leur table de routage. RIPng est le premier protocole de routage dynamique proposé pour IPv6 (RFC 2080). RIPng est une simple extension à IPv6 du protocole RIPv2 d'IPv4. Il en hérite les mêmes limitations d'utilisation (maximum de 15 sauts par exemple).

ISIS

IS-IS (*Intermediate System to Intermediate System*) est un protocole de routage interne à état de lien. Il a été standardisé par l'ISO (ISO 10589). C'est un protocole de niveau 3 (contrairement à OSPF et RIP, de niveau 4) qui s'appuie sur une couche 2 de type Ethernet 802.2. Cet élément mérite d'être signalé car cela rend ce protocole indépendant d'IP, que ce soit IPv4 ou IPv6. Ce protocole travaille sur deux niveaux de hiérarchie: les aires (niveau 1) et le *backbone* (niveau 2).

Un routeur IS-IS peut être:

- *level-1* (routage intra aire),
- *level-2* (routage inter aire),
- ou *level-1-2* (routage intra et inter aire).

Un routeur de niveau 1 n'a de voisins que dans son aire alors qu'un routeur de niveau 2 peut avoir des voisins dans une autre aire. Il n'y a pas d'aire de *backbone* (contrairement à OSPF). Le *backbone* est constitué de la réunion de tous les routeurs de *level-2* . Sur un réseau de type LAN, il y a élection d'un routeur désigné (DIS) qui a la charge de produire les annonces.

Afin de construire sa topologie, IS-IS utilise 3 types de messages:

- les messages HELLO permettant de construire les adjacences;
- les messages LSP (*Link State Protocol*) permettant d'échanger les informations sur l'état des liens;
- les messages SNP (*Sequence Number Packet*) permettant de confirmer la topologie.

Pour élaborer ces messages, IS-IS se base sur l'utilisation d'éléments d'informations indépendants appelés TLV (Type, Longueur, Valeur). Le message est ainsi constitué d'un en-tête suivi d'une liste de TLV. Chaque TLV véhicule une information propre, et est donc standardisée. L'exemple ci-dessous montre une TLV **Protocoles supportés** faisant partie d'un message HELLO, informant les voisins des protocoles supportés par l'émetteur du paquet:

- 0x81 0x02 0xcc 0x8e
    - Le premier octet donne le type de la TLV. Il s'agit ici du type 0x81, c'est-à-dire **Protocoles supportés** .
    - Le second octet donne la longueur en octets de la TLV: ici, les deux octets qui suivent.
    - Les autres octets composent la valeur de la TLV. Ici, nous avons deux octets indiquant des numéros de protocoles supportés (NLPID: *Network Layer Protocol IDentifier*): 0xCC pour IPv4 et 0x8E pour IPv6.

OSPFv3

Le troisième protocole de routage interne, basé sur l'algorithme du plus court chemin (SPF, *Shortest Path First* , ou algorithme de Dijkstra), s'appelle OSPF (*Open Shortest Path First*). Relativement plus complexe à mettre en oeuvre que RIPng, il est beaucoup plus efficace dans les détections et la suppression des boucles dans les phases transitoires. Ce protocole est basé sur plusieurs sous-protocoles, dont un qui permet une inondation fiable du réseau. Les routeurs possèdent alors chacun une copie des configurations de tous les routeurs présents sur le réseau, et peuvent calculer simultanément le plus court chemin pour aller vers l'ensemble des destinations.

Pour réduire la durée des calculs, et surtout pour éviter un recalcul complet des routes à chaque changement de configuration, OSPF offre la possibilité de découper le réseau en aires. Une aire principale appelée *backbone* relie toutes les autres aires. Les réseaux trouvés dans une aire donnée sont envoyés aux autres aires par les routeurs qui sont en frontière d'aire.

OSPF a été adapté à IPv6 (RFC 2740); la version est passée de 2 à 3. La plupart des algorithmes implémentés dans OSPFv2 ont été réutilisés en OSPFv3. Bien évidemment, certains changements ont été nécessaires en vue de l'adaptation aux fonctionnalités d'IPv6.

BGP

BGP-4 est le protocole de routage externe actuellement utilisé pour le routage global de l'Internet IPv4 (le numéro de version 4, identique pour BGP et IP, est une pure coïncidence) [2] . Compte tenu de sa criticité, ce protocole est l'objet d'évolutions constantes. L'une d'entre elles est le RFC 4760 qui rend BGP-4 "multi-protocole" en introduisant la notion de famille d'adresses (ex. IPv4, IPv6, IPX...) et de sous-famille d'adresses (ex. *unicast* , *multicast*). Le RFC 2545 précise l'usage des extensions multi-protocoles pour le cas d'IPv6.

L'adaptation multi-protocole de BGP-4 est assez simple car elle ne concerne que les trois attributs dont le format dépend de l'adresse, soit:

- NLRI: *Network Layer Reachability Information* (suite de préfixes);
- NEXT_HOP: Adresse IP où il faut router les NLRI;
- AGGREGATOR: Adresse IP du routeur qui a fait une agrégation de préfixes.

Pour réaliser pratiquement cette adaptation, BGP4+ introduit deux nouveaux attributs:

- MP_REACH_NLRI: *Multiprotocol Reachable NLRI* ,
- MP_UNREACH_NLRI: *Multiprotocol Unreachable NLRI* ,

qui indiquent que l'on annonce des informations de routage autres que les routes *unicast* IPv4. Ces attributs codent en premier le type de famille et de sous-famille d'adresses, puis les attributs dont le format est spécifique. Les autres attributs (comme le chemin d'AS *Autonomous System*) sont codés et annoncés sans changement.

Les implémentations du RFC 4760 sont souvent appelées MP-BGP (pour faire référence à leur capacité de traitement des routes *multicast*) ou BGP4+ (pour faire référence à leur capacité de traitement de routes IPv6). Pour l'anecdote, le numéro de version du protocole n'a pas été modifié (en BGP-5 par exemple) car le passage de BGP-3 à BGP-4 rappelle trop de souvenirs douloureux à ceux qui l'ont mis en oeuvre. Les numéros d'AS utilisés pour IPv4 servent aussi pour IPv6.

Conclusion

Cette activité vous a présenté le principe du routage IP, et la table de routage. Cet élément essentiel de la couche réseau, présent dans chaque noeud de l'Internet, indique à un routeur qui a un paquet à transmettre à qui doit être remis ce paquet: à un routeur local, indiqué dans la table de routage comme prochain associé à l'adresse de destination contenue dans la paquet, ou directement à la destination. La configuration correcte de la table de routage est donc importante aussi bien sur les routeurs que sur les stations.

Cette configuration peut se faire manuellement par l'administrateur du réseau: on parle alors de routage statique. Afin de pouvoir s'adapter à l'évolution du réseau, les tables de routage peuvent être mise à jour par des protocoles de routage dynamique permettant de propager les modifications de la topologie du réseau. Les tables de routage sont mises à jour automatiquement au fur et à mesure des changements dans le réseau.

