Université de Versailles Saint-Quentin-en-Yvelines
Laboratoire PRiSM

Rapport de Stage de DEA

Simulation des Instabilités de BGP

Houssame YAHIAOUI

Encadrement : Jean-Michel FOURNEAU

Sommaire

Table des figures

Chapitre 1

Introduction

Internet a énormement changé depuis son avènement durant les années 1960s. Évoluant d'un simple réseau de 4 nœuds connectés par des circuits à 56Kbits/s, en une architecture centrée autour du *backbone* ARPANET, puis autour du *backbone* NSFNET, pour enfin se développer durant les années 1990s en en un large réseau ou les ISPs (*Internet Service Provider*) sont en compétition commerciale, chacun disposant de son propre *backbone* qu'il utilise pour offrir des services de transit de données aux réseaux qui souhaitent se connecter à Internet.

An final, l'architecture d'Internet à progressé jusqu'à se transformer en un réseau distribué et possédant une forme de hierarchie d'ISPs. En effet, les ISPs se partagent en trois catégories selon la taille et la connectivité du réseau qu'ils possèdent : les ISPs *Tier I* sont de très grand réseaux (de niveau international) presque entièrement interconnectés entre eux. Ces ISPs fournissent un service de transit aux ISPs *Tier II* (de niveau national) qui garantissent eux même un service de connectivité aux ISP *Tier III*. Ces derniers possèdent des réseaux de petites tailles (de niveau régional), comparées aux deux autres catégories d'ISPs, et offre la connectivité à la majorité des ordinateurs générant le véritable traffic de l'Internet.

Chacun de ces réseaux (*backbone* d'ISP ou réseau hôtes) utilisant tous la pile protocolaire TCP/IP pour communiquer, il a fallut trouver un moyen de garantir le routage des données de n'importe quel point vers n'importe quel autre point de l'Internet. C'est le rôle des protocoles de routage, qui du fait de la structure de l'Internet, sont séparés en deux catégories : les protocole de routage intra-domaine IGP (*Internal Gateway Protocol*) garantissent la consistence du routage à l'intérieur des réseaux des ISPs ; et les protocoles de routage inter-domaines EGP (*External Gateway Protocol*) qui se chargent de faire communiquer les différents *backbones* entre eux. Parmi les protocoles de cette dernière catégorie, le plus largement utilisé aujourd'hui est le protocole BGP (*Border Gateway Protocol*).

De fait, depuis son apparition, BGP est devenu le protocole de routage inter-domaine "officiel" de l'Internet. Selon la terminologie BGP, les *backbones* des ISPs, mais aussi les réseaux de leurs clients, constituent des "systèmes autonomes" (AS : *Autonomous System*), interconnectés par des liens inter-domaine. Grâce à BGP, ces ASs peuvent se communiquer des informations d'accessibilité à propos de leurs réseaux respectifs, ainsi que des informations d'accessibilité aux réseaux que chacun peut atteindre. Cette communication se fait par l'échange de messages. Ainsi, tout routeur BGP qui reçoit une nouvelle information de routage se doit, si cette information altère sa propre table de routage, de la communiquer aux autres routeur. Par ce moyen, l'information à propos de l'apparition ou la disparition d'un réseau d'un AS est transmise successivement des routeurs de cet AS à leurs voisins, puis aux voisins de ceux-ci, et ainsi de suite jusqu'à être diffusée à tous les routeurs concernés.

De plus, BGP permet, pour chaque AS, de choisir les routes vers les destination par un mécanisme différent du traditionnel "plus court chemin" des autres protocoles de routage. Au lieu de cela, BGP permet un choix par *politiques de routage* implémentant les accords commerciaux de connectivité entre ISPs. Cette stratégie particulière, en plus de la structure fiable, extensible et réactive du protocole, ont poussé la majorité des ISPs à l'adopter comme protocole de routage inter-domaine.

Mais l'explosion de croissance de l'Internet alliée à l'amenuisement de la structure hierarchique de l'Internet, ont généré de nombreux problèmes au niveau du routage inter-domaine. Certains de ces problèmes sont dus à l'augmentation du volume du traffic, mais d'autres problèmes, et cela n'a été découvert que récement, sont dus à la conception même, ou aux implémentations du protocole BGP. Variation fréquentes des tables de routage (Instabilité), lenteur de la convergence de l'algorithme de routage de BGP, augmentation du volume des échanges BGP, sont autant de problèmes qui heurtent le routage inter-domaine et qui induisent des désordres sur le traffic de l'Internet.

La résolution de ces problèmes, ou du moins l'atténuation de leurs conséquences, passe obligatoirement par l'étude extensive et détaillée du protocole BGP, et ce par tous les moyens disponibles : études théoriques, études empiriques (analyse du comportement de BGP par la visualisation des données qu'il génère) et experimentales, et simulations.
- Dans ce travail, l'objectif était de concevoir un simulateur des instabilités

de BGP. Ce document constitue une présentation du travail réalisé.

Le chapitre 2 fournit une introduction au protocole BGP. Il pose les bases du protocole tels qu'ils sont exposé dans spécification, mais présente aussi quelques détails d'implémentation particuliers, et deux des extensions du protocole BGP.

Le chapitre 3 expose quelques uns des problèmes constatés sur Internet et relatif à BGP. Cela comprend autant les difficultés que rencontre BGP (instabilité du routage) que les problèmes inhérents à la conception même du protocole (lenteur de la convergence et oscillations persistantes de routes). De plus, il expose un type de problème, récement découvert, qui émane d'une méthode de diminution de l'instabilité de BGP.

Le chapitre 4 présente quelques uns des simulateurs existants relatifs au protocole BGP. Certains ont été spécifiquement conçus pour l'étude du protocole ou de son comportement ; d'autres ont été réalisé, notamment pour tester la valeur d'améliorations proposées pour BGP.

Le chapitre 5 expose une proposition d'architecture pour un simulateur des instabilités de BGP. Il y est présenté un modèle de routeur BGP, mais aussi une proposition de nouvelle topologie réaliste pour effectuer les simulations sur BGP, ainsi que l'algorithme pour les générer.

Enfin, le chapitre 6 clora la présentation par une conclusion du travail réalisé et quelques idées quand à l'implémentation du simulateur ou à son extension.

Chapitre 2

Présentation du protocole BGP

2.1 Qu'est ce que BGP ?

2.1.1 Introduction

Durant les dernières années, Internet s'est étendu au point de regrouper plusieurs milliers de domaines, interconnectés entre eux. Chaque domaine est constitué d'un nombre potentiellment très grand de réseaux et de routeurs qui les relient. Certains domaines constituent de véritables épines dorsales (*backbone*) sur lesquelles transitent les informatons échangées entre les autres domaines. D'autres systèmes ne sont que les nœuds terminaux de la hiérarchie d'Internet, tel que les providers d'accès par réseau commuté. Tous ces domaines, quelque soit leur étendues, leurs tailles, ou le nombre de leurs connexions, possèdent deux point communs : ils utilisent tous un (ou plusieurs) protocole de routage **intra-domaine** (**IGP** : *Interior Gateway Protocol*) pour assurer le convoyage des données entre les hôtes au sein du domaine ; et ils utilisent un protocole de routage **inter-domaine** (**EGP** : *Exterior Gateway Protocol*) pour assurer la connectivité entre les domaines, c'est à dire pour fournir un moyen d'échange de données entre différents domaines.
Le protocole BGP (*Border Gateway Protocol*) fait partie de la seconde catégorie de protocole de routage. Il a succédé au protocol EGP (*External Gateway Protocol*) [Mil84] qui ne pouvait être utilisé que sur des topologies de domaines strictement hiérarchiques (donc, sans circuit dans le graphe des domaines). Le protocole BGP a passé par plusieurs versions, chacune ajoutant ou retirant des spécificités au protocole (tel que certains timers, l'identificateur BGP,...). La version actuellement déployée est la version 4 ; elle a été officielement publiée dans le RFC 1771 [RL95]. Le protocole BGP4 fonctionne sur des topologies de domaines quelconques, supporte le mode d'adressage

CIDR (*Classless inter-domaine Routing*) [FLYV93], l'aggregation de routes, et permet le routage par politiques indépendantes [RG95].
Dans ce qui suit nous allons d'abord définir quelques termes et concepts essentiels à la présentation de BGP. Nous pourrons, alors, expliquer plus précisement ce qu'est BGP, les possibilités de ce protocole, son fonctionnement, ainsi que les méthodes qu'il emploit.

2.1.2 Lexique de termes

Système Autonome (**AS** : Autonomous System) La définition classique d'un système autonome est l'**union d'un ensemble de réseaux et de routeurs administrés par une unique entité** [Bei02]. Le plus souvent, un AS désigne un ensemble de routeurs (reliant des réseaux appartenant à l'AS) régis par la même administration technique, en utilisant un IGP et des métriques communes pour router les paquets au sein de l'AS, et en utilisant un EGP pour le routage inter-AS [RL95]. On utilisera aussi le terme de "domaine" pour désigner de telles rassemblements.

***Speaker* BGP** BGP utilise le concept de *speaker* pour publier les informations de routage. Un *speaker* réside dans une machine appartenant à l'AS (le plus souvent un routeur). Ce speaker convient avec les speakers des ASs voisins des routeurs de bordure à utiliser comme point d'entrée/sortie pour des destinations exterieurs à l'AS. Les décisions des speakers sont dictées par des **politiques**, spécifiques à chaque AS.

Politique de routage Une politique de routage est un ensemble de règles qui dictent l'attitude de l'AS envers les informations de routage qu'il reçoit. Ainsi, ces règles sont utilisées par les speakers BGP de l'AS pour déterminer les informations de routage à accepter, pour influencer le processus de choix du chemin vers une destination et pour décider des informations de routage à envoyer vers les autres speakers.

***Peer* BGP** Le terme *peer* peut être traduit littéralement par "correspondant". En effet, le *peer* d'un speaker BGP est un autre speaker BGP auquel il communique et duquel il reçoit des informations de routage. Il doit, pour cela, exister un **session** établie entre ces deux speakers.
Les *peers* d'un speaker BGP peuvent être classés selon leur appartenance à l'AS du speaker : un peer **interne** réside dans le même AS que le speaker ; un peer **externe** réside dans un AS voisin de l'AS du speaker. Une des différences importantes entre ces deux types provient de la connectivité exigée entre peers. Ainsi, deux peers externes (résidant dans deux ASs différents) doivent se trouver sur le même sous-réseau

physique ; alors que deux peers internes (résidant dans le même AS) peuvent se trouver n'importe ou à l'interieur de l'AS, du moment qu'il peut exister un canal de communication entre eux.

Session BGP Une session BGP est une relation entre deux speakers BGP qui désirent s'échanger des informations de routage. Elle débute par l'établissement d'une connexion au niveau du protocole de transport (TCP), puis par l'établissement de la session même. Cela permet de positionner l'échange dans un contexte précis, et de réguler les transfers.

2.1.3 Définition de BGP

BGP est un protocole de routage inter-domain (inter-AS) à vecteur de chemin (*path vector protocol*) [Ste99]. Le terme inter-AS exprime le fait que BGP est utilisé pour communiquer des informations de routage **entre** les ASs. BGP construit un graphe de systèmes autonomes, en utilisant les informations échangées entre les speakers d'ASs voisins. Du point de vue de BGP, l'Internet en entier constitue un unique graphe de domaines, chacun étant identifé par un unique **numéro de système autonome** (*ASN*). Dans ce graphe, BGP établira un chemin entre chaque paire AS désirant communiquer, en garantissant l'absence de cycles dans ces chemins. La dénomination *protocole à vecteur de chemin* provient justement de la méthode d'évitement de cycles utilisée [1].

2.2 Caractéristiques de BGP

La fonction première de BGP est de permettre à des systèmes communiquants de s'échanger des informations d'accessibilité aux réseaux qui résident dans chaque système ainsi que les réseaux externes que connait chaque système. C'est à dire que deux AS peuvent se communiquer, grâce à BGP, les destinations (réseaux) que chacun peut atteindre à travers l'autre.

BGP possède plusieurs caratéristiques qui lui sont propres :

1. BGP utilise pour ses transmissions un protocole de transport fiable, et exige que ce protocole ne rompe la connexion qu'après avoir délivré les dernières données (déconnexion explicite). Dans l'environnement TCP\IP, le protocole utilisé par BGP est le protocole TCP.
   Cela permet à BGP de ne plus s'inquiter des problèmes de pertes et de retransmission. Et, contrairement aux protocoles qui utilisent un

[1] la section 2.6 explique cette méthode

transport de type UDP ou IP, BGP ne peut effectuer de *broadcast* ni de *multicast* (diffusion simultanée d'informations vers de multiples destinataires).

2. Un speaker BGP n'entre en relation avec un autre speaker, que s'ils ont tous deux été configurés pour se reconnaitre. Il n'y a pas de découverte automatique de voisins, tout doit obligatoirement être spécifé manuellement pour BGP. C'est ce qui permet l'utilisation de protocole de transport tel que TCP (la découverte automatique fait souvent appelle au *broadcast* ou au *multicast*).
3. BGP utilise un algorithme à vecteur de chemin pour éviter les cycles dans ses routes. De plus, BGP n'utilise pas les métriques habituellement utilisées pour les protocole de routage (type plus court chemin ou moindre nombre de *Hops*). Au lieu de cela, il adopte une architecture basée sur des politiques de routage indépendament déployées sur les différents ASs de l'Internet.
4. Contrairement à la majorité des autres protocoles de routage, BGP n'adopte pas une stratégie d'envoi d'informations de routage en continue (périodiquement). Un speaker BGP n'envoi de message à ses peers que lorsque il détecte une mise à jour de ses propres informations de routage. A cause de cela, chaque speaker BGP ne transmet à ses peers que les routes qu'il utilise lui-même.
5. Un speaker BGP conserve, pour chaque destination, les routes qu'il a reçu de chacun de ses peers. En sauvegardant les chemins alternatifs vers une destination, en plus du meilleur chemin, BGP peut rapidement réagir à la perte du chemin primaire. Cela devrait raccourcir les périodes d'instabilité des tables de routage.
6. Dans un AS voisin de plusieurs autres ASs, et si cet AS est configuré pour servir de transit aux flux des autres ASs, BGP opère à deux niveaux (voir la figure 2.1) :
   - BGP est utilisé pour transporter des informations de routage entre les ASs. Cette utilisation est alors appellée **External BGP** (**E-BGP**).
   - BGP est utilisé pour distribuer des informations provenant de l'extérieur d'un AS entre les *speakers* de cet AS. Cette utilisation de BGP est alors appellée **Internal BGP** (**I-BGP**).

   E-BGP et I-BGP constituent en fait le même protocole (ils utilisent les mêmes messages), mais ils obéissent à différentes règles quand à la distribution des informations de routage.
7. En transmettant une route vers l'un des ses peers, un speakers y adjoint un ensemble d'informations sur la route, sous la forme d'**attributs**

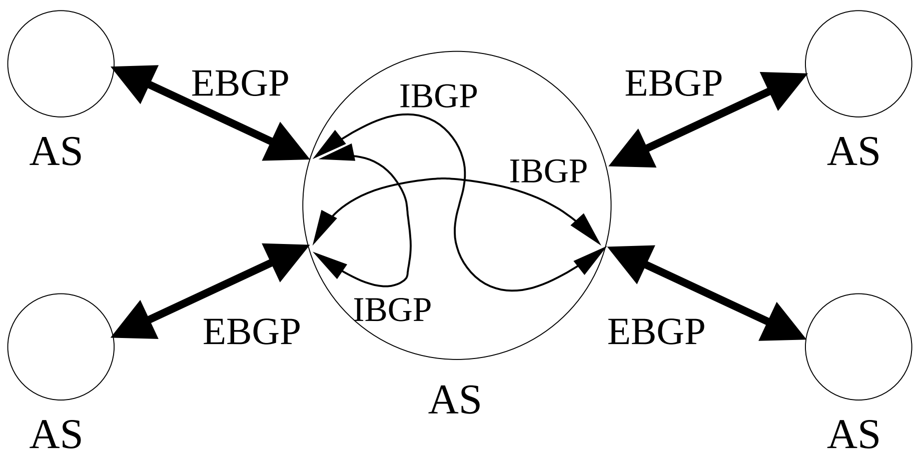

Fig. 2.1 – Positionnement de eBGP et de iBGP

de chemin. Ces attributs, associées à une route, fournissent un moyen pour le récepteur d'une annonce sur cette route, de prendre différentes décisions (rejet, acceptation, modification,...).

2.3 Fonctionnement de BGP

Le protocole BGP fonctionne par échange de **messages**. Un speaker désirant transmettre une information de routage à l'un de ses peers devra, donc, lui transmetre un **message BGP**. Ces messages transportent essentiellement deux types d'information : le speaker peut apprendre une nouvelle route vers une destination, auquel cas il émet une **annonce de route** vers ses peers (*route announcement*) ; sinon le speaker peut apprendre que la route qu'il utilisait est devenu inopérante, auquel cas il émet à ses peers un **rejet de route** (*route withdrawal*).

Avant d'effectuer ces échanges, deux speakers doivent établier une session BGP. Au cours de la session, ils doivent s'envoyer les informations que chacun à appris de ses autres peers (à condition qu'eux même utilisent les routes ainsi apprises), ils doivent s'envoyer régulièrement des messages pour s'assurer de la présence du lien entre eux, et ils doivent s'informer des causes des erreurs que chacun peut rencontrer.

2.3.1 Ordre chronologie des opérations

Quand on ordonne les actions à effectuer par BGP selon le moment ou sont effectuées ces actions, on obtient l'ordre suivant :

1. Les deux speakers forment d'abord une connexion entre eux au niveau du protocole de transport (TCP).
2. Ils s'échangent des messages OPEN pour ouvrir la connexion, établir le session, et négocier les paramètres de cette session.
3. Le flot de données initialement échangé est la table de routage BGP entière : chaque speaker doit, par une suite de messages UPDATE, informer son peer de **toutes** les destinations qu'il connait, ainsi que les routes pour y accéder.
4. Quand l'un des speaker détecte une modification de sa table de routage, il doit émettre un message UPDATE vers son peer. Ces messages provoquent une modification incrémentale des informations chez le peer, c'est à dire que les données du message sont interpretées en fonction des données précédemment reçus de l'émetteur.
5. Des messages KEEPALIVE sont périodiquement envoyés pour s'assurer de la pérennité de la connexion. L'absence de réception de messages KEEPALIVE pendant une trop longue période signifira la perte de la connexion entre les speakers et la terminaison de la session.
6. Lorsque BGP détecte une erreur, une condition spéciale ou une rupture de connexion, le speaker subissant l'erreur doit envoyer un message NOTIFICATION pour informer son peer des conditions de l'erreur ; puis ils cloturent tout deux la session, avant de fermer la connexion de transport entre eux. Dans ce cas, chaque speaker doit absolument rejeter **toutes** les routes qu'il a reçu de son peer.
7. Après une déconnexion, et si les speaker sont configurés pour produire ce comportement, les deux peers peuvent tenter de rétablir la session ; ils devront alors démarrer à partir du début de la procédure.

2.3.2 Automate à états finis de BGP

BGP maintient pour chaque session BGP un **état** dont les valeurs possibles sont décrites dans l'automate à états finis de BGP [RL95]. Cet automate décrit aussi les transitions entre ces états, les causes déclenchant ces transitions, et leurs effets. Ce qui suit est un exposé des états qui composent cet automate, et des transitions entre eux :

Idle State état initiale de l'automate. Dans cet état, BGP refuse toutes les connexions entrantes, et aucune ressource n'est allouée au peer. Quand BGP détecte un évènement *Start Event* (initié par un script, par exemple), il doit initialiser toutes les ressources BGP, déclencher le timer *ConnectRetry*, initier une connexion TCP vers le peer BGP et passer à l'état **Connect State**.

Connect State Dans cet état, BGP est en attente de la terminaison de la connexion du protocole de transport. Si cette connexion réussi, BGP désactive le timer *ConnectRetry*, complète l'initialisation, envoi un message OPEN vers son peer et passe à l'état **Open Sent State**. Par contre, si cette connexion échoue, il faut réinitialiser le timer *ConnectRetry*, et passer à l'état **Active State**. Si le timer *ConnectRetry* expire, il faut le réinitialiser et réinitier une connexion TCP vers le peer tout en écoutant une éventuel connexion entrante.

Active State BGP est en attente d'une connexion TCP entrante. Si elle réussi, BGP désactive le timer *ConnectRetry*, envoi un message OPEN au peer, positionne son timer *HoldTimer* et passe à l'état **Open Sent State**. Si aucune connexion n'est détectée avant l'expiration du timer *ConnectRetry*, BGP doit réinitialiser ce timer, initier une connexion TCP et rester à l'état **Active State**.

Open Sent State Dans cet état, BGP est en attente d'un message OPEN de la part de son peer. Quand il le reçoit, il envoi vers son peer un message KEEPALIVE et positionne le timer *KeepAliveTimer*. La valeur du timer *HoldTimer* est modifiée en fonction de la valeur négociée par les messages OPEN.L'état passe alors à **Open Confirm State**.

Open Confirm State Dans cet état, BGP est en attente d'un message KEEPALIVE de confirmation. Quand il le reçoit, l'état passe à **Established State**. Tout autre évènement (expiration du timer *HoldTimer*, réception d'un message NOTIFICATION, déconnexion au niveau transport,...) provoque l'émission d'un message NOTIFICATION puis le passage à l'état **Idle State** (annulation de l'ouverture de session).

Established State Cet état indique l'établissement complet de la session BGP. Les peers peuvent s'échanger des messages UPDATE, KEEPALIVE ou NOTIFICATION, tant que dure la session. La réception d'un message KEEPALIVE provoque le repositionnement de *HoldTimer*. A chaque fois que *KeepAliveTimer* expire il provoque l'émission d'un message KEEPALIVE et le repositionnement du timer.
Toute erreur détectée dans le traitement des messages, ainsi que l'expiration de *HoldTimer* provoquent l'émission d'un message NOTIFICA-

TION indiquant la source de l'erreur, la fermeture de la session et de la connexion et le passage à l'état **Idle State**.

2.3.3 Modèle générale de routage de BGP

La figure 2.2 expose le schéma générale de fonctionnement de BGP. On y remarque la séparation qui existe entre la table de routage IP et la table de routage BGP. En fait BGP stocke séparement les routes qu'il reçoit de chacun de ses peers, les routes qu'il utilise (les meilleurs des routes reçues de ses peers) et les routes qu'il doit transmettre à ses peers. Les tables utilisées par BGP sont appellées RIBs (*Routing Information Base*), et sont de trois types : Adj-RIB-In, Adj-RIB-Out et Loc-RIB [2].

