Recherche dynamique de topologie pour les réseaux IPv6

Isabelle Astic, Olivier Festor, André Schaff

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Recherche dynamique de topologie pour les réseaux IPv6

Isabelle ASTIC — Olivier FESTOR — André SCHAFF

N° 4706

Janvier 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4706--FR

Recherche dynamique de topologie pour les réseaux IPv6

Isabelle ASTIC , Olivier FESTOR , André SCHAFF

Thème 1 — Réseaux et systèmes
Projet MADYNES

Rapport de recherche n° 4706 — Janvier 2003 — 45 pages

Résumé : Être capable de fournir une image juste de la topologie du réseau est une nécessité pour toute plate-forme de gestion. Alors que la solution à ce besoin n'est pas évidente pour les réseaux IPv4, elle est encore plus difficile pour les réseaux IPv6, à cause de la taille des adresses qui interdit toute méthode itérative. Dans ce document, nous étudions ce sujet en (1) identifiant quels sont les éléments des algorithmes de recherche de topologie pour les réseaux IPv4 qui ne peuvent pas être réutilisés en IPv6, (2) énumérant tous les protocoles et tous les éléments spécifiques à IPv6 qui nous permettraient de résoudre ce problème, (3) proposant une nouvelle application basée uniquement sur des services IPv6 qui peuvent être utilisés dans toute plate-forme de supervision pour assurer une découverte de niveau 3 de la topologie d'un réseau.

Mots-clés : IPv6, recherche dynamique de topologie de niveau réseau, supervision de réseau, supervision hierarchique

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

Dynamic Topology Discovery for IPv6 Networks

Abstract: Being able to maintain an accurate image of the network topology is one of the basic requirements of any management platform. While the solution to this requirement is not obvious in IPv4 networks, it becomes even harder in IPv6 networks mainly because of the huge address space which prevents the use of any iterative method. Within this report, we address this subject by (1) identifying the parts of algorithms already used in IPv4 networks that can be reused in IPv6 networks, (2) enumerating all protocols and addressing features of an IPv6 network that make this problem challenging and, (3) proposing a novel application based solely on IPv6 services that can be used in any management platform to ensure discovery of layer 3 topology.

Key-words: IPv6, dynamic network layer topology discovery, network management, hierarchical management

Table des matières

1 Introduction **5**

2 Etude des outils existants **6**
2.1 Les outils pour les réseaux IPv6 6
2.2 Les outils pour les réseaux IPv4 6
2.2.1 Outils de niveau 3 6
2.2.2 Outils de niveau 2 11
2.3 Conclusion 11

3 Evolution des services de recherche de topologie vers les réseaux IPv6 **12**
3.1 Evolution de l'architecture d'adressage 12
3.2 Evolution des standards et protocoles ICMP et SNMP 14
3.2.1 Le protocole ICMP 14
3.2.2 Le standard SNMP 15
3.2.3 Conséquences sur la portabilité des services de niveau 2 16
3.3 Etude de la portabilité des services de niveau 3 17
3.3.1 Portage de la phase de recherche exhaustive 17
3.3.2 Portage de la phase de construction du squelette du réseau 17
3.4 Conclusions 19

4 Recherche des solutions **19**
4.1 Solutions à la phase de recherche exhaustive 19
4.1.1 Principe de base de la solution 19
4.1.2 Essais de propagation : utilisation de l'option de routage 20
4.1.3 Essais de propagation : utilisation du protocole de recherche de voisin 21
4.1.4 Conclusions 21
4.2 Solutions à la phase de construction du squelette du réseau 22
4.2.1 Complétude des informations 22
4.2.2 Trouver toutes les interfaces d'un nœud 25
4.2.3 Associer toutes les interfaces d'un nœud 25
4.3 Conclusions 25

5 Le service de recherche dynamique de topologie pour les réseaux IPv6 **26**
5.1 Architecture proposée 26
5.2 Algorithme de l'agent local 27
5.3 Algorithme de l'agent global 29

6 Implémentation du service de recherche dynamique de topologie **30**
6.1 Implémentation de l'agent global 30
6.1.1 Le fichier *agentCM.c* 31
6.1.2 Le fichier *fonctionCM.c* 32

6.2 Implémentation de l'agent local 33
6.2.1 Le fichier *agentloc.c* 34
6.2.2 Le fichier *traitreq.c* 35
6.2.3 Le fichier *traitliste.c* 36
6.3 Envoi des informations entre agent local et agent global 36
6.4 Présentation de la plate-forme 36
6.5 Choix de l'implantation des agents locaux 38
6.6 Résultats obtenus 38

7 Limites de l'outil **39**
7.1 Limites architecturelles 39
7.2 Limites fonctionnelles 41
7.3 Limites de portabilité 41

8 Conclusions **41**

1 Introduction

La gestion d'un réseau, qu'il soit IPv4 ou IPv6, est un sujet très vaste. Il comprend entre autre la gestion du trafic (statistiques, accounting), la gestion des dysfonctionnements, la gestion des configurations, la gestion de l'accès au réseau. Cependant quelque soit le domaine, il nécessite toujours une connaissance exacte de la topologie du réseau. En conséquence, toutes les plate-formes commerciales de supervision de réseaux IPv4 (telles HP OpenView, ou Tivoli) disposent d'un service de recherche dynamique de topologie. L'arrivée de réseaux IPv6 natifs amène donc naturellement le besoin d'adaptation des services de recherche dynamique de topologie pour ces réseaux. C'est pourquoi nous avons cherché à savoir ce qui existait déjà dans ce domaine, pour des réseaux physiquement connectés.
Cette étude, que nous présentons dans le chapitre suivant, nous a permis de constater que pour les réseaux IPv6, n'existait qu'un seul outil d'affichage de topologie, et aucun outil de recherche dynamique. De plus, cet outil gère uniquement les backbones disposant de routeurs de type CISCO et utilisant BGP4+ pour protocole de routage. A l'inverse, les outils de recherche dynamique de topologie de réseaux IPv4, présentés également dans le chapitre suivant, gèrent les backbones et les LANs et travaillent en environnement hétérogène. C'est pourquoi, nous avons décidé de définir un service de recherche dynamique de topologie pour les réseaux locaux IPv6, en partant des outils existant en IPv4.
Pour cela, il convenait, dans un premier temps, d'analyser ces outils. Cette étude décrite au chapitre 3, nous a permis de séparer les différentes phases des algorithmes et d'analyser l'éventualité de leur portabilité au dessus d'IPv6. Nous avons ainsi établi la liste des problèmes à résoudre afin de pouvoir utiliser ces services pour des réseaux IPv6.
Le chapitre 4 présente les solutions que nous avons apportées à ces différents problèmes. Toutes les solutions cherchées l'ont été dans le souci d'utiliser l'outil dans un environnement complètement hétérogène. C'est pourquoi, elles sont toutes issues des RFCs (Request For Comments) et des standards définissant IPv6, que nous présenterons également dans ce chapitre.
Ces solutions nous ont amené à définir une nouvelle architecture pour ce service et à définir de nouveaux algorithmes que nous présenterons dans le chapitre 5.
Afin de les valider, nous avons créé un prototype que nous avons implémenté sur notre plateforme IPv6. Cette démarche est décrite dans le chapitre 6 et présente en détail l'architecture et les algorithmes des différents éléments la constituant.
Le chapitre 7 précise les limites connues de ce prototype.
Ce document s'achève par la conclusion où nous reprenons les points forts de cette étude et où nous présentons les évolutions et améliorations possibles du service proposé.

2 Etude des outils existants

2.1 Les outils pour les réseaux IPv6

Le seul outil permettant d'afficher une topologie de réseau IPv6 est l'`ASPath-tree`[1]. Il permet de faire afficher les liens existant entre les routeurs `CISCO`[2] d'un backbone reliés par le protocole de routage BGP4+. Pour cela, il :

- se connecte à chaque routeur du réseau,
- récupère les informations concernant les voisins de ce routeur dans la table de routage BGP4+,
- recrée les liens entre les différents routeurs afin d'afficher la topologie du backbone.

Cet outil a l'avantage d'être le seul disponible actuellement sur le marché. Cependant, il a été défini pour répondre à un besoin précis, qui était de faire afficher la topologie du `6Bone`[3], équipé uniquement de routeurs CISCO. Il ne peut donc pas être utilisé dans sa configuration actuelle pour afficher la topologie d'un backbone hétérogène disposant de routeurs de marques différentes. D'autre part, comme il nécessite la connaissance des login et password d'accès sur chaque routeur géré, il ne peut être utilisé sur un réseau à large échelle et génère un trou de sécurité. Enfin, le besoin de connaissance *a priori* de ces informations n'en font certainement pas un outil de recherche dynamique.

2.2 Les outils pour les réseaux IPv4

Pour les réseaux IPv4, de nombreux outils de découverte dynamique de topologie existent. Presque toutes les plate-formes de supervision fournissent leur propres outils pour cette fonction. Quelques uns travaillent au niveau des AS (utilisant BGP par exemple), d'autres travaillent à l'intérieur des AS. Ce sont ces derniers que nous allons décrire.

A l'intérieur d'un AS, deux types de services de découverte de réseaux existent :

- ceux qui construisent les topologies de niveau 3 (appelé aussi outils de niveau 3), déterminant les nœuds et les routeurs du réseau et les connectant avec des routes issues des tables de routage,
- ceux qui construisent les topologies de niveau 2 (ou outils de niveau 2), découvrant tous les nœuds, routeurs et commutateurs du réseau, et les reliant avec des liens physiques.

2.2.1 Outils de niveau 3

J. Schönwälder et H. Langendörfer ([SL93]) d'une part, Lin *et al* ([LLC00]) d'autre part, proposent chacun un exemple d'outils de niveau 3. Le premier utilise uniquement UDP et ICMP, tandis que le second utilise SNMP (Simple Network Management Protocol) et ICMP

[1] http ://carmen/IPv6.tilab.com/IPv6/tools/ASpath-tree
[2] http ://www.cisco.com
[3] http ://www.6bone.net

(Internet Control Message Protocol). Nous allons présenter ces deux protocoles avant de présenter chacun des algorithmes.

L'architecture SNMP (RFC 1157 [CFSD90]) : Elle est la plus couramment utilisée pour la supervision des réseaux IP. Elle comprend un protocole client/serveur (ou manager/agent dans la terminologie SNMP), le protocole SNMP.
L'agent s'exécute sur l'entité à gérer. Il répond aux questions du manager concernant l'état de cette entité. Cet état est fourni par un ensemble d'éléments (ou *objets* dans la terminologie SNMP) consignés dans une MIB (Management Information Base).
Une MIB est constituée d'objets simples ou structurés (appelés également *tables*), spécifiées suivant les règles du SMIv2 (Structure of Management Information version 2 [MPS99a] et [MPS99b]). Les tables elle-mêmes sont constituées d'objets simples ou de tables. Plusieurs types de MIB existent :

- celles définies par les RFC qui permettent de gérer tout type d'équipement dialoguant via le protocole IP. La principale est la MIB II qui gère principalement les différents protocoles TCP/IP : IP ([McC96a], TCP ([McC96b], UDP ([McC96c] par exemple. D'autres modélisent le comportement de familles d'équipements (par exemple la MIB des ponts, ou Bridge MIB ([DLR+93]), etc...).
- celles définies par les constructeurs et spécifiques à leur matériel, par exemple la MIB CISCO.

Un agent peut implémenter plusieurs MIBs.
Pour plus de renseignements, se reporter aux différents RFCs cités ci-dessus et définissant le protocole SNMP et ses MIBs.

Le protocole ICMP : Il fait partie intégrante du protocole IP. Il est définit par le RFC 792 ([Pos81]), mis à jour par le RFC 950 ([MP85]). Il permet de contrôler le fonctionnement d'IP par :

- un contrôle de flux : ICMP peut demander l'arrêt temporaire d'émission de datagrame,
- un avertissement en cas d'erreur d'adressage : il retourne un message d'erreur lorsqu'une destination (port, machine ou réseau) ne peut pas être atteinte,
- la redirection d'une machine vers un autre routeur,
- le test de l'état d'un équipement (fonctions Echo Request et Echo Reply).

C'est le premier protocole de contrôle d'IP.

L'algorithme de Schönwälder et Langendörfer : Cet algorithme collecte en premier lieu toutes les informations qu'il peut obtenir du réseau et les analyse ensuite.
Ainsi, dans un premier temps, il recherche l'ensemble des adresses en service dans un espace d'adressage précis et donné au préalable. Cette recherche s'effectue grâce à un *ICMP Echo Request* exhaustif vers toutes les adresses possibles de l'espace d'adressage donné.
Ensuite, il recherche l'ensemble des routeurs permettant de relier ces différentes adresses à l'aide de la fonction traceroute.

Pour associer la topologie d'adressage et la topologie de routage trouvées, l'algorithme cherche la liste des préfixes utilisés sur le réseau en envoyant un *ICMP Mask Request* aux différentes interfaces et routeurs détectés. Comme certains systèmes peuvent posséder plusieurs interfaces, il envoie ensuite un paquet UDP vers un port supposé non utilisé pour essayer d'obtenir les adresses des autres interfaces : l'adresse source retournée dans le paquet d'erreur ICMP donne quelquefois la valeur de l'adresse de l'interface qui a reçu le paquet. Toutes ces informations sont ensuite analysées pour définir les différents sous-réseaux du réseau et pour trouver :

- les routeurs qui relient ces différents réseaux,
- les nœuds dans les sous-réseaux et leurs différentes adresses s'ils possèdent plusieurs interfaces.