Références bibliographiques

1. ↑ Rubino, G. et Toutain, L. (2000). Techniques de l'ingénieur. Routage dans les réseaux Internet
2. ↑ Balakrishnan, H. et Feamster, N. (2005), Lecture notes. Interdomain Internet Routing.

Pour aller plus loin

RFC et leur analyse par S. Bortzmeyer: Vous pouvez approfondir vos connaissances sur les protocoles de routage en consultant les liens suivants:

RIPng:

- Article dans le livre "IPv6, Théorie et Pratique"
- RFC 2453 : RIP Version 2
- RFC 4822 : RIPv2 Cryptographic Authentication

ISIS:

- Article dans le livre "IPv6, Théorie et Pratique"
- ISO-IEC 8473 Information technology — Protocol for providing the connectionless-mode network service: Protocol specification

OSPF:

- Article dans le livre "IPv6, Théorie et Pratique"
- RFC 5340 : OSPF for IPv6 (Analyse par S.Bortzmeyer)
- RFC 7503 : OSPFv3 Autoconfiguration (Analyse par S.Bortzmeyer)

BGP:

- Article dans le livre "IPv6, Théorie et Pratique"
- RFC 2545 : Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing
- RFC 3849 : IPv6 Address Prefix Reserved for Documentation
- RFC 4760 : Multiprotocol Extensions for BGP-4 (Analyse par S.Bortzmeyer)
- RFC 5963 : IPv6 Deployment in Internet Exchange Points (IXPs) (Analyse par S.Bortzmeyer)

Activité 24: Le mécanisme d'extension de l'en-tête IPv6

Introduction

Les extensions de l'en-tête IPv6 visent à ajouter des fonctionnalités supplémentaires à l'acheminement d'un paquet dans l'Internet. Ces extensions vont impliquer le traitement de ces fonctionnalités au niveau réseau:

- soit par le destinataire du paquet IPv6,
- soit par les routeurs intermédiaires en charge de l'acheminement du paquet IPv6.

De nombreuses extensions ont été définies, afin d'assurer des fonctions comme:

- le routage par la source,
- la gestion de la fragmentation,
- la confidentialité des communications (mécanisme *ipsec*),
- etc.

Le mécanisme d'extension de l'en-tête IPv6 est assez souple pour pouvoir inclure d'autres fonctionnalités futures. Dans cette activité, nous allons nous intéresser au principe du traitement des extensions au moyen d'exemples simples et démonstratifs de ce mécanisme.

Principe des extensions IPv6

Les extensions peuvent être vues comme un protocole 3.5, entre les couches 3 (réseau) et 4 (transport) du modèle OSI. En effet, à part l'extension de proche-en-proche, qui est traitée par tous les routeurs traversés, les autres extensions ne sont traitées que par le destinataire du paquet (i.e. celui spécifié dans le champ adresse de destination du paquet IPv6). Une extension a une longueur multiple de 8 octets (64 bits). Elle commence par un champ `Next header` qui définit sur 1 octet le type d'extension ou de protocole qui suit. Pour les extensions de longueur variable, l'octet suivant contient la longueur de l'extension en mots de 8 octets, le premier mot n'étant pas compté. Par exemple, une extension de 16 octets aura une longueur de 1.

Si d'un point de vue théorique les extensions sont supérieures aux options d'IPv4, dans la réalité très peu sont utilisées à grande échelle et restent du domaine de la recherche.

Nota: *Dans la pratique, l'extension la plus couramment rencontrée est probablement l'option de fragmentation à la source. En effet, certains protocoles applicatifs sur UDP, tel que NFS, supposant que la fragmentation existe au niveau réseau, produisent des messages de taille quelconque sans se soucier du MTU. Comme il n'est pas envisageable de modifier ces applications largement déployées, la couche réseau IPv6 doit être capable de gérer la fragmentation. IPv6 impose, simplement, que cette dernière se fasse à la source.*

Une présentation illustrée des extensionsIPv6 peut être consultée sur le site de Cisco [1] .

Le champ *Next Header*

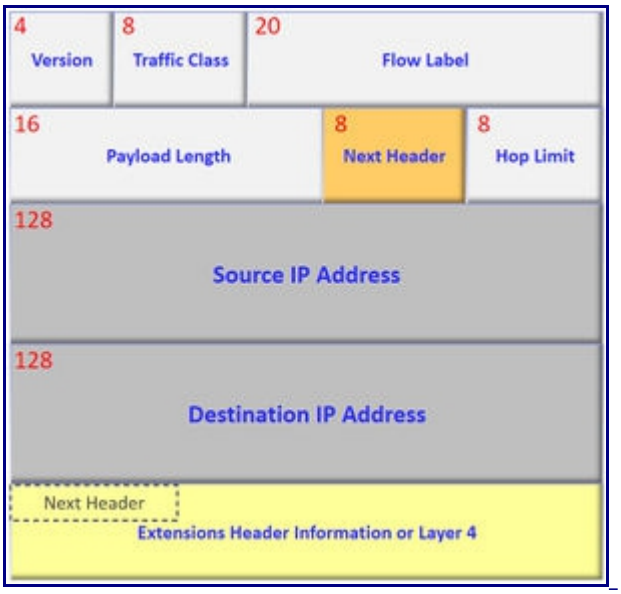

Figure 1: Localisation du champ `Next Header` dans l'en-tête IPv6.

Le champ `Next Header` de l'en-tête IPv6, comme illustré sur la figure 1, identifie généralement le protocole de niveau supérieur comme, par exemple, le transport TCP/UDP. Mais, dans le cas des extensions, plusieurs mécanismes particuliers sont également disponibles parmi la liste suivante:

Valeurs du champ en-tête suivant pour IPv6

valeur	**Hexa**	**Protocole ou *Extension***
0	0x00	*Proche-en-proche*
4	0x04	IPv4
6	0x06	TCP
17	0x11	UDP
41	0x29	IPv6
43	0x2b	*Routage*
44	0x2c	*Fragmentation*
50	0x32	*Confidentialité*
51	0x33	*Authentification*
58	0x3a	ICMPv6
59	0x3b	Fin des en-têtes
60	0x3c	*Destination*
132	0x84	SCTP
135	0x87	*Mobilité*

136	0x88	UDP-lite
140	0x8c	*Shim6*

Intégration des extensions d'en-tête dans le paquet IPv6

Quand il y a plusieurs extensions dans un même datagramme, les extensions sont placées selon un ordre qui dépend de leur portée:

- extensions impliquant tous les routeurs intermédiaires: *Hop-by-Hop* ;
- extensions impliquant seulement certains routeurs désignés: *Routing* ;
- extension impliquant le destinataire: *Authentication* , *Encapsulating Security Payload* , Fragmentation, Destination.