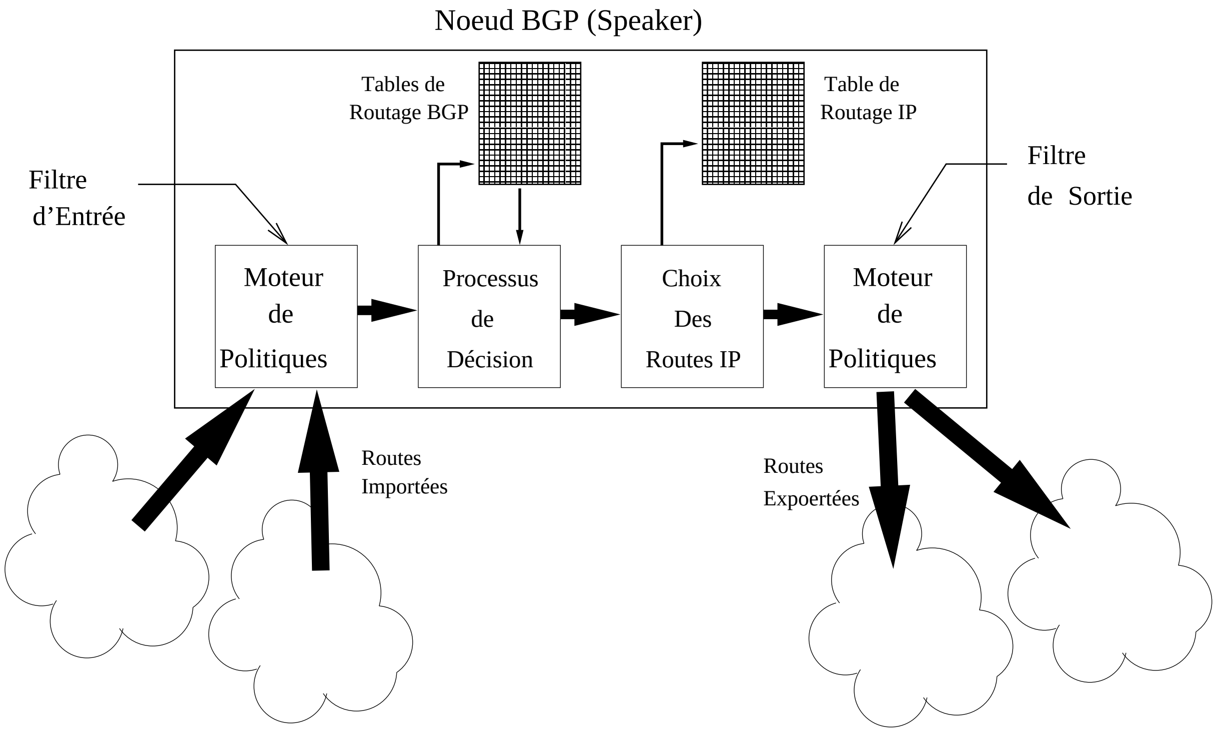

Fig. 2.2 – Schéma de fonctionnement d'un speaker BGP

Comme le montre la figure 2.2, à la réception d'un message de nouvelle route (ou de rejet de route), le message subit d'abord un filtrage à l'entrée du nœud BGP (speaker). Ce filtrage se présente sous la forme d'un composant logiciel qui implémente une politique de routage et permet d'examiner toutes

[2] voir la section 2.4

les informations contenues dans le message pour prendre une décision (rejet ou non du message, modification ou non de son contenu). Dans la majorité des implémentations de BGP, la politique se présente sous la forme de **règles de décision**. Chaque règle spécifie certaines conditions à remplir par les données du message, et les actions à effectuer quand ces conditions sont remplies.

La seconde opération de BGP consiste en l'application du processus de décision. Ce processus est chargé de décider du chemin à prendre pour atteindre la destination spécifiée dans le message. Si plusieurs chemins sont disponibles, les champs de ces chemins sont examinés et le "meilleur chemin" est choisit. [3]Ces routes sont alors placées dans la table de routage IP et peuvent, dès lors, servir au *forwarding* des paquets qui passent par le routeur.

BGP doit alors publier les routes qu'il utilise aux autres peers BGP, c'est à dire qu'il ne publie à ses peers que les "meilleurs" de ses routes. Il faut alors faire passer les routes à publier par un filtre de sortie qui, en appliquant une politique de routage similaire à celle d'entrée, décide des routes à envoyer et des éventuelles modifications à y apporter avant de les transmettre à ses peers.

Les speakers BGP se transmettent les informations de routage sous forme de messages UPDATE. Chaque message peut contenir une ou plusieur destinations accédées par le même chemin, ainsi que le chemin qui y mène et des informations supplémentaires à propos du chemin. De plus, ces messages peuvent transporter des informations de rejet de route spécifiant que le chemin publié auparavant n'est plus utilisable. Il faudra au récepteur d'un tel rejet, changer de route s'il existe un chemin alternatif, sinon il lui faudra totalement invalider la destination.

BGP permet à un speaker d'informer ces peers qu'une route doit être rejetée (*withdrawned*) par trois méthodes :

1. Publier explicitemet la destination dans la partie WITHDRAWN ROUTES d'un message UPDATE à envoyer aux peers.
2. Envoyer aux peers une route de replacement avec la même destination (champs *Network Layer Reachability Information* identique). Cette méthode correspond à une suppression implicite de la route précédement publiée.
3. La session BGP entre deux speakers peut être cloturée, ce qui supprime implicitement **toutes** les routes que chaque speaker à appris de l'autre.

[3]Les opérations menées par le processus de décision sont exposés dans la section 6

2.4 Stockage des chemins BGP

BGP stocke au niveau de chaque speaker BGP trois sortes d'informations : les routes qu'il a reçu de ses peers, les routes qu'il a choisit comme chemin pour atteindre les destinations associées aux routes, et les routes qu'il a décidé d'annoncer à ses peers, pour que les peers puissent atteindre ces destinations et transitant par lui, s'ils le désirent.
Ces informations sont stockées sous forme de tables répondant au nom de RIB (*Routing Information Base*), et sont de trois types [RL95] :

1. La table **Adj-RIB-In** contient les routes apprises par des messages UPDATE entrants, elles représentent donc des routes fournies en entrée au processus de décision de BGP. Un Adj-RIB-In ne contient que les routes apprises que d'**un seul** peer. Chaque Adj-RIB-In liste toutes les plus récentes routes publiées par le peer associé. Une nouvelle route indiquant la même destination qu'une ancienne route, provoque un rejet implicite de l'ancienne route et son remplacement par la route nouvellement arrivée.
   Les messages tels qu'il sont stockés dans ces tables peuvent présenter des différences par rapport aux messages originalement reçus des peers, du fait des politiques appliqués sous forme de filtre *ingress* (filtre d'entrée).

2. Le **Adj-RIB-Out** contient les informations que le processus de décision destine à la publication vers ses peers. Ces informations seront donc structurées en messages UPDATE à émettre vers le peer. Comme pour le Adj-RIB-In, chaque Adj-RIB-Out est associé à l'un des peers du speaker, et les informations qu'il contient peuvent différer des informations contenus dans le Loc-RIB, du fait des politiques de routage appliquées par le filtre *egress* (filtre de sortie).

3. Le **Loc-RIB** contient les informationss de routage locales obtenues en appliquant le processus de décision sur les données des Adj-RIB-In des différents peers. Cette table contient donc les "meilleurs" chemins, selon les critères du speaker, pour atteindre les destinations associées. Le critère de choix dépend des politiques de routage adoptées par chaque speaker. Mais, en général, ces choix se font selon l'identité de l'AS du peer publiant la route, ou la longeur en nombre de saut (*Hop number*) du chemin menant de l'AS du peer vers l'AS hébergeant la destination.

2.5 Les messages BGP

2.5.1 Formats des messages BGP

Tous les messages BGP sont formés d'un **entête commun** suivit d'un **corps de message** variant en taille et en structure selon le type de message. L'entête commun est composé de trois champs :

- Le champs **Marker**, dont la longeur est de 16 octets, est utilisé à des fins de synchronisation ou de sécurité. Sa valeur dépend du message envoyé et du type de sécurité employée. Dans le cas de message OPEN, tous les bits du champs Marker sont à 1. Au cours de la session, ce champs prend une valeur que le récepteur peut prédire en fonction du type de sécurité utilisée.
- Le champs **Length** contient, sur deux octets, la taille de tout le message, entête compris, en nombre d'octets. Ce champs ne peut prendre que les valeurs comprises entre 19 (message contenant l'entête seul) et 4096 (taille maximale d'un message BGP).
- Le champs **Type**, codé sur un seul octet, indique le type du message émis. Il ne peut prendre que les valeurs 1 (message OPEN), 2 (message UPDATE), 3 (message NOTIFICATION) et 4 (message KEEPALIVE).

Le corps du message est composé d'une suite d'octets structurés selon le type du message. Dans ce qui suit, sont décrit les quatres types de messages et leurs structures.

Structure du message OPEN

Le message OPEN est le premier message échangé après l'établissement de la connexion TCP. Il est composé de 6 champs :

- Le champs **Version** transporte, sur un octet, la version du protocole utilisée par l'émetteur du message. Cela permet à un couple de peers BGP de s'accorder sur la version de BGP qu'ils utiliseront dans la session.
- Le champs **My Autonomous System** spécifie le numéro d'AS de l'émetteur du message (*Autonomous System Number*, **ASN**). Chaque domaine sur Internet dispose un ASN unique qu'il utilise dans ses transactions BGP. La longeur de deux octets du champs My Autonomous System permet de différencier 65536 ASN.
- Le champs **Hold Time** transporte, sur deux octets, le nombre de secondes que propose l'émetteur pour le **hold timer**. Cette valeur correspond à la période maximum de temps que peut passer un speaker sans recevoir de messages de son peer (messages UPDATE ou KEEPALIVE)

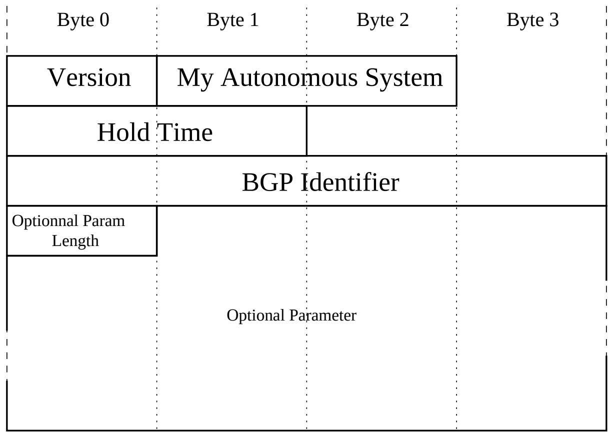

FIG. 2.3 – Schéma du message OPEN

avant qu'il ne considère que la session, entre eux, est perdue. La valeur choisie pour ce timer est le minimum entre la valeur localement configurée et celle reçue par message. La spécification autorise une valeur nulle signifiant qu'il n'y aura pas d'échange de messages KEEPALIVE (mode opératoire fortement déconseillé, car cela ne permet pas de détecter les pertes de connectivité).

- Le champs **BGP Identifier** (4 octets) transporte l'identité BGP de l'émetteur. En général, elle correspond à l'une des adresses IP du speaker, comme une adresse IP associée à une interface virtuelle du routeur BGP. Cette identité doit être unique et utilisée pour toutes les sessions BGP qu'établira ce speaker.
- Le champs **Optional Parameter Length** (1 octet) indique en octets la longueur du champs **Optional Parameter**. Il peut être à 0, si ce dernier est vide.
- Le champs **Optional Parameter** est de longeur variable. Il contient, quand il est présent, une liste de paramètres optionnels. Chaque paramètre doit être sous la forme d'un tuple

$$< ParameterType, ParameterLength, ParameterValue >$$

Parameter Type identifie, sur un octet, les paramètres individuels. *Parameter length* (1 octet) donne la longeur en octets du sous-champs *Parameter Value* suivant. Enfin, *Parameter Value* est un champs de longeur variable, dont la valeur est interpretée selon le type du paramètre. Actuellement, seul le type 1 est définie, il est utilisé pour le transport d'informations d'authentification.

Structure du message UPDATE

Le message UPDATE est le message utilisé entre les peers pour l'échange des informations de routage. A chaque fois qu'un speaker désire annoncer un préfixe de route ou rejeter un préfixe précédement annoncé, il utilise un message UPDATE. Chaque message UPDATE ne peut convoyer qu'une seule annonce de route, mais cette route peut mener vers de multiples destinations, qui résident toutes obligatoirement dans le même AS. Par contre, chaque message peut transporter un nombre quelconque de routes rejetées (*route withdrawals*).

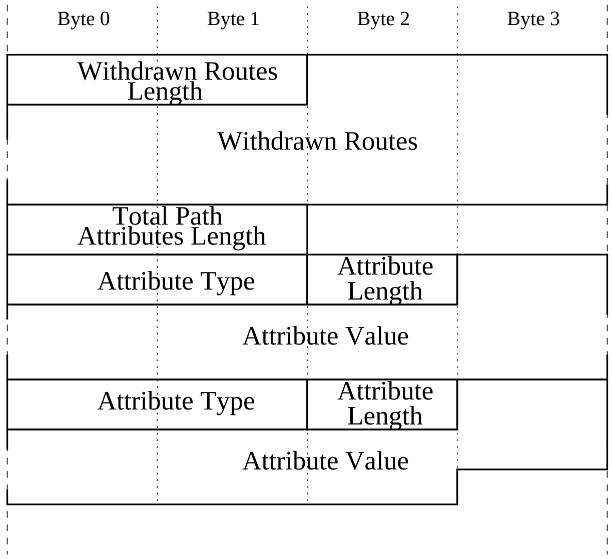

FIG. 2.4 – Schéma du message UPDATE

Chaque message UPDATE comporte les champs suivants :

- **Withdrawn Routes Length**, est un champs sur deux octets qui indique en octets la longeur du champs **Withdrawn Routes**. Si ce champs est à 0, aucune route n'est explicitement rejetée.
- Le champs **Withdrawn Routes** est de longeur variable, et contient une liste de préfixes d'adresses IP désignant les routes annoncées auparavant par le speaker émetteur de ce message, et qui ne sont plus utilisables. Chaque préfixe est encodé sous la forme

  $$< Length, Prefix >$$

  Le sous-champs *Length* donne, en bits, la longeur du préfixe. Le sous-champs *Prefix* contient le préfixe d'adresse IP, complété par assez de bits à 0 pour obtenir un nombre de bits multiple de 8.

– Le champs **Total Path Attributes Length** indique, sur 2 octets, la longeur du champs **Path Attributes**. Une valeur nulle indique qu'il n'y a pas de routes à annoncer.
– Le champs **Path Attributes** est de longeur variable. Il est composé d'une suite d'attributs de chemin. Tous les attributs de chemins sont appliquables à toutes les destinations spécifiées dans le le champs NLRI du message. Chaque attribut consiste en un triplet de longeur variable, dont la structure est :

$$< AttributeType, AttributeLength, AttributeValue >$$

Attribute Type est un sous-champs comprenant un octet de *Flags* et un octet *Attribute Type Code*. L'**Attribute Type Code** est un numéro assigné par l'IANA, et qui identifie de manière unique l'attribut. L'octet *Flags* contient quatre valeurs binaires décrivant l'attribut, et quatre bits inutilisés. A partir du bit de poids fort vers le bit de poids faible, la signification des flags est :

1. Le premier bit, **Optional Bit**, indique si l'attribut est **optionnel** (bit à 1) ou **obligatoirement reconnu** (bit à 0). Un attribut **obligatoirement reconnu** doit être reconnus par toutes les implémentations de BGP.
2. Le second bit, **Transitive Bit**, doit être mis à 1 pour les attributs reconnus. Pour les attributs optionnels, ce bit indique si l'attribut est **transitif** (bit à 1) et qu'il doit, donc, être passé aux autres peers BGP lors de l'annonce de cette route ; ou si il est **non transitif** (bit à 0) et que l'attribut ne doit pas être transmis aux peers, lorsque le speaker leur communique cette route.
3. Le troisième bit, **Partial Bit**, indique si un attribut optionnel, transitif contient une information **complète** (bit à 0) ou **partielle** (bit à 1). En fait ce flag permet de savoir si tous les ASs sur lesquels est passé cet attribut, ont reconnus cet attribut. En effet, un attribut optionnel transitif, peut ne pas être reconnus par une implémentation mais quand même retransmis sans modification. Ce qui produirait, sous certaines conditions, des attributs contenant des informations partielles.
4. Le quatrième bit, **Extended Length Bit** indique la la longeur du champs **Attribute Length** est sur **un** octet (bit à 0) ou **deux** octets (bit à 1).

Le second sous-champs de l'attribut estle sous-champs **Attribute Length**, qui indique la longeur en octets du sous-champs **Attribute Value**. Ce

sous-champs peut tenir sur un ou deux octets en fonction du flag **Extended Length Bit**. Le sous-champs **Attribute Value** contient la valeur de l'attribut ; cette valeur est traitée en fonction des valeurs des champs **Attribute Type** et **Attribute flags**. Tous les attributs et leurs significations sont traités dans la sous-section 2.5.2.
- Le dernier champs d'un message UPDATE est le champs NLRI (**Network Layer Reachability Information**). Il contient une liste de destinations sous forme de préfixes d'adresses IP. La longeur de ce champs peut être calculée à partir de la longeur du message, celle de l'entête, la longeur totale des attributs et la longeur de routes rejetées. Notez que tous les attributs contenus dans la liste d'attributs du message s'appliquent à **toutes** les destinations listées dans ce champs. Ce qui signifie qu'une implémentation de BGP ne peut regrouper plusieurs destinations dans une même annonce de route BGP, que si ces destinations ont exactement les mêmes attributs.

Structure du message KEEPALIVE

Le mécanisme utilisé par BGP pour s'assurer qu'une connexion entre peers est toujours présente est basé sur l'échange périodique de messages KEEPALIVE entre les peers. En fait, les peers doivent s'envoyer des messages KEEPALIVE à un rythme qui empêche le timer **HoldTimer** d'expirer. La spécification de BGP propose un interval de temps entre les envois de messages KEEPALIVE égale à un tiere de la valeur du timer **Hold Timer**. Chaque message KEEPALIVE ne comporte que l'entête du message BGP, avec la bonne valeur du champs **Type** (4, pour signifier KEEPALIVE). La longeur de ce message est toujours de 19 octets.

Structure du message NOTIFICATION

Un message NOTIFICATION est envoyé immédiatement après la détection d'une erreur, et cela juste avant la fermeture de la connexion TCP. Cela permet de fournir aux administrateurs une indication sur la cause de l'erreur. Chaque message NOTIFICATION contient, en plus de l'entête, les champs :
- **Error Code** : (1 octet) indique le type d'erreur qui est survenu.
- **Error Subcode** : (1 octet) fournit des informations plus spécifiques à propos de la nature de l'erreur
- **Data** : est de taille variable. Sa longeur peut être calculée à partir de la valeur du champs **Length** (taille totale du message).Ce champs transporte des données interpretées en fonction du code d'erreur et du souscode d'erreur. Ces données sont utilisées pour diagnostiquer les

raisons de la notification d'erreur.

2.5.2 Les attributs de chemins du message UPDATE

Les attributs de chemin (ou de route) sont l'élément le plus important de BGP. Ils transportent, en effet, la plus grande part des informations de routage échangées entre les peers BGP. Les attributs ont été encodé de manière à pouvoir facilement ajouter de nouveaux attributs, sans avoir à changer totalement le protocole. Dans ce qui suit, on décrit les attributs faisant partie de la spécification de base du protocole BGP. Les attributs introduits par la suite sont exposés plus loin dans le text (à la section 2.7.2 et à la section 2.7.1).

Chaque attribut ne peut être présent qu'une seule fois dans chaque message BGP. L'absence ou la présence de chaque attribut dépend des caractéristiques de chaque attribut, et des informations qu'il transporte. Certains de ces attributs doivent être reconnus par **toutes** les implémentations de BGP (*well-known attribute*), les autres ne sont pas obligatoirement reconnus (*optionnal attribute*). Parmi les attributs obligatoirement reconnus, il est des attributs qu'il faut inclure **dans chaque** message BGP (*mandatory attribute*) ; les autres pouvant être absents des messages (*discretionary attribute*). Enfin, les attributs peuvent être classés selon qu'il doivent ou pas être passés aux autres peers, quand ils sont présents dans un message (*transitive attribute*).

L'attribut ORIGIN

C'est un attribut reconnus de toute implémentation et est obligatoire dans chaque message. Son *type code* est de 1, et il est transporte, sur un octet, une information sur l'origine des données de routage. En fait, le *speaker* BGP qui a généré ces données, doit renseigner sur la manière dont ces données ont été injectées dans BGP. Pour cela, l'attribut peut prendre trois valeurs :

- IGP (1) indique que le préfixe a été appris d'un IGP.
- EGP (2) indique que le préfixe a été appris à travers un protocole EGP.
- INCOMPLETE (3) indique que le préfixe provient d'autre chose qu'un IGP ou EGP. Cela est surtout utilisé pour les routes configurées de manière statique.

L'attribut AS_PATH

Cet attribut contient la liste des ASs qui ont vu passer cette information de routage. En effet, chaque AS qui annonce ce préfixe doit ajouter à cet

attribut **son** numéro d'AS, en début de liste. Cette liste a deux rôles : elle permet à tout système autonome de découvrir si l'information de routage lui est revenu après avoir traversé une boucle dans la topologie des ASs (algorithme de détection de cycles) ; et elle joue un rôle dans l'algorithme de sélection de chemin du protocole.
L'attribut AS_PATH est obligatoirement reconnu de toute implémentation, et il est présent dans chaque message d'annonce de route. Il possède le *type code* 2. Cet attribut est encodé sous la forme d'une séquence de *segments d'AS-PATH*. Chaque segment est sous la forme

$$< PATHSegmentType, PATHSegmentLength, PATHSegmentValue >$$

Le champs *PATH Segment Type* peut prendre deux valeurs :
- AS-SET (1) contient un ensemble désordonné de numéros d'AS.
- AS-SEQUENCE (2) contient une liste ordonnée de numéros d'AS.

En fait, la majorité des AS_PATH contiennent un segment AS-SEQUENCE unique. Le but des AS-SETs est de permettre l'aggregations de routes qui contiennent des AS_PATH différents. Ainsi, l'aggregation de deux routes contenant des AS_PATH différents produira une route contenant un AS_PATH composé d'AS-SETs et d'AS-SEQUENCEs. Les AS-SEQUENCEs proviendront des suites de numéros d'AS contenus dans les deux routes, dans le même ordre. Les AS-SETs proviendront des numéros d'AS contenus dans l'une des deux routes, ou ceux présents dans les deux routes dans des ordres différents.

Un speaker BGP qui veut annoncer une information de routage, doit respecter les contraintes suivantes :
- Si le speaker doit propager une information de routage vers un de ces peers *internes* (appartenant au même AS que lui), il ne doit pas modifier l'attribut AS_PATH.
- Si le speaker doit propager une route vers un peer externe, il doit joindre à l'AS_PATH son numéro d'AS, de la manière suivante :
  1. Si le premier segment de l'AS_PATH est un segment AS-SEQUENCE, le système local y ajoute son numéro d'AS comme le dernier élément (le plus à gauche) ;
  2. Si le premier segment de l'AS_PATH est un segment AS-SET, alors le système local ajoute à l'AS_PATH un nouveau segment AS-SEQUENCE, comprenant son numéro d'AS.
- Si le speaker génère la route, il doit envoyer un message UPDATE avec un attribut AS_PATH vide vers ses peers internes, et un messages UPDATE avec un attribut AS_PATH contenant un AS-SEQUENCE comprenant son numéro d'AS uniquement, vers ses peers externes.

Les deux autres champs de chaque segments d'AS_PATH sont le *PATH Segment Length*, qui indique le nombre de numéro d'AS dans le segment, et le *PATH Segment Value* qui contient la liste de numéro d'ASs.

L'attribut NEXT_HOP

Selon le paradigme du *Hop-by-Hop* utilisé sur l'Internet, un routeur doit diriger les données, émises vers une certaine destination, vers le prochain routeur qui les rapprochera le plus de cette destination. Tout speaker BGP doit, donc, inclure dans chaque annonce de route vers un préfixe, l'adresse de prochain hop pour atteindre ce préfixe. L'attribut NEXT_HOP est utilisé pour désigner cette adresse de prochain saut. L'adresse IP du routeur de bordure, incluse dans cet attribut devra être utilisée comme prochain hop de **toutes** les destinations listées dans le message UPDATE.
L'adresse spécifiée dans cet attribut doit respecter certaines règles :

- Elle ne doit pas correspondre à l'adresse du peer vers lequel est émise cette annonce.
- Elle ne doit pas correspondre à l'adresse IP servant d'identité au speaker émettant cette annonce (mais peut correspondre à l'une de ces autres adresses IP, s'il en possède plusieurs).
- Le routeur de bordure possédant cette adresse, doit être interne au domaine du speaker et l'interface associée à cette adresse doit partager un sous-réseau avec le speaker et son peer ; ou
- Le routeur de bordure auquel correspond cette adresse, doit être externe au domaine du speaker, le speaker doit avoir appris cette adresse de l'un de ses peers et l'interface associée à cette adresse doit partager un sous-réseau avec le speaker et son peer.