L'algorithme de Lin *et al* : Si le service de recherche de topologie précédent s'attache d'abord à récolter l'ensemble des informations qu'il peut obtenir avant de construire la topologie du réseau, celui-ci construit la topologie du réseau sitôt qu'il la découvre.
Connaissant une première liste d'adresses IP, cet algorithme y cherche une interface en service, en utilisant un *ICMP Echo Request*.
Il va ensuite chercher le préfixe du sous-réseau auquel appartient cette interface à l'aide d'un *ICMP Mask Request*.
Puis, il va raccorder ce sous-réseau au nœud où se trouve le service de recherche de topologie par un traceroute ou une requête SNMP. Il obtient alors la liste des routeurs jusqu'à cette interface, et en particulier le routeur par défaut du sous-réseau auquel l'interface appartient. Ce dernier devient le premier élément de la liste des routeurs R. L'algorithme débute ensuite une phase itérative de recherche de tous les nœuds se trouvant sur le réseau :

1. prendre le premier élément de la liste,
2. faire la liste des sous-réseaux connectés à ce routeur à l'aide de requêtes SNMP. Ces informations sont obtenues par les objets *ipRouteDest*, *ipRouteMask* de la table *ipRouteTable* de la MIB II,
3. pour chacun des sous-réseaux trouvés, chercher et mémoriser toutes les interfaces qui y sont connectées (*ICMP Echo Request* vers toutes les adresses IP possibles du sous-réseau),
4. trouver tous les routeurs directement connectés au routeur courant à l'aide du protocole SNMP. Ces informations sont données par les objets *ipRouteNextHop* de la table *IpRouteTable* de la MIB II. Chaque nouveau routeur est inséré dans la liste des routeurs R,
5. revenir au premier point

Si un routeur ne dispose pas d'agent SNMP, la recherche utilise alors l'algorithme de Schönwälder et Langendörfer, pour compléter ses informations.
Prenons un exemple et appliquons l'algorithme de Lin *et al* au réseau présenté dans la figure 1.

Nous supposerons que :

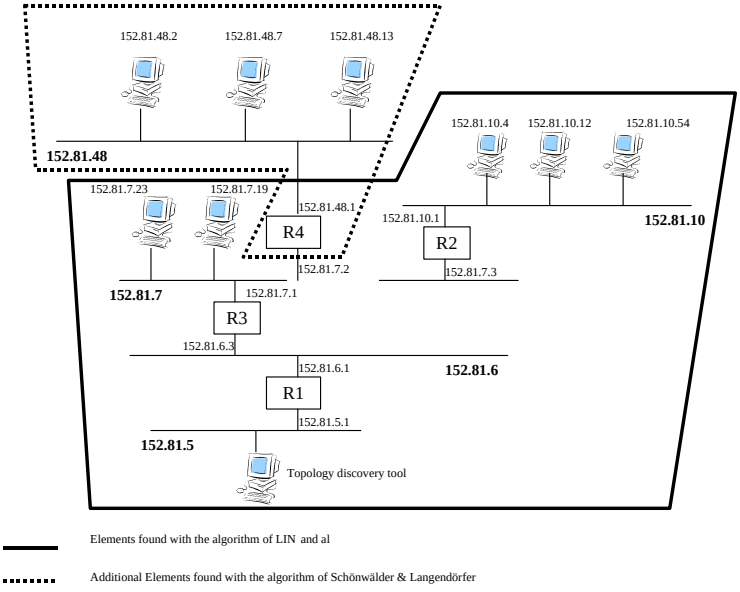

FIG. 1 – Exemple d'application de l'algorithme de Lin *et al.*

- la liste initiale contient les nœuds suivants : {152.81.7.4, 152.81.7.6, 152.81.7.23, 152.81.48.10, 152.81.48.7, 152.81.10.6, 152.81.10.96},
- tous les sous-réseaux sont des sous-réseaux de classe C,
- tous les routeurs, sauf R4, disposent d'un agent SNMP et d'une MIB II,
- l'outil de recherche de topologie est installé sur le sous-réseau 152.81.5/24,

L'algorithme de Lin *et al* effectue d'abord un *ICMP Echo Request* sur toutes les adresses contenues dans la liste, jusqu'à ce qu'une interface associée à l'une de ces adresses réponde. Dans notre exemple, ce sera l'interface d'adresse 152.81.7.23 qui répondra. Il enverra alors à cette interface un *ICMP Mask Request*, pour obtenir le masque du sous-réseau auquel elle est connectée. Il recevra pour réponse 255.255.255.0.
Une requête SNMP ou un traceroute, toujours vers cette interface, lui apprend ensuite que le routeur R3 est le routeur par défaut du sous-réseau 152.81.7/24. La liste de routeurs R n'est plus vide et contient maintenant un élément (R3).
La boucle SNMP/ICMP décrite ci-dessus commence :

- R3 devient le routeur courant (étape 1).
- L'étape 2 identifie deux sous-réseaux connectés (152.81.6/24 et 152.81.7/24).

- L'étape 3 découvre 2 nœuds connectés sur le premier des sous-réseaux (152.81.6.1 et 152.81.6.3) et 5 nœuds connectés sur le second sous-réseau (152.81.7.1, 152.81.7.2, 152.81.7.3, 152.81.7.19, 152.81.7.23).
- La liste de tous les routeurs connectés à R3 est {R1,R2,R4} (étape 4). R devient alors {R1,R2,R4}.
- L'algorithme retourne à l'étape 1 (étape 5) et choisit alors R1 pour routeur courant (étape 1).
- Il découvre un autre sous-réseau (152.81.5/24) (étape 2).
- Les systèmes connectés à ce sous-réseau sont : celui d'interface 152.81.5.1 et le nœud supportant le service de découverte de topologie (étape 3).
- L'objet *ipRouteNextHop* fourni par l'agent SNMP se trouvant sur le routeur courant, indique que le seul routeur suivant est le routeur R3, que nous connaissions déjà. Il n'est donc pas inséré dans la liste qui contient maintenant {R2,R4}.

- R2 devient le nouveau routeur courant (étape 1).
- L'algorithme nous permet de découvrir le réseau 152.81.10/24 (étape 2),
- et les nœuds 152.81.10.1, 152.81.10.4, 152.81.10.12 et 152.81.10.54 (étape 3).
- L'étape 4 fournit les routeurs R3 et R4, déjà connus. Donc seul le routeur R4 reste dans la liste R. Comme il ne dispose pas de MIB, cette partie de l'algorithme ne peut pas découvrir plus d'informations.

La figure 1 indique les éléments découverts jusqu'à présent : ils sont contenus dans la boucle en trait continu.
Pour aller au delà, il faut utiliser l'algorithme de Schönwälder et Langendörfer. Celui-ci effectue d'abord un "ping" exhaustif sur toutes les adresses possibles comprises entre la valeur 152.81.48.1 et 152.81.48.253. Il découvre ainsi les nœuds : 152.81.48.1, 152.81.48.2, 152.81.48.7 et 152.81.48.13.
Puis le traceroute permettra de découvrir tous les routeurs traversés entre le nœud supportant l'outil de découverte de topologie et un nœud quelconque du sous-réseau 152.81.48/24. Ainsi, il découvrira les routeurs R1, R3 et R4, permettant alors de rattacher ce dernier sous-réseau à ceux déjà connus (voir Figure 1).

Cet algorithme présentait l'inconvénient d'induire un trafic important et de découvrir l'ensemble de la topologie dans un délai assez long. Afin de résoudre ces problèmes, H-C Lin a proposé de l'inclure dans une architecture hierarchique et distribuée (cf [LWW01]). Des serveurs intermédiaires utilisent l'algorithme de Lin *et al*, afin de trouver la topologie d'une partie du réseau. Ces topologies sont envoyées, sur demande uniquement, à un gestionnaire d'entreprise qui les associe afin d'obtenir la topologie globale. L'implémentation qui en a été faite prouve qu'effectivement le trafic induit est bien moindre et que le temps de calcul de la topologie, puisque réparti entre chaque serveur, est beaucoup plus court.

2.2.2 Outils de niveau 2

Les outils de niveau 2, opérant au niveau de la couche liaison et données, se subdivisent en deux groupes : ceux qui utilisent SNMP et ICMP, comme l'algorithme de niveau 2 de Lin *et al* [LLC00], et ceux qui utilisent les propriétés mathématiques des graphes, comme l'algorithme de Y. Breitbart *et al* [BGM+00], ou celui de Lowekamp, O'Hallaron et Gross [LOG01].

L'algorithme de Lin *et al* permet de séparer les simples nœuds, des routeurs et des commutateurs, et de découvrir les liens physiques qui les relient. Pour cela, il va d'abord chercher tous les nœuds en service sur le réseau, en utilisant un *ICMP Echo Request* exhaustif. Il sépare ensuite les nœuds, des ponts et des commutateurs, en demandant, aux agents SNMP de chacun des nœuds trouvés, l'objet *sysServices* du groupe *system* de la MIB II. Lorsque cet objet précise que le nœud fournit un service de niveau lien, l'algorithme demande alors :

- son adresse physique (objet *dot1dBaseBridgeAddress* du groupe *dot1dBase* de la Bridge MIB),
- les ports qui sont utilisés (objets *dot1dBasePort*, *dot1dBasePortIfIndex* et *dot1dBaseNumPorts* de la table *dot1dBasePortTable* de la Bridge MIB),
- l'adresse physique des systèmes directement connectés à celui étudié (fournis par la table *ipNetToMediaTable* de la MIB II).

Les algorithmes de Breitbart *et al*, de Lowekamp *et al* résolvent le problème de la recherche des connexions physiques entre les nœuds avec des outils mathématiques.
Ils modélisent tous les deux le réseau en un graphe mathématique où les nœuds sont les systèmes du réseau et les arcs sont les liens physiques. Avec des hypothèses de départ plus ou moins fortes, et en utilisant SNMP pour obtenir les informations de Niveau 2 dont ils ont besoin, ils peuvent déterminer les arêtes possibles et celles qui ne le sont pas. Les adresses physiques des nœuds et leur type (pont, nœud ou commutateur) sont obtenus grâce à des requêtes SNMP afin d'obtenir l'objet *ipRouteNextHop* de la table *ipRouteTable*. Ces algorithmes font la distinction entre un commutateur et un routeur en :

- vérifiant l'existence ou non de la MIB des ponts (Bridge MIB),
- lisant l'objet *ipForwarding*. Si le nœud est un routeur, sa valeur est 1. Si c'est un commutateur ou un simple hôte, sa valeur est 0.

2.3 Conclusion

Les services de topologie IPv6 existant ne sont prévus pour fonctionner que dans des contextes homogènes tant du point de vue du type de nœuds (routeurs CISCO) que du type de protocole d'interconnexion entre ces nœuds (BGP 4+). En conséquence, ils ne sont définis que pour traiter des réseaux de type backbone et ne permettent pas à proprement parler d'offrir une découverte automatique de la topologie de ce type de réseau puisqu'ils nécessitent une liste pré-établie de routeurs à gérer.

Par contre, les services étudiés pour les réseaux IPv4 travaillent tous dans un environnement hétérogène et pourraient être utilisés pour des réseaux de type backbone ou réseaux locaux. Ils offrent pour la plupart une solution de recherche dynamique de la topologie des réseaux, seule la solution distribuée nécessitant l'intervention d'un opérateur pour la récupération des informations stockées dans chacun des serveurs.
C'est pourquoi, nous avons réalisé notre étude à partir des services de découverte de topologie associés aux réseaux IPv4.

3 Evolution des services de recherche de topologie vers les réseaux IPv6

Afin d'étudier la portabilité des services de recherche dynamique de topologie existant, nous devons d'abord vérifier que les standards ou protocoles sur lesquels ils sont construits, sont définis pour les réseaux IPv6, et donc, dans un premier temps, comprendre les évolutions majeures qu'IPv6 apporte par rapport à IPv4.

3.1 Evolution de l'architecture d'adressage

La principale raison de la définition d'IPv6 est le manque d'adresses disponibles pour le réseau IPv4.
Une adresse IPv4 est définie sur 32 bits et régie par des systèmes de classe. Selon la classe utilisée, la partie significative de l'adresse dévolue au réseau varie de 8 à 24 bits, et donc, par voie de conséquence, celle dévolue à l'identifiant de l'interface varie de 24 à 8 bits. Cette architecture d'adresse n'est plus ni performante, ni suffisante avec la popularité grandissante du réseau internet. La forte demande de connexion au réseau se traduit par l'utilisation toujours plus grande d'adresses. Leur nombre diminuant dangereusement, des solutions transitoires ont été définies comme l'architecture d'adressage CIDR (*Classless Inter-Domain Routing* [FLYV93]) et l'adressage privé (solution NAT, *Network Address Translation* [SE01]). Mais ces solutions ne font que repousser le problème, sans le résoudre. C'est pourquoi, la création d'un nouveau protocole Internet a été décidée et a abouti à la spécification en 1995 du nouveau protocole IPv6, RFC 1883 [Dee95], et à la spécification d'une nouvelle architecture d'adressage, RFC 1884 [Hin95]. Ces deux RFCs ont eux-mêmes été mis à jour en 1998 par, respectivement, le RFC 2460 ([DH98]) et le RFC 2373 ([HD98]).

Une adresse IPv6 est codée sur 128 bits et dispose d'une partie préfixe et d'une partie identifiant d'interface. Le préfixe est codé sur 64 bits dans le format aggrégé, seul format utilisé pour le moment au niveau global (RFC 2073 [RLH+97], mis à jour par le RFC 2374 [HOD97]) (cf figure 2).

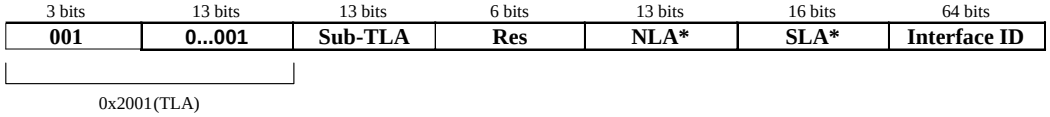

FIG. 2 – Format des adresses IPv6

où :

TLA (Top level Aggregator), définit comment sont attribués les préfixes. Le TLA à 0x0001 précise que ce rôle est dévolu aux unités régionales (RIPE-CC, APNIC, ARIN, IANA),

Sub-TLA, détermine un grand opérateur. Si la valeur du TLA vaut 0x001, ce sont les autorités régionales qui l'attribuent,

NLA* (Next Level Agregator), représente des opérateurs intermédiaires. Il est fourni par l'opérateur défini par le Sub-TLA. Il peut être découpé par ce dernier comme il le souhaite (d'où le *),

SLA* (Site Level Agregator), détermine la topologie de site. Il est donné par l'administrateur du site et peut être découpé à son gré.