La figure 2 montre la souplesse avec laquelle plusieurs extensions peuvent être chaînées. Chaque extension contient dans son en-tête un champ `en-tête suivant` et un champ `longueur` . Le premier paquet ne contient pas d'extension; le champ `en-tête suivant` pointe sur ICMPv6. Le second paquet ne contient pas d'extension; le champ `en-tête suivant` pointe sur TCP. Le troisième paquet contient une extension de protection qui pointe ensuite sur UDP. Dans le dernier paquet, une extension de routage, qui pointe sur une extension de fragmentation, pointe finalement sur ICMPv6.

- L'enchaînement des extensions se fait dans un ordre bien déterminé. Par exemple, une extension concernant tous les routeurs sur le chemin (*Hop-by-Hop*) devra forcément se trouver en première position. Si elle se trouvait à la suite d'une extension *Destination* , elle ne pourrait être lue, l'extension *Destination* n'étant interprétée que par le destinataire du paquet.
- Si cet enchaînement d'extension offre beaucoup plus de souplesse que les options d'IPv4, il rend difficile la lecture des numéros de port. Il faut en effet lire tout l'enchaînement d'extension pour arriver au protocole de niveau 4. Ceci a servi de justification à l'identificateur de flux qui permettait de refléter au niveau 3 un flux particulier et évitait de dérouler l'enchaînement. Bien entendu, les pare-feux devront vérifier les numéros de ports.
- Les extensions peuvent être vues comme un protocole 3.5 (entre la couche 3 et la couche 4). En effet, à part l'extension de "proche en proche", qui est traitée par tous les routeurs traversés, les autres extensions ne sont traitées que par le destinataire du paquet (*i.e.* celui spécifié dans le champ `adresse de destination` du paquet IPv6).

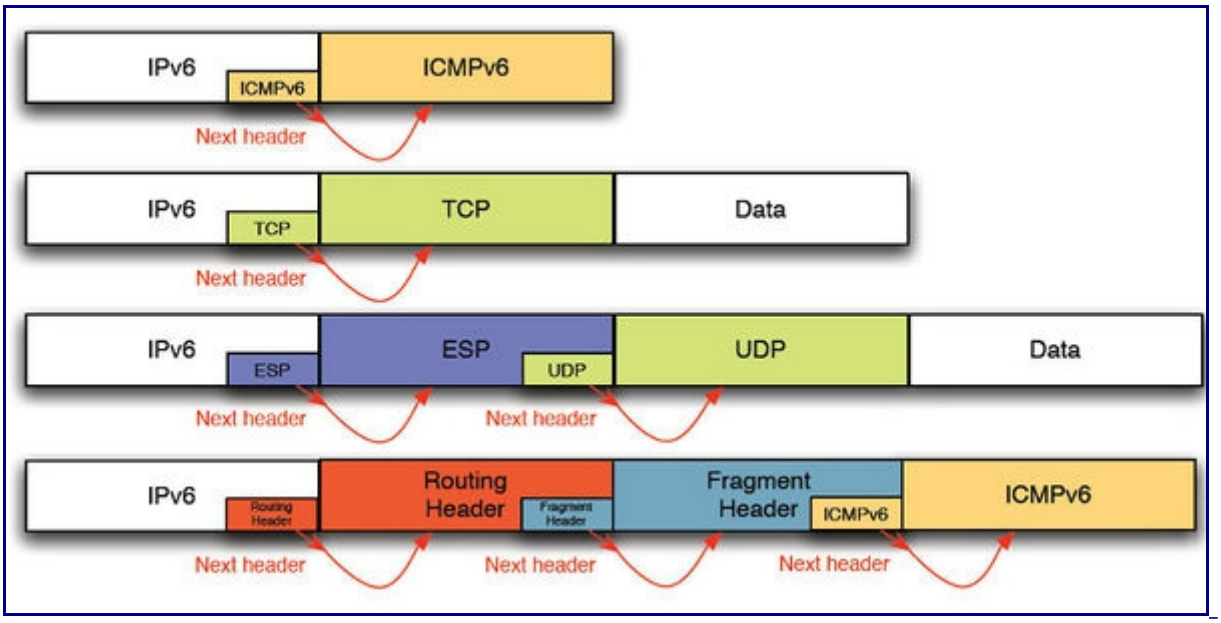

Figure 2: Enchaînement d'extensions.

Quelques exemples

Fragmentation

La fragmentation, telle qu'elle est pratiquée dans IPv4, n'est pas très performante. Initialement, elle servait à rendre transparente les limitations physiques des supports de transmission. Dans IPv4, quand un routeur ne peut pas transmettre un paquet à cause de sa trop grande taille, et si le bit DF *don't fragment* est à 0, il découpe l'information à transmettre en fragments. Or, le réseau IP étant un réseau à datagrammes, il n'y a pas de possibilité de contrôler les fragments. Deux fragments successifs peuvent prendre deux chemins différents et, par conséquent, seul le destinataire peut effectuer le ré-assemblage. En conséquence, après la traversée d'un lien impliquant une fragmentation, le reste du réseau ne voit passer que des paquets de taille réduite.

Il est plus intéressant d'adapter la taille des paquets à l'émission. Ceci est fait en utilisant les techniques de découverte du MTU (voir RFC 8201 : Mécanisme de découverte du PMTU). En pratique, une taille de paquets de 1 500 octets est presque universelle.

Il existe pourtant des cas où la fragmentation est nécessaire. Ainsi, une application telle que NFS sur UDP suppose que la fragmentation existe et produit des messages de taille quelconque. Comme on ne veut pas modifier ces applications, la couche réseau d'IPv6 doit aussi être capable de gérer la fragmentation. Pour réduire le travail des routeurs intermédiaires, la fragmentation se fera au niveau de la source et le ré-assemblage par le destinataire. Les informations utiles pour ces mécanismes (identification du paquet fragmenté, place du fragment) sont transportées dans une extension de fragmentation, représentée dans la figure 3.

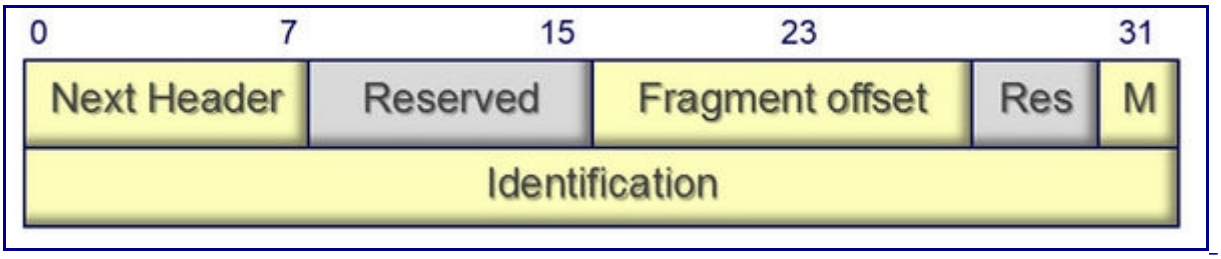

Figure 3: Format de l'extension de fragmentation.