Cet attribut possède le *type code* 3. C'est un attribut reconnu et obligatoire qui tient toujours sur 4 octets ; lesquels seront directement placés derrière le champs Type Code (pas de champs pour la longeur de l'attribut). De plus, la spécification de BGP [RL95], conseille de ne pas altérer la valeur de cet attribut lorsque la route est annoncée à un peer interne.

L'attribut MULTI_EXIT_DESCRIMINATOR

Cet attribut est utilisé pour établir une discrimination entre les liens externes (inter-ASs) reliant deux ASs voisins. Plus clairement, cet attribut permet de choisir, parmi plusieurs points d'entrée/sortie d'un AS voisin, le point d'entrée optimale pour atteindre une destination dans cet AS ou au delà. Concrètement, cet attribut transporte une valeur, entière, non signée, sur 4 octets, qui est une métrique exprimant un degré de préférence pour le

lien reliant les deux ASs.
D'après la spécification de BGP, la valeur de cet attribut ne doit être utilisée que lorsque tous les facteurs, plus prioritaires, de comparaisons de route sont égaux. Il faut alors préférer la route qui possède le point d'entrée/sortie à la métrique la plus basse. Et afin de s'assurer de l'homogénéité du routage au sein d'un AS, l'attribut MULTI_EXIT_DESCRIMINATOR doit être propagé vers les peers internes d'un speaker, mais ne doit absolument pas être propagé vers les peers externes.
L'attribut MULTI_EXIT_DESCRIMINATOR dispose du *type code* 4 ; c'est un attribut optionnel non transitif. Il tient toujours sur quatre octets, et représente un entier non signé.Il faut absolument se souvenir que cet attribut ne peut être utilisé qu'entre des routes apprises d'un même AS. Comparer les métriques de deux routes originaires de deux ASs différents ne fournit aucune réelle information car ces domaines disposent de topologies et de politiques de routage différentes, donc de critères d'établissement des valeurs du MED complètement différents.

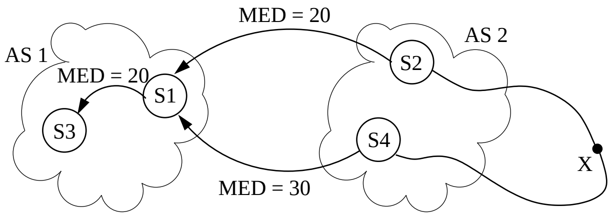

Fig. 2.5 – Scéma d'utilisation du MED : la route vers X annoncée par S2 et par S4 à S1 sont indéffirentiables par d'autres moyens. S1 choisit alors la route dont le MED est le plus petit (dans ce cas S2) et transmet cette route à S3 avec le MED inchangé

L'attribut LOCAL_PREF

Cet attribut correspond au *type code* 5, c'est un attribut obligatoirement reconnu de toutes les implémentations, mais optionnels dans les messages UPDATE. Il est representé par un entier non signé sur quatre octets.
Cet attribut est utilisé pour la sélection des routes au sein des ASs. Tout AS qui reçoit une route d'un peer externe et qui veut la propager vers ses peers internes, doit calculer le degré de préférence de cette route et l'ajouter au message UPDATE sous forme d'un attribut LOCAL_PREF. A la réception de plusieurs routes, un speaker choisit la route dont la valeur de l'attribut LOCAL_PREF est la plus grande. Il est, cependant, indispensable de ne pas

propager cette métrique vers les peers externes, car la valeur calculée de cet attribut concerne exclusivement l'AS local, et les politiques qu'il utilise.

L'attribut ATOMIC_AGGREGATE

Lorsqu'un AS reçoit des annonces de routes pour deux destinations *overlappées* (une des destinations est un sous-ensemble de l'autre), que les attributs de routes sont différents, et que l'AS décide d'annoncer la route la moins spécifique sans annoncer la route la plus spécifique (c'est à dire qu'il ne veut annoncer que la route vers le préfixe désignant le sur-ensemble), alors le speaker BGP ayant effectué cette decision doit attacher un attribut ATOMIC_AGGREGATE à la route. Un speaker recevant une telle route, saura immédiatement qu'il ne faut pas désaggreger le préfixe pour créer des préfixes plus spécifiques, car la présence de l'attribut ATOMIC_AGGREGATE signifie que les chemins vers certains sous-ensemble de la destination aggrégée peuvent traverser des ASs absents de l'attribut AS_PATH.
Cet attribut doit être reconnu par toutes les implémentations, mais n'est pas obligatoirement présent dans chaque message. Il correspond au *type code* 6, et est de longeur nulle. En effet, il indique par sa simple présence que le préfixe est un aggregat atomique (qui ne peut donc pas être fractionné).

L'attribut AGGREGATOR

Quand un speaker exécute une aggregation de route sur quelques unes des routes qu'il a reçu des ces peers, il peut décider de joindre à la route ainsi créée, un attribut AGGREGATOR afin d'indiquer l'identité du speaker qui a effectué l'aggregation. Cet attribut est optionnel et transitif, et comprend toujours le numéro d'AS du speaker suivi de l'identité BGP du speaker. Sa longeur est, donc, toujour de six octets. Son *type code* est le 7.

2.6 Traitement des messages UPDATE

Après l'établissement de la session BGP entre deux peers, ils peuvent commencer à s'envoyer des messages UPDATE afin de s'informer mutuellement des destinations qu'ils peuvent atteindre. Chaque message UPDATE reçu par l'un des speakers subit un traitement particulier qui aura pour résultat un éventuel changement de la table de routage des routeurs de bordure de l'AS du speaker, et la propagation de l'information contenue dans ce message, vers les autres peers du speaker.
Tout message UPDATE reçu par un speaker passe d'abord, comme tout autre message BGP, par un test de validité. Ce test examinera successivement tous

les champs et attributs du message, pour s'assurer de la correction des informations qu'ils transportent. Une erreur détectée lors de cette phase provoque l'émission d'un message NOTIFICATION vers le peer, avant la fermeture de la session.
Ensuite, le message subit un filtrage qui applique les politiques de routage de l'AS aux routes en entrée du noeud BGP. Ce filtrage a, en général, pour conséquence l'acceptation de la route contenue dans le message, le refus de cette route ou l'application de modifications sur les informations du message avant l'acceptation de la route.
Après le passage par le filtre *ingress*, le message UPDATE subit un traitement différencié selon son contenu :

- Si le message contient des routes à rejeter (le champs *Withdrawn Routes* est non vide), alors les routes annoncées auparavant par le peer émetteur du message, et dont les destinations sont contenus dans le champs *Withdrawn Routes*, doivent être supprimées du Adj-RIB-In correspondant à ce peer. Si, pour une destination, la meilleur route est ainsi supprimée, le speaker BGP devra exécuter son processus de décision.
- Si le message contient l'annonce d'une route, celle-ci doit être ajoutée au Adj-RIB-In associé au peer émetteur du message, puis les actions suivantes sont executées :
  1. Si le NLRI (*Network Layer Reachability Information*) de la route correspond à celui d'une route déja contenue dans l'Adj-RIB-In, alors la nouvelle route remplace l'ancienne (rejet implicite de l'ancienne route). BGP doit alors exécuter son processus de décision.
  2. Si la nouvelle route indique une destination absente de l'Adj-RIB-In, alors cette route doit être incluse dans l'Adj-RIB-In et BGP doit exécuter son procesus de décision.
  3. Si la route possède les mêmes attributs qu'une route déja présente dans l'Adj-RIB-In, et qu'elle indique une destination plus spécifique que l'ancienne route ,alors aucune action ne doit être entreprise.
  4. Si la nouvelle route indique une destination moins spécifique que la destination associée à une route présente dans l'Adj-RIB-In (*Overlapping Routes*), alors BGP doit exécuter son processus de décision sur l'ensemble des destinations décrites uniquement par la route la moins spécifique, c'est à dire les destinations appartenant aux préfixe le moins spécifique sans appartenir au préfixe le plus spécifique.

Pendant ce traitement, le speaker doit vérifier à tout moment que la route reçu n'est pas une route qu'il a généré lui même ou une route qu'il connaisse déja. Cela peut se produire si l'information de routage a effectué un cycle dans le graphe des ASs pour lui revenir. Le speaker doit, donc, vérifier **l'absence de son ASN de l'attribut AS_PATH de la route**. Toute route dont l'AS_PATH contient l'ASN du speaker doit immédiatement être supprimée. C'est ce qui constitue l'**algorithme d'évitement de cycles de BGP**. L'étape suivante, dans le traitement des messages BGP, concerne l'application du processus de décision aux routes apprises par les peers, afin de sélectionner, pour chaque destination apprise, la route à utiliser pour atteindre cette destination (la "meilleur" route, selon les critères du speaker).

2.6.1 Le processus de décision de BGP

Une composante importante de tout protocole de routage est l'algorithme utilisée par ce protocole pour sélectionner parmi tous les chemins possible vers une destination, celui qu'il choisira pour convoyer les paquets vers cette destination. BGP ne fait pas exception ; ainsi,le processus de décision de BGP a pour rôle de sélectionner parmi les routes de tous les Adj-RIB-Ins, une route pour chaque destination qu'il connait. La sélection se fait selon la politiques de routage configuré chez le speaker, et la sortie de ce processus est formé par les routes utilisées en local pour le *forwarding* vers les destinations et par les routes à annoncer aux peers et qui seront stockées dans les Adj-RIB-Outs.

Le processus de décision opère sur toutes les routes contenues dans chaque Adj-RIB-In, et est responsable de :

- la sélection de la meilleure route pour chaque destination. L'attribut NEXTHOP de cette route fournit alors l'adresse du routeur vers lequel sont convoyés les paquets destinés à cette destination.
- la sélection des routes à annoncer aux speakers du même AS que le système local.
- la sélection des routes à annoncer aux speakers des ASs voisins.
- l'aggrégation des routes et la réductions des informations à transmettre aux peers.

Le processus de **sélection** de routes est formalisé par la définition d'une fonction prenant comme arguments les attributs d'une routes donnée, et retournant une valeur entière non-négative exprimant le **degré de préférence pour cette route**. D'après la spécification de BGP4, cette fonction de calcul du degré de préférence ne doit pas prendre en compte l'existance ou la non-existance d'autres routes vers la destination, ni utiliser les attributs d'autres routes que celle dont elle calcul le degré de préférence.La sélection des routes consistera en l'application individuelle de la fonction de calcule du degré de préférence sur chaque route pratiquable — suivi par le choix de la route ayant

le plus haut degré de préférence.
Le processus de décision se déroule en trois phases distinctes :

Phase 1 : Calcul du degré de préférence

La première phase du processus de décision est invoquée à chaque fois qu'une nouvelle route est apprise, ou qu'une nouvelle route remplace ou rejète une ancienne route. Ce processus est alors appliqué aux routes nouvellement arrivées.
Dans cette phase, le speaker BGP doit déterminer un degré de préférence pour chaque nouvelle route. Si la route est apprise d'un peer interne du speaker (peer résidant dans le même AS que le speaker), alors le degré de préférence peut être soit extrait de l'attribut LOCAL_PREF de la route, soit calculé localement. Dans le cas des routes apprises de peers externes, le degré de préférence doit obligatoirement être calculé localement selon les informations de politiques préconfiguées.
Cette phase se termine lorsque toutes les routes de tous les Adj-RIB-Ins possèdent un degré de préférence. BGP fait, alors, appel à la phase 2 du processus de décision.

Phase 2 : Choix de la meilleure route

Cette phase du processus de décision devra considerer **toutes** les routes contenues dans **tous** les Adj-RIB-Ins (c'est à dire les routes apprises des peers *internes* et *externes*). En fait, cette phase constitue un processus indépendant qui doit envisager toutes les destinations vers les quelles il existe une route pratiquable dans au moins l'un des Adj-RIB-Ins ; ensuite il doit, pour chaque destination, appliquer successivement les règles suivantes jusqu'à l'obtention d'une seule route, la meilleure pour cette destination :

1. Choisir la route qui a le plus haut degré de préférence parmi toutes les routes disponibles. S'il existe plusieurs routes de même plus haut degré de préférence, il faut passer à l'étape suivante ;
2. Choisir la route qui possède le plus petit AS_PATH, c'est à dire la route avec l'AS_PATH contenant le moins d'ASN. S'il existe plusieurs routes possédant toutes le plus court chemin, il faut passer à l'étape suivante ;
3. Sélectionner parmi les routes restantes originaires du même AS, celle qui possède la plus petite valeur de l'attribut MULTI_EXIT_DISCRIMENATOR.S'il existe encore plusieurs routes originaires d'ASs différents, passer à l'étape suivante ;

4. Choisir la route dont le coût d'accès vers le routeur, désigné par l'attribut NEXT_HOP de la route, est le plus bas. Il faut, donc, choisir la route dont la métrique d'après le protocole de routage interne est la plus basse. S'il existe plusieurs routes dont le coût est identiquement bas, BGP doit passer à la régle suivante ;
5. Choisir la route annoncée par la peer BGP **externe** dont l'identificateur BGP est le plus bas. Dans cette phase, on considère les identificateurs BGP comme de simples valeurs entières, positives, sur 4 octets. S'il n'existe aucune route annoncée par un peer externe, il faut appliquer la dernière règle.
6. Sélectionner la route annoncée par le peer BGP **interne**, dont l'identificateur BGP est le plus petit.

À l'issue de cette procédure, BGP aura sélectionné exactement **une** route pour chaque destination qu'il connait. Pour une destination donnée, le speaker devra installer la route correspondante dans le Loc-RIB, en remplacement de n'importe quelle route vers la même destination qui y était présente. BGP devra déterminer pour la route l'adresse de prochain saut pour atteindre l'adresse IP inclue dans l'attribut NEXT_HOP de la route. Cette adresse est obtenue de l'IGP en lui fournissant l'adresse contenue dans le NEXT_HOP. L'adresse IP ainsi obtenue, est installée dans le Loc-RIB comme adresse de prochain saut immédiat pour atteindre la destination.
Il peut arriver que l'unique route vers une certaine destination soit supprimée de l'Adj-RIB-In d'un peer, du fait d'un rejet explicite de route (*route withdrawal*). Par conséquent, l'application du processus de décision ne produirait aucun chemin vers cette destination. BGP doit dans ce cas de figure supprimer la route du Loc-RIB et émettre, vers tous ses peers, un message de rejet de route concernant la destination.

Phase 3 : Annonce aux peers des routes sélectionnées

La phase 3 du processus de décision constitue une procédure indépendante des deux autres phases, et est invoquée lorsque survient l'un de ces évènements :

- La terminaison de la phase 2 du processus de décision, mais uniquement si au moins l'une des routes du Loc-RIB a changé.
- Le changement d'une des routes du Loc-RIB vers une destination locale à l'AS.
- Le changement d'une route générée localement, par d'autres moyens que BGP.
- L'établissement d'une nouvelle session BGP avec un nouveau peer.

En cet instant, la phase 3 du processus de décision est chargée de générer pour toute nouvelle entrée dans le Loc-RIB, une entrée correspondante dans tous les Adj-RIB-Outs. À cette occasion, le filtre de politique en sortie est appliqué, de la même manière que le filtre en entrée, afin de déterminer exactement les routes à annoncer et avec quels attributs.

La mise en place des routes à annoncer dans les Adj-RIB-Outs peut amener le processus de décision à réduire les informations à annoncer par des techniques telles que l'aggregation de routes. L'aggregation de route consiste en la combinaison des caractéristiques de plusieurs routes de telle sorte qu'une seule route puisse être annoncée. Le but d'une telle opération est de réduire la visibilité des préfixes très spécifiques au niveau global de l'Internet, en permettant aux ASs de composer l'annonce de routes vers plusieurs destinations en une annonce d'une route vers une destination moins spécifique englobant les préfixes les plus longs. Cela devrait diminuer le nombre de réseaux globalement visibles au niveau de l'Internet, et permettre ainsi de réduire la taille des tables de routage et d'améliorer les performences du routage interdomaine.

Quand la phase 3 du processus de décision termine la mise à jour des Adj-RIB-Outs, le speaker BGP doit exécuter le processus d'envoi des messages UPDATE vers ses peers. Ce processus se chargera de la remise des messages au bon interlocuteur, au bon moment.

2.6.2 Processus d'annonce des routes

Ce processus est responsable de l'envoi des messages UPDATE à tous les peers du speaker. Cela concerne autant les peers internes qu'externes d'un speaker. Les procédures pour les deux types de peers étant légèrement différentes, il faut différencier le processus chargé de remettre les messages UPDATE aux peers appartenant au même AS que les speaker, du processus chargé de remettre ces messages aux speakers externes à l'AS du speaker.

Le Processus de Mise-à-Jour interne est concerné par la distribution des informations d'accessibilité aux peers internes du speaker. Ce processus ne redistribue un message reçu par le speaker que si ce message provient d'un peer externe. Dans ce cas, il doit propager l'information vers **tous** ses peers internes dès la réception de l'information. Dans le cas ou le speaker reçoit un rejet de route, il devra supprimer de son Adj-RIB-In la route correspondante, ainsi que de son Loc-RIB, si la route était sélectionnée comme meilleure route. De plus, il devra annoncer une route de remplacement s'il en existe une, ou sinon inclure la destination à rejeter dans un message UPDATE dans la partie réservée aux rejets de routes.

Le Processus de Mise-à-Jour externe s'occupe de la distribution des messages UPDATE aux peers externes du speaker. Comme pour le processus de MAJ interne, le processus de MAJ externe doit informer ses peers de toute nouvelle route, ainsi que de toutes les routes rejetées (explicitement par l'insertion de la destination dans la partie *withdrawn routes* d'un message UPDATE, ou implicitement par l'annonce d'une nouvelle route pour une destination dont on a auparavant annoncé une autre route). La différence réside dans les instants d'envoi des messages UPDATE. En effet, le processus de MAJ interne ne fait subir aucune attente aux messages qu'il émet, alors que le processus de MAJ externe doit temporiser certains messages avant de les émettre.

En fait, BGP utilise deux **timers** de limitation des envois de messages UPDATE, afin de borner le taux d'émission des messages UPDATE dans le but de ne pas créer trop de traffic BGP.

Minimum Route Advertisement Interval représente l'interval de temps minimum qui doit s'écouler entre deux envois successifs de messages UPDATE d'un speaker vers l'autre. La spécification conseille l'utilisation d'un timer MRAI par peer et par destination, mais tolère l'utilisation d'un timer par peer seulement, pour cause de consommation mémoire.
Ce timer sert aussi à minimiser les mises-à-jour successives trop rapides. En effet, si plusieurs routes sont sélectionnées pour la même destination durant le temps d'attente d'un message UPDATE au niveau du timer, alors seul le dernier message en date doit être émis. Cela permet de réduire les fluctuations rapides d'accessibilité de certaines destinations.

Minimum AS Origination Interval représente le temps minimal qui doit s'écouler entre deux annonces de routes concernant des destinations locales à l'AS du speaker.

Il est à noter que l'attente due à ces timers ne doit être infligée qu'aux messages d'annonce de route. Les messages de rejets de routes doivent être émis immédiatement. Cela permet, lorsqu'une destination devient inaccessible juste après qu'elle soit annoncée comme accessible, d'empecher la propagation trop rapide de l'information d'accès de cette destination dans tout le réseau.

Toutes les implémentations doivent appliquer ce type de limitation du rythme d'envoi des messages UPDATE. Le seul moment où une telle limitation n'est pas appliquée est lors de l'établissement d'une nouvelle session BGP. Les nouveaux peers doivent s'échanger la totalité de leurs connaissances respectives. Une limitation telle que celle du MRAI handicaperait beaucoups les nouvelles sessions, en allongeant considérablement le temps consacré à cette tâche.

Il est à noter que la spécification a aussi conseillé de varier la valeur des timers utilisés (*jittered timers*) pour éviter la synchronisation d'un groupe de routeurs. A cet effet, il faut, pour chaque routeur, multiplier la valeur localement configurée du timer (pour le MRAI, la valeur par défaut est de 30 secondes) par une valeur aléatoire comprise dans l'interval $]0.75, 1]$.

2.7 Routage BGP intra-domaine

Quand un speaker BGP reçoit une information de routage et qu'il sélectionne la route transportée par ce message comme étant la meilleur route vers une certaine destination, il doit propager cette information vers tous les peers auquel il est connecté. Ainsi, un speaker recevant une information de routage d'un peer externe devra transmettre cette information vers ses peers internes.

La méthode de détection de cycles utilisée par BGP permet de repérer les cycles dans le graphe des ASs du réseau. Malheureusement, au sein d'un AS, il est impossible de détecter les boucles traversées par les messages BGP entre les peers internes. Aussi, BGP utilise une règle très simple pour remedier à ce problème : tout speaker qui reçoit un message d'un peer interne **ne doit** pas propager ce message vers ses autres peers internes. Bien sur, il peut transmettre cette information vers les peers externes à sa convenance.
À cause de cette règle, et pour conserver la consistance du routage au sein de l'AS, il faut que le speaker établisse une connexion BGP avec **tous** les autres speakers appartenant à l'AS. On obtient finalement, à l'interieur de chaque AS, une session I-BGP entre chaque couple de speaker de l'AS (figure 2.6).
Notez que les speakers internes ne doivent pas être connectés directements par des sous-réseaux, comme c'est le cas pour les peers de deux ASs différents. Au lieu de cela, il suffit que chaque speaker puisse établier une connexion TCP avec les autres, peut importe les liens physiques utilisés par ces connexions.
On obtient, alors, à l'intérieur de chaque AS, une sorte de clique dont les noeuds sont tous les speakers de l'AS, et dont les arcs sont constitués par les sessions I-BGP établies entre les speakers. La figure 2.6 montre comment

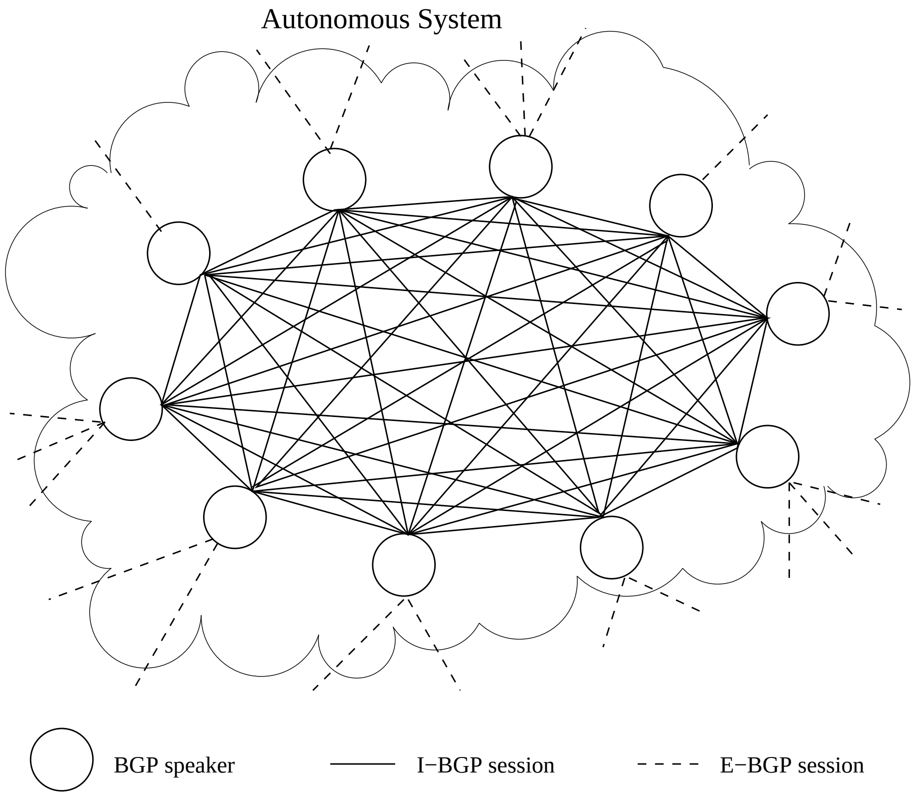

FIG. 2.6 – Interconnexion complète des speakers BGP d'un AS

doivent être connectés les speakers d'un AS pour satisfaire la condition de consistance du routage à l'interieur de cet AS.
Un tel déploiment est facilement réalisable dans les ASs de petites tailles, contenant quelques speakers seulement. Mais il devient problématique de créer de grands ASs, contenant plusieurs dizaines de speakers ou plus, par cette méthode. En effet, il faudrait, alors, établir chez chaque speaker une session pour chacun des speakers de l'AS, en plus des éventuelles sessions établies vers des peers externes à l'AS. Cette solution, en plus de d'exiger une grande capacité mémoire (une table Adj-RIB-In, une table Adj-RIB-Out, les timers ... pour chaque peer du speaker) surcharge fortement le CPU du speaker, qui devra consacrer énormement de son temps aux traitements BGP. Deux solutions ont été proposées pour résoudre ce problème de scalabilité :

La réflexion de route [BCC00] permet à certains speakers BGP de rompre la règle de non-réemission vers des peers internes des messages provenant de peers internes.