D'autre part, le RFC 2374 définit des adresses de portées réduites et non plus forcément globales. Actuellement, sont définies :

- des adresses lien-local qui ne peuvent être utilisées que pour dialoguer avec des nœuds connectés sur le même lien que le nœud courant,
- des adresses de site, utilisées pour dialoguer avec tous les nœuds d'un même site (par exemple le LORIA),
- des adresses d'organisation qui sont utilisées pour dialoguer avec un système appartenant à une même organisation (par exemple l'INRIA).

Toutes ces adresses de portées réduites sont définies par un préfixe particulier. Une adresse lien-local aura un préfixe ne contenant que des 0 exceptés les deux premiers octets qui seront égaux à 0xFE80 (notation du préfixe : `FE80::/64`). Une adresse de site aura un préfixe égal à `FEC0::/48`, les 16 derniers octets servant à définir les sous-réseaux du site.

Actuellement, les notions de site et d'organisation sont très peu utilisées dans le monde IPv6 car elles posent des problèmes de différenciation d'adresses. En effet, contrairement à une adresse globale qui est définie de façon unique, une adresse de site, par exemple, peut se retrouver sur deux, voire plusieurs sites différents. Un nœud de bordure entre deux sites peut donc avoir du mal à rediriger un paquet émis vers une adresse de site, s'il possède deux routes possibles vers cette adresse, une dans chaque site. Le draft [DHJ+02] définit une solution pour tenter de donner localement un caractère unique à ces adresses de portées réduites mais elle n'est actuellement qu'à l'état d'étude. C'est pourquoi, par la suite, seules

les adresses globales et lien local seront recherchées.

Cette modification majeure de l'architecture d'adressage a des répercussions au niveau des protocoles et standards définis pour IPv4, et en particulier sur ceux mis en oeuvre dans les services de recherche dynamique de topologie.

3.2 Evolution des standards et protocoles ICMP et SNMP

Dans l'étude effectuée à la section précédente, nous avons pu noter que les services de recherche dynamique de topologie, qu'ils soient de niveau 2 ou 3, se servaient du protocole ICMP et de l'architecture SNMP (protocole SNMP, MIB II et MIB des ponts). Nous étudions dans les sous-sections suivantes leur évolution vers les réseaux IPv6.

3.2.1 Le protocole ICMP

Parce qu'un nouveau protocole IP est défini, il faut réactualiser le protocole de contrôle qui lui est associé. C'est le rôle des RFC 2463 ([CD98]), 2710 ([DFh99]) et 2461 ([NNS98]) qui redéfinissent ICMP pour le protocole IPv6.

L'expérience tirée d'ICMPv4 a permis de mieux spécifier les fonctions de contrôles. Celles-ci sont séparées en deux groupes :

- le premier contient les messages d'indication d'erreur, renvoyés lorsqu'une erreur est détectée lors de la réception d'une PDU. Ils sont définis dans le RFC 2463,
- le second contient :
  - les messages de type information : Echo Request et Echo Reply, définis également dans le RFC 2463,
  - les messages de gestion de groupe multicast, définis dans le RFC 2710.

Les messages de type Echo permettent de tester l'état des nœuds et la connectivité entre deux nœuds. Ils sont donc utilisés à des fins de diagnostic et dans la fonction "ping".

La refonte du protocole ICMP pour les réseaux de type IPv6 a permis d'y intégrer d'autres éléments, comme le protocole de recherche de voisins (RFC 2461), successeur d'ARP pour IPv4.
Ce protocole est utilisé par un nœud, sur un lien, afin de construire son adresse IP et de déterminer le routeur qui pourra router ses paquets vers leur destination. Il comprend 5 messages :

Router solicitation (Solicitation de routeur) : il est émis vers l'adresse multicast regroupant l'ensemble des routeurs d'un lien (`FF02::2`). Il permet à un équipement d'obtenir des informations de routage, dès le démarrage. L'émetteur peut préciser son adresse physique dans le champ option du message.

Router advertisement (Annonce du routeur) : il peut être une réponse à un message de solicitation mais est aussi émis périodiquement par le routeur. Il contient toujours

pour adresse source, l'adresse lien-local du routeur. Dans le cas d'une émission périodique, l'adresse destination est l'adresse multicast définissant tous les noeuds du lien (`FF02::1`). Deux champs option sont intéressants : celui contenant l'adresse physique du routeur, et celui donnant des informations sur le préfixe (valeur de celui-ci, durée de vie...).

Neighbor solicitation (Solicitation de voisin) : il est utilisé lorsqu'une station émet pour la première fois vers un voisin dont il connaît l'adresse IP et dont il ne connaît pas encore l'adresse physique. Ce message contient l'adresse IP de l'équipement recherché et en général l'adresse physique de la source.

Neighbor advertisement (Annonce du voisin) : il peut être la réponse au message précédent ou être émis régulièrement sur le réseau. Dans le cas d'une émission régulière, l'adresse destination est l'adresse multicast `FF02::1`. Un drapeau positionné dans l'entête du message permet de déterminer si l'émetteur de ce message est un routeur. Si le message n'est pas une réponse à une solicitation, un champ indique l'adresse IPv6 de l'émetteur. Une option, obligatoire dans le cas d'une réponse à une solicitation, facultative sinon, précise son adresse physique.

Redirect (Indication de redirection) : il est envoyé par un routeur et permet :
- de fournir à une station, l'adresse du routeur le plus approprié pour faire parvenir son message à destination,
- d'indiquer qu'une route directe existe entre l'émetteur et le récepteur lorsque les deux stations sont sur un même lien mais disposent de préfixes différents.

Tous les paquets émis par le protocole de recherche de voisins possèdent un TTL de 255.

3.2.2 Le standard SNMP

Nous avons vu que l'architecture SNMP repose sur deux éléments qui sont le protocole de communication SNMP et les MIBs (cf 2.2.1, p 7).

Le protocole SNMP est défini au dessus du protocole UDP, et ne contient pas d'information liées à l'adressage. Il est donc peu soumis aux problèmes liés au portage sur un réseau IPv6. Les seules modifications à faire sont celles liées à l'API des sockets qui permet de préciser si l'application utilisera SNMP au dessus d'IPv4 ou au dessus d'IPv6. D'autre part, il est toujours possible d'utiliser SNMP au dessus d'IPv4 pour lire des informations définies dans des MIBs gérant des réseaux IPv6. Le protocole lui-même ne cause donc pas de problème.

Les MIBs contiennent la modélisation des objets permettant de gérer le réseau. La modification de l'architecture d'adressage des réseaux IPv6, faisant passer la taille d'une adresse IP de 32 à 128 bits, implique une modification des conventions textuelles définissant les adresses IP. Cette évolution entraîne des modifications majeures dans certaines MIB, notamment dans la MIB II et peut-être dans la MIB des ponts (Bridge MIB). Pour le moment,

seules les implications sur la MIB II sont à l'étude. Deux évolutions ont été présentées successivement au sein de l'IETF : gérer les réseaux IPv4 et IPv6 de façon séparée ou unifiée. L'approche séparée a été définie en 1998 par les RFCs suivants :

- le RFC 2851 [DHRS00] qui présente les nouvelles conventions textuelles qui serviront à définir les objets gérant les réseaux IPv6.
- le RFC 2465 [HO98b] qui définit ces objets. Il est l'équivalent du RFC 2011 pour IPv4.
- le RFC 2466 [HO98a] qui définit les objets permettant de gérer le protocole ICMPv6
- le RFC 2452 [Dan98a] qui définit les objets permettant de gérer TCP au dessus d'IPv6. C'est l'équivalent du RFC 2012 pour IPv4.
- le RFC 2454 [Dan98b] qui définit les objets permettant de gérer UDP au dessus d'IPv6. C'est l'équivalent du RFC 2013 pour IPv4.

Mais cette première approche définissait IPv6 comme un autre protocole, différent d'IPv4, alors qu'il le remplacera. D'autre part, elle entraînait la duplication de plusieurs tableaux, notamment pour gérer TCP et UDP, alors qu'ils contenaient, fonctionnellement parlant, les mêmes objets.

C'est pour cela qu'un groupe de travail s'est créé, le groupe de travail `O&MA`[4] (Operations and Management Area), afin de modifier les MIBs existantes pour IPv4, pour qu'elles soient capables de gérer à la fois des réseaux IPv4 et IPv6. Dans ce but, il a publié en 2001 de nouveaux drafts ([FHST01b], [FHKS01], [FHK+01], [FHST01a]) qui semblent faire un consensus et qui s'appuient sur les nouvelles conventions textuelles définies dans le RFC 3291 ([DHRS02]).

Cette seconde solution semble celle qui sera adoptée. Quoiqu'il en soit, la MIB II permettant de gérer les réseaux IPv6 n'est toujours pas standardisée (et encore moins la MIB des ponts). D'autre part, cette instabilité de la MIB gérant IPv6 fait que peu de MIBs sont implémentées sur les systèmes et en particulier sur les routeurs. Ainsi, le standard SNMP est très peu disponible actuellement pour les réseaux IPv6. C'est pourquoi, nous ne l'utiliserons pas dans cette étude.

3.2.3 Conséquences sur la portabilité des services de niveau 2

Les services de recherche dynamique de topologie de niveau 2 sont basés à la fois sur des considérations mathématiques et sur le standard SNMP. Si les considérations mathématiques sont indépendantes du protocole IP, nous avons vu qu'il n'en est pas de même pour le standards SNMP. Les services de niveau 2, tels que définis actuellement, ne sont donc pas portables sur IPv6.

Par contre, les services de niveau 3 proposaient des solutions utilisant d'autres protocoles que SNMP. L'étude de leur portabilité sur les réseaux IPv6 est donc justifiée.

[4] http ://www.ops.ietf.org

3.3 Etude de la portabilité des services de niveau 3

Nous avons vu qu'ils travaillaient selon deux phases : une phase qui peut être nommée *recherche exhaustive*, qui permet de déterminer l'ensemble des nœuds du réseau géré, et une phase de *recherche de squelette du réseau*, qui permet de connaître la façon dont ces nœuds sont connectés. Nous allons donc regarder, phase par phase, si ces différentes parties des algorithmes sont utilisables telles quelles pour les réseaux IPv6.

3.3.1 Portage de la phase de recherche exhaustive

Elle est basée sur la recherche dans une plage d'adresses de toutes les adresses utilisées. Cette plage d'adresse est définie par la partie identifiant d'interface de l'adresse IPv4. Selon la classe de cette adresse, l'identifiant peut avoir une longueur variant de 8 à 24 bits. Ainsi, pour tester toutes les adresses possibles appartenant à une plage d'adresse, il conviendra de faire entre 2**8 et 2**24 pings.
Mais ce qui était vrai pour IPv4 ne l'est plus pour IPv6. En effet, dans le format d'adressage aggrégé, l'identifiant d'interface possède une longueur de 64 bits (cf 3.1, p 12). En conséquence, pour un réseau IPv6, obtenir la liste des adresses utilisées dans chaque sous-réseau revient à faire 2**64 pings, ce qui est impossible.

3.3.2 Portage de la phase de construction du squelette du réseau

L'étude des services de recherche de topologie pour les réseaux IPv4 nous a montré que cette phase pouvait s'effectuer selon deux types d'algorithmes, l'un utilisant SNMP, l'autre utilisant le traceroute.

La solution SNMP. Nous avons vu précedemment que nous ne pouvions envisager de solution reposant sur SNMP, à cause de l'indisponibilité des MIBs, et en particulier de la MIB II. Cependant, même si elle l'était, la recherche itérative de tous les routeurs du réseau, effectués en demandant l'objet *ipRouteNextHop* de la table *ipRouteTable* de la MIB II n'est plus réalisable. En effet, la plupart des protocoles de routage IPv6 existant mémorisent dans l'objet *ipRouteNextHop* une adresse lien local (cf RFC 2080 [MM97] §2.1. et RFC 2740 [CFM99] §2.5). Cette particularité est nettement visible dans la figure 3, extrait de la table de routage BGP d'un routeur CISCO IPv6. La plupart des entrées de voisins sont spécifiées à l'aide d'une adresse lien local de l'interface correspondante (cf les adresses en gras).
Autrement dit, l'objet *ipRouteNextHop* ne contiendra qu'une adresse locale et non une adresse globale qui permettrait un questionnement au delà du lien sur lequel est connecté l'agent SNMP. Donc, la connaissance de l'adresse du prochain élément ne permet plus de questionner celui-ci afin de connaître les éléments qui lui sont directement connectés et ainsi, de connaître la topologie globale du réseau. C'est pourquoi le service de recherche dynamique de topologie des réseaux IPv6 n'utilisera pas le contenu de la table de routage.

```
IPv6 Routing Table - 227 entries
Codes: C-Connected, L-Local, S-Static, R-RIP, B–BGP Timers: Uptime/Expires
B 2001:200:12A::/48 [20/5] via FE80::10:FF22:141C:22, ATM2/0.1015,
3:03:48/never
B 2001:200::/35 [20/3] via FE80::10:FF22:141C:22, ATM2/0.1015, 1w1d/never
L 2001:660:80:4005::2/128 [0/0] via ::, ATM2/0.1015, 1w1d/never
C 2001:660:80:4005::2/64 [0/0] via ::, ATM2/0.1015, 1w1d/never
B 2001:660:80:4006::/64 [20/1] via FE80::202:7EFF:FE57:1000,
FastEthernet1/0.106,1w0d/never
B 2001:660:80::/41 [20/1] via FE80::10:FF22:141C:22, ATM2/0.1015, 1w1d/never
L FE80::/64 [0/0] via ::, Null0, 1w1d/never
```

FIG. 3 – Extrait d'une table de routage BGP4+

Ainsi, même si l'architecture SNMP avait été disponible au dessus d'IPv6, seule la solution du traceroute (traceroute6 pour IPv6) était, de toutes façons, envisageable.

La solution traceroute6. Traceroute6 fonctionne comme le traceroute pour IPv4. Les modifications apportées par IPv6 n'ont donc pas d'influence majeure sur cette fonctionnalité. Cependant, l'utilisation de la fonction traceroute6 pour notre service pose plusieurs problèmes :

- d'une part, à cause de la partialité des informations obtenues. En effet, nous l'avons vu, plusieurs types d'adresses sont définies dans l'architecture d'adressage d'IPv6. Or, le traceroute6 ne nous permettra de connaître qu'un seul type d'adresse pour chaque interface : celui qui sera préféré lors de l'émission.
- d'autre part, le traceroute6 ne nous retourne qu'une seule adresse par nœud. Un nœud comportant plusieurs interfaces ne sera connu que par l'une d'entre elles (cf chemin (1) figure 4, qui ne retournera que l'interface I2). La possibilité de faire des traceroute6 dans le sens contraire pourrait permettre de découvrir toutes les interfaces d'un nœud à multiples cartes réseaux (cf chemin (2) figure 4 qui retournerait I1). Mais se pose alors un troisième problème : comment savoir que ces interfaces, obtenues par des recherches non coordonnées, appartiennent au même nœud ?