- Le champ longueur de l'extension est remplacé par un champ réservé, positionné pour l'instant à 0. Un champ de longueur est, car l'extension tient sur un seul mot de 64 bits et le premier mot n'est pas compté dans le calcul de longueur des extensions.
- Le `fragment offset` indique, lors du ré-assemblage, la position à laquelle les données du fragment doivent s'insérer. Ceci permet de résoudre les problèmes dus au déséquencement éventuel des datagrammes. Ce champ étant sur 13 bits, la taille des fragments, excepté le dernier, doit être multiple de 8 octets (alignement sur frontière de mots de 64 bits),
- Le bit `M` (More data) est à 1 sur les fragments intermédiaires et à 0 sur le fragment final,
- Le champ `identification` permet de repérer les fragments appartenant à un même paquet initial pour une source et une destination données.

Le bit DF (Don't Fragment), de l'en-tête IPv4 n'est plus nécessaire. Si un paquet est trop grand, il y aura rejet par le routeur et émission d'un message ICMP.

Dans IPv6, l'en-tête et les extensions qui concernent les routeurs intermédiaires (pour l'instant proche en proche et routage par la source) sont recopiées dans chaque fragment, tandis les autres extensions et l'en-tête de niveau 4 ne seront présents que dans le premier fragment.

Le mécanisme de gestion de la taille d'un paquet IPv6, selon les différents cas possibles, est approfondi dans l'activité suivante.

Extensions d'authentification (AH) et de confidentialité (ESP)

L'extension d'authentification (AH: Authentication Header) décrite dans le RFC 4302 permet de s'assurer que l'émetteur du message est bien celui qu'il prétend être. Elle sert aussi au contrôle d'intégrité pour garantir au récepteur que personne n'a modifié le contenu d'un message lors de son transfert sur le réseau. Elle peut optionnellement être utilisée pour détecter les rejeux.

Le principe de l'authentification est relativement simple. L'émetteur calcule un authentificateur sur un datagramme et l'émet avec le datagramme sur lequel il porte. Le récepteur récupère cette valeur et vérifie qu'elle est correcte. C'est-à-dire, si un code d'authentification de message [2] est utilisé, il lui suffit de calculer de son côté le code sur le même datagramme à l'aide de la clé symétrique partagée et de le comparer avec le code reçu. Si le mécanisme de signature numérique est employé, le récepteur doit alors récupérer la signature, la déchiffrer avec la clé publique de l'émetteur et comparer le condensat ainsi obtenu avec celui calculé de son côté sur le datagramme reçu. Si les deux codes, ou les deux condensats diffèrent, soit l'émetteur ne possède pas la bonne clé, soit le message a subi des modifications en chemin.

L'extension ESP *Encapsulating Security Payload* décrite dans le RFC 4303 permet de chiffrer l'ensemble des paquets ou leur partie transport et de garantir l'authentification et l'intégrité de ces paquets. Cette extension permet optionnellement de détecter les rejeux (à condition que le service d'authentification soit assuré) et garantit de façon limitée la confidentialité du flux.

Pour obtenir ces services de sécurité, il est nécessaire, avant d'émettre un paquet IP sur le réseau, de chiffrer les données à protéger, de calculer un authentificateur, et d'encapsuler ces informations dans l'en-tête de confidentialité ESP. Cela nécessite bien entendu l'existence d'une association de sécurité précisant, entre autres, le ou les algorithme(s) de chiffrement, la ou les clé(s), et un indice de paramètres de sécurité. Pour en savoir plus sur les extensions de sécurité, nous renvoyons le lecteur vers le livre [3] .

Segment Routing Header (SRH)

Cette extension est une des briques qui permet la mise en oeuvre de l'ingénierie de trafic et de la qualité de service dans les réseaux IPv6. Certaines applications ont besoin que le délai, le taux de perte des paquets et le débit ne dépassent pas un certain seuil. Par exemple, une session de vidéo conférence peut nécessiter que le délai de bout en bout reste inférieur à 100ms, que le débit disponible soit au minimum de 2Mbit/s et que le taux de perte reste inférieur à 1%. La solution habituellement mise en oeuvre pour le respect de ces métriques de qualité de service est RSVP-TE [4] . RSVP est un protocole de réservation de ressources sur les routeurs du chemin entre une source et une destination. La composante d'ingénierie de trafic permet notamment de choisir explicitement le chemin qui sera suivi par le paquet. Lorsqu'une application a besoin de réserver des ressources, un circuit MPLS est établi et le protocole RSVP réserve les ressources sur l'ensemble des routeurs qui composent le chemin. La figure 4 résume ce comportement, le nœud A est la source du flot de vidéo conférence destiné à B, elle demande l'installation d'un chemin via MPLS. Le contrôleur détermine que le chemin le plus adapté est celui transitant par G,H,I,K,N source de vidéo conférence. Elle effectue ensuite une réservation de ressource via RSVP. Chacun des routeurs G,H,I,K,N doivent accepter la requête.

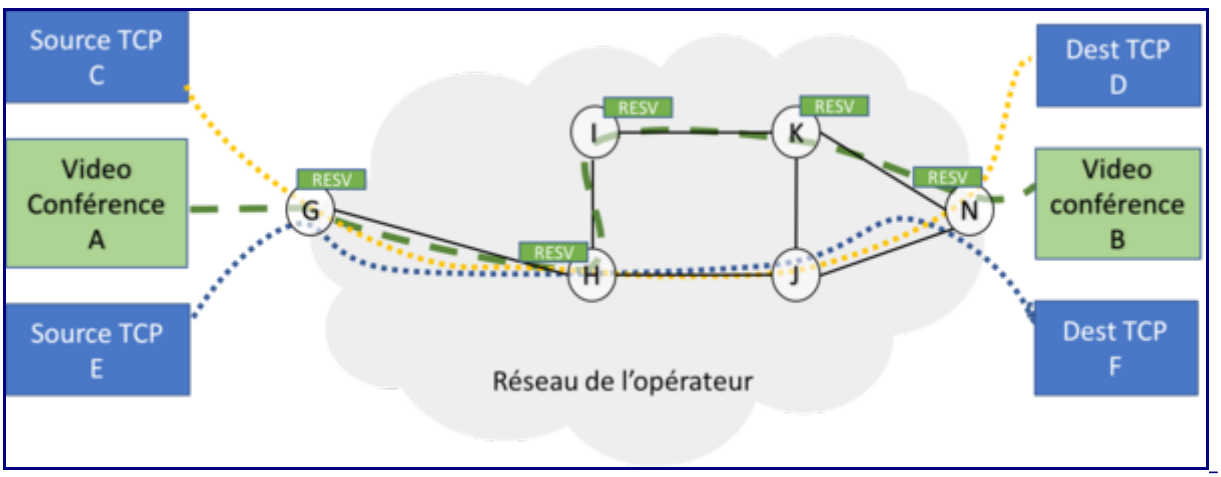

Figure 4: Exemple d'utilisation de RSVP-TE pour assurer la qualité de service.