Les confederations d'ASs [TMS01] permettent de *fragmenter* les grands

ASs, de manière transparente à l'environnement exterieur de l'AS.

2.7.1 La réflexion de routes dans BGP

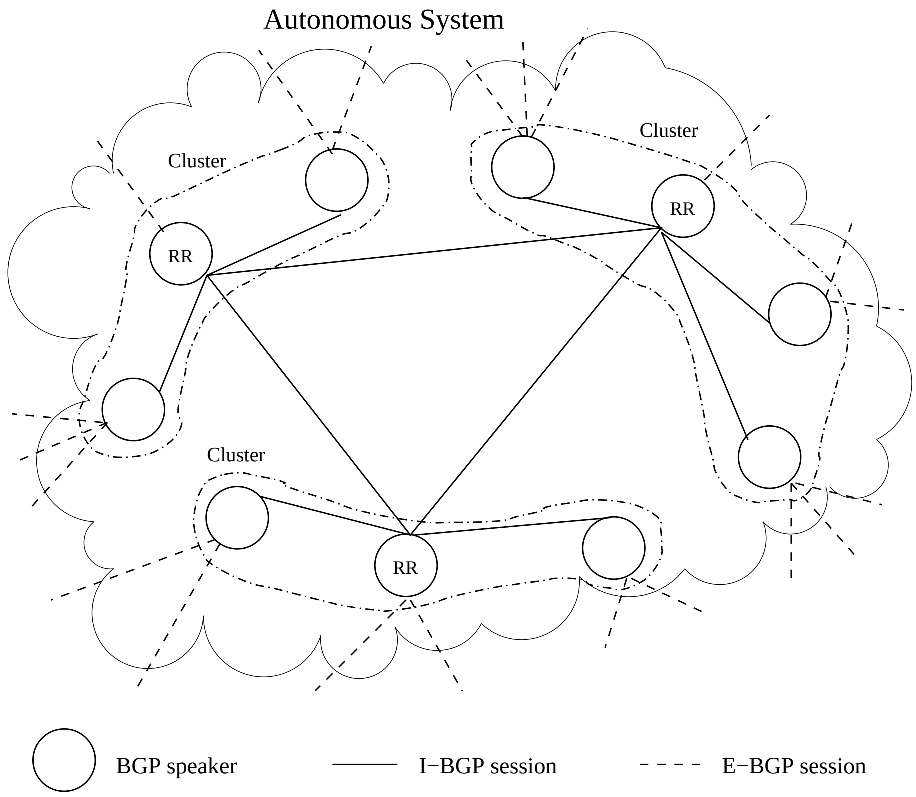

FIG. 2.7 – La réflexion de Routes

Le but de la réflexion de route (*Route Reflection*) est de réduire la connectivité des speakers au sein d'un AS, en relachant la règle selon laquelle un speaker ne doit pas réémettre vers ses peers internes une information qu'il a reçu d'un peer interne. La réflexion de route introduit la notion de **réflecteur de routes** (*Route Reflector*, RR). Un réflecteur de routes est un speaker qui **peut annoncer à ses peers internes les routes apprises de ses peers internes**. Les reflecteurs de routes peuvent, donc, renvoyer à leurs peers internes des messages en provenances de ses autres peers internes. Les speakers qui ne sont pas des réflecteurs de route appliquent les procédures I-BGP, telles qu'elle sont exposées ci-haut.
Comme le montre la figure 2.7, les peers d'un réflecteur de routes sont de deux types :

- Les peers *clients* du RR sont les peers du RR qui appliquent le I-BGP traditionnel (pas de réflexion de routes) et qui appartienne au *cluster* du RR.
- Les peers *non-clients* du RR sont les peers du RR qui n'appartiennent pas au *cluster* du RR. Ces peers peuvent être de simple speakers appliquant l'I-BGP conventionnel, comme ils peuvent constituer les RRs d'autres clusters.

Pour appliquer la réflexion de routes dans l'AS, il faut que :
– Tous les peers non-clients d'un RR soient entièrement connectés entre eux.
– Les clients d'un RR ne doivent pas être connectés avec d'autres RRs. Chaque client ne doit appartenir qu'à un seul *cluster*.
– Chaque speaker client doit transmettre une information de routage vers ses peers internes, uniquement si cette informatin provient d'un peer externe (par E-BGP).
– Chaque RR doit appliquer cet algorithme , pour les routes qu'il a décidé de transmettre vers l'intérieur de l'AS :
  1. Si la route provient d'un peer non-client, il faut la transmettre vers tous les peers clients.
  2. Si la route provient d'un peer client, il faut réflechir la route vers tous les peers non-clients, ainsi que vers tous les peers clients autres que l'origine de l'information.
  3. Si la route est apprise par E-BGP, il faut transmettre cette route vers tous les peers client et vers tous les peers non-clients.

La mise en œuvre de la réflexion de routes exige aussi la mise en place d'un dispositif d'évitement de cycles au sein de l'AS. En effet, par le principe même de la réflexionde routes, une information annoncée dans l'AS par un speaker peut être réfléchie de RR en RR jusqu'à lui revenir. Pour éviter cela, il a été définit deux nouveaux attributs à ajouter aux messages BGP :
- L'attribut ORIGINATOR_ID est un attribut optionnel, non transitif, de *type code* 9, et de 4 octets en longeur. Cet attribut doit être créé par un RR. Il y entrepose l'identité du speaker qui a généré l'information dans cet AS. Puis, les RRs ne doivent jamais, renvoyer une route au speaker désigné dans l'attribut ORIGINATOR_ID de la route.
- L'attribut CLUSTER_LIST est optionnel, non-transitif, de *type code* 10. Il est constitué d'une liste de valeurs CLUSTER_ID représentant le chemin de réflexion traversé par la route dans l'AS. Cette liste est remplie à fur et à mesure par les RRs des différents clusters, chaque fois qu'un RR propage la route d'un peer client vers un peer non-client. Cet attribut permet de détecter les cycles effectués par les routes dans

les ASs.

2.7.2 Les confèderations d'ASs dans BGP

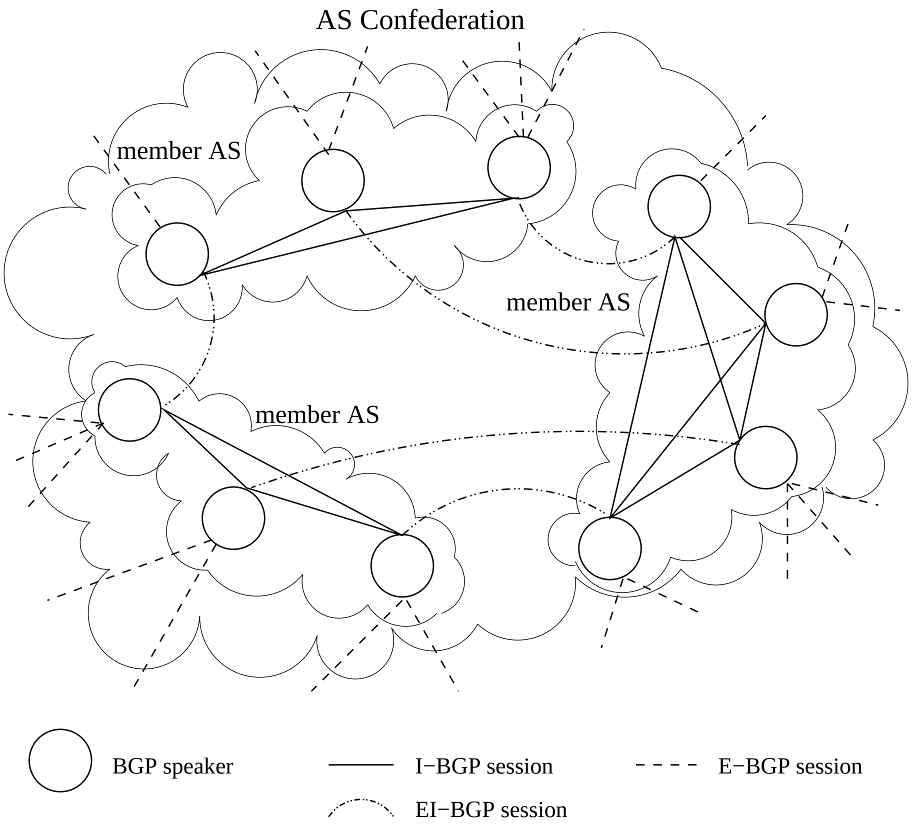

FIG. 2.8 – Les confèderations d'AS

La seconde solution au problème de connectivité complète des speakers au sein d'un même système autonome, passe par l'utilisation de **confédérations de systèmes autonomes**. Une confédération d'ASs peut être définie comme "une collection de systèmes autonomes, annoncée aux ASs extérieurs à la confédération comme un seul AS, c'est à dire en utilisant un seul ASN". La confédération dispose d'un **identificateur de confédération d'ASs**, qui est en fait un ASN identifiant la confédération entière comme une seule entité. Chaque AS de la confédération est un **AS membre**, et dispose d'un **numéro d'AS membre**, qui est un ASN visible uniquement dans la confédération.

La figure 2.8 expose les différentes interconnexions dans l'application des confédérations d'ASs. Au sein d'une confédération, les ASs membres communiquent vers l'exterieur de la confédération grâce à des sessions E-BGP. A l'interieur d'un AS membre, les speakers interagissent normalement via des sessions I-BGP. Cela exige une connectivité complète des speaker **à l'interieur de chaque AS membre**. Enfin, les speakers appartenant à des ASs membres différents communiquent par des sessions E-BGP particulières, notées EI-BGP.
La différence entre E-BGP et EI-BGP réside dans la façon dont s'indentifient les speakers. Pour un peer externes à l'AS d'un speaker, mais interne à la confédération, le speaker s'exprime en tant que représentant de l'AS membre (grâce à son numéro d'AS membre). Alors que pour un peer totalement externe à la confédératrion, le speaker se déclare en utilisant l'identificateur de confédération.
Cette application du principe "diviser pour régner" permet une diminution considérable du nombre de sessions I-BGP. Paradoxalement, cette division peut augmenter la taille des attributs AS_PATH des routes transitants par des confédérations d'ASs. C'est pour cela qu'il a été décidé de ne pas exposer la composition interne des confédérations, en limitant la visibilité des numéros d'AS membres aux frontières de la confédération.

Dans ce but, la spécification a ajouté deux nouveaux types de segments à l'attribut AS_PATH :

- le type de segments AS_CONFED_SEQUENCE contient une liste ordonnée de numéros d'AS membres de la confédération traversée par le message UPDATE
- le type de segments AS_CONFED_SET contient un ensemble désordonné de numéros d'AS membres de la confédération traversée par le message UPDATE.

A partir de là, le comportement des speakers est modifié pour prendre en compte leur appartenance à une confédération :

– Un speaker membre d'une confédération utilisera l'identificateur de confédération d'ASs dans ses transactions avec les speakers étrangers à la confédération. Mais il utilisera le numéro d'AS membre, pour ses échanges avec les speakers appartenant à la confédération.
– Un speaker doit effectuer les tests visant à détecter les cycles dans les segments AS_SEQUENCE et AS_SET ; mais aussi dans les AS_CONFED_SEQUENCE et AS_CONFED_SET.
– Un speaker doit modifier l'attribut AS_PATH des routes, comme suit :

1. si la route est propagée vers un peer appartenant à l'AS du speaker,

aucune modification n'est appliquée.

2. si la route est propagée vers un peer exterieur à l'AS du speaker, mais appartenant à la même confédération, le speaker ajoute son numéro d'AS membre au segment AS_CONFED_SEQUENCE en début de l'attribut AS_PATH, et il en crée un si nécessaire.
3. si la route est propagé vers un peer exterieur à la confédération, le speaker doit supprimer de l'AS_PATH tout segment AS_CONFED_SEQUENCE ou AS_CONFED_SET, puis ajouter son identificateur de confédération d'ASs au segment AS_SEQUENCE en début de l'attribut AS_PATH, et il en crée un si nécessaire.
4. si un speaker veut générer une nouvelle route, il applique les mêmes règles que ci-dessus, à ceci près, que l'attribut AS_PATH ne doit contenir aucun segment AS_SEQUENCE, ni AS_SET pour les peers de la même confédération, et doit contenir un unique AS_SEQUENCE renfermant seulement l'identificateur de confédération du speaker.

Grâce à ce principe de masquage du contenu des confédérations, il devient possible de créer des ASs d'une ampleur considérable, tout en limitant la consommation des ressources dans chaque speaker (Les clique I-BGP à l'interieur de chaque AS membre sont de tailles résonables).

Chapitre 3

Les problèmes du Routage inter-domaine

Ce chapitre a pour but de présenter quelques uns des problèmes du routage inter-domaine. Les origines de certains de ces problèmes ont déja été identifiés, et des recherches sont actuellement menées pour y remedier. Mais il existe malheureusement, des problèmes dont les sources ne sont pas, ou sont mal connues pour le moment. Pour tous ces types de problèmes, les recherches actuelles tentent d'apporter des solutions, soit en proposant des modifications au protocole existant (la version 4 du protocole BGP actuellement déployé), soit en repensant totalement le mode de fonctionnement du protocole de routage inter-domaine.

3.1 L'Instabilité du Routage

Labovitz et ses co-auteurs on définis, dans [LMJ98], l'instabilité du routage comme étant **des changements rapides de l'accessibilité des réseaux et des informations de topologie**. Selon cette définition, toute fluctuation rapide d'accessibilité à un réseau donné constiturait une instabilité de routage. Dans le cas de BGP, cette fluctuation se traduit par des alternances d'annonces d'une route vers des destination, et d'annonces de perte de chemin vers ces destinations.

Un des problèmes majeurs induit par l'instabilité du routage, est la propagation de ces instabilités à de large portions de l'Internet. Ainsi, si la connectivité vers un réseau donné n'arrête pas de fluctuer, le speaker BGP chargé d'annoncer la route vers cette destination ne sessera d'émettre alternativement des messages d'annonce et des messages de rejets de route. Les peers de ce speaker qui reçoivent ces messages ferons de même pour leurs propres peers ; ils enverront alternativement des messages informant de l'apparition d'une route puis de sa disparition. L'instabilité se répand de cette manière à travers tout le réseau, jusqu'à provoquer parfois des effets extrêmes : la perte de la connectivité pour des portions entières de l'Internet. Mais dans la majorité des cas, ces instabilités peuvent générer trois types d'effets **visibles** :

1. l'augmentation des taux de perte des paquets sur le réseau : Cela est principalement dû au fait que certains routeurs ne possèdent plus, à certains moment d'informations pour router les paquets, et les suppriment tout simplement. Cela se rencontre surtout au niveau des routeurs de l'Internet qui ne possèdent pas de routes par défaut (*default-free routeurs*).
2. l'augmentation de la consommation de ressources sur l'infrastructure de l'Internet : D'après le fonctionnement de BGP, un speaker qui perd un chemin vers une destination tente de trouver un chemin alternatif. Contrairement au cas ou un réel chemin alternatif existe, la perte de connectivité vers un réseau pousse les routeurs BGP à explorer de faux chemins de routage, de plus en plus nombreux et de plus en plus longs (en terme de longeur de l'AS_PATH) jusqu'à atteindre une limite dictée par la topologie du réseau (voir la sous-section 3.2 pour plus de détails sur ce phénomène). A cause de cela, l'instabilité du routage provoque de grosses consommations de ressources. Puisqu'elle ne cesse de provoquer ce type de perte de connectivité, elle entraîne très souvent des augmentations de l'utilisation de la mémoire au niveau des routeurs, et augmente la charge des CPU des routeurs par les traitements BGP supplémentaires qu'elle suscite.
3. l'augmentation du temps de convergence du réseau : Dans la plupart des cas, les fluctuations d'accessibilité d'un certain réseau sont limités dans le temps. Mais les effets de ces fluctuations peuvent perdurer sur le réseau, longtemps après que le lien vers le réseau se soit stabilisé.

Plus généralement, l'instabilité du routage provoque un dégradation des performances de bout-en-bout du réseau (perte, augmentations de délais,...) et une dégradation de l'efficacité global de l'Internet (allongement du temps de convergence, surcharge des routeurs par les activités BGP, ...) [LMJ98].

Pour tenter de quantifier l'ampleur de l'instabilité, Labovitz a différencié les messages UPDATE circulant selon qu'ils apportent une information sur un chagement réel de topologie sur l'Internet, ou qu'ils constituent des informations redondantes [LMJ98]. Tous les messages appartenant à la seconde catégorie ont été imputé à **un comportement pathologique de BGP**. Les observations de cette étude ont montré un ordre de grandeur du volume d'échange de messages **plusieurs fois supérieur à celui attendu**. En effet, dans un WAN stable et optimale, les routeurs BGP ne devraient générer que des messages UPDATE , peu fréquents, pour s'informer des changements de politiques et des réseaux nouvellement connectés.A l'inverse, l'étude a montré de très grandes quantités de messages UPDATE, dont 99% ont été identifié comme pathologiques.

En tentant de localiser la source de ces informations redondantes, Labovitz a mis en avant plusieurs explications possibles à ces comportements pathologiques [Li03] :

1. **Les décisions d'implémentations des constructeurs de routeurs :** A cause de la relative liberté laissé par la spécification aux constructeurs sur certains poins d'implémentations, il peut exister des implémentations respectant totalement la spécification de BGP, mais qui exhibent un comportement inattendu. L'exemple le plus frappant, est l'implémentation appelée "*stateless BGP*" [LMJ98]. Afin de réduire la consommation mémoire des routeurs BGP, cette implémentation ne sauvegarde pas les derniers messages émis vers les peers d'un speaker. Cela conduit les routeurs à ré-émettre certains messages plusieurs fois vers les mêmes peers, sans en avoir conscience. Et bien que parfaitement conforme à la spécification de l'IETF, cette implémentation est en partie responsable des énormes volumes d'UPDATEs constatés par Labovitz dans [LMJ98].
2. **Les erreurs de configurations de BGP/IGP et leurs interactions :** La configuration des protocoles de routage n'est pas une tâche triviale. Cela est d'autant plus vrai avec BGP, pour lequel il existe un grand nombre de paramètres et de politiques à configurer. La mise en œuvre de configurations pour l'interaction BGP/IGP comporte une difficulté supplémentaire, à cause des filtres complexes à établir. De plus, ce type d'interactions est seulement visibles des routeurs de bordure internes à l'AS concerné, alors qu'elles peuvent affecter des routeurs à l'exterieur du domaine. Ainsi, une erreur de configuration de ces interactions peut provoquer des instabilités répandues sur tout l'Internet. Par exemple, en 1997, une mauvaise configuration dans un petit ISP (AS7007) a provoqué des cassures pour de larges portions de l'Internet, et ce pour des durées allant jusqu'à 24 heures !
3. **Les bugs Software et effondrement des routeurs :** Chez certains routeurs possédants des architectures de cache des routes, une augmentation du nombre de messages UPDATE et de leurs fréquence d'arrivée provoquera de fréquents défauts de cache. Ce qui accroit la charge du CPU, la latence dûe aux remplissages des caches, et les pertes

de paquets. Au pire, certains routeurs ne pourront supporter cette augmentation de la charge, et subiront un effondrement. Dans d'autres cas, la charge supplémentaire des CPU empechera certains routeurs de maintenir l'échange de messages KEEPALIVE avec leurs peers. Ces peers considéreront alors la session comme perdue, et rejeteront toutes les routes qu'ils ont appris de ces routeurs, ce qui les conduira à choisir d'autres routes qu'ils devront publier, provoquant ainsi une nouvelle augmentation du nombre d'UPDATEs circulant sur le réseau. Depuis la parution des résultats publiés par Labovitz et ses co-auteurs [LMJ98], la plupart des routeurs maintiennent toute leur table de routage en mémoire, limitant par la le comportement pathologique cité ci-haut.

4. **La synchronisation des timers :**La spécification de BGP conseille l'utilisation de timers MRAI dont la valeur est variée par la multiplication de la valeur configurée par une valeur aléatoire comprise dans l'interval $[0.75, 1[$. L'ignorance de cette règle par certaines implémentations peut causer un groupe de routeurs BGP à se synchroniser : tous les routeurs émettent simultanement leurs messages UPDATE, provoquant les uns chez les autres la réception simultanée d'un grand nombre de messages. Cette surcharge subite peut conduire certains des routeurs au crash, ou les empecher de maintenir les sessions BGP avec leurs peers.
5. **la mapping "Métriques IGP-Valeurs MED" :** Dans [LMJ99], Labovitz a découvert que certains ASs attribuaient la valeurs des attributs MED des routes qu'il exportent vers d'autres ASs, en fonction des valeurs des métriques attribuées au chemin du routeur d'entrée de l'AS au routeur de sortie de l'AS, par l'IGP interne à l'AS, comme le montre la figure 3.1 Le problème avec ce type d'attribution des valeurs MED

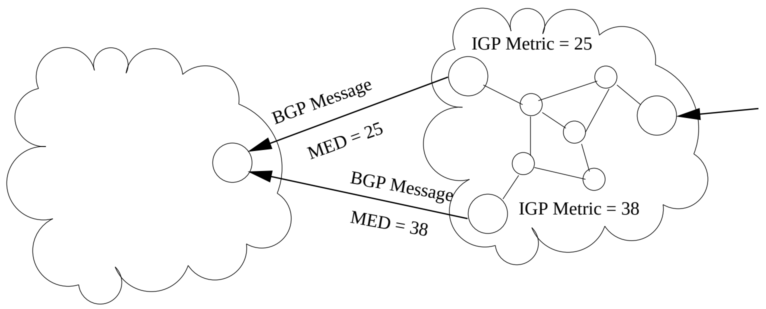

FIG. 3.1 – Mapping des métriques IGP en attributs MED

réside dans la création d'un sensibilité du routage inter-domaine aux évènements internes (normalement cachés) d'un AS. Ainsi, si l'IGP de l'AS décide une modification des valeurs des poids ou réagit à la perte d'un lien par la recherche d'un nouvel état stable, ces modifications deviennent "visibles" à l'extérieur de l'AS. Dans le meilleur des cas, cette modification ne changera rien aux choix des ASs voisins. Mais dans le pire des cas, une altération bénigne du poids d'un arête de la topologie interne à l'AS, peut mener un AS voisin à modifier ses routes. Heureusement, les valeurs des attributs MED ne sont pas transmis au delà de l'AS voisin. Malgré tout, cela peut générer l'émission d'un nombre important de messages UPDATE, qui, s'ils ne provoquent pas de changement importants des données du récepteur, peuvent le mener à émettre lui aussi de nouveaux messages UPDATE, et ainsi de suite.