3.4 Conclusions

Le portage des services de recherche dynamique de topologie ne peut se faire actuellement que pour les services de niveau 3, le protocole SNMP nécessaire aux outils de niveau 2 n'étant pas encore disponible.
Le portage des outils de Niveau 3 est possible grâce à l'utilisation de la fonction traceroute6, ce qui implique la résolution des quatre problèmes suivants :

- Comment trouver l'ensemble des nœuds du réseau étudié ?
- Comment trouver toutes les adresses d'une interface ?

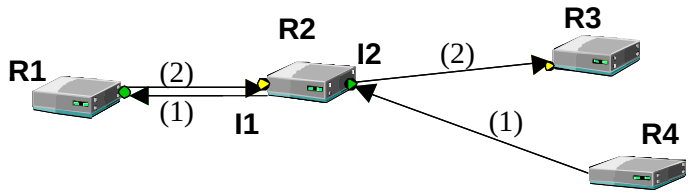

FIG. 4 – Problème de l'association des multiples interfaces d'un même nœud

- Comment trouver toutes les interfaces d'un nœud ?
- Comment associer les interfaces appartenant à un même nœud ?

4 Recherche des solutions

Les réseaux sont des environnements hétérogènes (nœuds ou routeurs, systèmes d'exploitation différents, matériel différents). Les solutions que nous devons trouver doivent donc être valides quelque soit l'environnement. C'est pour cela que nous ne chercherons les solutions que dans les RFC et les standards. Ils sont l'assurance d'une interopérabilité maximale.

4.1 Solutions à la phase de recherche exhaustive

4.1.1 Principe de base de la solution

En plus du format des adresses unicast, le RFC 2374 ([HOD97]) définit le format des adresses multicast. Tout comme les adresses unicast, elles sont formées d'une partie préfixe codée sur 2 octets et d'une partie identifiant d'interface (cf figure 5).

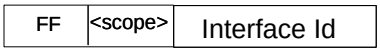

FIG. 5 – Format des adresses multicast IPv6

Le premier octet du préfixe vaut toujours FF, afin de signifier que l'adresse est de type muticast.
Le second octet (scope)est découpé en deux parties de 4 bits chacun.
Les 4 premiers bits définissent la permanence de l'adresse. S'ils valent 0, c'est que l'adresse est permanente c'est à dire, définie par une entité de type `IANA`[5] (Internet Assigned Numbers Authority), etc.... S'ils valent 1, elle est non permanente.
Les 4 bits suivants définissent la portée de l'adresse multicast. S'ils valent :

[5] http ://www.iana.org

- 2, le groupe est défini sur un lien,
- 5, il est défini sur un site,
- C, il est défini sur une organisation,
- E, le groupe est de portée mondiale.

La partie identifiant d'interface définit, quant à lui, le groupe multicast.

Le RFC 2374 définit certains de ces groupes, comme celui regroupant tous les nœuds (identifiant égal à 1), ou tous les routeurs (identifiant égal à 2). Ainsi, l'adresse multicast `FF02::1` désigne tous les nœuds d'un lien et l'adresse `FF02::2` tous les routeurs du lien, tandis que l'adresse `FF05::1` regroupe tous les nœuds d'un site, et l'adresse `FF05::2` tous les routeurs d'un site.

En utilisant la fonction ICMPv6 *Echo Request* avec pour adresse destination l'adresse multicast `FF02::1` (*tous les nœuds du lien*), il serait alors possible de découvrir l'ensemble des interfaces connectées au même lien que l'interface émettrice.

Cependant, cette solution, utilisée telle quelle, nous oblige à rester au niveau du lien puisque l'adresse multicast utilisée pour l'adresse destination ne permet de dialoguer qu'avec des nœuds du même lien que l'interface émettrice. Nous avons donc cherché à l'utiliser pour obtenir des informations au delà du lien sur laquelle l'interface émettrice est connectée, autrement dit, à propager la solution au delà du lien local.

4.1.2 Essais de propagation : utilisation de l'option de routage

Suite aux expériences issues d'IPv4, le format des entêtes IPv6 a été simplifié et de nouvelles options sont définies : ainsi, l'option de routage par la source qui permet d'imposer à un paquet une route différente de celle inscrite dans la table de routage (RFC 2460 [DH98]). En indiquant dans cette option l'adresse destination et la liste des routeurs jusqu'à la destination finale, et en adressant le paquet vers le premier routeur voulu, l'émetteur peut donc ainsi faire parcourir aux données transmises le chemin qu'il désire. Le champ TTL (Time To Live) devra être positionné en conséquence.

Maintenant, supposons que nous envoyons un tel message ayant :
- pour adresse source l'adresse du nœud supportant le service de recherche dynamique de topologie,
- pour adresse destination le routeur par défaut du lien,
- contenant en option de routage l'adresse multicast `FF02::1`.

Lorsque le routeur destinataire recevra le paquet, il devrait le propager sur les autres liens sur lequel il est connecté. L'adresse source du message étant une adresse globale, les réponses des nœuds de ces liens seraient revenues au service de découverte de topologie. En utilisant ce procédé pour tous les routeurs du lien (obtenus par un ICMP *Echo Request* vers l'adresse `FF02::2`), il est possible de trouver tous les nœuds connectés à tous les routeurs du lien sur lequel est connecté notre outil. En reproduisant cet algorithme pour les routeurs

connectés aux routeurs du 1er lien, puis aux liens suivants, etc..., il aurait été possible de trouver la topologie du réseau, en cercles concentriques autours du lien initial. Mais le RFC 2460 interdit les adresses multicast dans l'option de routage. Donc cette solution n'est pas envisageable.

4.1.3 Essais de propagation : utilisation du protocole de recherche de voisin

L'utilisation de ce protocole avait été envisagée conjointement avec l'utilisation d'une option de routage en envoyant un message de type *Neighbor solicitation* avec un TTL permettant le passage à d'autres sous-réseaux que celui auquel appartient le service de recherche de topologie, et contenant l'adresse *tous les nœuds du lien* dans l'option de routage. Mais la remarque faite précédemment sur l'utilisation de l'option de routage, plus l'obligation d'avoir un TTL de 255 pour les messages du protocole de recherche de voisins, font que cette solution n'est pas possible.

4.1.4 Conclusions

L'utilisation de la fonction ICMPv6 *Echo Request* nous conduit à effectuer cette recherche au niveau de chaque lien, sans possibilité de la propager au delà. Cet algorithme nous fournit donc une liste de liens sans nous indiquer comment ces liens sont connectés les uns aux autres. Cette méthode ne nous dispense donc pas de la phase de construction du squelette du réseau.

D'autre part, cette méthode ne nous fournit que des informations partielles sur les interfaces. En effet, l'adresse multicast `FF02::1` est une adresse de portée locale. Or, le RFC 2463 [CD98] précise bien qu' ICMPv6 doit répondre à toute demande par une requête dont l'adresse source est de même portée que l'adresse destination du paquet reçu. Autrement dit, la réponse à une requête *ICMPv6 Echo Request* vers l'adresse multicast *tous les nœuds du lien*, sera un paquet *ICMP Echo Response* dont l'adresse source sera une adresse lien local.

Afin d'obtenir un peu plus d'informations sur les interfaces découvertes, l'outil peut tout de même utiliser le protocole de découverte de voisins. En envoyant un message de type *Neighbor solicitation* vers l'adresse lien-local obtenue par la requête *ICMPv6 Echo request*, le service de découverte de topologie recevra un message de type *Neighbor advertisement* contenant l'adresse physique du voisin dans la partie option de ce message. De cette manière, le service de découverte de topologie obtient deux renseignements qui sont l'adresse lien-local des voisins et leur adresse physique.

4.2 Solutions à la phase de construction du squelette du réseau

Nous avions vu au chapitre précédent (cf 3.3.2, p 17) qu'actuellement, la seule possibilité offerte à un service de découverte dynamique de topologie pour trouver le squelette du réseau qu'il gère, est l'utilisation de la fonction traceroute6. Mais cela impliquait de résoudre les problèmes suivants :

- la complétude des informations trouvées par le traceroute6,
- trouver toutes les interfaces d'un nœud,
- associer les différentes interfaces d'un même nœud,

4.2.1 Complétude des informations

Ce problème est en fait équivalent à celui que nous venons de rencontrer pour la phase de recherche exhaustive de topologie. En effet, le traceroute6 fournit des informations que nous qualifierons de globales (le plus généralement, l'adresse globale et, avec l'aide du DNS, le nom associé à chaque adresse), et l'ICMPv6 *Echo Request* fournit des adresses de type local (adresse physique et adresse lien locale). Le problème à résoudre devient donc : comment associer ces deux couples d'information ? Comment savoir que cette adresse globale est associée à cette adresse locale ?

Une solution : *IPv6 Node Information Query*. L'extension du protocole ICMPv6 définie par Matt Crawford, et intitulée *IPv6 Information Query* ([Cra02]), peut résoudre ce problème. En effet, elle permet de remplir les fonctions d'un DNS lorsqu'un serveur de ce type n'est pas présent sur le réseau, mais elle permet également d'obtenir les adresses de domaines non globaux comme les adresses de site ou lien local rarement inscrites dans les DNS. Deux requêtes ont été rajoutées au protocole ICMPv6. Ce sont :

- le *Node Information Query*, qui permet de préciser quel type d'information est demandé (adresse, nom, par exemple),
- le *Node Information Reply*, sa réponse.

Ce protocole ne peut être utilisé qu'à destination d'adresses autres que des adresses globales, pour des raisons de sécurité.

Mais ce protocole n'est actuellement qu'à l'état de draft et souffre donc de peu d'implémentations. Pour ces raisons, nous avons décidé de ne pas l'utiliser.

Autre solution : utilisation des procédures de configuration des interfaces. Pour tenter de résoudre ce problème, il nous fallait revenir à la définition d'une adresse IPv6, et en particulier à la façon dont elle peut etre configurée.

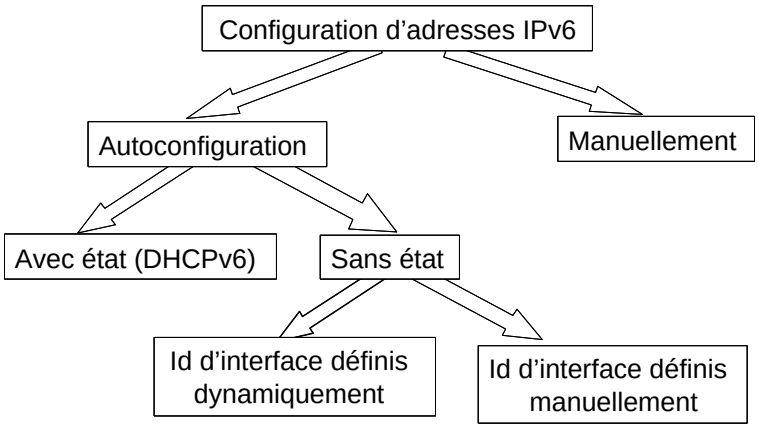

FIG. 6 – les différents types d'autoconfiguration

Une adresse IPv6 peut être configurée manuellement ou dynamiquement (cf figure 6). Dans le cas d'une configuration dynamique, celle-ci peut s'effectuer à l'aide d'un serveur de configuration externe (autoconfiguration avec état), ou de façon autonome par l'interface elle-même, c'est l'autoconfiguration sans état.

Autoconfiguration avec état. Actuellement, le seul protocole d'autoconfiguration avec état en cours de spécification est le protocole DHCPv6. Toujours à l'état de draft [DPV+02], les seules informations de configuration qui y sont définies sont les informations liées aux adresses d'interface et aux timers d'attente ou de retransmission de requêtes DHCP (option Identity association). Mais la spécification est ouverte afin que d'autres informations de configuration puissent être définies ultérieurement, comme les adresses des serveurs DNS ou des serveurs NTP.
Lorsqu'une station désire obtenir les adresses IPv6 d'une ou plusieurs de ses interfaces, elle recherche d'abord l'adresse du serveur susceptible de lui retourner les meilleures informations. Pour cela, elle envoie un *DHCP sollicit* à l'adresse multicast lien local `FF02::1:2` (tous les serveurs DHCP du lien). Si un serveur la reçoit, il répond directement. Sinon, un relais installé sur le lien du nœud la prend en compte et la transmet :

- soit à un serveur se trouvant sur le site, grâce à l'adresse multicast `FF05::1:3` (tous les serveurs DHCP du site),
- soit vers une liste d'adresses de serveurs préprogrammée.

Chaque serveur recevant cette sollicitation répond en indiquant son "empressement" à configurer la station (*DHCP advertise*). La station choisira le serveur DHCP proposant l'"empressement" le plus grand. Il lui enverra alors directement une demande de configu-

ration (*DHCP request*), indiquant pour quelles interfaces le nœud désire se configurer. Il recevra en retour une réponse (*DHCP reply*), contenant les adresses allouées par le serveur DHCP pour ces interfaces et les paramètres de configuration de ces adresses (temps de vie...). D'autres requêtes permettent la gestion des adresses, comme, par exemple, les maintenir en vie lorsque le temps de vie arrive à expiration, ou permettent des demandes d'informations de configuration autres que les information d'adressage (par exemple, les adresses des serveurs DNS).

Autoconfiguration sans état. L'autoconfiguration sans état est décrite dans le RFC 2462 ([TN98]). Dans ce cas, l'interface définit elle-même sa propre adresse. dans le cas d'un simple noeud, l'interface attend de recevoir du routeur par défaut sur le lien, le ou les préfixe(s) défini(s) sur ce lien. Dans le cas d'un routeur, le préfixe utilisé sur le lien connecté à l'interface est défini manuellement. Reste à l'interface à déterminer la partie identifiant de l'adresse. Pour cela, elle a deux possibilités :

- soit cet identifiant a été défini manuellement, dans un fichier par exemple. Dans ce cas, elle l'associe au préfixe et le tout constitue une des adresses de l'interface,
- soit cet identifiant n'est pas défini manuellement, auquel cas, elle doit utiliser un algorithme permettant de le calculer de manière unique.