L'inconvénient de cette méthode est qu'avec RSVP, les réservations doivent être fréquemment renouvelées et les routeurs doivent garder des états relatifs à chacun des flots. A chaque fois qu'un routeur reçoit un paquet, il doit chercher dans la liste des flots le comportement à y associer. Cela limite le passage à l'échelle (le nombre de flots supportés). De plus, l'établissement des circuits MPLS est coûteux en temps, en états, et en communication. Pour finir, si un des équipements ou des liens qui compose le chemin n'est plus disponible, le temps requis pour établir un nouveau chemin peut être trop important face aux besoins des flots [5] .

Le concept de segment routing a été proposé pour palier à ces désavantages. Il permet de réduire l'intelligence des équipements du réseau. Un contrôleur choisit le chemin qui respecte les besoins de qualité de service du noeud. Le chemin à suivre ainsi que le traitement à associer aux paquets sont ensuite explicitement listés dans leurs en-têtes. Ces traitements à appliquer au paquet sont appelées segments. Un segment peut être l'adresse d'un routeur par lequel le paquet devra transiter, un lien à emprunter ou encore un service fourni par le routeur sur lequel transite le paquet. La séquence de segments que doit suivre le paquet peut être implémentée via une liste de label MPLS ou dans le cas d'en réseau IPv6 via l'option Segment Routing de l'en-tête [6] . Contrairement à MPLS, le segment routing avec IPv6 ne requière pas que tous les équipements traversés par le paquet soient compatibles avec cette option. Cela permet un déploiement progressif et la traversée de plusieurs types de réseaux. Notons qu'il n'est pas nécessaire d'inscrire l'intégralité du chemin dans la liste. Les règles de routage classique de l'IGP s'appliquent jusqu'au prochain nœud explicitement listés dans l'en-tête. Dans le cas de la figure 5, pour suivre le chemin G,H,I,K,N, il est seulement nécessaire d'ajouter I et B dans la liste. Lorsque le paquet est généré par A, l'adresse de destination est celle de I. Lorsque le paquet atteint I, l'adresse de destination est remplacée par la prochaine dans la liste c.a.d. B. Le paquet suit le plus court chemin entre I et B en l'occurrence I,K,N. Le surcoût induit par la taille de l'extension de l'en-tête est donc limité.

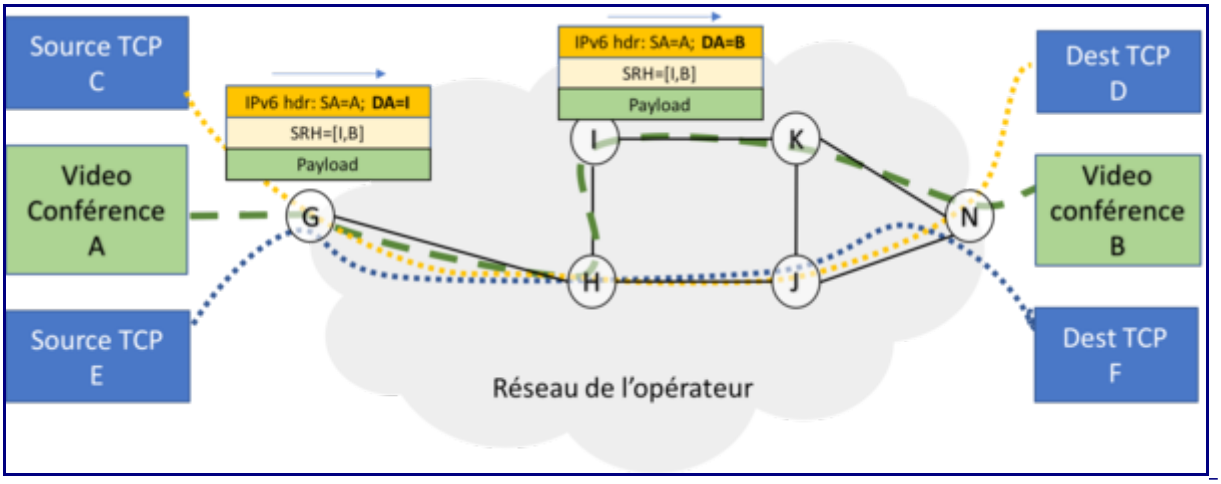

Figure 5:Exemple d'utilisation du Segment Routing pour l'ingénierie de trafic et la qualité de service.

Le projet de RFC prévoit qu'un segment soit représenté sur 128bit, c.a.d. autant qu'une adresse IPv6. Le nombre de segments inclus dans l'option SRH peut être déduit du champ `Hdr Ext Len` qui indique le nombre d'octets de l'option moins un. Le prochain segment à appliquer est indiqué par le champ `Segments Left` qui indique nombre de segments restant à appliquer au paquet. Le segment courant est inscrit dans le champ adresse de destination de l'en-tête IPv6. Lorsque le traitement associé à un segment est achevé, le prochain segment à traiter est copié dans le champ adresse de destination et le champ `Segments Left` est décrementé.

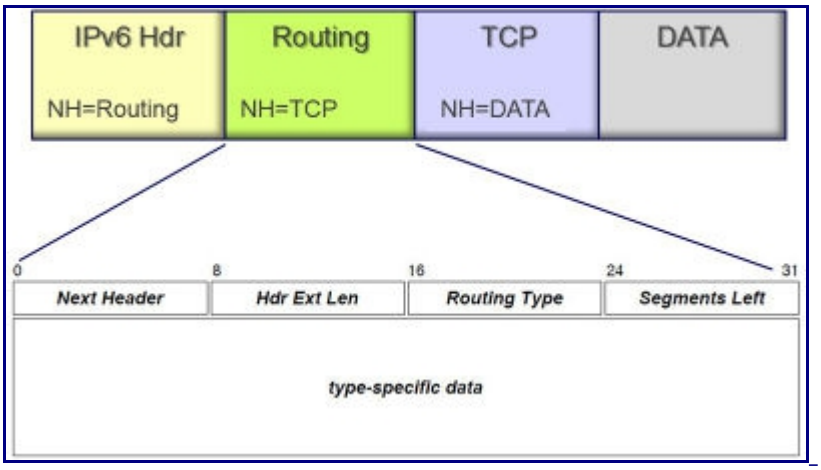

Figure 6: Extension de routage par la source.

L'extension SRH est très proche de l'en-tête de routage défini dans le RFC 8200 de IPv6. L'en-tête de routage devait notamment permettre le routage à la source (voir la figure 6) qui consiste à lister les routeurs que devaient traverser le paquet au cours de son acheminement. Cette pratique a été dépréciée en 2007 par la RFC 5095 pour des raisons de sécurité. En effet cela permettait de générer de la congestion sur certains chemins, des attaques de déni de service, des contournements des règles de filtrages de certains pare-feu ou encore d'utiliser des machines publiques accessibles (en DMZ) pour accéder à des machines privées. [7] .