En 1998, une méthode a été proposé pour limiter la propagation de certaines instabilités [VCG98]. Cette méthode, connue sous le nom de *Route Flap Damping*, a pour but de **supprimer** l'annonce d'une route instable par un routeur qui soit proche de la source, jusqu'à ce que la route devienne stable [Ste99].

3.1.1 *Route Flap Damping*

Le *Route Flap Damping* [VCG98] est une méthode de restriction des annonces de routes, permettant de distinguer les routes stables (celles qu'il est permit d'annoncer aux voisins) des routes instables (celles qu'il vaut mieux ne pas annoncer avant qu'elles ne se stabilisent). Le *Route Flap Damping* n'est pas une extension du protocole BGP, il s'apparenterai plutôt à une politique, un peu particulière. Cette méthode permet à un speaker BGP de prendre en compte **la stabilité passée d'une route** pour décider d'annoncer ou non, la route en question.

A cet effet, Le *Route Flap Damping* associe un **pénalité** à chaque route. Cette pénalité est incrémentée si la route oscille (est annoncée comme disponible par un peer, puis est rejetée), et elle est décrementée dans le temps suivant un certain taux. Puis, si la pénalité d'une certaine route atteint une certaine limite, elle est supprimée, c'est à dire qu'elle ne sera plus candidate au choix des routes et qu'elle ne sera pas annoncée aux voisins. Et la route restera supprimée jusqu'à ce que la pénalité atteigne un certain seuil (la pénalité est décrementée dans le temps, comme le montre la figure 3.2).

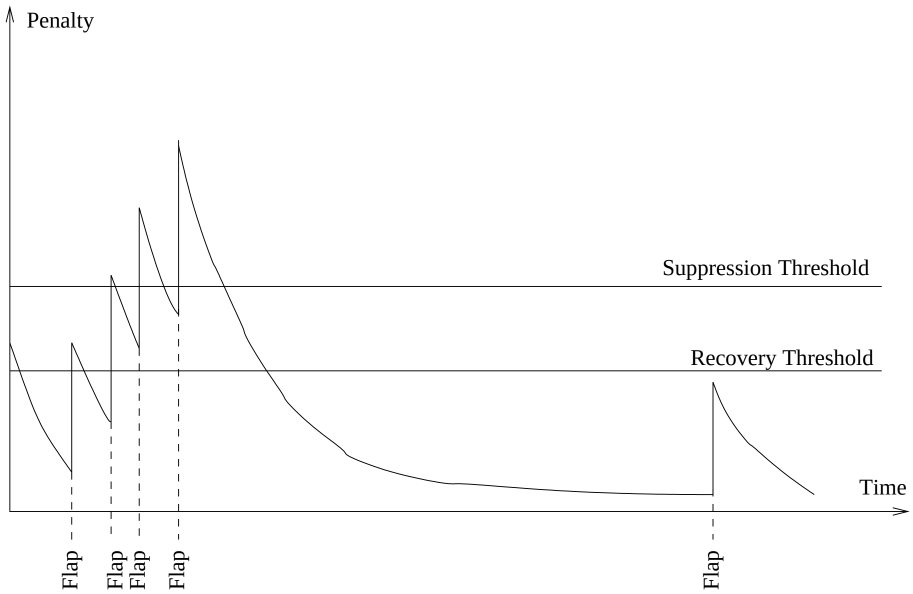

FIG. 3.2 – Graph de progression de la pénalité associée à une route

3.2 La lenteur de la convergence de BGP

La lenteur de la convergence est problème bien connu du routage sur Internet. A l'inverse des réseaux commutés, qui peuvent récuperer d'une erreur en quelques millisecondes, ou des routage IGP, qui prennent jusqu'à quelques dizaines de millisecondes pour converger, le routage sur Internet peut prendre plusieurs minutes pour converger [LABJ00].

En dehors des effets de la lenteur de la convergence, il est intéressant de s'attacher à l'étude de son origine. En effet, durant une longue période, l'origine de cette lenteur a été imputée aux mécanismes de files d'attente chez les routeurs et aux traitements CPU de ces routeurs. Il est, cependant, apparu que cette lenteur est inhérente à l'utilisation d'un protocole à vecteur de chemin, c'est à dire que la lenteur est imputable à la conception même du protocole. En effet, Il a été prouvé que les protocoles à vecteurs de distance (*distance vector protocols*) possèdent pour caractéristique de converger lentement. En étant dérivé des algorithmes à vecteurs de distance, les algorithmes à vecteur de chemins ne peuvent totalement se libérer de certaines limitations, même si la transmission d'informations de route permet d'en éviter d'autres (*loop* de routage, duplication de routes,...) [Li03].

Pour illustrer ce concept, la figure 3.3 montre un exemple complet des échanges de messages entre speakers BGP dans une clique d'ASs. Pour sim-

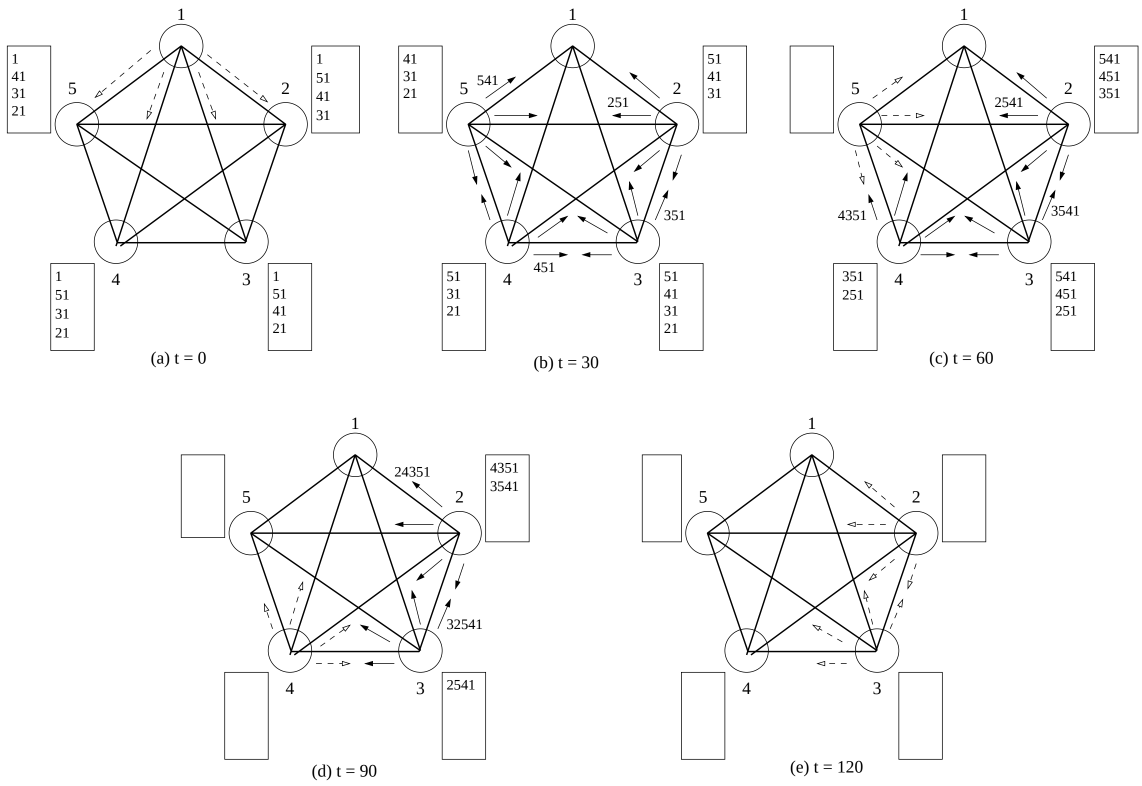

FIG. 3.3 – Comportement de BGP dans le cas d'un rejet de route dans une clique d'ASs

plifier, chaque AS ne contient qu'un seul speaker identifié par un nombre (R1, R2, R3, R4 et R5), qui identifie aussi l'AS. De plus, les timers MRAI possèdent tous la valeur 30 secondes, et sont synchronisés. On supposera que l'AS 1, a auparavant, annoncé une route vers une destination à l'interieur de cet AS. Après la convergence de BGP, les tables de routage se trouvent dans l'état montré par la figure 3.3(a). La table accolés de chaque nœud représente la table de routage BGP de ce nœud. Les routes y sont ordonnées, par ordre de préference, de haut (la plus préférée des routes) en bas (la moins préférée). Par exemple, le nœud 3 préfère accéder à la destination par l'émission de ces paquets directement vers l'AS 1. Son second choix, exprime le fait que les paquets seraient envoyés vers l'AS 5 avant d'être envoyés à l'AS 1.

A l'instant t = 0, l'AS 1 annonce que la destination n'est plus disponible. les restes des figures montrent l'évolution dans le temps des tables de routage et les messages échangés. Le rejet de route de l'AS 1 provoque le rejet de toutes les routes annoncées au préalable par l'AS 1 (toutes les entrées "1" des tables de routage). Chaque AS choisit alors sa second route préférée, lui accole son propre ASN et l'annonce à tous ses voisins (figure 3.3(b)). Mais les messages sont bloqués par le timer MRAI, et ils ne peuvent être émis que 30 secondes plus tard (valeur par défaut du MRAI). Quand l'AS 5 reçoit de chacun de ses peer une nouvelle route, il écrase l'ancienne. Mais chacune de ces nouvelles routes contient l'ASN 5, l'AS 5 les rejète,donc, et décide d'envoyer un message de rejet de route vers tous ses peers (figure 3.3(c)). Pendant ce temps, les ASs 2, 3 et 4 reçoivent les uns des autres des routes qui écrasent les anciennes. Ils se retrouvent avec des routes de longeur 3 ; qu'il émettent dès que leurs timer MRAI le permet (figure 3.3(c)). Cette fois-ci, c'est au tour de l'AS 4 de ne plus posséder de routes, il envoit, donc, un message de rejet de route. Entre temps, les ASs 2 et 3 ont appris des chemins de longeurs 4 et tentent d'en informer leurs voisins (après expiration des MRAI)(figure 3.3(d)). Lorsque les ASs 2 et 3 reçoivent le message que chacun à émis vers l'autre, ils perdent tout chemin vers la destination, et émettent donc des messages de rejet de route (figure 3.3(e)).

Cet exemple illustre parfaitement la durée de convergence d'un système BGP. Il est à noter qu'aujourd'hui, le cœur d'Internet est composé d'un pseudo-clique comportant plus de 15 ASs, et que le comportement montré dans la figure exploite la synchronisation des timers. Dans un cas réel, le temps de covergence serait encore plus long.

3.3 Les oscillations persistentes des routes

Lors de la création de BGP, il a été décidé de ne pas contraindre le protocole à effectuer des choix de route de plus court chemin. Au lieu de cela, BGP permet aux speakers d'effectuer des choix dictés par des politiques de routage. Le problème est que le choix de ces politiques, au niveau des différents ASs, est effectué de manière totalement indépendante, ce qui pourrait mener à des conflits entre ces politiques[Li03].

Govidan et *al.* ont montré, dans [VGE00], que certains choix de politiques de routage pouvaient entrener des **oscillations persistantes de routes**. Cela correspond aux cas ou un groupe d'AS ne cesserait de faire osciller le choix de leurs routes, à cause des politiques qu'ils ont adopté.

La figure 3.4, tiré de [VGE00], montre un exemple de choix de politiques qui génerent des oscillations qui peuvent perdurer extrêmement longtemps. Dans cet exemple, l'AS central annonce une destination x, vers tous les autres ASs. Ces derniers (D0, D1 et D2) possèdent donc une route directe vers x. Mais, les routeurs r0, r1 et r2, appartenant aux ASs D0, D1 et D2 repectivement, appliquent des politiques telles qu'il préfèrent la route annoncée par

	r0	r1	r2
D0	–	r0	r2
D1	r0	–	r1
D2	r2	r1	–

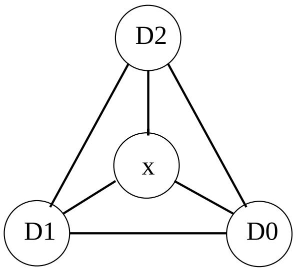

FIG. 3.4 – Exemple de politiques générant des oscillations de routes

le routeur qui les précèdent au lieu de leurs propres routes (D0 préfère r2 au lieu de sa propre route, D1 préfère r0 au lieu de sa route et D2 préfère r1 à sa route directe). Sous ces conditions, chaque routeur oscillera entre sa route directe et la route annoncée pas le routeur qui le précède, comme le montre la figure 3.5. L'étape (h) de cette figure montre le même état que l'étape (b), ce qui pousse la séquence à continuer, et empêche le système de converger.

Bien qu'il y'ait peu de chance que de telles configurations existent sur l'Internet, la possibilité de rencontrer des politiques conflictuelles ne sont pas nulles. Ce qui rend les oscillations de routes susceptibles d'exister, n'importe ou dans le réseau, et à n'importe quel moment.

3.4 La lenteur du temps de convergence induite par le *Route Flap Damping*

L'idée principale du *Route Flap Damping* est de pouvoir supprimer l'annonce des routes qui oscillent plus qu'un certain seuil. Mais il est apparu que l'exploration de routes alternatives après le rejet d'une route, peut allonger le temps de convergence de routes relativement stables [MGeRK02]. Ainsi, Varghese a montré, dans [MGeRK02], que cette méthode pouvait augmenter les temps de convergence de route relativement stable de plus d'une heure par rapport à leurs temps de convergence sans application du*Route Flap Damping*.

La raison invoquée pour expliquer ce comportement provient du mécanisme d'exploration de routes alternatives lors d'une suppression de route. Ainsi, les routeurs qui apprennent que leur route principale n'est plus valide, tentent de trouver une autre route vers la destination, au lieu de conclure immédiatement à l'innaccessibilité totale à la destination. Ce processus qui permet à BGP de trouver rapidement de nouvelles routes lorsque ces routes

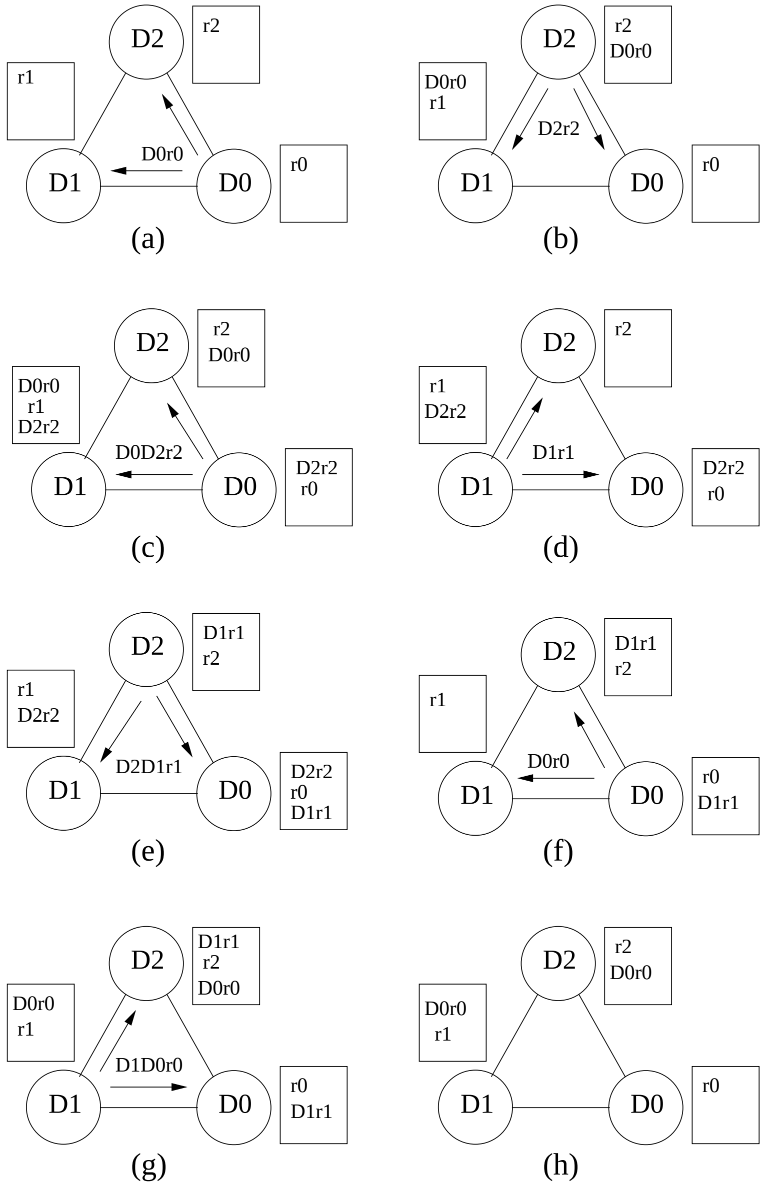

FIG. 3.5 – Étapes de l'oscillation de routes

principales sont détruites, provoque une augmentation du nombre de messages UPDATE au delà du seuil de suppression de l'algorithme de *Route Flap Damping*. Ce qui empeche (ou retarde considérablement) la future diffusion d'une réelle annonce de nouvelle route vers la destination.

La figure 3.6, tirée de [MGeRK02], expose un exemple de situation, ou le *Route Flap Damping* provoque un allongement, inutile, du temps de conver-

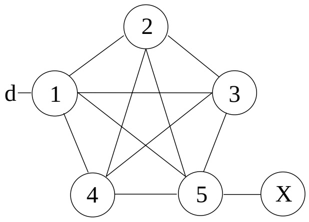

FIG. 3.6 – Exemple de réseau affecté par la lenteur de convergence générée par le *Route Flap DAmping*

gence. La destination, *d*, est annoncée par l'AS 1 à tous ces peers. A la fin de la période de convergence, on obtient les tables de convergence exposées dans la première ligne de la figure 3.7.

La figure 3.7 montre dans la troisième colonne, les états des tables de routage des nœuds 2, 3, 4 et 5. Chaque nœud est représenté avec une liste de routes : une route de chacun de ses peers. Dans chaque liste, l'èlément précédé d'une étoile est la meilleur route pour ce nœud. La quatrième colonne montre les messages traités à chaque étape, et l'impact de ces messages sur les tables de routage.

Le point le plus important de cet exemple, est que le nœud X reçoit une multitude de messages en provenance de l'AS 5, à propos de la destination. X considère ces messages comme des instabilités. Ainsi, dans l'exemple, si le nœud AS1 oscille seulement deux fois, par le rejet de la route vers d, puis l'annonce de sa disponibilité, alors le nœud X supprimera la route de ses tables. En considérant uniquement les messages émis par 5 à destination de X, on notera la présence de trois messages UPDATE et un rejet de route(*route withdraw*). Ces quatres messages sont considérés par X comme des oscillations de la route vers d. Selon [MGeRK02], et en utilisant les paramètres de routage de Cisco, la pénalité associée à d chez X a atteint 2500. Selon ces mêmes paramètres, le seuil de suppression est de 2000. La route est, donc, supprimée et elle ne peut être réutilisée qu'environ 15 minutes plus tard. Et parceque X ne peut atteindre d qu'à travers l'AS 5, la connectivité de X est alterée à cause du *Route Flap Damping* [Li03].

Step	Time	Routing Tables	Messages Processed
0	N/A	Steady State 2(*1,31,41,51) 3(21,*1,41,51) 4(21,31,*1,51) 5(21,31,41,*1)	
1	N/A	1 withdraw the route 2(−,*31,41,51) 3(*21,−,41,51) 4(*21,31,−,51) 5(*21,31,41,−)	1−>{2,3,4,5}W
2	N/A	Annoucement from 2 2(−,31,41,51) 3(−,−,*41,51) 4(231,*31,−,51) 5(231,*31,41,−)	2−>{1,3,4,5}[231]
3	N/A	Annoucement from 3 2(−,−,*41,51) 3(−,−,*41,51) 4(231,321,−,−51) 5(231,321,*41,−)	3−>{1,2,4,5}[321]
4	N/A	Annoucement from 4 2(−,−,−,*51) 3(−,−,421,*51) 4(231,321,−−,*51) 5(*231,321,421,−)	4−>{1,2,3,5}[421]
5	30	Annoucement from 5 2(−,−,−,−) 3(−,−,*421,521) 4(*231,321,−−,521) 5(*231,321,421,−)	5−>{1,2,3,4,X}[521]
6	N/A	Withdrawal from 2 2(−,−,−,−) 3(−,−,*421,521) 4(−,*321,−,521) 5(−,*321,421,−)	2−>{1,3,4,5}W
7	N/A	Annoucement from 3 2(−,−,−,−) 3(−,−,*421,521) 4(−,−,−,*521) 5(−,*3421,421,−)	3−>{1,2,4,5}[3421]
8	N/A	Annoucement from 4 2(−,−,−,−) 3(−,−,−,*521) 4(−,−,−,*521) 5(−,*3421,4231,−)	4−>{1,2,3,5}[4231]
9	60	Annoucement from 5 2(−,−,−,−) 3(−,−,−,−) 4(−,−,−,*5231) 5(−,*3421,4231,−)	5−>{1,2,3,4,X}[5231]
10	N/A	Withdrawal from 3 2(−,−,−,−) 3(−,−,−,−) 4(−,−,−,*5231) 5(−,−,*4231,−)	3−>{1,2,4,5}W
11	N/A	Withdrawal from 4 2(−,−,−,−) 3(−,−,−,−) 4(−,−,−,*5231) 5(−,−,−,−)	4−>{1,2,3,5}[45231]
12	N/A	Withdrawal from 5 2(−,−,−,−) 3(−,−,−,−) 4(−,−,−,−) 5(−,−,−,−)	5−>{1,2,3,4,X}[53421]

Fig. 3.7 – Evolution des tables de routage d'une clique de 5 éléments

Chapitre 4

Les simulateurs de BGP

Ce chapitre présente quelqu'uns des simulateurs existants du routage inter-domaine basé sur BGP. Ne pouvant exposer la totalité des simulateurs existants du protocole BGP, nous nous sommes concentrés sur les simulateurs ayant pour vocation l'étude des instabilités du routage inter-domaine [NCM02], ses causes [Pre03][LYPN02]ou ses effets sur le traffic Internet [Quo04][FWR04]. Plusieurs de ces simulateurs ont pour but de montrer l'efficacité, ou plus souvent, les effets de certaines méthodes d'amélioration de BGP sur le routage inter-domaine [Pre03] [HK03], ainsi que l'impact de certaines décisions d'implémentation sur l'instabilité du routage inter-domaine [Pre03][FBT04][HK03].