Dans le cas d'une interface connectée sur un réseau Ethernet, l'identifiant peut être obtenu à partir de l'adresse MAC de la carte réseau. Comme cette adresse MAC est définie sur 48 bit, et non sur 64, elle est séparée en deux parts égales : un bit de la première partie est modifié afin de préciser que cet identifiant à une dimension globale, la seconde partie est inchangée. Entre les deux parties, deux octets sont ajoutés, de valeur FFFE, afin d'obtenir un identifiant de longueur égale à 64 bits ([CD98]).

Ainsi, dans le cas d'une configuration avec état, le service de recherche dynamique de topologie peut obtenir la configuration complète de toutes les interfaces se configurant ainsi. Il lui suffit pour cela d'écouter les paquets qui passent sur le réseau. En relevant toutes les demandes de configuration et les réponses associées, il obtient la configuration de toutes les interfaces se configurant de cette manière.
Dans le cas d'une configuration sans état, l'outil peut aussi tenter de découvrir l'adresse globale associée à l'adresse lien local qu'il connait. Le service de recherche dynamique de topologie peut reconstruire l'adresse globale possible, de la même façon que le fait l'interface, et peut la tester en envoyant vers cette adresse reconstruite un message de type ICMPv6 *Echo Request*. S'il obtient une réponse, c'est que l'adresse est utilisée.

Cependant, dans tous les autres cas, aucune solution universelle ne peut être trouvée. Nous verrons lors de la présentation de l'implémentation que nous avons faite, qu'il est toutefois possible, dans un cas très particulier de trouver cette adresse globale.

Troisième solution : utilisation conjointe du traceroute6 et du *Router solicitation* Une troisième façon d'associer deux éléments d'information est d'avoir un élément commun aux deux. Pour pouvoir associer d'une part les informations globales obtenues par le traceroute6 (adresse IPv6 globale) et d'autre part, les éléments locaux obtenus par le recherche exhaustive des interfaces d'un lien (adresse lien local et adresse physique), il faudrait pouvoir associer à l'adresse IPv6 globale soit une adresse lien local, soit une adresse physique. C'est chose possible pour les routeurs grâce au *Router solicitation*. En effet, nous avons vu (cf 3.2.1, p 14), que la réponse à un tel message était un *Router advertisement* pouvant contenir dans son champ option l'adresse physique du routeur. Ainsi, en synchronisant l'envoi d'un *Router solicitation* vers l'adresse IPv6 globale du routeur et la réception du *Router advertisement* réponse, il sera possible d'associer cette adresse IPv6 avec son adresse physique, et par là même, son adresse lien local.

4.2.2 Trouver toutes les interfaces d'un nœud

Ce problème est résolu de façon indirecte par la phase de recherche exhaustive des nœuds. En effet, nous avons vu que la seule solution pour obtenir tous les nœuds du sous-réseau était d'obtenir tous les nœuds d'un lien et d'effectuer la recherche pour tous les liens du réseau (cf 4.1, p 19). Ainsi, toutes les interfaces connectées au réseau sont trouvées et donc, en particulier, les interfaces appartenant à un même nœud.
Par contre, cela ne résoud pas le dernier problème qui est l'association des différentes interfaces d'un même nœud.

4.2.3 Associer toutes les interfaces d'un nœud

Nous avons cherché différentes solutions dans les RFCs et standards définissant IPv6, mais nous n'avons trouvé aucune solution. Tant que l'*IPv6 Node Information Query* de Matt Crawford ne sera pas un standard ou que la MIB II ne sera pas implémentée, il n'y aura pas de solution complète. La seule que nous ayons pu trouver est liée à l'implémentation que nous avons faite de nos algorithmes et à l'implantation de ceux-ci sur les différents éléments du réseau. Ces différents points sont évoqués ultérieurement (cf section 6.1.2, p 33 et section 6.5, p 38).

4.3 Conclusions

Ainsi, la découverte du squelette du réseau ne peut pas être résolue pour le moment de façon complète. Nous avons vu qu'il n'était pas toujours possible d'associer l'ensemble des adresses d'une même interface. Lorsqu'un des éléments constituant l'adresse est défini manuellement, il n'y a actuellement pas toujours de solutions. D'autre part, si nous sommes sûrs de pouvoir trouver toutes les interfaces de tous les nœuds du réseau, gràce à la nouvelle architecture que nous proposons dans le chapitre suivant, nous ne sommes pas certains de pouvoir déterminer à quels nœuds appartiennent ces interfaces.

5 Le service de recherche dynamique de topologie pour les réseaux IPv6

5.1 Architecture proposée

La phase de recherche exhaustive des nœuds du réseau nous a conduit à implanter, sur chacun des liens, un service qui permet de trouver l'ensemble des interfaces attachées à ce lien. Cependant, le service de recherche dynamique de topologie ne peut pas se limiter à ces éléments puisqu'alors, nous n'aurions qu'une vision locale du réseau, qu'une liste de lien, comme nous l'avons vu à la section 4.1, p 19. Il est donc nécessaire de disposer d'un autre outil, capable de relier tous ces liens entre eux. C'est pourquoi nous définissons une architecture hiérarchique à deux niveaux (cf figure 7).

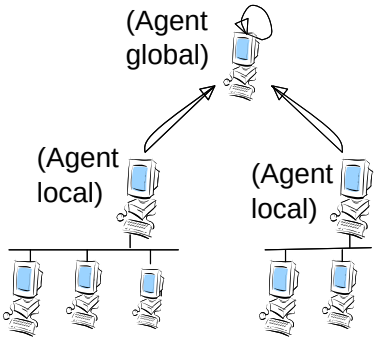

FIG. 7 – Architecture à deux niveaux

Un **agent local (AL)** installé sur un nœud ne découvre la topologie que d'un seul des liens auquel ce nœud est raccordé. Pour découvrir et gérer tous les liens raccordés à un nœud N, il faut donc implanter autant d'agents locaux que de liens. Ils peuvent être tous implantés sur ce nœud N mais peuvent être aussi implantés sur d'autres nœuds, chacun connecté à un des liens à gérer.
L'architecture comprend donc plusieurs ALs dont la mission est de trouver :

- l'ensemble des interfaces connectées au lien géré,
- pour chacune d'elle :
  - son adresse lien local,
  - son adresse physique,
  - son adresse globale. Le protocole DHCPv6 n'étant qu'à l'état de draft, l'adresse globale de l'interface ne sera cherchée que par reconstruction de l'adresse tel que nous l'avons présenté dans la chapitre 4.2.1,
  - le type du nœud associé (routeur ou hôte),

- son nom DNS,
- les préfixes utilisés sur le lien.

Cet agent local peut avoir soit un algorithme passif, c'est à dire se contenter d'écouter ce qui passe sur le réseau, soit un algorithme actif, c'est à dire envoyer des requêtes spécifiques pour obtenir les informations ciblées.
L'avantage d'un algorithme actif est que, bien qu'il augmente le risque de congestion et de collisions sur le lien, il permet d'obtenir à coup sûr les informations ciblées (l'adresse lien local des interfaces et leur adresse physique, par exemple).
A l'inverse, un algorithme passif ne génère pas de trafic supplémentaire sur le lien local. De plus, l'écoute des messages transitant sur le réseau peut permettre de recevoir des messages de type *Neighbor advertisement* non solicités dans lesquels peuvent être inclus l'adresse globale IPv6 de l'interface ayant émis ce message (cf [NNS98] § 4.4). Mais ces messages ne sont pas des messages périodiques et sont généralement envoyés lorsqu'une interface change de configuration réseau. Donc si la configuration du réseau ne change pas, l'agent local peut attendre longtemps pour obtenir les adresses globales des interfaces connectées au lien.

L' **agent global (AG)** est unique pour l'ensemble du réseau. Son rôle est de :
- recevoir toutes les informations en provenance des agents locaux,
- trouver le squelette du réseau,
- corréler ces différentes informations,
- afficher ou mettre à jour la représentation graphique du réseau.

Les agents locaux doivent connaître une adresse IPv6 de l'agent global. Pour cela, l'agent global se configure avec une adresse de type anycast. Ce type d'adresse est défini par le RFC 2374 ([HOD97]). Elle ne possède pas de format particulier, mais elle est définie comme anycast lors de la configuration de l'interface. Plusieurs interfaces peuvent être configurées avec cette adresse, mais, à la différence des adresses multicast, seule la première, au sens de la métrique du protocole de routage utilisé, recevra les paquets émis vers cette destination.

Le protocole utilisé pour le transfert des informations entre les agents locaux et l'agent global peut être un simple protocole au dessus d'UDP car :
- les informations sont transmises régulièrement entre l'agent local et l'agent global
- l'utilisation de TCP serait trop contraignante : en cas de perte de message, les paquets qui seront retransmis seront strictement identiques à ceux perdus. Or, il peut s'avérer que l'agent local dispose alors d'informations plus récentes qu'il devra attendre pour transmettre.

5.2 Algorithme de l'agent local

L'algorithme de l'agent local que nous avons défini est à la fois actif et passif. Il débute par une phase active, afin de prendre connaissance de l'ensemble des nœuds existants sur le lien et tenter de découvrir leur adresse globale. Á la fin de cette phase active, il envoie l'ensemble des informations recueillies vers l'agent global. Puis, il continue par une phase passive où il écoute le trafic généré par les protocoles ICMPv6 et de recherche de voisins. Cette phase passive lui permet :

- de prendre connaissance le plus rapidement possible d'interfaces nouvellement connectées sur le réseau ou d'interfaces n'ayant pas voulu répondre aux messages ICMPv6 mais générant du trafic. Toute nouvelle information est alors envoyée immédiatement à l'agent global afin qu'il mette à jour sa topologie et la représentation qu'il en a faite.
- de limiter le trafic en n'envoyant que les nouvelles informations reçues et non pas l'ensemble de toutes les informations connues par l'AL

Afin d'assurer la mise à jour des informations mémorisées par l'AG, et en particulier, faire disparaître des informations qui ne seraient plus utiles, l'AL recommence une phase active. C'est la réception des nouveaux vecteurs en provenance des AL qui provoquera la modification de la base de données de l'AG. L'algorithme de l'AL est donc une boucle infinie alternant phases actives et phases passives.

Lors de son initialisation, l'agent local recherche les informations concernant l'interface du noeud sur lequel il se trouve, connectée au lien à gérer. Il effectue ensuite un traceroute6 vers l'extérieur du réseau local et un second vers l'agent global. Ces deux traceroute6 permettent d'obtenir les adresses IPv6 globales des deux routeurs connectés sur le lien géré et se trouvant sur une de ces deux routes. Par l'utilisation d'un *Router advertisement*, l'agent local peut tenter alors de déterminer leur adresse physique pour pouvoir ultérieurement corréler ces informations avec les informations d'ordre local (adresse lien local et adresse physique) qu'il obtiendra par l'algorithme de recherche exhaustive (cf 4.2.1, p 24).

L'algorithme de l'agent local est donc le suivant (les étapes 2 à 8 marquent la phase active tandis que les étapes de 9 à 11 marquent la phase passive) :

1. recueillir les informations locales au nœud sur lequel l'algorithme est implanté : l'adresse lien local, l'adresse physique, l'adresse globale IPv6 et le nom de l'interface,
2. envoyer un traceroute6 vers l'extérieur du réseau local et vers l'agent global. Pour chaque routeur trouvé, envoyer un *Router solicitation* et attendre le *Router advertisement* associé,
3. envoyer un ICMPv6 *Echo Request* vers l'adresse multicast *tous les nœuds du lien*,
4. armer une temporisation,
5. attendre la fin de la temporisation ou l'arrivée d'un message,
6. si un message arrive :
   - si le message est un ICMPv6 *Echo Reply*, envoyer une *Neighbor solicitation* vers l'adresse source du message reçu,
   - si le message est un *Neighbor advertisement*, lire l'adresse IPv6 lien local, l'adresse physique contenue dans le message (s'il y en a une) et le type du nœud (hôte ou routeur). Mettre à jour toute information encore inconnue dans la base de données,
   - si le paquet est un *Router advertisement*, lire l'adresse IPv6 reçue, l'adresse physique de l'émetteur (s'il y en a une), le préfixe du sous-réseau. Mettre à jour toute information encore inconnue dans la base de données,

- si le paquet est une *Neighbor solicitation* ou une *Router solicitation*, lire l'adresse IPv6, l'adresse physique de l'émetteur (si elle est présente). Mettre à jour toute information encore inconnue dans la base de données,

7. si le temporisateur expire :
   - si le préfixe est connu, essayer de trouver les adresses globales IPv6 des nœuds dont seules les adresses lien local sont connues, en supposant que les interfaces associées sont des interfaces autoconfigurées sans état. Envoyer toutes les informations à l'AG. Armer le temporisateur. Aller en 9,
   - si le préfixe est encore inconnu, armer le temporisateur de nouveau,
8. aller en 5,
9. attendre la fin de la temporisation ou l'arrivée d'un message,
10. si un message arrive :
    - si le paquet est un *Neighbor advertisement*, lire l'adresse IPv6, l'adresse physique de l'émetteur (si elle est présente dans le paquet) et son type (routeur ou nœud). Chercher le nom de l'interface si necessaire. Envoyer toute nouvelle information à l'agent global,
    - si le paquet est un *Router advertisement*, lire l'adresse IPv6 et l'adresse physique de l'émetteur, ainsi que le préfixe du sous-réseau. Chercher le nom de l'interface trouvée. Envoyer toute nouvelle information à l'agent global,
    - si le paquet est une *Neighbor solicitation* ou une *Router solicitation*, lire l'adresse IPv6 et l'adresse physique de l'émetteur. Chercher le nom de l'interface associée si nécessaire. Envoyer toute nouvelle information à l'agent global,
11. si le temporisateur expire, aller en 4,
12. aller en 9.