Pour être acceptée, l'extension SRH doit donc garantir qu'elle ne serait pas vulnérable à ces attaques. Plusieurs cas d'utilisation sont décrits. Lorsque l'extension SRH est générée au sein du réseau de l'opérateur (e.g. routeur de bordure), les attaques listées dans la RFC 5095 sont inopérantes. Les routeurs de bordures doivent simplement filtrer les paquets provenant de l'extérieur pour éviter des hôtes s'attribuent des privilèges ou provoquent de la congestion sur certains liens. Lorsque les SRH sont générées en dehors du réseau de l'opérateur, le projet de RFC prévoit l'utilisation d'un mécanisme d'authentification de l'en-tête défini par le RFC 2104 (option HMAC). La clé de chiffrement de l'en-tête HMAC est distribuée au sein des noeuds autorisés à générer ou manipuler des extensions d'en-têtes SRH.

Conclusion

Cette activité vous a décrit les différents mécanismes qui enrichissent les fonctions disponibles au niveau de la couche réseau: routage, fragmentation, sécurité, etc. Ces mécanismes tirent

parti de la possibilité d'ajouter des champs supplémentaires à l'en-tête IPv6 grâce aux extensions d'en-tête. Ces extensions sont ajoutées par les extrémités de la communications et sont transparentes pour les routeurs (exception faite des extensions de type *Hop-by-Hop*). De plus, le mécanisme de chainage par le champ *Next Header* permet d'ajouter des extensions de manière souple.

L'usage des extensions est encore assez limité sur l'Internet. Certaines fonctionnalités ont été dépréciées (comme l'extension de routage RH0) et d'autres peinent à se développer (comme la mobilité IPv6). La présence potentielle de ces extensions doit être cependant pris en compte dans le traitement des paquets, notamment le filtrage sur les valeurs du champ *Next Header* de l'en-tête IPv6.

Références bibliographiques

1. ↑ Cisco (2006) White paper. IPv6 Extension Headers Review and Considerations
2. ↑ Code d'authentification de message, Article Wikipedia
3. ↑ G. Cizault. livre "IPv6, Théorie et Pratique". Chapitre sur la Sécurité
4. ↑ Packet Pushers, Novembre 2016. Deep dive in the RSVP-TE protocol
5. ↑ Cisco, SR Traffic Engineering
6. ↑ Previdi al. IPv6 Segment Routing Header (SRH) (Draft)
7. ↑ Philippe BIONDI et Arnaud EBALARD, CanSecWest, 2017 . IPv6 Routing Header Security

Pour aller plus loin

Vous pouvez approfondir vos connaissances en consultant les documents de ce paragraphe. RFC et leur analyse par S. Bortzmeyer:

- RFC 4302 : IP Authentication Header
- RFC 4303 : IP Encapsulating Security Payload (ESP)
- RFC 5095 : Deprecation of Type 0 Routing Headers in IPv6 Specification Analyse
- RFC 7045 : Transmission and Processing of IPv6 Extension Headers Analyse
- RFC 7872 : Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World Analyse
- RFC 8200 : Internet Protocol, Version 6 (IPv6) Specification Analyse
- RFC 8201 : Path MTU Discovery for IP version 6 Analyse

Activité 25: La taille des paquets IPv6

Introduction

La notion de datagramme, sur laquelle repose le protocole IP, implique un découpage des données provenant de la couche transport afin que ces données puissent être transportées dans des paquets. Ces paquets sont ensuite encapsulés dans des trames avant d'être émis sur le support physique. Le protocole IP est lui-même transporté par la couche liaison sur des supports physiques qui peuvent être de natures différentes, impliquant des tailles de trames de longueur variable. La gestion de la taille du paquet sur le chemin est donc nécessaire pour permettre la transmission des données de manière transparente d'un bout à l'autre de l'Internet. Ce problème de la taille du paquet à transférer existe tout aussi bien en IPv4 qu'en IPv6. La solution pour régler le problème est différente entre IPv4 et IPv6 [1] .

Cas nominal (taille paquet inférieure à la PMTU)

La couche réseau a pour tâche de placer les segments provenant de la couche transport (données utiles + en-tête transport) dans des paquets. Ces paquets sont ensuite encapsulés dans des trames (PDU de niveau 2) sur le support physique (cf. Figure 1). Ce support, selon sa nature, définit une taille maximale du champ données de la trame:

- Ethernet (1500 octets),
- PPPoA (1468 octets),
- MPLS (de 1500 à 65535 octets),
- etc.

Cette taille fixe donc, pour la couche réseau, la taille maximale de données pouvant être placées dans un paquet. Elle est appelée MTU *Maximum Transmission Unit* .

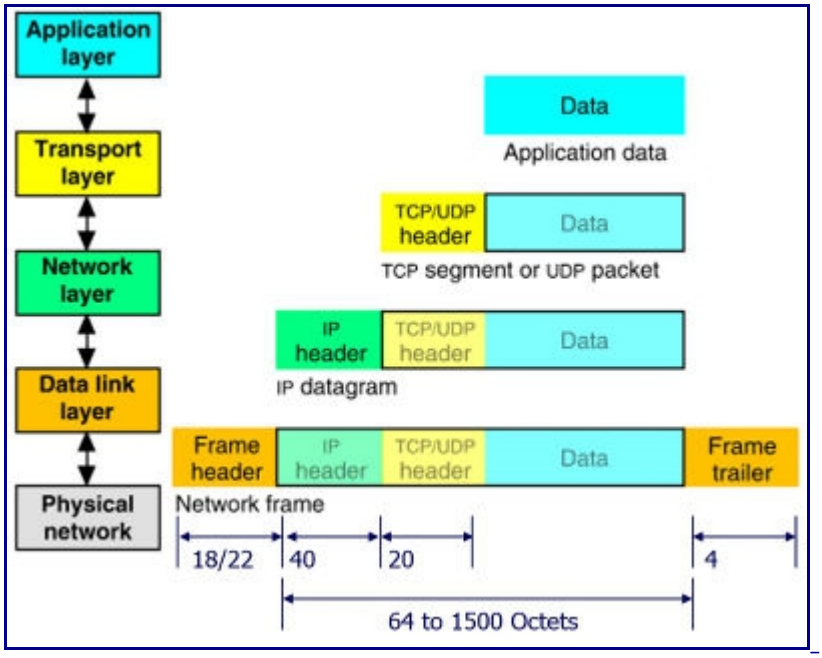

Figure 1: Encapsulation IP dans Ethernet.

Un paquet IP est cependant amené à voyager sur plusieurs supports de natures différentes,

chacun imposant une taille maximale (ou MTU) différente. Pour pouvoir parcourir son chemin jusqu'à sa destination, le paquet doit donc avoir une taille inférieure ou égale à la plus petite taille autorisée par l'ensemble des liens traversés (cf. Figure 2). Cette taille est de ce fait appelée PMTU *Path Maximum Transmission Unit* ou unité maximale de taille de transfert sur le chemin.