4.1 Le Simulateur SSF.OS.BGP4

Exposé dans [Pre03], ce simulateur a été conçu dans le but d'observer en détail, et de tenter d'expliquer, le comportement du protocole BGP, dans différentes topologies de réseaux, et face à différents évènements. L'objectif étant de pouvoir observer de très près le phénomène de convergence après qu'un routeur du réseau annonce un changement dans ses informations de routage ou après que survienne un changement physique dans le réseau (connexion d'un nouveau routeur, disponibilité d'une nouvelle destination, *link failure*, *crash* de routeur, ...). En effet, l'étude est parti de la constation que BGP convergeait beaucoups plus lentement que ce qui était attendu. Cette lenteur ayant un impact direct sur les performances de bout-en-bout du traffic circulant sur Internet, il était intéressant de pouvoir anaylser la conduite de BGP pour pouvoir remédier aux problèmes que pose le routage inter-domaine.

Le dessein de l'étude était aussi de répondre à certaines questions spécifiques, apparues durant la tentative des auteurs de répertorier les éventuelles sources des problèmes de BGP :

Question 1 : Est-il bénefique pour BGP d'appliquer une limitations du taux d'envoi des informations d'inaccessibilité (*withdrawals*) ? et si oui, quand ? Cette question est apparu car un grand nombre d'implémentations paraissent appliquer une telle limitation alors qu'elle était fortement déconseillée par la spécification [RL95].

Question 2 : Est-il bénefique pour BGP d'appliquer une détection de cycles plus aggressive ? En effet, BGP détecte les boucles d'annonces de routes en comparant **à la réception** le contenu de l'AS_PATH de la route reçue à l'ASN du récepteur. Et il serait peut-être intéressant de faire une comparaison **avant** d'émettre les messages (comparer l'AS_PATH du message à émettre avec l'ASN du destinataire).

Question 3 : Quelle est la valeur optimale du timer de limitation du rythme d'envoi des messages UPDATE (*MinRouteAnnounceInterval* ou **MRAI**) ? En effet, la valeur par défaut proposée par la spécification (30 secondes, la plus utilisée) n'a absolument pas été justifiée, et l'impact d'une valeur différente n'est pas connus.

Question 4 : Est-ce que le fait de randomiser la valeur du timer du rythme d'envoi des UPDATE augmente les performances ? Les effets de la sychronisation d'un groupe de routeurs sont peu connus, et l'étude de la variation des timers permetterait affirmer ou d'infirmer l'utilité de cette méthode.

Question 5 : Est-ce que la limitation du rythme d'envoi par peer et par destination améliore les performences par rapport à la limitation par peer seulement ? Les contraintes d'utilisation de mémoire dans les routeurs imposent de limiter le nombre de timers. Mais l'effet de l'utilisation de timers pour chaque destination et pour chaque peers n'est pas connu.

Question 6 : A quel point est ce que la charge du réseau influe sur la convergence de BGP ? Cette question est apparue suite aux études montrant une augmentation du nombre d'UPDATEs sur le réseau durant les périodes de fortes activités du traffic [WZPa].

Question 7 : Comment est affectée la convergence de BGP par les formes de topologie du réseau ? Internet peut être vu comme un assemblage de réseaux de formes diverses. Quel est, donc, le comportement de BGP dans chaque forme de topologie ?

4.1.1 Modèle de simulation

Le modèle simulé de BGP à été implementé sur le simulateur SSFNet. SSFNet est un simulateur basé sur Java, avec lequel il est possible de modéliser de grands réseaux assez complexes. Le paquetage comprend le noyau du simulateur, un ensemble de composants réseau réutilisables, et *open source*, ainsi qu'un langage de configuration (**DML** : Domain Modeling Language). DML est utilisé pour construire des modèles en combinant les composants réseau. Le modèle de BGP implementé sur SFFNet, dans le paquetage SSF.OS.BGP4, est entierement conforme à la spécification du protocole BGP 4 [RL95]. Il inclus aussi quelques extensions spécifiques à certaines implémentations commerciales (*Withdrawal Rate Limiting*, timers MRAI continues) et des détails comportementaux introuvables sur d'autres implémentations (Sender Side Loop Detection, timers par peer par destination, *random last-resort tie-breaking*).

Dans ce qui suit, on détaillera les éléments du modèles de simulation de BGP, les éléments d'implémentation spécifiques à certaines distribution commerciales du protocole et la manière dont ils ont été integrés au modèle ; ainsi que les nouvelles modifications proposées par les concepteurs du simulateur :

Délais de traitements SSF.OS.BGP4 comprend son propre modèle pour imposer le délais normalement généré par différentes utilisation du CPU. Il permet en particulier d'imposer des valeurs aux délais induits par le traitements des messages entrants, et aux délais des messages sortants. En fait, le délais imposé au traitement de chaque message BGP est une modélisation de la charge du routeur. Attendu qu'il n'est pas nécessaire de définir exactement la charge du routeur mais plutôt ses effets sur un speaker BGP, tout message traité dans le modèle se voit retardé durant un interval de temps, représentant le retard que subit le traitement d'un message BGP dans un routeur du fait des autres tâches de ce routeur (convoyage du traffic utilisateur, traitements IGP, autres traitements BGP,...). C'est, donc, un délais **induit par la charge du routeur** (*workload induced delay*).

Variation des valeurs des timers La spécification du protocole BGP a imposé de randomiser certaines timers (en particulier *MinRouteAdvertisementInterval*), c'est à dire varier les valeurs qu'ils prennent en multipliant les valeurs de configuration par un nombre aléatoire (généralement choisit dans l'interval $[0.75, 1.00[$ de manière uniforme). Comme toute implémentation conforme à la spécification, SSF.OS.BGP4 applique cette variation aléatoire, mais permet aussi de la désactiver.

Limitation du rythme d'envoi des messages UPDATE Le MRAI (*Minimum Route Advertisement Interval*) spécifie l'interval de temps minimum qui doit s'écouler entre l'envoi de deux annonces successives à propos de la même destination vers le même peer. Il n'est malheureusement pas possible de disposer d'un timer par peer et par destination dans chaque speaker BGP (le nombre de destinations dans l'Internet actuel dépassant les 100000). A cause de ce type de problèmes, la spécification a autorisé l'utilisation d'un timer par peer, et ce pour toutes les destinations. Même si toutes les implémentations actuelles de BGP utilisent cette méthode, personne ne connais les différences de l'utilisation d'un timer par destination pour chaque peer par rapport à l'utilisation d'un timer par peer seulement. Dans le simulateur, les deux approches ont été implémentées, afin de pouvoir comparer.

Politiques de routage La spécification de BGP n'a imposé aucune métrique d'évaluation des chemins, mais a permit l'utilisation de politiques de routage. Dans les implémentations commerciales, ces politiques fonctionnent majoritairement comme des filtres de route. Sous sa forme la plus élaborée, le filtrage de route est permit à deux endroits : le filtre *ingress* est appliqué aux routes apprises par les peers d'un speaker, les filtre *egress* permettent de filtrer les routes à annoncer vers less peers d'un speaker. SSFOS.BGP4 implémente un ensemble de règles de filtrage, basées sur des expressions régulières correspondant aux attributs de chemin BGP. Par défaut (en l'absence de politique), BGP adopte le routage par plus court chemin.

Sender Side Loop Detection Dans la norme BGP, il est spécifié qu'un speaker BGP recevant une annonce de route, si il choisit cette nouvelle route, doit transmettre cette information à *tous* ces peers. Les peers devront alors tester la présence de leur numéro d'AS dans le chemin, afin d'éviter la présence de cycles dans le routage. Dans des études récentes, il a été suggéré d'utiliser des méthodes de détection de cycles plus aggressive : la méthode *split horizon* propose qu'un speaker ne renvoit pas l'information de routage au peer duquel il l'a apprise ; la méthode *sender side loop detection* suggère que chaque speaker vérifie, avant d'envoyer le message UPDATE, la présence ou l'absence de l'ASN du peer vers lequel il envoit l'annonce, dans le chemin de la route. Ces deux méthodes ont pour but de réduire le nombre de messages échangés, dans l'espoir de diminuer le temps de convergence.

Withdrawal rate limiting Alors qu'il est nécessaire d'appliquer une limitation au rythme d'envoi des messages UPDATE d'annonces de route, la spécification a indiqué qu'il ne fallait pas appliquer une telle mesure sur les messages de rejet de route (*withdrawals*). Cependant, nombre d'implémentations utilisées sur le réseau appliquent quand même aux messages de rejets les mêmes mesures de limitation que les messages d'annonces de route. Dans le simulateur, les deux alternatives ont été implementées, autorisant ainsi l'étude de l'impact de ce type de restrictions sur la convergence.

Timers MRAI continus À chaque envoi de message d'annonce de route vers un peer, un timer MRAI est positionné pour ce peer. Normalement, et suivant la spécification de BGP, la fin de la période d'attente du timer déclenche l'envoi du message en attente, si il y'en a un, puis le repositionnement du timer ; par contre, si il n'y avait pas de message en attente d'envoi, le protocole ne doit rien faire. Dans la plupart des implémentations actuelles de BGP, tout déclenchement du timer MRAI provoque son repositionnement, même en l'absence de messages en attente. Cette élément particulier de l'implémentation indique l'utilisation de *timers MRAI continus*. Dans le simulateur, cette option peut être activée ou désactivée.

Last resort Tie-breaking Dans le processus de décision de BGP, si plusieurs routes différentes, vers la même destination, ont le même degré de préférence, et que tous les critères de distinctions en désigne plus d'une, il faut choisir la route annoncée par le speaker ayant le plus petit identificateur BGP, comme étant la meilleure route. Cette méthode peut générer de grosses quantités de travail pour certaines parties du réseau, alors que d'autres sont moins occupées. Dans le simulateur, en plus de cette méthode imposée par la norme, deux autres méthodes ont été implementées : la première permet de choisir comme meilleur chemin, le premier appris (à condition que les routes apprises par la suite ne puissent être préférée par d'autres critères) ; la seconde fournit à la route nouvellement apprise 50% de chances d'être choisie en lieu et place de la route actuellement installée.

Evitement de duplication de messages Un des comportements pathologique observés de BGP, est l'envoi d'informations de routage successives identiques. L'évitement de tels comportements impliquerait la sauvegarde du dernier message envoyé vers chaque peer pour chaque destination, mais nécessiterait de grosses ressources mémoire. Dans SFF.OS.BGP4, l'option de sauvegarde des derniers messages est implémentée mais peut être désactivée.

Monitoring SSFNet dispose d'un système de monitoring intégré afin de faciliter l'observation des simulations. Le simulateur de BGP utilise cette structure pour fournir les informations durant les simulations sous forme de données textuelles ou binaires. De nombreux évènements survenant durant les experiences peuvent être visualisés, tels que les évènements d'établissement de connexion, de transfers d'informations, d'émissions et de réception de données, d'expiration des timers,...

4.1.2 Validation du modèle de BGP

Il n'existe, pour l'heure, aucun test de validation des implémentations BGP. A cause de cela, et afin de valider l'implémentation du protocole sur le simulateur SSFNet, un ensemble de tests a été élaboré afin de s'assurer de la conformité de `SSF.OS.BGP4` à la spécification du protocole. Les tests comprennent de nombreux scénarios, visant à tester le comportement de l'implémentation face à aux évenements courants dans les réseaux simples. Cela couvre , entre autres, la maintenance des sessions, les annonces de routes ainsi que les rejets de routes (*withdrawals*), la réflexion de route et les commuications BGP internes aux ASs. Des configurations plus complexes sont utilisées pour tester le comportement géneral du modèle en terme de convoyage de données de bout-en-bout.

4.1.3 Experimentations réalisées avec le simulateur

Description des éléments de la simulation

Durant les expériences sur la convergence de BGP et afin de pouvoir observer les comportements intéressants de la convergence, plusieurs simplifications ont été réalisées. Ainsi, chaque AS ne comprend qu'un seul speaker BGP, qui est en faite le seul routeur de cet AS. Une topologie de n AS contient, alors, n routeurs R_1, R_2, ..., R_n, interconnectés, chacun appartenant à un AS différent. Les topologies ne contiennent qu'une seule destination accessible, d, et elle est toujours annoncée par l'AS numéro 1, c'est à dire par le routeur R_1. La simplification de la représentation des AS a pour but d'ignorer les interactions entre BGP et les protocoles intra-domaine, ainsi que les effets du IBGP.

Topologies des simulations

Le but de le recherche était d'obtenir des résultats applicables à l'Internet. Dans cet optique, il a été décidé de simuler des topologies simplifiées correspondant à des sous-graphes du graphe de l'Internet. Ces sous-graphes sont suffisament simples pour qu'on puisse expliquer les résultats obtenus (au moins partiellement), tout en correpondant à des parties de réseau actuel. Les experiences sont catégorisés en familles, correspondant aux formes des topologies. La taille d'une topologie, appartenant à l'une de ces familles correspond au nombre de systèmes autonomes présents dans cette topologie.

La première famille d'experiences comprend les topologies en **ligne**. Une topologie en ligne, de n AS, comprend les routeurs de R_1, à R_n, avec des liens entre les routeurs R_i et $R_i + 1$ pour i = 1,2,...,n-1.

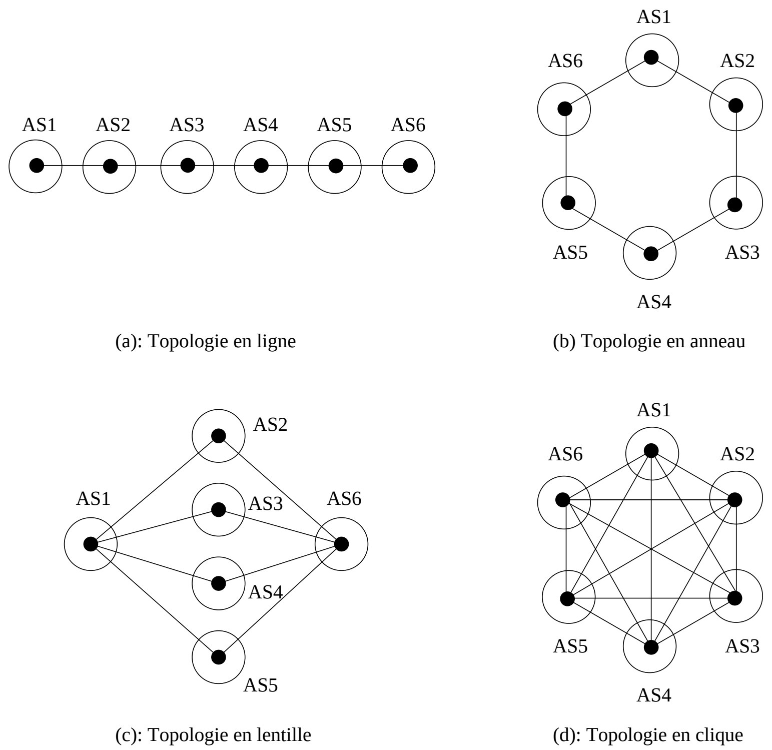

FIG. 4.1 – Familles de topologies de réseaux pour les expériences sur SSFNet

La seconde famille de topologies concerne les topologies en **anneau**, et correspondent aux cycles dans le réseau Internet. Un anneau de taille n contient les routeurs R_1, à R_n, avec des liens entre les routeurs R_i et $R_i + 1$ pour $i =$ 1,2,...,n-1 ainsi qu'un lien entre R_1 et R_n.
La troisième famille de topologies englobe les topologies en **lentille** (*focus topologie*). Ces topologies sont retrouvent sur Internet du fait du *multihoming* des réseaux clients (connexion d'un réseau client vers plusieurs providers). Une topologie en lentille de taille n, interconnecte les deux routeurs R_1 et R_n à travers *n-2* AS intermédiaires R_2, R_3, ..., $R_n - 1$. Chaque routeurs intermédiaire R_i (i=2, 3,..., n-1), n'est connecté qu'à R_1 et R_n.
La dernière famille de topologies est plus representative du réseau cœur de l'Internet, qui est fortement connecté. Cette famille contient les topologies en **clique** de diverses tailles. Une clique de taille n est obtenue en créant, pour chaque routeur R_i un lien vers tous les autres routeurs R_j avec $j = 1, 2, .., n$ et $j \neq i$.

Méthodologie et points d'observation

Chaque test effectué sur le simulateur comprend deux phase. La première phase débute lorsque le routeur d'origine, R_1, annonce la destination d, et se termine lorsque plus aucun routeur n'émet de message UPDATE. Cette phase, nommée phase *up*, a pour but de montrer les réactions de BGP face à l'annonce d'une nouvelle route. La seconde phase, nommée phase *down*, commence après la fin de la phase *up*, et est déclenchée par l'émission d'un message de rejet de route à propos de d, par le routeur R_1. De la même manière que pour la phase *up*, la phase *down* se termine au traitement du dernier message UPDATE du réseau. Dans cette phase, on veut étudier le comportement de BGP face à une perte de connectivité vers une destination.
Avant toute experience, le réseau dispose d'une durée de temps pour établir les connexions physiques, ainsi que les sessions BGP entre chaque deux speakers interconnectés. La phase *up* ne débute que lorsque tout le réseau est stable après la période détablissement de connexions.
Aucune politique de routage complexe n'est utilisée durant les tests. En effet, la plupart des politiques existantes ne sont que des politiques restrictives, implementées sous forme de filtres empechant la propagation de routes depuis, ou vers, certains AS. Au lieu de mettre en place une telle politique entre AS, il est plus simple de ne pas établire de liens entre deux routeurs appliquant cette politique. La topologie physique des simulations (topologie établie par des liens) correspondra, alors, à la topologie logique de certains réseaux (topologie sous contraintes de politiques). Par exemple, dans les topologies en lentille réelles (*focus topology*), il est fréquent que les routeurs intermédiaires soient interconnectés physiquement, mais les politiques de routage invalident ces liens. Dans les experiences du simulateur, les *effets* de ces politiques sont directement obtenues en ne créant pas de liens entre les routeurs intermédiaires.
Les experiences ont été regroupées en famille, dont le dénominateur commun est la forme de la topologie du réseau utilisé. Pour chaque famille F, il existe une fonction I_F qui construit le modèle SSFNet d'une instance de cette famille. Cette fonction prend comme paramètre le vecteur $\vec{p}$, qui contient, entre autres, la taille de la topologie, la valeur du MRAI utilisé, des booléens indiquant l'utilisation de diverses options (SSLD, variation du MRAI, WRL, MRAI continus), la méthode de résolution d'égalités utilisées (*Last resort Tie breaking*), et la graine du génerateur de nombre aléatoires. Étant donné les paramètres $\vec{p}$, on execute le modèle SSFNet $I_F(\vec{p})$, et on enregistre pour chaque experience, les données suivantes :

1. $U_A(\uparrow, \vec{p})$: represente le nombre messages d'annonce de route échangés durant la phase *up*, pour la configuration $\vec{p}$.

2. $U_W(\uparrow, \vec{p})$: represente le nombre messages de rejet de routes échangés durant la phase *down*, pour la configuration $\vec{p}$.
3. $U_A(\downarrow, \vec{p})$: represente le nombre messages d'annonce de route échangés durant la phase *up*, pour la configuration $\vec{p}$.
4. $U_W(\downarrow, \vec{p})$: represente le nombre messages de rejet de route échangés durant la phase *down*, pour la configuration $\vec{p}$.
5. $U(\uparrow, \vec{p})$: represente le nombre globale de messages UPDATE échangés durant la phase *up*, pour la configuration $\vec{p}$.
   $U(\uparrow, \vec{p}) = U_A(\uparrow, \vec{p}) + U_W(\uparrow, \vec{p})$.
6. $U(\downarrow, \vec{p})$: represente le nombre globale de messages UPDATE échangés durant la phase *down*, pour la configuration $\vec{p}$.
   $U(\downarrow, \vec{p}) = U_A(\downarrow, \vec{p}) + U_W(\downarrow, \vec{p})$.
7. $T(\uparrow, \vec{p})$: represente le temps de convergence durant la phase *up*, pour la configuration $\vec{p}$.
8. $T(\downarrow, \vec{p})$: represente le temps de convergence durant la phase *down*, pour la configuration $\vec{p}$.

Pour chaque famille d'experience, et pour chaque phase de chaque experience, l'auteur a tenté d'établir une formulation théorique des grandeurs ci-haut, en fonction des paramètres du vecteur $\vec{p}$. Quand cela n'était pas possible, il tentait de borner ces grandeurs entre des grandeurs fonctions de $\vec{p}$. La validité de ces formulations était ensuite prouvée en comparant les courbes des formules théoriques aux courbes obtenues par simulation.

4.2 Le Simulateur C-BGP

C-BGP [Quo04] [Quo03] est un simulateur efficace du processus de décision de BGP. Il peut être utilisé pour l'experimentation de modifications sur le processus de décision,ainsi que de nouveaux attributs BGP. Il peut être utilisé sur des topologies de même ordre de grandeur que l'Internet.

C-BGP est entièrement écris en langage C, et possède les caractéristiques suivantes :

- C-BGP ne fonctionne pas par la transmission de messages BGP sur des connexions TCP simulées. Au lieu de cela, C-BGP assure une remise fiable des messages. L'ordonnancement temporel exacte des remises est assuré par l'utilisation d'une file de messages globale et unique.
- Les éléments suivants de la spécification, sont implementés dans C-BGP :
  1. Le processus de décision complet de BGP.
  2. Les filtres d'importation et d'exportation de routes.
  3. Les *communities* de redistribution de routes.
  4. La réflexion de routes.
- C-BGP implémente les attributs : LOCAL_PREF, AS_PATH, NEXT_HOP, ORIGIN, MED, COMMUNITIES, EXTENDED_COMMUNITIES et REDISTIBUTION_COMMUNITIES. La valeur de l'attribut NEXT_HOP correspond toujours à l'adresse IP du routeur propageant la route. Cette contrainte est dictée par l'absence d'IGP dans le simulateur. De plus, C-BGP implémente des attributs non-standars utilisés pour le transport d'information sur la qualité des routes (QoS_DELAY et QoS_BANDWIDTH).
- C-BGP est facilement configurable par une interface de commandes similaire à celle de CISCO.
- C-BGP ne simule pas les timers de BGP (MRAI, MinOriginatingRouteTimer). Les messages sont, donc, émis dès leur construction.
- C-BGP dispose d'une méthode simplifiée de gestion de session, il n'utilise pas l'automate complexe de BGP.
- C-BGP n'implémente pas le *Route Flap Dampening*.
- Afin de pouvoir effectuer les simulations les plus réalistes possibles, C-BGP peut charger des topologies de l'Internet produites par l'Université de Berkeley [Aga]. Il peut aussi sauvegarder tous les échanges de messages dans le format MRTD [MRT], charger des tables de routage réelles au format MRTD ou sauvegarder les tables qu'il a généré dans ce même format.

4.2.1 Modèle de simulation

Modélisation des routeurs BGP

Chaque routeur BGP (*speaker*) dans C-BGP est une entité indépendante qui peut annoncer des routes à ces voisins et rejeter ces routes. Chaque routeur BGP dispose de tous les composants existants dans un speaker réel. Un routeur contient **un** Loc-RIB qui stocke les meillers routes du routeur. Le routeur associe à **chacun de ses peers** un Adj-RIB-In et un Adj-RIB-Out. Le routeur dispose aussi de filtres configurables pour chaque peer. Les filtres pour un peer sont appliqués aux routes importées de ce peer (filtre d'importation) et aux routes annoncées à ce peer (filtre d'exportation). Enfin, le routeur intégre le processus de décision de BGP sous forme d'une séquence de règles, appliquées aux routes reçues des routeurs adjacents.