5.3 Algorithme de l'agent global

L'agent global (AG) mémorise toutes les informations en provenance des différents ALs, découvre le "squelette" du réseau et corrèle l'ensemble de ces informations. Son algorithme est le suivant :

1. découvrir les informations réseaux concernant le nœud sur lequel l'AG est implanté : ses adresses IPv6 globales et lien local, son adresse physique et son nom,
2. trouver tous les routeurs connectés aux mêmes liens que lui, en envoyant un *Router Solicitation* vers l'adresse multicast `0xFF02::2`. Le *Router Solicitation* doit donc être envoyé à partir de chaque interface de l'AG,
3. attendre les messages en provenance de l'AL et des autres routeurs du lien,
4. rechercher dans les informations déjà reçues si celles que nous venons de recevoir n'ont pas déjà été enregistrées. Si ce n'est pas le cas, les sauvegarder. Il convient ici de vérifier d'une part que des informations ont déjà été reçues en provenance de cet AL, et d'autre part, qu'il n'existe pas de nouvelles données à l'intérieur de ces informations,

5. sauvegarder l'interface émettrice des informations en provenance de l'AL,
6. corréler cette information avec celles déjà mémorisées. L'enregistrer si elle n'est pas présente,
7. effectuer un traceroute6 en direction de l'AL et relier les adresses des routeurs trouvés ainsi avec les informations déjà mémorisées,
8. mettre à jour la topologie,
9. aller en 3.

L'étape 5 permet à l'AG de trouver les multiples interfaces d'un nœud car l'interface émettrice de l'AL n'est pas forcément l'interface qui est connectée au lien géré par cet AL. La corrélation faite à l'étape 6 permet à l'AG d'associer immédiatement les adresses IPv6 trouvées individuellement grâce au traceroute6 avec celles déjà mémorisées grâce aux informations en provenance des ALs. Cela simplifie énormément la construction de la topologie faite à l'étape 8, mise à jour au fur et à mesure de chaque réception d'information fournies par les ALs.

6 Implémentation du service de recherche dynamique de topologie

6.1 Implémentation de l'agent global

Nous appelerons *vecteur*, l'ensemble des informations reçues d'un AL.
Nous appellerons Base de Données des ALs (ou *BDLA*,) l'ensemble des vecteurs reçus. Elle est constituée d'enregistrements tels que décrits dans la figure 8).

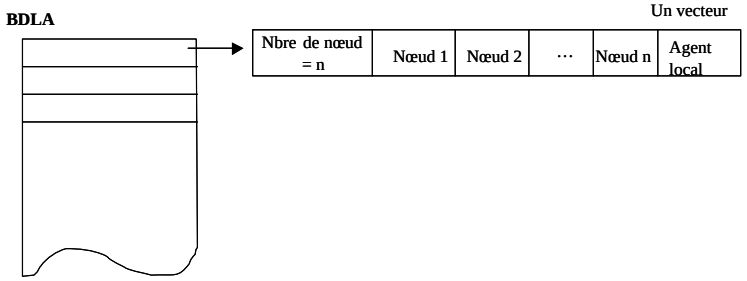

FIG. 8 – structure de la BDLA

Conjointement à cette BDLA, est créée une base de données des interfaces (ou *BI*), ensemble des interfaces mémorisées par l'AG. Cette base de données des interfaces est la pièce maîtresse de la recherche dynamique de topologie car elle permet de retrouver rapidement si une interface a déjà été reçue ou pas. Elles est constituée à partir des informations contenues

dans les vecteurs mais aussi à partir de celles reçues dans les traceroute6.
La recherche pouvant s'effectuer sur l'adresse IPv6 globale, l'adresse lien local, l'adresse physique ou le nom du nœud, cette BI est en fait composée de quatre listes. Une table de hashage est associée à chacune des listes afin de faciliter et d'accélérer la recherche des informations.
Ces deux éléments, BDLA et BI, sont nécessaires pour effectuer la corrélation de toutes les informations reçues par l'AG.

Actuellement, la construction de la topologie s'effectue uniquement lors de l'affichage. La structure du réseau local n'est pas mémorisée entièrement. Seules sont mémorisées les relations entre les interfaces d'un même lien (elles appartiennent au même vecteur) et les relations entre les routeurs détectées lors de la réalisation des traceroute6. Ce sont elles qui permettent l'affichage de la topologie complète du réseau.

L'adresse anycast définie pour adresser l'agent global est obtenue à partir du préfixe du réseau géré, fourni lors de la configuration du service de recherche de topologie. Celui-ci est défini par un préfixe particulier, englobant selon le principe de l'adressage aggrégé, l'ensemble des sous-réseaux le constituant. Ainsi, un ensemble de sous-réseaux définis par les préfixes `2001:660:301:32/64`, `2001:660:301:33/64`, `2001:660:301:34/64`,..., `2002:660:301:40/64` feront parti du réseau `2001:660:301::/58`. L'adresse anycast est obtenue en concaténant le préfixe du réseau avec l'identifiant d'interface de valeur 0x0a. Dans l'exemple ci-dessus, l'adresse anycast de l'agent global opérant sur ce réseau sera donc `2001:660:301::a`.

Les fichiers les plus importants dans l'implémentation de l'agent global sont :
- le fichier *agentCM.c*. C'est celui qui contient la fonction *main* et la fonction d'attente des informations en provenance des ALs,
- le fichier *fonctionCM.c*, qui analyse et sauvegarde les informations reçues dans le vecteur. Elle contient aussi les fonctions liées à l'affichage de la topologie,
- le fichier *BDLA.c*, qui contient toutes les fonctions nécessaires à la création, suppression, recherche et mise à jour des enregistrements dans la BDLA,
- le fichier *hash.c*, qui gère les quatre tables de hashage.

6.1.1 Le fichier *agentCM.c*

Les fonctions importantes de ce fichier sont :
- la fonction *main*, qui initialise l'AG,
- la fonction *attente_LA*, qui lit les informations en provenance des ALs,

La fonction main(). Son rôle est le suivant :
- initialiser l'AG. Cela signifie :
  - créer et configurer la socket de réception des informations en provenance des ALs et la socket servant à chercher l'ensemble des routeurs connectés au nœud courant.

Cette dernière est définie avec un timout et filtre les messages de type *Router Advertisement*,
  - récupérer les informations concernant le nœud local (adresses IP, adresse physique et nom DNS),
  - construire l'adresse anycast valable sur le domaine traité et configurer le nœud courant avec cette adresse.
- rechercher l'ensemble des routeurs connectés au nœud local. Cette recherche permet d'une part d'amorcer la liste des routeurs mais aussi d'aider à la construction de la topologie puisque les liens distants du nœud courant par un TTL de 2 seront facilement liés au nœud courant. Enfin, cela permet quelquefois de pouvoir associer une adresse IPv6 globale à certains routeurs dont seule l'adresse IPv6 lien local aurait été découverte par l'AL (cf 4.2.1, p 24).
- en boucle infinie, appeler la fonction d'attente de lecture des informations en provenance des ALs et mettre à jour l'affichage de la topologie.

La fonction attente_LA(). Cette fonction se met en attente sur la socket de réception des vecteurs. Lorsqu'un vecteur est reçu, elle appelle la fonction qui s'occupe de l'analyse du vecteur (*enregistrer_data()*, cf ci-après).

6.1.2 Le fichier *fonctionCM.c*

Le rôle de ce fichier est, entre autre, de gérer toutes les relations de l'AG avec le réseau et d'effectuer un premier traitement sur les informations reçues. A partir de ces informations, la construction de la topologie est possible. C'est pour cela que l'on y trouvera les fonctions :
- *enregistrer_data* : qui effectue l'analyse des vecteurs,
- *send_rs* : qui envoie les requêtes de type *Router solicitation* sur une interface du réseau,
- *traiter_ra* : qui récupère les informations intéressantes contenues dans le *Router advertisement* (adresse globale et adresse physique),
- *afficher_hote* : qui affiche toutes les interfaces connectées à un nœud donné, ainsi que les nœuds reliés à ce même nœud par un traceroute6,
- *correler_BDLA* : qui vérifie que les informations reçues ont bien déjà été mémorisées et associées au même AL,
- *routeurs_connectes* : qui retourne la liste des routeurs connectés à toutes les interfaces du nœud local.

Certaines de ces fonctions ne présentent pas de difficultés, c'est pourquoi nous ne les présenterons pas toutes.

La fonction enregistrer_data(). C'est cette fonction qui vérifie que le vecteur a déjà été reçu. Si ce n'est pas le cas, elle crée un nouvel enregistrement dans la BDLA. Qu'il soit nouveau ou non, l'enregistrement est mis à jour en fonctions des données reçues, s'il y a lieu. Elle mémorise également l'adresse émettrice du vecteur, si elle est différente de l'adresse de

l'interface de l'AL connecté au lien géré.
Elle effectue ensuite un traceroute6 vers l'AL émetteur du vecteur afin de connaître l'ensemble des routeurs constituant le chemin vers cet AL.

La fonction afficher_hote(). Cette fonction permet l'affichage récursif de la topologie du réseau local à partir du nœud sur lequel est installé l'AG. Á partir d'un nœud, elle affiche les informations concernant toutes les interfaces connectées à toutes les interfaces de ce nœud, puis, s'appelle récursivement, pour tous les routeurs directement connectés à ce nœud.

La fonction correler_BDLA(). L'AG reçoit des informations de différentes sources : les ALs principalement, mais aussi des traceroute6 ou des *Router solicitation*. Certaines informations sont complètes (c'est à dire, contiennent les adresses globales, locales, physiques et les noms DNS des nœuds), mais certaines sont partielles (par exemple le résultat des traceroute6). Il convient donc, à un moment donné, de relier les informations reçues par différentes sources mais concernant une même interface. C'est le but de cette fonction, qui est en fait la fonction d'appel d'un ensemble de fonctions de corrélation.
Elle recherche d'abord dans la BI s'il existe une interface mémorisée associée au moins à une des informations recueillies dans le vecteur (adresse globale, lien local, physique ou nom). Comme plusieurs enregistrements de la table de hashage peuvent répondre à ces sélections, la fonction recherche, dans les listes obtenues, celles qui appartiennent au vecteur reçu. Le résultat de cette première recherche est au maximum quatre éléments : l'un associé à l'adresse globale, le second associé à l'adresse lien local, le troisième associé à l'adresse physique et enfin le quatrième associé au nom DNS. Une seule structure sera gardée, regroupant l'ensemble des informations qui avaient été mémorisées séparémment.

6.2 Implémentation de l'agent local

Les trois fichiers principaux, nécessaires à l'implémentation de l'algorithme sont :
- le fichier *agentloc.c*. C'est celui qui contient la fonction *main*,
- le fichier *traitreq.c*, qui contient les fonctions de traitement des requêtes reçues,
- le fichier *traitlist.c*, qui contient les fonctions de recherches et de traitements dans la liste des hôtes, nœuds et routeurs mémorisés.

Nous appelons :

nœud : tout élément du réseau possédant une interface IPv6 sur le lien étudié et dont on ne sait pas encore s'il est routeur ou pas,

hôte : tout élément du réseau possédant une interface IPv6 sur le lien étudié et n'est pas un routeur,

routeur : tout élément du réseau possédant une interface IPv6 sur le lien étudié et qui est un routeur.

Tout élément connecté sur le réseau appartient à une seule de ces listes. Tant que le type de l'élément connecté n'a pas été déterminé, il appartient à la liste des nœuds. Il

passera ensuite dans l'une ou l'autre des deux autres listes, en fonctions des messages reçus le concernant (voir plus loin le fichier *traitreq.c*).

6.2.1 Le fichier *agentloc.c*

Il contient quatre fonctions importantes qui sont :
- la fonction *main* qui est la fonction d'initialisation,
- la fonction *readloop*, qui est la fonction d'attente des informations en provenance du réseau,
- la fonction *procv6*, qui traite les requêtes reçues,
- et enfin, la fonction *sig_ recept*, qui gère les interruptions du timer.

La fonction main(). Son rôle est le suivant :
- initialiser les différents paramètres et structures de données nécessaires à la mémorisation des informations qui seront recueillies,
- ouvrir les sockets d'émission et de réception,
- lire les informations d'adressage concernant les différentes interfaces du nœud sur lequel se trouve l'AL,
- envoyer le premier message d'*Echo Request* vers tous les nœuds du lien et appelle la fonction de lecture des informations en provenance du lien,
- armer également le timer qui permettra à l'agent local de passer successivement de la phase active à la phase passive et réciproquement.

La fonction readloop(). Elle attend les réponses du premier *Echo Request* effectué par la fonction main, sur l'interface émettrice. Pour chaque message reçu, s'il provient de l'interface correspondant à celle d'émission, la fonction de traitement des requêtes est appelée (cf ci-après, fonction *proc_ v6()*). Cette phase d'attente de message ne s'effectue que lorsque l'agent local est en phase active. La fin de la phase active s'arrête lorsque les trois éléments suivants sont vérifiés :
- fin de la temporisation de phase active,
- un préfixe a été reçu pour le lien étudié,
- une adresse IPv6 globale a été testée pour toutes les interfaces trouvées.

En effet, lorsque le temporisateur de fin de phase active échoue et qu'un préfixe a bien été reçu pour le lien, l'ensemble des informations reçues sont relues afin de vérifier qu'à toutes les adresses IPv6 lien local reçues, est associée une adresse globale. Si ce n'est pas le cas, le passage dans la phase passive ne s'effectuera que lorsqu'une adresse globale aura été cherchée. Pour cela, l'adresse globale possible définie à partir du préfixe recueilli sur le réseau et de l'identifiant d'interface contenu dans l'adresse lien local est testée par un *Echo Request*. Si elle répond, c'est que l'adresse est valide et elle est alors conservée avec les autres informations. Si elle ne répond pas, après un certain temps, l'agent local passe à l'enregistrement suivant. Lorsque tous les enregistrements de la BDLA ont été vérifiés, l'agent local débute la phase passive.

La fonction proc_v6(). Elle teste le type de requête reçu et appelle la fonction de traitement de requête correspondant. Ces fonctions de traitement de requêtes se trouvent dans le fichier *traitreq.c*.