Pour des considérations d'efficacité, il est généralement préférable que les informations échangées entre équipements soient contenues dans des datagrammes de taille maximale. Une trop petite taille de paquet a pour effet d'augmenter la charge supplémentaire des en-têtes par rapport aux données transportées, ainsi que d'augmenter le nombre de paquets à traiter dans les routeurs. Au moment de créer des paquets, la couche réseau essaie donc de respecter au maximum la PMTU.

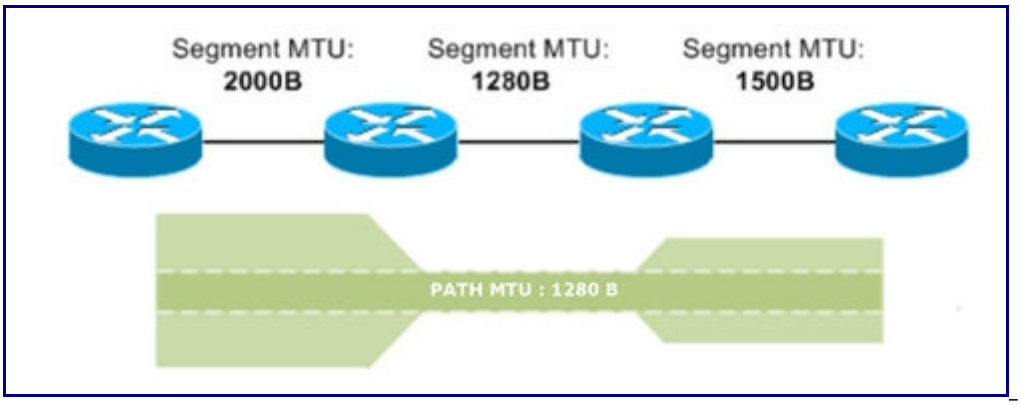

Figure 2: Path MTU.

Il est à noter qu'IPv6 impose une valeur minimale pour la MTU au niveau réseau (et donc pour la PMTU pour un chemin), valeur fixée à 1280. Cette limite a pour objectif d'éviter qu'un lien imposant une MTU très faible n'implique la transmission de petits paquets pour tous les chemins empruntant ce lien. Si un support physique impose une taille inférieure à 1280, il est nécessaire de mettre en place une couche d'adaptation pour la couche réseau. C'est le cas par exemple pour les réseaux IEEE 802.15.4, imposant une MTU de 81 octets, pour lesquels la couche d'adaptation pour IPv6 6LowPAN (RFC 4944) a dû être spécifiée. L'imposition d'une taille mimimale de MTU qui ne doit pas être fragmentée est une nouveauté en IPv6. La bonne taille du paquet IPv6 dépend du protocole de transport. Pour TCP, la valeur de 1280 octets montre de bons résultat [2] .

Découverte de la PMTU

Cependant, la valeur de la PMTU n'est pas forcément connue à l'envoi d'un premier paquet vers une destination quelconque. L'émetteur du paquet fait alors la supposition que la taille maximale vers cette destination est égale à celle du support physique sur lequel il est connecté, c'est-à-dire la MTU du réseau d'accès.

L'acheminement du paquet IPv6 s'effectue normalement jusqu'au premier routeur rencontrant une incompatibilité entre la taille du paquet à transmettre et la taille maximale autorisée sur le support physique. Le routeur est alors dans l'incapacité de traiter correctement le paquet. Le routeur supprime le paquet et utilise alors un message de signalisation (reposant sur ICMPv6,

qui sera décrit dans la séquence 3) pour informer, l'émetteur du paquet, du problème d'acheminement. Ce message indique également la taille recommandée pour que les paquets soient acheminés correctement.

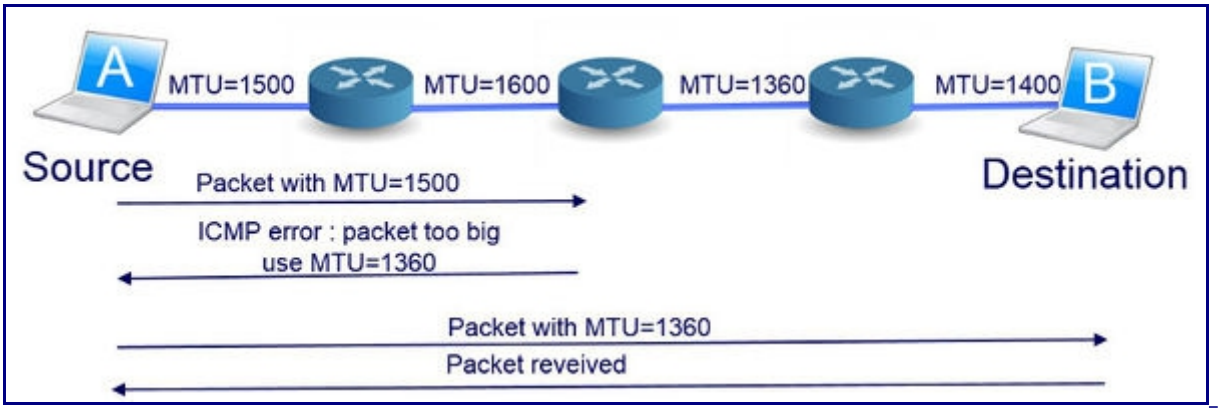

Figure 3: Négociation pour la Path MTU Discovery.

La couche réseau de l'émetteur, à la réception de ce message, enregistre la valeur recommandée de taille de paquet (cf. Figure 3). Cette valeur est la PMTU pour tous les prochains paquets vers cette même destination. La couche de transport comme TCP retransmet les données perdues ou l'application produit des nouvelles données à transmettre dans un paquet à la taille adaptée.

Ce processus peut être répété si d'autres liens imposent des tailles maximales de transmission encore inférieures. L'émetteur enregistrera les valeurs recommandées successives jusqu'à arriver à la taille maximale autorisée sur le chemin. Les paquets suivants qui respecteront cette taille seront alors acheminés sans problème. Ce mécanisme de découverte de la taille maximale de transmission sur le chemin est spécifié dans le RFC 8201 .

NOTA: Attention au filtrage d'ICMPv6 ! *Contrairement à une pratique couramment répandue en IPv4,* ***il ne faut jamais filtrer l'ensemble des messages ICMPv6 en entrée d'un réseau (en particulier le message "Paquet trop grand").*** *Le filtrage peut introduire des effets néfastes sur le fonctionnement du réseau.*

Cas où taille paquet supérieure à la PMTU: besoin de fragmentation IPv6

Il existe cependant des cas où la couche réseau ne peut pas adapter la taille des données à transmettre à la taille maximale autorisée sur le chemin. C'est le cas par exemple des messages transportés sur UDP pour le système de fichiers NFS. Ces messages peuvent avoir une taille supérieure à celle autorisée sur le support.