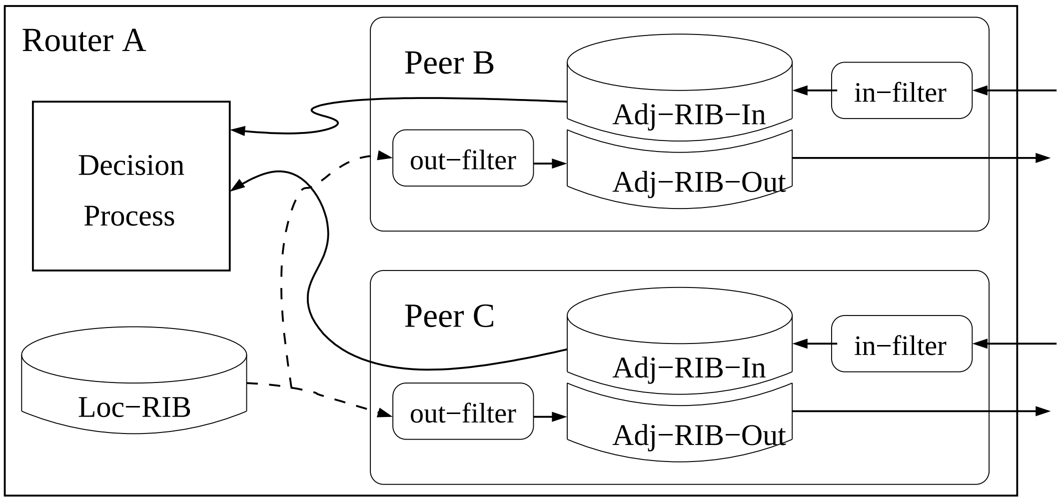

Fig. 4.2 – Modèle de router dans C-BGP

Comme le montre la figure 4.2, le filtre d'importation est appliquée à toute route provenant d'un peer. Puis, le processus de décision est appliqué aux routes provenant de tous les Adj-RIB-In et désignant la destination arrivée dans le nouveau message. Le processus de décision est implémenté sous la forme d'une suite de règles appliquées succèssivement à l'ensemble de routes vers une destination, jusqu'à ce qu'il n'en reste plus qu'une seule - la meilleure route vers la destination. Ces règles sont ordonnées comme suit :

1. Préférer la route qui possède la plus haute valeur de l'attribut LOCAL_PREF.

2. Si l'ensemble de routes candidates comporte plus d'une route, Préférer la route qui possède l'AS_PATH le plus court.
3. S'il reste toujours plus d'une route candidate,préférer une route selon la valeur de l'attribut ORIGIN. Les routes générées par IGP doivent être préférées à celles générées par EGP qui doivent être préférées aux routes d'origines INCOMPLETE.
4. S'il reste toujours plus d'une route dans l'ensemble, sélectionner la route qui a la valeur la plus basse de l'attribut MED.
5. Si l'ensemble comprend toujours plus d'une route, préférer les routes apprises par les sessions E-BGP aux routes apprises par des sessions I-BGP.
6. S'il reste toujours plus d'une route, sélectionner la route avec la métrique IGP vers son NEXT_HOP la plus basse. Et puisqu'il n'y a pas d'IGP dans C-BGP, il faut que les valeurs de ces métriques soient configurées pour chaque routeur.
7. La règle finale de rupture d'égalités calcule une valeur de hachage d'après les AS_PATHs des routes restantes. La routes avec la plus basse valeur de hachage est sélectionnée. S'il reste quand même plus d'une route après l'application du hachage, le processus de décision compare les AS_PATHs des routes et recherche la première différence, puis il choisit la route dont l'ASN dans la position de différence est le plus bas.

A l'issue de cette procédure, il est impossible d'avoir plus d'une route par destination. Cette route est alors mise dans le Loc-RIB, puis soumise aux filtres d'exportation pour être annoncées aux peers du routeur. C-BGP applique une méthode de détection de loop du coté de l'émetteur en vérifiant avant chaque annonce si l'ASN du peer est présent dans l'AS_PATH de la route à annoncer.

4.2.2 Modélisation du réseau

Le réseau dans C-BGP est représenté par un graphe. Les routeurs du réseau sont les nœuds du graphe, les arcs étant les liens de communication entre les routeurs. La représentation en mémoire de ce graphe influe fortement sur les performences du simulateur, il a été décidé de choisir une structure de données offrant un compromis entre efficacité (recherches fréquentes de noeuds pour la remise des paquets) et fléxibilité (à causes de la variabilité de la taille des topologies, du nombre de ses nœuds et de la distribution des arcs). Pour cela, il a été décidé d'utiliser une structure de données en

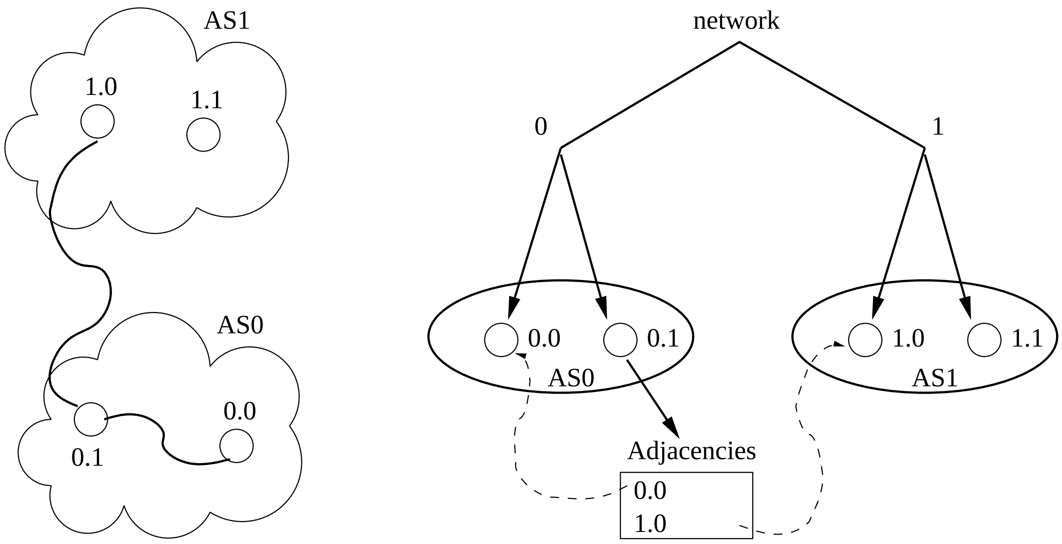

Fig. 4.3 – Exemple de stockage du graph des routeurs BGP dans C-BGP pour des nœuds identifiés par 2 bits

arbre binaire. Chaque nœud étant identifié par un nombre sur 32 bits, la profondeur maximale de l'arbre est de 32. Les nœuds du graphe sont les feuilles de l'arbre binaire. Chaque feuille est associée à une liste contenant les numéros des nœuds adjacents au nœud représenté par la feuille. La liste de numéros de nœuds est implémentée sous la forme d'une table triée pour accélerer la recherche (voir figure 4.3).

Grâce à cette représentation, une topologie avec un grand nombre de routeurs et de liens, peut être étudiée avec des temps d'accès aux nœuds acceptables, et une consommation mémoire raisonnable.

4.2.3 Modélisation des filtres de routes

Les filtres de routage dans C-BGP sont utilisés pour deux choses : d'une part ils servent à choisir parmi les routes reçues des peers, celles qui intégreront l'Adj-RIB-In associé à ce peer (acceptation de route), et à sélectionner parmi les meilleurs routes du speaker, celles à exporter vers les peers ; d'autre part, ces filtres servent à modifier les attributs des routes (assignation d'un LOCAL_PREF, ajout d'un attribut COMMUNITIES, ...).

Les filtres d'importation et d'exportation fonctionnent de la même manière : le principe du filtre est de vérifier un ensemble de conditions, puis si ces conditions sont réunis d'effectuer un ensemble d'actions. De manière formelle, chaque filtre est composé d'une suite de **règles**. Chaque règle est sous la forme :

$$< Predicat \Rightarrow Ensembled'actions >$$

. Le prédicat est une composition de conditions sur les attributs de la route. La composition se fait par les opérateurs AND, OR et NOT. L'ensemble d'actions comprend aussi bien des modifications des attributs que des actions spéciales : DENY pour refuser une route et ACCEPT pour l'accepter.

Les règles formant le filtre sont appliquées séquentiellement à la route, jusqu'à ce qu'elle soit acceptée, refusée, ou qu'il n'y ait plus de règles à appliquer, auquel cas la route est implicitement acceptée.

4.3 Le Simulateur à Grande-Echelle de Hao et Koppol

Le but de Hao et Koppol était d'élaborer un simulateur du protocole de routage BGP permettant d'obtenir un comportement très proche de celui du protocole réel. Et partant de la constatation que la convergence de BGP est fortement influencée par la topologie (taille et connectivité) et les timers du protocole, il a été décidé de concevoir un simulateur de BGP fonctionnant sur des topologies à l'échelle de l'Internet (10000 ASs ou plus) et de conserver dans l'abstraction du modèle de simulation, les temporisations par timers de BGP [HK03].

Dans la conception d'un simulateur à large échelle, il y'a deux aspects importants à prendre en compte : la disponibilité de topologies reproduisant le graphe de l'Internet, et la capacité du simulateur à accepter de grandes topologies. Le premier aspect est résolu par la grande disponibilité de cartographies de l'Internet, cartographies qui peuvent diffèrer dans certaines portions (problème inhérent aux méthodes de génération des topologies de l'Internet). Le second aspect concerne les capacités d'expansion du simulateur. Cela touche la configuration des topologies, la simulation au niveau plan de données, les questions de consommation mémoire des tables de BGP, les évènements à gérer par le simulateur, etc. Le point le plus important étant la consommation mémoire induite par les différentes tables de BGP, il a été proposé une méthode de stockage originale, qui réduit fortement cette consommation.

Le troisième apport de cette étude est une méthode d'estimation de l'impact de la convergence sur le plan de données. Les auteurs ont proposé de nouvelles métriques, refletant beaucoups mieux que le temps de convergence du réseau, l'effet de la convergence de BGP sur le forwarding des données utilisateur.

4.3.1 Modèle de simulation

Modélisation du protocole BGP

Dans ce simulateur, le topologie considérée est le graphe des ASs. Plus précisement, chaque nœud de la toologie est un AS, ne contenant **qu'un seul speaker BGP**. Les liens reliant les ASs sont le sessions E-BGP établies entre les speakers de différents ASs. Les sessions I-BGP est leurs échanges ne sont pas simulés.

Le simulateur est construit à partir du simulateur à évènements discrets

SSFNet, mais contrairement au simulateur construit par Premore [Pre03] [1], ce simulateur ne prend pas en consideration les aspects de communication sur TCP. La remise des messages BGP n'est, donc, pas influencée par une quelconque congestion.

Au niveau d'un AS, le simulateur n'implémente qu'un seul speaker BGP. Celui-ci pourra établir des sessions BGP avec les speakers des autres ASs. Dès lors qu'une telle session est établie, les speakers peuvent s'échanger des messages BGP d'annonce ou de rejet de route (messages UPDATE), selon la procédure normale de BGP. Le processus de décision est executé au niveau de chaque speaker et appliquant les règles de choix de routes telles qu'elles sont présentées dans la spécification. A l'issue de ce processus, la meilleur route est choisie, installée dans le Loc-RIB du routeur et propagée vers les peers.

L'imporation et l'exportation de route sont soumises aux politiques de routage, appliquées dans ce simulateur sous la forme de filtres d'imporation et d'exportation de routes. Chaque speaker dispose de filtres personnalisés pour chacun de ses peers.

Un autre aspect de BGP à prendre en compte est la simulation des timers et des délais de traitements et de transfers sur les liens physiques. Dans ce simulateur, le timer MRAI est simulé avec une valeur de base de 30 secondes et une variation aléatoire par la multiplication de cette valeur de base par un nombre compris dans l'interval $]0.75, 1]$. Chaque speaker possède son propre timer, mais la limitation est appliquée par peer, et non par peer et par destination. Le délai de traitement est considéré comme insignifiant devant la valeur du MRAI, il a donc été complètement ignoré. Mais le simulateur impose un délai de traversée des liens de 0.05 secondes.

Modélisation des politiques de routage

Hao et Koppol ont recensé trois types de politiques de routage pouvant être appliquées sur le Net :

1. Les politiques de transit : permettent à un AS de spécifier clairement qu'ils fournissent (ou pas) le transit aux paquets d'autres ASs vers une certaine destination. Cela est réalisé et permettant à l'AS d'exporter ses informations de routage sur la destination, vers les ASs pour lesquels il veut fournir le service de transit.
2. Les politiques locales de sélection de route : Ces politiques permettent à

[1] Ce simulateur est exposé dans la section 4.1

un AS de toujours préférer un de ses voisins pour les paquets d'une certaine destination, et ceux quel que soit la longeur de l'attribut AS_PATH.

3. Les politiques d'ajout d'ASs : (*AS prepending* ou *AS padding*) Il a été observé que la politique de choix de route majoritairement utilisée est celle du plus court chemin. Afin de tenter d'influencer le choix de routes d'autres ASs, les ASs appliquant des politiques d'ajout d'ASs rallongent artificiellement la longuer de l'attribut AS_PATH d'une route en y accollant plusieurs fois leurs ASN. De cette manière, le lien reliant deux ASs, et qui possède normalement un poids de 1, peut, par cette méthode possèder un poids supérieur. Ce qui fournit aux ASs le pouvoir d'infléchir les décisions de domaines lointains appliquant des politiques de plus court chemins.

Ces trois types de politiques sont implentés sur la topologie de simulation dans différents nœuds. Environ, 500 nœuds (choisis au hasard) sont configurés pour appliquer la politique d'ajout d'ASs. 600 nœuds (choisit au hasard) appliquent une politique locale de choix de route. Pour les politiques de transit, il a fallut identifier tous les nœuds *dual-homés* qui peuvent appliquer une politique de non-transit sans perte de connectivité pour certains autres nœuds, puis configurer ces nœuds pour exhiber un comportement de non-transit. Le reste des nœuds du réseau n'appliquent pas de politiques restrictives et accèptent toutes les routes qui leur sont envoyé. De même, ils annoncent toutes les routes qu'il choisissent à tous leurs voisins.

4.3.2 Nouvelles métriques d'évaluation de la convergence de BGP

Un aspect important dans l'étude des propriétés de convergence de BGP concerne les métriques utilisées pour mesurer l'effet de la convergence sur le *forwarding* des paquets (convoyage des paquets de données utilisateur à travers le réseau). La métrique la plus répandue dans ce type de simulation, le temps de convergence, mesure la durée de temps séparant le début d'une experience de l'instant ou il ne reste plus aucun message à traiter. Cette mesure ne donne malheureusment pas d'indications sur l'impact de la convergence sur le convoyage de données vers la destination. De ce fait, les auteurs de [HK03] proposent deux nouvelles mesures dont le but est d'évaluer les effets du comportement de BGP en période de convergence sur le plan de données. La première est *le temps de convergence du plan de données*, la seconde est *le temps moyen d'inaccessibilité*.

Le temps de convergence du plan de données mesure le temps entre le début d'une expérience et le moment ou tous les nœuds du réseau possèdent un chemin stable vers la destination. Un chemin stable d'un nœud vers une destination est obtenu par l'invarience du nexthop pour atteindre cette destination, dans le nœuds et dans tous les nœuds en aval. Ainsi, au cours de la convergence, le système peut atteindre un état stable sur le plan de données, même s'il reste encore des messages à traiter, que ces messages provoquent des changements sur les AS_PATH mais pas de chagements sur les NextHops.

En plus d'obtenir la convergence du plan de données le plus rapidement possible, il est souhaitable de minimizer le temps totale de perte de connectivité vers une destination. Le temps moyen d'inaccessibilitée sert de métrique pour capturer cette notion. L'approche utilisée pour estimer cette valeur est de vérifier la connectivité du chemin de données en partant de tous les nœuds vers la destination, et ce à des intervales alétoirement choisis. Le temps d'inaccessibilité pour un nœud est, alors, calculée en multipliant le rapport du nombre de vérifications qui ont montré un manque de connectivité au nombre totale de vérifications, par le délai de convergence sur le plan de données du nœud. La vérification de la connectivité de tous les nœuds est obtenue en appliquant récursivement le vérification ; ce qui facilite grandement la procédure.

4.3.3 Méthode global de stockage des tables BGP

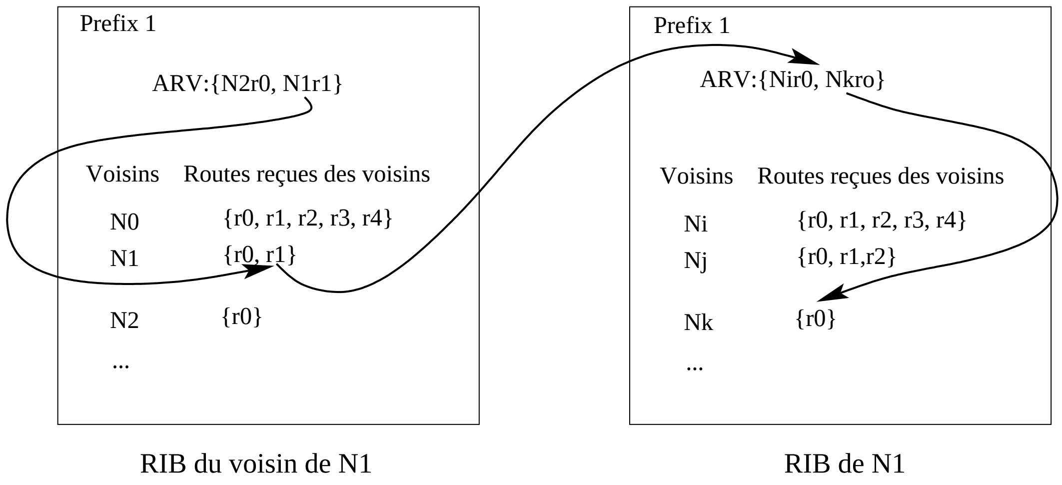

Fig. 4.4 – Fonctionnement du RIB globale dans le simulateur de Hao etKoppol

Au lieu d'adopter la représentation traditionnelle des RIBs pour chaque speaker (ce qui induit une grande consommation mémoire), [HK03] a décidé

de choisir une structure de données globale pour le stockage des routes importées et des meilleurs route. Ce choix est justifié par la taille des topologies utilisées (quelques milliers de nœuds, avec une moyenne de 4 liens par nœud). Aussi, il est indispensable de sélectionner la structure de données qui minimise le plus possible la consommation mémoire, et qui permette, ainsi, de simuler les topologies les plus grandes.
La structure de données utilisée consiste en un **RIB global** qui maintient l'union des RIBs de tous les speakers BGP du réseau. Comme le montre la figure 4.4, le RIB global conserve pour chaque destination, pour chaque speaker du réseau : la liste des voisins du speaker, une liste de route active (ou *active route vector*, ARV), et pour chaque peer du speaker, une liste de routes reçues à partir de ce peer. Ainsi, la liste de routes reçues du voisins N d'un speaker A contient la liste des annonces de routes émises par N vers A. Et la liste des routes actives d'un nœud A, contient la meilleur route vers la destination et les précedantes meilleurs routes vers cette destination que A a exporté vers ses peers.
En fait, ces deux types de liste sont des listes de pointeurs, comme le montre la figure 4.4. Dans un AS A (ou dans son unique speaker), la meilleur route vers la destination est stockée dans sa liste de routes actives en tant que **pointeur vers un élément de la liste des routes reçues de l'un de ses peers, N**. La liste des routes reçues de N, est elle même une **liste de pointeurs vers les éléments de la liste des routes actives de N**. Cette dernière liste contient des pointeurs vers les éléments des listes de routes reçues à partir des peers de N ; et ainsi de suite. La route active d'un AS vers un prefix qu'il a lui même généré est NULL. De cette manière, le chemin d'un AS vers une certaine destination est construite en parcourant successivement les listes de pointeurs, en commençant par la plus récente des entrées de la liste des routes actives de l'AS.
Le but de ces différentes listes est de pouvoir reconstruire le chemin de n'importe quel AS vers la destination. Quand le réseau est dans un état *stable*, les listes de routes actives ne devraient contenir qu'une seule entrée : la meilleur route vers la destination, et les listes de routes reçues ne devraient contenir que les plus récentes routes publiées par les peers. Mais en cours de convergence, il peut arriver qu'AS ne possède pas encore la meilleure route d'un de ses peers. Il faut, donc, que le peer conserve ses ancienne meilleurs route pour pouvoir à tout moment reconstruire le chemin d'un AS vers la destination.

4.3.4 Expirementations

La topologie utilisée pour les experimentations sur le simulateur est la topologie UMICH [JSW] qui est dérivée de l'union des tables de routage BGP

présentes sur plusieurs points de l'Internet. La topologie finale contient 11806 nœuds et 38781 liens. Parmi les nœuds de la topologie, 2897 sont connectés par un unique lien et 4757 sont dual-homés.
Dans chaque simulation, seule une destination est annoncée dans un AS choisit au hasard. Les MRAI sont réenclenchés périodiquament pour simuler la présence d'autres destinations dans le réseau. Les ASs sont configurés pour appliquer les politiques citées ci-haut. Les ASs qui n'utilisent pas ces politiques particulières utilisent un choix selon le plus court chemin.
Deux types d'évènements ont été étudiés : les pertes de connectivité entre ASs (*link down*) et l'augmentation des ajouts d'ASs (augmentation des poids des liens entre ASs, ce qui implique la concatenation d'un plus grand nombre d'ASN du speaker à l'AS_PATH). Dans les deux cas de figure, les procedures normales et améliorées de BGP ont été testées (BGP standard *vs.*, activation ou non du SSLD, du WRL, ...). L'étude a aussi différencié deux type d'experiences, selon le type de l'AS d'origine : *single homed origin* ou *dual homed origin*.

Enfin, les tests ont été effectué en utilisant différentes graines de générateur de nombres aléatoires. Les moyennes des résultats obtenus, ainsi, qu'un interval de confiance de 95%, ont été montrés et comparés, pour évaluer la perspicacité des améliorations à apporter au protocole BGP.

4.4 Le simulateur de convergence de BGP

Présenté dans [NCM02], ce simulateur a été conçu dans le but de tester les proprietés de convergence du routage inter-domaine. En effet, depuis l'introduction du protocole BGP, plusieurs études ont été menées sur ses proprietés de routage ; toutes ont concluent au comportement dommageable de BGP. Du fait de la liberté d'implémentation du protocole, plusieurs techniques ont été proposées pour agir sur la convergence du routage inter-domaine, et divers paramètres du protocole sont librement configurables. Les concepteurs du simulateur se sont intéressé aux interactions des différents paramètres de BGP dans les environnements à topologies dynamiques et aux effets de différentes techniques d'amélioration de la convergence sur les proprietés de convergence de BGP.

La mise en œuvre du simulateur s'est faite en quatre phases :

1. Le développement d'un simulateur à évènemets discrets du protocole BGP, avec un haut degré d'abstraction.
2. La vérification de l'exactitude de l'implémentation de BGP dans le simulateur ainsi que de l'exactitude du simulateur lui-même.
3. Les tests de différentes valeurs du MRAI. Cela inclus, les expériences sur les interactions entre les timers, la mise en place des timers pour certains messages BGP ou non, l'étude des différences entre timers synchronisés et timers non-synchronisés.
4. L'étude des proprietés de convergence en fonction de la taille du réseau. Les tests ont exclusivemnt porté sur des topologies en clique, puisque ce sont les topologies au comportement le plus complexe.