La fonction sig_recept(). Elle gère les signaux de fin de timer reçus du système.
Si le préfixe en cours sur le lien n'a pas été trouvé, elle réarme purement et simplement le timer.
S'il a été reçu et que l'agent local est en train de tester les adresses globales des interfaces détectées,

- si une modification récente des informations a été enregistrée, c'est à dire si l'on a reçu une adresse globale, le timer est réarmé.
- si aucune modification a été reçue depuis la dernière chute du timer, alors il y a peu de chance pour qu'il y en ait : la phase passive est décrétée.

Si un préfixe a été reçu et que les adresses IPv6 ont déjà été vérifiées, alors l'agent local passe en phase passive.

6.2.2 Le fichier *traitreq.c*

Il contient l'ensemble des fonctions servant à émettre et recevoir des requêtes ICMPv6. Toutes les fonctions de traitement des requêtes reçues possèdent la même architecture :

- analyse de la requête pour sauvegarder les informations intéressantes qu'elle contient (adresse IPv6 source, adresse IPv6 *cible*, adresse physique, type de l'élément : routeur ou simple hôte),
- recherche dans la liste des interfaces déjà trouvées si l'une des adresses reçues ne s'y trouve pas déjà. Corrélation des informations si plusieurs interfaces mémorisées correspondent aux informations reçues, création d'un nouvel élément dans la liste des nœuds si aucune interface ne correspond,
- mise à jour de la liste des interfaces en fonction des nouvelles informations mémorisées.

Les informations recueillies changent en fonction du type de message reçu :

Réception d'un *Neighbor advertisement*. Les informations que peut fournir ce type de message sont :

- l'adresse IPv6 de l'émetteur de ce message,
- dans le champ option, s'il existe, l'adresse physique de l'émetteur de la requête,
- dans le cas d'un message de type non sollicité, l'adresse IPv6 de l'émetteur de la requête, correspondant à l'adresse physique apparaissant dans les options.

Réception d'un *Echo Response*. L'unique information recueillie dans ce cas est l'adresse de l'interface source. Un *Echo Request* peut cependant permettre de détecter une interface nouvellement arrivée ou qui n'émettrait que périodiquement.

Réception d'un *Router advertisement*. Les informations contenues dans ce type de message sont :
- l'adresse IPv6 de l'émetteur de ce message,
- dans les champs options, s'il y en a :
  - l'adresse physique de l'émetteur,
  - le ou les préfixes en cours sur le lien étudié,
- l'émetteur est un routeur

Réception d'un *Neighbor* ou *Router solicitation*. Quelque soit le type de message, l'information contenue dans le champ option, s'il existe, est l'adresse physique de l'émetteur.

6.2.3 Le fichier *traitliste.c*

Ce fichier contient un ensemble de fonctions permettant d'insérer, retirer, rechercher un ou plusieurs éléments dans les quatre listes formant la BI (cf chapitre 6.1, p 31). La fonction la plus importante est la fonction de corrélation qui permet, à partir d'un pointeur sur une interface appartenant respectivement à chacune des listes, de corréler ces informations en les regroupant dans un seul pointeur et en le mémorisant dans la liste adéquate.

6.3 Envoi des informations entre agent local et agent global

Comme il a été dit précedemment (cf section 5.2, p 28), afin de permettre la réactualisation des données fournies à l'opérateur (perte de connectivité, perte d'un nœud, etc.), les ALs doivent mettre à jour et retransmettre périodiquement leurs informations. Ainsi, le protocole utilisé pour le transfert entre l'AL et l'AG est un simple échange asynchrone de données au-dessus d'UDP.
Pour chaque nœud qu'il gère, l'AL utilise une structure (de type *nœud*) dans laquelle il mémorisera les renseignements obtenus sur ce nœud. La taille de cette structure est un multiple de 64 bits. L'AL les trie selon qu'ils sont *routeur*, *hôte* ou de type non encore déterminé (type *attendu*), afin que l'AG puisse éventuellement faire des traitements spécifiques sur chaque type de nœud. Le format des paquets émis par l'AL vers l'AG est défini dans la figure 9.

6.4 Présentation de la plate-forme

Nous avons implémenté une maquette de ce service de recherche dynamique de topologie sur la plate-forme IPv6 du LORIA (cf figure 10). A l'exception de la passerelle IPv6 du LORIA (nommé Loria-ipv6-gateway), qui est un routeur CISCO 7200, tous les autres routeurs de la plate-forme sont des PC sous système FreeBSD 4.4.
Nous y avons installé plusieurs ALs et un AG. Cette maquette nous a permis de découvrir les adresses physiques, les adresses lien local, les adresses globales IPv6 et les noms de tous les nœuds et tous les routeurs du réseau, ainsi que chaque connexion de niveau 3.

Nombre de préfixes	Nombre d' « attendus »
Nombre d' « hôtes »	Nombre de « routeurs »
Nombre total de « nœuds »	Longueur de la structure
réservé	
Structure du nœud supportant l'agent local	
Structure de préfixe 1	
…	
Structure de préfixe n	
Structure de « nœud »	
…	
Structure de « nœud »	
Structure de « nœud »	
…	
Structure de « nœud »	
Structure de « nœud »	
…	
Structure de « nœud »	

Nombre de préfixes

Nombre d' « attendus »

Nombre d' « hôtes »

Nombre de « routeurs »

FIG. 9 – Format des paquets émis par l'Agent Local à destination de l'Agent Global

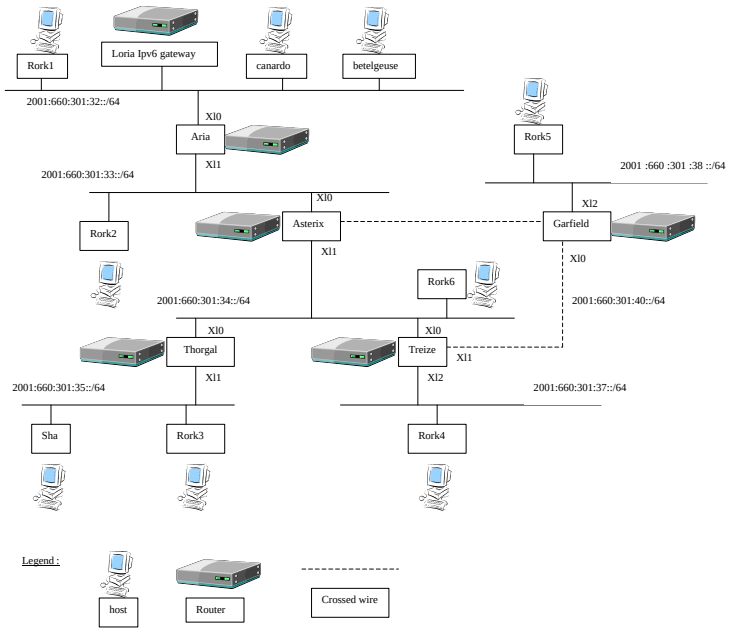

FIG. 10 – plan de la plate-forme de test

Rork est un serveur de fichier connecté à chaque sous-réseau. Pour éviter l'utilisation de multiples cartes Ethernet dans le PC, les connexions sont réalisées grâce à un VLAN. C'est pourquoi Rork apparaît sur plusieurs sous-réseaux avec la même adresse lien local.

6.5 Choix de l'implantation des agents locaux

Actuellement, chaque AL est lancé manuellement. Il peut être installé aussi bien sur un routeur que sur un simple nœud : la présence d'un AL gérant chacun des liens assure la connaissance de toutes les interfaces de tous les nœuds existant sur le réseau.
Cependant, pour pouvoir associer toutes les interfaces d'un même nœud, la solution choisie a été, d'une part de mémoriser au niveau de l'AG toutes les interface émettrices de vecteurs, d'autre part, de placer un AL par routeur pour gérer l'interface qui n'émettrait pas de vecteur.
Ainsi, prenons le schéma de la figure 10. L'AG est placé sur le routeur Aria. Un AL est placé sur Aria pour gérer le lien auquel son interface xl1 est raccordée. Cet AL découvre donc l'interface de Rork 2 et l'interface xl0 du routeur Astérix. En plaçant un AL sur Astérix pour gérer le lien raccordé à l'interface xl1 d'Astérix, nous sommes sûrs :

1. d'avoir les deux interfaces d'Astérix,
2. de recevoir les vecteurs en provenance de l'AL gérant le lien auquel Astérix xl1 est raccordé, depuis l'interface Astérix xl0,
3. et donc, de pouvoir associer Astérix xl0 et Astérix xl1 comme étant deux interfaces d'un même routeur.

Pour cette raison, sur la plate-forme, deux agents locaux ont été placés sur Aria (un pour l'interface xl0, l'autre pour l'interface xl1), et sur Treize (un pour l'interface xl1 et l'autre pour l'interface xl2). Un agent local a été installé sur Asterix, pour gérer le lien raccordé à l'interface xl1, un autre sur Thorgal (xl1) et un dernier sur Garfield (xl2).

6.6 Résultats obtenus

Toutes les interfaces de la plate-forme, à l'exception de celle appartenant à la passerelle IPv6 du LORIA s'autoconfigurent. Les ALs ont donc pu déterminer l'ensemble des adresses globales des interfaces qu'ils avaient à gérer (étape 7 de l'algorithme décrit au chapitre cf 7, p 29). L'adresse globale de la passerelle IPv6 du LORIA a pu etre trouvée grâce à l'étape 2 de l'algorithme de l'AL.

Sur la plate-forme, l'AG est implanté sur Aria. La première étape de son algorithme (cf chapitre 5.3, p 29) permet de découvrir son adresse physique, ses adresses globales et lien local, ainsi que le nom du nœud. L'étape 2 donne deux routeurs :

- l'un d'adresse lien local `fe80::230:b6ff:fe51:d41c`, dont l'adresse physique est 0-30-b6-51-d4-1c, connecté à Arial xl0,
- et le second d'adresse lien local `fe80::201:2ff:fee3:5fcc` et d'adresse physique 0-1-2-e3-5f-cc) connecté à Arial xl1.

Le vecteur reçu d'Aria xl0 permet de compléter les informations concernant le premier routeur et de savoir quels sont les autres interfaces connectées sur le meme lien qu'Arial xl0. Celui reçu d'Arial xl1 complète les données concernant le second routeur et donne les autres nœuds directement connectés à Arial xl1.
La réception du vecteur en provenance de Thorgal xl1 nous indique qu'il existe un autre lien et nous fournit l'ensemble des interfaces existant sur ce lien (étape 4).L'étape 6 révèle que Thorgal possède au moins deux interfaces. Le résultat du traceroute6 de l'étape 7 nous indique que le chemin pour aller d'Arial à Thorgal xl1 passe par Asterix xl0. Ce chemin est mémorisé. Il sera parfaitement identifié lorsque l'AG recevra le vecteur en provenance d'Asterix xl1. la seconde interface d'Asterix sera alors connue et l'AG pourra alors comprendre comment le lien sur lequel se trouve Thorgal xl1 est rattaché au reste de la topologie connue.

Un extrait des résultats obtenus par les agents locaux est fourni dans le Tableau 6.6. Il concerne les agents locaux établis sur Aria xl0, Aria xl1, Asterix xl1 et Thorgal xl1.

Sous-réseau(/64)	Nom (Type)	adresse IPv6 lien local	adresse IPv6 globale	adresse physique
2001 :660 :301 :32	Rork-1 (H)	fe80 : :201 :2ff :fee3 :6019	2001 :660 :301 :32 :201 :2ff :fee3 :6019	00 :01 :02 :e3 :60 :19
	Canardo (H)	fe80 :2b0 :d0ff :fe3f :8260	2001 :660 :301 :32 :2b0 :d0ff :fe3f :8260	00 :b0 :d0 :3f :82 :60
	Betelgeuse (H)	fe80 : :260 :8ff :fe50 :cbe9	2001 :660 :301 :32 :260 :8ff :fe50 :cbe9	00 :60 :08 :50 :cb :e9
	Ipv6-gateway (R)	fe80 : :230 :b6ff :fe51 :d41c	2001 :660 :301 :32 : :1	00 :30 :b6 :51 :d4 :1c
	Aria xl0 (R)	fe80 : :201 :2ff :fee3 :608a	2001 :660 :301 :32 :201 :2ff :fee3 :608a	00 :01 :02 :e3 :60 :8a
2001 :660 :301 :33	Rork (H)	fe80 : :201 :2ff :fee3 :6019	2001 :660 :301 :33 :201 :2ff :fee3 :6019	00 :01 :02 :e3 :60 :19
	Asterix xl0 (R)	fe80 : :201 :2ff :fee3 :5fcc	2001 :660 :301 :33 :201 :2ff :fee3 :5fcc	00 :01 :02 :e3 :5f :cc
	Aria xl1 (R)	fe80 : :201 :2ff :fee3 :605d	2001 :660 :301 :33 :201 :2ff :fee3 :605d	00 :01 :02 :e3 :60 :5d
2001 :660 :301 :34	Rork (H)	fe80 : :201 :2ff :fee3 :6019	2001 :660 :301 :34 :201 :2ff :fee3 :6019	00 :01 :02 :e3 :60 :19
	Treize xl0 (R)	fe80 : :201 :2ff :fee3 :6013	2001 :660 :301 :34 :201 :2ff :fee3 :6013	00 :01 :02 :e3 :60 :13
	Thorgal xl0 (R)	fe80 : :210 :4bff :fecd :e299	2001 :660 :301 :34 :210 :4bff :fecd :e299	00 :10 :4b :cd :e2 :99
	Asterix xl1 (R)	fe80 : :201 :2ff :fee3 :6015	2001 :660 :301 :34 :201 :2ff :fee3 :6015	00 :01 :02 :e3 :60 :15
2001 :660 :301 :35	Rork (H)	fe80 : :201 :2ff :fee3 :6019	2001 :660 :301 :35 :201 :2ff :fee3 :6019	00 :01 :02 :e3 :60 :19
	Sha (H)	fe80 : :2c0 :4fff :febb :af7a	2001 :660 :301 :35 :2c0 :4fff :febb :af7a	00 :c0 :4f :bb :af :7a
	Thorgal xl1 (R)	fe80 : :2c0 :4fff :fe67 :6b32	2001 :660 :301 :35 :2c0 :4fff :fe67 :6b32	00 :c0 :4f :67 :6b :32

7 Limites de l'outil

Cette étude et cette implémentation nous ont permis de déterminer différentes limites de cet outil. Elles peuvent être dues à l'architecture, c'est à dire inhérentes aux *a priori* que nous avons pu prendre, d'autres sont plutôt liées à l'implémentation que nous avons pu en faire. Ce sont alors des limites que nous qualifierons de fonctionnelles. Enfin, nous savons d'ores et déjà que certaines parties de notre code n'est pas portable. Ce sont des limites de portabilité.