La couche réseau n'a alors d'autre choix que de fragmenter ces données. Le principe de la fragmentation est de séparer un paquet, devant être émis avec une taille trop importante, en plusieurs paquets respectant la taille maximale autorisée. Ces paquets (ou fragments) sont émis et acheminés vers la destination comme n'importe quel autre paquet IP (cf. Figure 4). La couche réseau du destinataire se charge alors de reconstruire le paquet IP original pour que les données puissent être traitées.

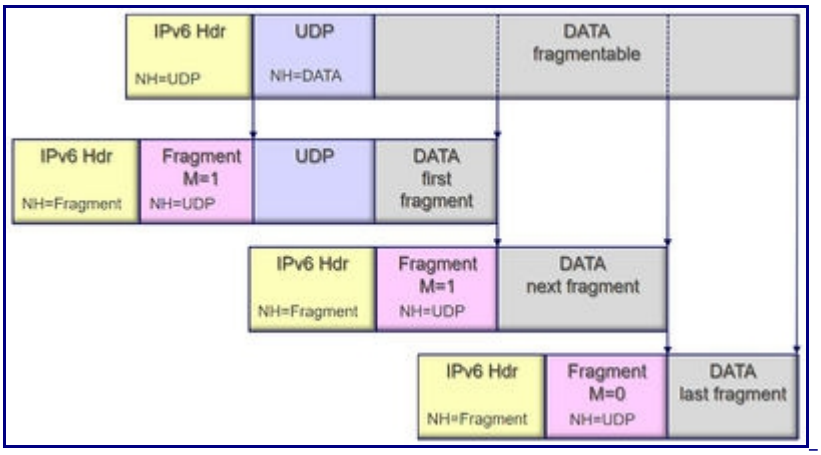

Figure 4: Fragmentation.

L'identification d'un fragment (à quel paquet appartient-il? quelle est la position relative de ce fragment?) est transmise dans une extension de fragmentation de l'en-tête IPv6. Le format de l'extension de fragmentation est donné figure 5.

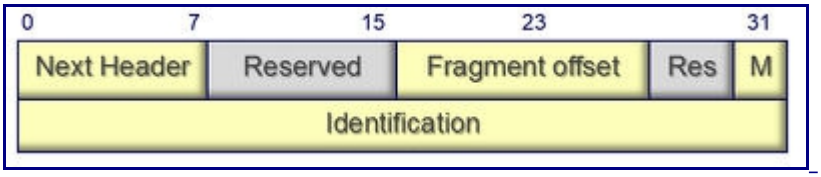

Figure 5: Format de l'extension de fragmentation.

- Le champ `Fragment offset` indique, lors du réassemblage, où les données doivent être insérées. Ceci permet de parer les problèmes dus au dé-séquencement dans les réseaux orientés datagrammes. Comme ce champ est sur 13 bits, la taille de tous les segments, sauf du dernier, doit être multiple de 8 octets.
- Le bit `M` , s'il vaut 1, indique qu'il y aura d'autres fragments émis.
- Le champ `identification` permet de repérer les fragments appartenant à un même paquet initial. Il est différent pour chaque paquet et recopié dans ses fragments.

Note: la fragmentation est permise en IPv4 au niveau des routeurs intermédiaires, leur donnant ainsi la possibilité de transmettre un paquet, même s'il est de taille supérieure à la MTU du lien suivant. Mais, dans ce cas, le mécanisme est jugé inefficace car il augmente la tâche des routeurs. En IPv6, grâce au mécanisme de découverte de la taille maximale de transmission, les routeurs intermédiaires ne fragmentent plus les paquets. Si la fragmentation est cependant nécessaire, cette tâche est déléguée aux extrémités de la communication.

Jumbogrammes

Une autre fonction optionnelle d'IPv6 est l'option "jumbogramme" dans une extension d'en-tête *Hop-by-Hop* , qui permet l'échange de paquets ayant une charge utile jusqu'à 4 Gio moins un (2^32 − 1 = 4 294 967 295 octets), en permettant l'utilisation d'un champ `longueur` de 32 bits. De tels paquets sont appelés "jumbogrammes".

Étant donné que TCP et UDP disposent de champs limités à 16 bits (`longueur` , `pointeur urgent`), le support des "jumbogrammes" IPv6 nécessite des modifications sur

l'implémentation des couches de protocoles Transport. Les "jumbogrammes" sont intéressants sur des liens qui disposent d'une MTU plus grande que 65583 octets (plus de 65535 octets de charge utile, plus 40 octets pour la taille fixe de l'en-tête, plus 8 octets pour l'en-tête d'extension *Hop-by-Hop*). Mais, à la date de rédaction de ce texte, aucun support de transmission ne permet une taille de trame aussi importante.

Conclusion

Cette activité vous a présenté les différents cas nécessitant une gestion particulière de la taille du paquet IPv6. Cette gestion est conditionnée à la valeur de l'unité maximale de transfert sur le chemin, ou PMTU. Si la taille du paquet à envoyer est inférieure à cette taille, le paquet pourra être transmis vers sa destination sans problème. Si par contre la taille du paquet est plus grande que la valeur de PMTU, soit un ajustement de la taille du segment de donnée est nécessaire au niveau de la couche transport, soit la fragmentation est obligatoire. Dans l'article en ligne [3] , l'auteur rapporte une autre idée: ne pas envoyer des paquets de plus de 1280 octets en dehors du réseau local. Ainsi en transmettant des paquets à la taille minimale de la PMTU, il n'y a plus de besoin de fragmenter et donc d'utiliser la découverte de la PMTU. C'est une solution un peu extrême pour contourner la non mise en oeuvre de la fragmentation. Nous reviendrons sur ce problème lors de la séquence suivante.

Références bibliographiques

1. ↑ Huston, G. (2016), January. Blog of AFNIC. http://blog.apnic.net/2016/01/28/evaluating-ipv4-and-ipv6-packet-frangmentation/

   Evaluating IPv4 and IPv6 packet fragmentation.
2. ↑ Huston, G. (2016), May. http://blog.apnic.net/2016/05/19/fragmenting-ipv6/ Fragmenting IPv6.
3. ↑ Bortzmeyer, S. (2010) Fragmentation IPv6: se résigner à couper à 1280 octets?

Pour aller plus loin

Vous pouvez approfondir vos connaissances en consultant les liens suivants:

- RFC 4821 : Packetization Layer Path MTU Discovery (Analyse par S.Bortzmeyer)
- RFC 4944 : Transmission of IPv6 Packets over IEEE 802.15.4 Networks
- RFC 6946 : Processing of IPv6 "atomic" fragments (Analyse par S.Bortzmeyer)
- RFC 8200 : IP version 6, Section 4.5 Fragment header (Analyse par S.Bortzmeyer)
- RFC 8201 : Path MTU Discovery for IP version 6 (Analyse par S.Bortzmeyer)