7.1 Limites architecturelles

Le service de topologie décrit ci-dessus admet certaines limites dues à ses spécifications. En effet, il est extrèmement difficile d'effectuer la corrélation entre les différentes informations trouvées. Deux cas ne peuvent pas être résolus à coup sûrs :

- L'association de deux interfaces appartenant à un même nœud,
- La découverte des adresses globales de toutes les interfaces.

D'autre part, trois autres limites connues du service présenté ici sont liées à son architecture même :

- le réseau physique doit être capable de supporter l'accès en multicast,
- l'utilisation de ce service suppose un minimum de connaissance de la topologie réelle du réseau,
- le service tel qu'il est défini actuellement admet des limitations quant aux nombres de nœuds pouvant être gérés.

Nous présentons ces limites ci-dessous.

L'association de deux interfaces appartenant à un même nœud. Dans ce cas, lors de notre implémentation, nous avons contourné le problème en installant les ALs sur les routeurs eux-mêmes, ce qui était possible puisque nous avions à faire à des PC. Mais nous ne pouvons pas installer de code sur des routeurs de type CISCO. Nous devons donc installer les ALs sur de simples PCs. Dans ce cas, les deux interfaces des routeurs seront découvertes, chacune par un AL gérant le lien sur lequel elle est connectée, mais la seule solution pour relier ces adresses est que :

1. l'AG dispose des adresses globales des différentes interfaces du routeur,
2. toutes les interfaces soient répertoriées au niveau du DNS et sous le même nom.

La découverte des adresses globales de toutes les interfaces. Dans ce cas, sauf en cas d'autoconfiguration des interfaces, nous ne pouvons être sûrs de pouvoir trouver les adresses globales des interfaces. L'utilisation d'un traceroute6 résoud quelquefois ce problème mais il n'est pas toujours possible de corréler l'adresse globale obtenue par le traceroute6 avec l'adresse lien local obtenue dans le vecteur.

Accès en multicast. L'utilisation du protocole de *Recherche de voisin* implique l'utilisation des adresses multicast *Tous les nœuds du lien* ou *Tous les routeurs du lien.* La conséquence est que le support physique et le protocole d'accès à ce support doivent impérativement être capable de gérer ce type d'adresses.

Connaissance *a priori* de la topologie du réseau. Sa structure hiérarchique, imposée par l'utilisation du protocole de *Recherche de voisins*, oblige à installer un AL sur chaque lien à gérer. Cela suppose la connaissance *a priori* de la topologie du réseau. D'autre part, cela impose que cet outil ne soit utilisé uniquement que sur un réseau local. Son utilisation sur un réseau de type backbone serait possible mais imposerait la présence d'un PC, sur lequel on installerait un AL, sur chaque lien du backbone.

Capacité du réseau. D'autre part, un tel algorithme n'est utilisable que si le lien qu'il gère ne possède pas trop de nœuds actifs. En effet, l'utilisation des requêtes ICMPv6 *Echo request* vers une adresse multicast pourrait, sinon, provoquer de nombreuses collisions. Un rapide calcul tenant compte du débit d'un lien (100 Mbits/s), du temps de réponse d'un

ping (environ 0,2 ms), nous conduit à définir un nombre limite de 100 nœuds sur le lien.

7.2 Limites fonctionnelles

Actuellement, la notion de *multi-préfixe* sur un lien n'est pas exploitée. Tous les préfixes existant sur le lien sont mémorisés mais seul le plus grand préfixe est utilisé pour définir les possibles adresses globales associées à une adresse locale.

La notion de *lifetime* n'est pas non plus prise en compte. Nous n'avons pas implémenté, par exemple, la notion d'adresse dépréciée. Plus généralement, les adresses, qu'elles soient IP ou physiques, sont considérées comme étant éternellement valides, sauf si un message de type *Router advertisement* prévient qu'une adresse IP vient de changer d'adresse physique.

Enfin, la mise à jour de la BDLA de l'AG lors de la réception périodique des vecteurs en provenance des ALs, n'est pas implémentée.

7.3 Limites de portabilité

La recherche des informations de configuration réseaux (adressse IPv6, adresses physiques) utilisent l'API que fournit le système d'exploitation pour accéder au noyau. Ces fonctions ne sont donc pas portables sur d'autres systèmes. Cependant, c'est la seule limite de portabilité du service.

8 Conclusions

Dans ce document, nous avons montré que le service de découverte dynamique de topologie tel qu'il existait pour les réseaux IPv4 ne pouvait pas être repris tel quel pour les réseaux IPv6. Ce résultat est d'autant plus important que ce service est massivement mis en œuvre dans les plate-formes de supervision commerciales utilisées pour les réseaux IPv4. L'utilisation implicite de la longueur de l'identifiant d'interface en est une des raisons majeures, ainsi que la non disponibilité de SNMP.

Pour pouvoir disposer d'un service équivalent pour les réseaux IPv6, nous devions trouver des solutions permettant notamment de gérer un réseau hétérogène. C'est ce que nous avons fait en cherchant à spécifier un outil basé uniquement sur des standards. Nous avons proposé un nouveau service, défini sur une architecture hierarchisée mettant en jeu un agent local sur chacun des liens du réseau local à gérer, et un agent global construisant la topologie de ce réseau à partir des informations recueillies à partir des agents locaux.

L'implémentation faite sur notre plate-forme possède certes encore quelques limites. Mais une évolution de la maquette afin de gérer les multiples préfixes disponibles sur le lien ainsi que le temps de vie des adresses, permettrait de supprimer les limites fonctionelles. La limite architecturelle liée à la connaissance *a priori* de la topologie du réseau pourrait être levée par

l'utilisation de protocoles actifs permettant de télécharger et d'installer automatiquement les agents locaux sur les liens. De même, les limites liées à la complétude des informations et à l'association des multiples interfaces d'un même nœud disparaîtront lorsque les différents éléments nécessaires seront standardisés. L'utilisation du protocole défini par Matt Crawford ([Cra02]) permettra d'obtenir, pour toute interface, l'ensemble de ses adresses. La standardisation de la MIB II permettra d'utiliser SNMP et donc d'obtenir les adresses IPv6 globales et lien local de toutes les interfaces d'un nœud.

Cependant, si ces deux protocoles résoudront les derniers problèmes de complétude rencontrés par le service de recherche dynamique de topologie que nous proposons, ils ne le remettraient pas en question. En effet, pour des raisons de sécurité, les requêtes du protocole *IPv6 Node Information Query* ne peuvent être émises que vers une adresse lien local, interdisant d'effectuer une recherche au delà du lien. D'autre part, comme nous l'avons vu dans la section 3.3.2, p 17, SNMP ne permettra pas non plus d'accéder au delà du lien sur lequel l'émetteur de la requête SNMP est connecté, car les tables de routage contiennent essentiellement des adresses IPv6 lien local. Donc, quelque soit le protocole utilisé, l'architecture du service de recherche dynamique de topologie devra être hierarchisée, avec un agent travaillant au niveau du lien pour récolter toutes les informations concernant ce lien, et un autre agent travaillant à un niveau global pour synthétiser toutes les informations recueillies par les agents locaux.
Notre service de découverte dynamique de topologie, par l'utilisation du protocole ICMPv6, permet la découverte la plus complète à ce jour de la topologie du lien. L'utilisation de SNMP et de l'algorithme de Crawford lui permettrait simplement d'être plus complet.

Références

[BGM+00] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschartz. Topology discovery in heterogeneous IP networks. In *IEEEE/INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume xxvi+1826, 2000.

[CD98] A. Conta and S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. Technical report, IETF, December 1998. RFC 2463.

[CFM99] R. Coltun, D. Ferguson, and J. Moy. OSPF for IPv6. Technical report, IETF, December 1999. RFC 2740.

[CFSD90] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin. Simple network management protocol (snmp). Technical report, IETF, May 1990. RFC 1157.

[Cra02] M. Crawford. "ipv6 node information queries, draft-ietf-ipngwg-name-lookups-09.txt". Technical report, IETF, 2002.

[Dan98a] M. Daniele. IP Version 6 Management Information Base for the Transmission Control Protocol. Technical report, IETF, December 1998. RFC 2452.

[Dan98b] M. Daniele. IP version 6 Management Information Base for the user Datagram Protocol. Technical report, IETF, December 1998. RFC 2454.

[Dee95] R. Deering, S.and Hinden. Internet Protocol Version 6 (IPv6) Specification. Technical report, IETF, December 1995. RFC 1883.

[DFh99] S. Deering, W. Fenner, and B. haberman. Multicast Listener Discovery (MLD) for IPv6. Technical report, IETF, October 1999. RFC 2710.

[DH98] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specifications. Technical report, IETF, December 1998. RFC 2460.

[DHJ+02] S. Deering, B. Haberman, T. Jinmei, E. Nordmark, A. Onoe, and B. Zill. IPv6 Scoped Address Architecture, draft-ietf-ipngwg-scoping-arch-04.txt. Technical report, IETF, June 2002.

[DHRS00] M. Daniele, B. Haberman, S. Routhier, and J Schoenwaelder. Textual conventions for internet network addresses. Technical report, IETF, June 2000. RFC 2851.

[DHRS02] M. Daniele, B. Haberman, S. Routhier, and J Schoenwaelder. Textual Conventions for Internet Network Addresses. Technical report, IETF, 2002. RFC 3291.

[DLR+93] E. Decker, P. Langille, A. Rijsinghami, , and K. McCloghrie. Definition of Managed Objects for Bridges. Technical report, IETF, July 1993. RFC 1493.

[DPV+02] R. Droms, C. Perkins, B. Volz, M. Carney, and T. Lemon. Dynamic Host Configuration Protocol for IPv6 (DHCPv6), draft-ietf-dhc-dhcpv6-28.txt. Technical report, IETF, November 2002.

[FHK+01] B. Fenner, B. Haberman, McCloghrie K., J. Schoenwaelder, and D. Thaler. "management information base for the user datagram protocol (udp), draft-ietf-ipngwg-rfc2013-update-01.txt". Technical report, IETF, 2001.

[FHKS01] B. Fenner, B. Haberman, McCloghrie K., and J. Schoenwaelder. "management information base for the transmission control protocol (tcp), draft-ietf-ipngwg-rfc2012-update-01.txt". Technical report, IETF, 2001.

[FHST01a] B. Fenner, B. Haberman, J. Schoenwaelder, and D. Thaler. "ip forwarding table mib, draft-ietf-ipngwg-rfc2096-update-00.txt". Technical report, IETF, 2001.

[FHST01b] B. Fenner, B. Haberman, J. Schoenwaelder, and D. Thaler. "management information base for the internet protocol (ip), draft-ietf-ipngwg-rfc2011-update-00.txt". Technical report, IETF, 2001.

[FLYV93] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR) : an Address Assignment and Aggregation Strategy. Technical report, IETF, September 1993. RFC 1519.

[HD98] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. Technical report, IETF, July 1998. RFC 2373.

[Hin95] S. Hinden, R.and Deering. Internet Protocol Version 6 (IPv6) Addressing Architecture. Technical report, IETF, December 1995. RFC 1884.

[HO98a] D. Haskin and O. Onishi. Management Information Base for IP Version 6, ICMP Group. Technical report, IETF, December 1998. RFC 2466.

[HO98b] D. Haskin and S. Onishi. Management Information Base for IP Version 6 : Textual Conventions and General Group. Technical report, IETF, December 1998. RFC 2465.

[HOD97] R. Hinden, M. O'Dell, and S. Deering. An IPv6 Aggregatable Global Unicast address Format. Technical report, IETF, January 1997. RFC 2374.

[LLC00] H-C Lin, S-C Lai, and P-W Chen. Automatic topology discovery of IP networks. In *IEICE Trans. Inf. & Syst.*, volume E83-D, January 2000.

[LOG01] B. Lowekamp, D.R. O'Hallaron, and T.R. Gross. Topology Discovery for large ethernet networks. In *ACM/SIGCOMM*, 2001.

[LWW01] Hwa-Chun Lin, Yi-Fan Wang, and Chien-Hsing Wang. Web-based Distributed Topology Discovery of Ip Networks. In *Proceedings of the 15th International Conference on Information Networking*, January 2001.

[McC96a] K. McCloghrie. Management Information Base for the Internet Protocol (IP). Technical report, IETF, November 1996. RFC 2011.

[McC96b] K. McCloghrie. Management Information Base for the Transmission Control Protocol (TCP). Technical report, IETF, November 1996. RFC 2012.

[McC96c] K. McCloghrie. Management Information Base for the User Datagram Protocol (UDP). Technical report, IETF, November 1996. RFC 2013.

[MM97] G. Malkin and R. Minnear. RIPng for IPv6. Technical report, IETF, January 1997. RFC 2080.

[MP85] J.C. Mogul and J. Postel. *Internet Standard Subnetting Procedure, RFC 950, STD 5*, August 1985.

[MPS99a] K. McCloghrie, D. Perkins, and J. Schoenwaelder. Structure of Management Information v2 (SMIv2). Technical report, IETF, April 1999. RFC 2578, STD 58.

[MPS99b] K. McCloghrie, D. Perkins, and J. Schoenwaelder. Textual conventions for SMIv2. Technical report, IETF, April 1999. RFC 2579, STD 58.

[NNS98] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). Technical report, IETF, December 1998. RFC 2461.

[Pos81] J. Postel. Internet Control Message Protocol, RFC 792, STD 5, September 1981.

[RLH+97] Y. Rekhter, P. Lothberg, R. Hinden, S. Deering, and J. Postel. An IPv6 Provider-Based Unicast Address Format. Technical report, IETF, November 1997. RFC 2073.

[SE01] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Tr. Technical report, IETF, January 2001. RFC 1631.

[SL93] J. Schönwälder and H. Langendörfer. How to keep track of your network configuration. In *LISA*, November 1993.

[TN98] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. Technical report, IETF, December 1998. RFC 2462.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399