4.4.1 Le modèle de simulation

Le modèle de simulation représente une abstraction du modèle réel de routage par BGP. Ainsi, certains aspects fondamentaux de BGP sont compris dans le modèle de simulation, mais d'autres détails ont été ignoré,tel que les mécanismes de transports sous-jacents(TCP\IP). Le modèle a été implémenté dans le simulateur *RouteSim.* écris en Java, ce dernier comporte un moteur de simulation, des entités modélisant des composants tels que les routeurs ou les liens, et des évènements représentant des changements d'états du système.Les entités composants le modèle de simulation sont chargés à partir d'un fichier textuel. Les évènements peuvent, soit être créés par certaines entités, soit être injectées depuis l'extérieur du modèle de simulation. Les entités routeur représentent chacun un speaker BGP établissant des sessions E-BGP. Pour conserver la simplicité du modèle, les sessions I-BGP n'ont pas été prises en compte. Chaque AS est, donc, constitué de son unique routeur BGP. Les liens entre routeurs modélisent des connections point-à-point *full duplex* qui délivrent des paquets entre des paires de routeurs. Ces liens garantissent la livraison des paquets en un délai dont la durée est indépendante de la taille du paquet. Cela modélise le comportement d'un protocole de remise fiable. Le modèle du système forme, donc, un graphe, comprenant des routeurs formant les nœuds du graphe et des liens entre routeurs pour les arêtes. La simulation est effectuée sur le plan de contrôle du routage uniquement. Seuls les échanges d'informations de routage sont modélisés. Le convoyage des données utilisateur n'est pas pris en compte, mais son effet est mis en œuvre sous forme de gigue de délai des transmission des paquets de routage. Le modèle de simulation ne comporte qu'une seule destination. Cette destination est annoncée à partir d'un nœud choisit au sein réseau. La charge du routeur (charge du CPU) est modélisée par un délai aléatoire uniforme imposé à chaque paquet. Seuls les messages UPDATE sont compris dans la simulation. Cela impose l'utilisation d'un modèle de simulation sans gestion de session. Toutes les sessions sont, donc, toujours dans l'état *Established*. Au niveau de chaque routeur BGP modélisé, il existe un timer MRAI pour chacun de ses peers. Seuls les annonces de routes sont astreintes au retards imposés par le MRAI. Les rejets de routes ont immédiatement transmis. Les routeurs peuvent disposer de politiques de routage individuelles. Ces politiques sont mises en places sous forme de chemins d'AS préférées aux autres chemins plus cours. Enfin, l'ordre initial d'annonce ou de rejet de route peut provenir d'un fichier de configuration ou d'un évènement injecté de l'extérieur.

4.4.2 La mise en place des expérimentations

Toutes les expériences ont porté sur des topologies en clique. Chaque AS (ou son unique routeur) est connecté à l'ensemble des autres ASes du modèle. Les expériences ont surtout porté sur l'étude des effets de la variation de la valeur du MRAI sur la convergence, et l'impact de l'utilisation de méthode d'amélioration de la convergence. Les méthodes étudiées sont :

- *Sender Side Loop Detection*-SSLD : mise en œuvre de BGP ou chaque routeur doit vérifier l'absence de l'ASN d'un peer d'un chemin avant de lui envoyer ce chemin.
- *Split Horizon*-SH : implémentation ou un speaker ne renvois pas une annonce de route au peer d'ou elle provient.

Le délai imposé par la latence du CPU a été aléatoirement sélectionné dans l'interval [0.01, 0.1] secondes. La latence due au lien de transport a pris la valeur 0.01 secondes, sans variations.
Toutes les expériences ont été effectuées de la même manière : un routeur est

d'abord configuré pour annoncer une destination, puis pour la rejeter ensuite. Le temps entre l'annonce et le rejet du routeur source a été choisit de manière à obtenir, avant le rejet de destination, une situation stable du réseau, où chaque routeur possède une route stable vers la destination. Puis les mesures ont calculé le temps de convergence (période de temps séparant le rejet de la destination, de l'instant de traitement du dernier message UPDATE) et le nombre de messages d'annonce de route et de rejet de route. Il est à noter qu'aucun routeur n'a appliqué de filtrage de route. Tous les routeurs ont donc sélectionné leurs meilleurs chemin selon le nombre d'ASs dans l'AS_PATH.

4.4.3 Résultats et conclusions

Sur les experiemenations réalisées avec 15 ASes, sans SSLD ni SH, il est apparu qu'il existait une valeur du MRAI qui minimise le temps de convergence et le nombre de messages échangées. Cela confirme les résultats de [Pre03]. Sur une telle configuration, la valeur du MRAI optimale était de 8 secondes. Pour des valeurs supérieurs à cette valeur optimale, le temps de convergence croît linéairement avec la valeur du MRAI, le nombre de messages restant constant. Pour les valeurs inférieurs à cette valeur optimale, le temps de convergence et le nombre de messages prennent de très grandes valeurs qui augmentent avec la diminution de la valeur du MRAI. Les expériences ont permit de confirmer la borne inférieur du temps de convergence, $(n-3) * MRAI$, calculée dans [Pre03].

Un autre résultat de ces expériences concerne la synchronisation des timers MRAI des routeurs. La valeur optimale du MRAI n'est pas influencée par la synchronisation des timers. Les tests réalisés avec des timers asynchrones montrent une augmentation du temps de convergence en fonction de la valeur du MRAI légèrement plus faible qu'avec des timers synchronisés. Les expériences sur l'utilisation de SSLd et de SH montrent que l'utilisation de l'une de ces méthodes diminuent le nombre d'annonces de route échangés ainsi que le temps de convergence. La diminution la importante provient de SSLD, mais, ce dernier provoque une augmentation du nombre de rejets de routes, ce qui produit un nombre de messages global approximativement égale dans les trois situation (RSLD,SSLD ou SH).

4.5 Le simulateur BGP++

Le simulateur BGP++ [Dim] est un simulateur du protocole BGP construit sur l'environnement de simulation *ns2* [**?**]. L'approche adoptée pour élaborer ce simulateur est totalement inédite : au lieu de construire le simulateur à partir d'un modèle du protocole, il a été décidé d'incorporer une implémentation réelle de BGP dans le simulateur. L'implémentation choisie pour la simulation est le démon BGP du logiciel de routage GNU *Zebra* [ZEB]. L'avantage de cette approche est que le simulateur hérite automatiquement des détails et des fonctionnalités du logiciel originel, tout en minimisant l'effort de développement.
Zebra est une implémentation *Open Source* de plusieurs protocoles de routage (RIP, OSPF,IPv6, SNMP,BGP, ...). La réelle modularité de conception de *Zebra*, a permit de n'exploiter que le processus BGP du projet pour la création du simulateur. De plus, la vaste utilisation qu'en a fait la communauté d'Internet sur les infrastructures réelles, permet certifier le réalisme du simulateur construit à partir de *Zebra*.
ns2 est un simulateur à évènements discrets largement utilisé, qui fournit les détails d'implémentation de beacoups de protocoles de routage. Ainsi, l'association de *ns2* et de *Zebra* fournit un modèle de simulation très détaillé de BGP

4.5.1 Méthode d'intégration du software *Zebra* dans le simulateur

Comme cité auparavant, le simulateur BGP++ a été construit à partir du démon *bgpd* de *Zebra*. La logique du code, ainsi que les algorithmes implémentés ne sont pas changés. Cependant, il a fallu supprimer ou modifier les aspects du software liés au système d'exploitation, tel que les appels système de blocage et de communication. En fait, les principales modifications ont concerné :

- Le passage de l'implémentation en langage C du démon *bgpd* à une version C++ pour le simulateur
- La transition d'un ordonnencement basé sur les processus à un ordonnencement à évènements discrets.
- Le changement des communication via des sockets BSD à des communications utilisant les implémentations TCP simulées sur *ns2*.
- L'utilisation de routines non-bloquantes dans le simulateur au lieu des routines bloquantes de *Zebra*.
- L'aspect le plus important de la transformation a concerné la capacité du simulateur à lancer plusieurs démons BGP simultanément, attendu

que *Zebra* n'a été conçu que pour initier un seul démon BGP à la fois sur une certaine machine.

Ces modifications ont été apportée durant les quatres phases du processus d'intégration de *bgpd* dans le simulateur *ns2* :

1. La convertion du code C de *Zebra* en C++. Cela fut réalisé par la création d'une classe, appellée BGP, qui contient la majorité du code C du démon. Toutes les fonctions C furent transformées en méthodes membres de la classe, et toutes les variables globales furent incorporées en tant qu'attributs membres de la classe.

2. Le remplacement du *scheduler* de *bgpd* par le *scheduler* à évènements discrets de *ns2*. Les applications réseaux typiques fonctionnent selon un modèle simple : elles s'exécutent jusqu'à ce qu'elles rencontrent une routine de blocage. Pour incorporer l'application au simulateur, il faut modifier le *scheduler* de l'application pour qu'il communique avec l'ordonenceur de la file des évènements. Cela ce fait en modifiant l'application de telle sorte qu'elle s'exécute normalement jusqu'à rencontrer une routine de blocage, puis au lieu de bloquer l'exécution, elle doit rendre la main au *scheduler* de la file. L'application reprend la main lorsque la routine de blocage la déverrouille.
   Dans *Zebra*, la routine de blocage utilisée est l'appel système *select()*, qui doit recevoir en entrée une liste de descripteurs fichiers et une valeur de *timeout*. Cette routine bloque jusqu'à ce que l'état de l'un des fichiers change ou que le timeout expire. Pour intégrer *bgpd* dans le simulateur, les appels à select() sont remplacés par une mise en file d'un évènement d'expiration du timeout dans la file des évènements. Si le simulateur rencontre un évènement d'arrivée de paquet, il supprime de la file l'évènement de timeout et donne le contrôle au démon BGP.
   Afin de modéliser le temps d'occupation du CPU par les traitements BGP, à chaque fois qu'un démon BGP simulé complète une opération,il sélectionne une **période d'activité** (*busy-period time*). Durant cette période, le démon ne peut exécuter aucune opération, ni répondre à aucune invocation. Tous les invocations sont bufferisées dans l'ordre de leurs arrivées, puis les traitements associées à la première invocation sont exécutés à la fin de la période d'attente, ce qui génére une nouvelle période d'attente, ... et ainsi de suite.

3. La substitution de l'implémentation de TCP de *ns2* aux sockets BSD de *bgpd*. Cette étape consiste en le remplacement de tous les aspects de communication utilisant les sockets BSD par leurs correspondants dans les modèles simulées par *ns2*.
   Un autre probléme traité dans cette étape concerne le passage d'une

instance du démon BGP vers de multiples instances. En particulier, il a fallu mettre en place un système de configuration et de log individuel, pour permettre la différenciation des processus BGP.

4. Le remplacement des appels système dans *Zebra* par des fonctions du simulateurs et la suppression du code non-obligatoire. Par exemple, *Zebra* possède une interface de communication *telnet*. Cette option a été supprimée et remplacée par un service équivalent permettant de communiquer avec les démons BGP simulés lors de l'exécution.

4.5.2 Validation et vérification du simulateur BGP++

L'argument, pour affirmer la validité du simulateur BGP++, se base sur le procédé de construction du simulateur. En effet, BGP++ n'est pas conçu à partir d'un modèle abstrait d'un phénomène existant, mais il est élaboré à partir d'un software réel. De ce fait, la validité du simulateur provient de la validité du software et de la validité de la méthodologie d'intégration. *Zebra* est un logiciel de routage largement utilisé, et l'intégration consiste en le remplacement des aspects relatifs à l'OS dans *Zebra* par des dispositifs correpondants dans *ns2*. *ns2* et *Zebra* étant validés et amplement employés, les auteurs concluent à la validité du modèle de BGP fournit par BGP++. La vérification du simulateur a consisté en l'évaluation du degré de justesse du simulateur par rapport au software originel. Les premiers scénarios de tests ont permis d'établir la capacité du simulateur à effectuer plusieurs des tâches indispensables dans BGP : la gestion des sessions (établissement, maintient et fermeture), la distribution des routes et l'application correcte de l'algorithme du choix de routes, l'usage correct des aspects du protocole relatifs aux politiques de routage , l'utilisation des services de monitoring, l'emploi des dispositifs avancées de BGP(les conféderations, la réflexion de routes, la négociation de paramètres, ...). Ensuite, des tests ont été menés pour comparer le comportement de BGP++ par rapport au comportement de *Zebra*. Pour cela, il a fallu mettre en place une petite configuration de routeurs *Zebra*. Pour chaque configuration, l'objectif était de comparer les valeurs des métriques observés sur la configuration de routeurs *Zebra* (nombre de messages échangés, temps de convergence), et les valeurs des mêmes métriques dans une topologie de simulation. Les résultats obtenus montrent que BGP++ génére effectivement les mêmes comportements que ceux du banc d'essai, confirmant par là, la justesse de l'implémentation du simulateur.

4.5.3 Mesures des performances du simulateur

Pour la simulation de grandes topologies, l'aspect le plus influant sur les performances du simulateur est la consommation mémoire des simulations. Afin de mesurer cette consommation, un scénario de simulation paramétré est créé. chaque simulation contient n routeurs, interconnectés en clique de taille n. Chaque routeur appartient à un AS séparé qui annonce p préfixes. Après la stabilisation du réseau, chaque routeur dispose de n*p destinations dans ses tables de routage. Sur les simulations, il a été observé que l'augmentation du nombre de préfixes, c'est à dire la taille des tables de routage, limite fortement la taille de la topologie (quelques dizaines de routeurs). Au contraire, l'utilisation de tables de routage limitées permetteraient de simuler plusieurs milliers de routeurs de manière efficace. Les prochaines modifications apportées au simulateur tenteront, donc, de réduire la consommation mémoire des tables de routage, afin de réaliser les simulations les plus réalistes possible (au point de vu taille des topologies, et nombre de réseaux annoncés).

4.6 Le simulateur ns-BGP

Proposé par T. Feng, R. Ballantyne et L. Trajkovic, ce simulateur présenté dans [FBT04] implémente la dernière version du protocole de routage inter-domaine, BGP-4, dans le simulateur *ns-2*. En fait, la mise-en-œuvre de du modèle de simulation sur *ns-2* a été effectué à partir de l'implémentation de BGP-4 sur SSFNet. Le modèle de BGP utilisé pour le portage est celui du simulateur SSF.OS.BGP4, conçu par Griffin et Premore [Pre03]. Ce modèle étant entirement programmé en JAVA en utilisant une approche purement objet, il a fallu porter la hiérachie de classes de SSF.NET.BGP4 vers le simulateur *ns-2*. L'implémentation du protocole BGP réalisée dans le simulateur ns-2 est conforme aux spécifications du protocole BGP-4 (RFC 1771) et comporte quelques uns des éléments optionnels du protocole, tel que les attributs MED, Aggregator et Community. En plus de cela, il comporte plusieurs dispostifs experimentaux : la détection de boucle chez l'émetteur *SSLD*, la limitation du rythme d'envoi de rejets de route *WRL*, la fixation du MRAI (sans variations) et la disposition d'un timer MRAI par peer et par destination. Enfin, le simulateur implémente aussi la réflexionde routes, et dispose de ce fait des attributs OriginatorID et ClusterList.

4.6.1 Conception et implémentation de ns-BGP

Les classes constituant le simulateur ns-BGP sont dérivées de la hiérarchie de classe existante de ns-2, et plus particulièrement de la structure de routage **unicast** de ce simulateur. Cette structure est constituée de deux plans : le plan de **contrôl** et le plan d'**acheminement** (*forwarding plan*). Le plan d'achemminement des données est responsable de la classification est du convoyage des paquets vers les nœuds destinations. Il est composé de divers types de *classifiers* et de *routing modules*, interconnectés. Dans ce qui suit, nous présentrons d'abord brievement la structure de routage classique de *ns-2* ; puis, nous décrirons l'architecture de routage de ns-BGP.

Structure de routage unicast de *ns-2*

Comme le montre la figure 4.5 L'architecture de routage unicast de ns-2 est composée d'un ensemble de classificateurs (*classifieurs*) et de modules de routages (*Routing modules*). Les classificateurs ont pour rôle de délivrer les paquets entrants dans un nœud à l'agent approprié, ou vers le liens de sortie. Les modules de routage doivent gérer les classificateurs des nœuds et fournir une interface vers le plan de contrôle. La figure 4.5 montre deux types de classificateurs (schématisés par des trapèse) :

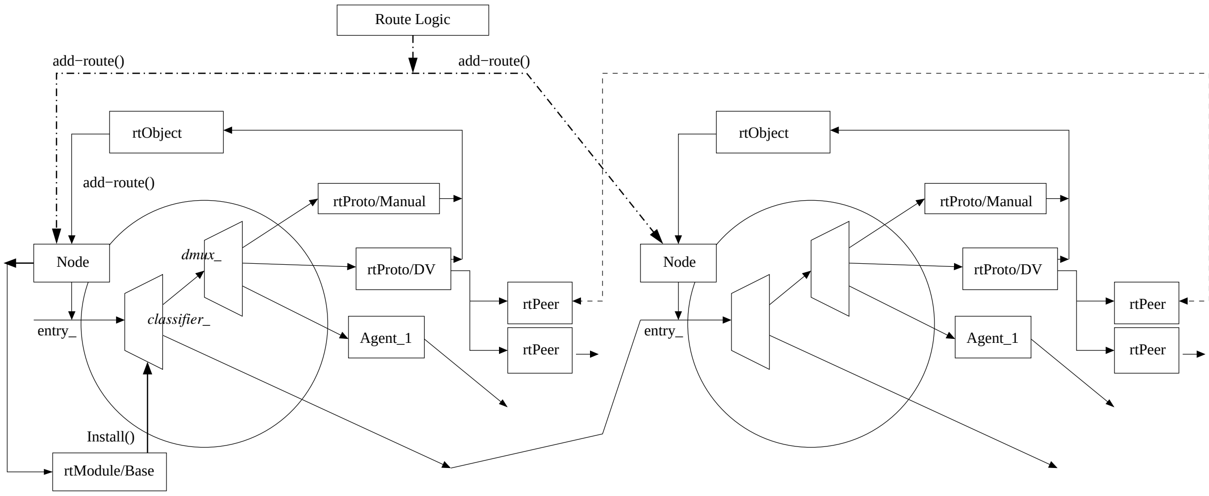

FIG. 4.5 – Structure de routage Unicast de ns-2

- le **classificateur d'adresse** (*classifier_*) examine l'adresse de destination de chaque paquet entrant dans le nœud, puis il livre cet paquet au classificateur *dmux_* si le nœud est la destination du paquet. Sinon, le *classifier_* envoi ce paquet sur un lien sortant.
- le **classificateur de ports** (*demux_*) remet les paquets, qu'il reçoit du *classifier_*, à l'agent qui correspond au numéro de port transporté par ces paquets.

Le plan de contrôle se charge du calcul des routes, ainsi que de la creation et de la maintenance des tables de routage. Il implémente aussi les algorithmes spécifiques de routages du modèle de simulation. Ce plan de contrôle comprend quatre composants :

le composant *Route logic* représente la table de routage créée et gérée de manière centralisée.

les composants *Route object* sont uniquement employés dans la simulation de routage dynamique. Lorsqu'un *Route Object* est associé à un nœud, il agit comme une entité coordinatrice entre les différentes instances de routage de ce nœud.

le composant *Route peer* agit comme un conteneur utilisé par le protocole de routage (*Routing Protocol*), il permet de stoquer les adresses des peers, les métriques et les préferences pour chaque route annoncé par le peer.

le composant *Routing protocol* implémente un algorithme de routage spécifique, tel que les algorithmes à vecteurs de distance, ou à états de liens.

Architecture de routage de ns-BGP

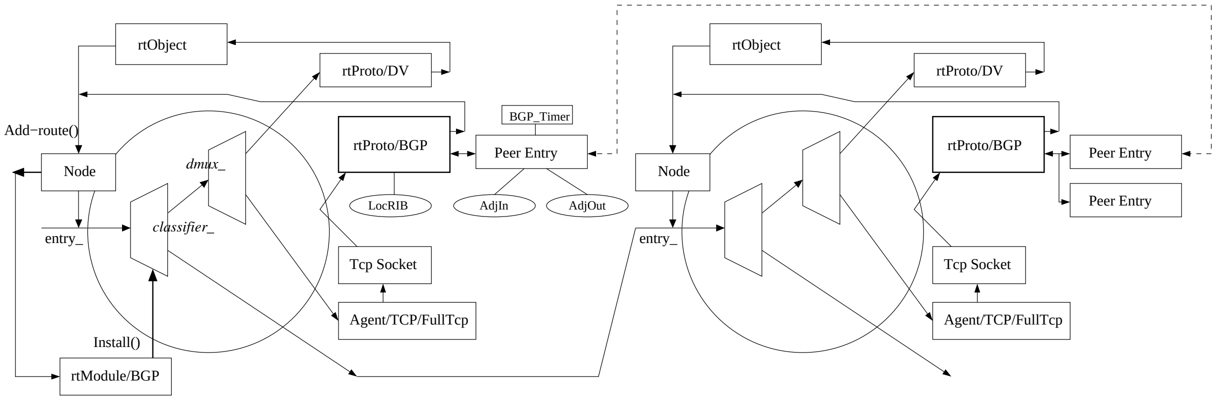

FIG. 4.6 – Structure de routage du simulateur nsBGP

La structure du nœud, dans ns-BGPest, est construit à partir de la structure du nœud unicast de *ns-2* et du modèle SSF.OS.BG de SSFNet. Le modèle de SSFNEt a été convertit et integré à *ns-2*. En plus de cela, il a fallu :

– fournir le support des *sockets* pour maintenir la structure de SSF.NET.BGP4. Pour cela, la couche d'implémentation des *sockets* de SSFNET, *TcpSocket*, a été portéé vers *ns-2*.
– fournit un support de l'adressage IPv4, et de l'acheminement des paquets. Cela fût obtenus par le remplacement du classificateur d'adresses classique par un nouvel classificateur, *IPv4Classifier*.
– supporter la transmission de données utilisateur. Pour cela, il a fallu modifier *FullTcpAgent*, qui est l'agent TCP pour *TcpSocket*.

La figure 4.6 montre l'architecture de nsBGP. Sur ce schéma, le classificateur *classifier_* est un *IPv4Classifier*. Un nouveau module de routage, *rtModule/BGP*, gère l'*IPv4Classifier* et remplace le module de routage de base. *TcpSocket*, qui encapsule les services TCP en une interface sockets, a été ajouté à l'agent *FullTcpAgent*. Un nouveau protocole de routage, *rtProto/BGP*, se base uniquement sur *TcpSocket* pour la transmission de ses messages. Le protocole *rtProto/BGP* possède un *PeerEntry* pour chacun de ses peers (internes ou externes). *PeerEntry* est chargé de l'établissement et de la fermeture de la session avec le peer qui lui est associé. Il prend aussi en charge l'échange des messages avec ce peer, et il intègre pour cela un

Adj-RIB-In (*AdjIn*), un Adj-RIB-Out (*AdjOut*) et une variable *BGP_Timer*. Pour clore la description du simulateur, nous présentons, dans ce qui suit, les quatre classes les plus importantes de nsBGP :

- **TcpSockets** Cette classe est une implémentation de l'API des sockets utilisés pour les communications réseaux. L'implémentation est assez similaire de celle du système UNIX. Elle inclus, pour cela, les fonctions classiques des sockets : *bind, listen, connect, close, read,* et *write*. La mise en œuvre de cette classe a exigé la construction, et l'incorporation à *ns-2* d'autres classes, en particuliers la classe *TcpData* qui contient les données utilisateurs transmises. Enfin, l'agent *FullTcpAgent* de *ns-2* a subit des modifications afin qu'il puisse transmettre et recevoir des données utilisateurs, et informer le *TcpSocket* correspondant des changements d'états de TCP.
- **IPv4Classifier** Le classificateur *IPv4Classifier* est dérivé du *Classifier* de *ns-2*. Il a été implémenté en tant que **classe dual de *ns-2*** (Mise en œuvre en C++ **et** en OTcl). Son rôle consiste à examiner l'adresses de destination d'un paquet, et à choisir dans la table de routage du classificateur la route possèdant le préfixe le plus long qui corresponde à cette adresse.
- **rtModule/BGP** Le nouveau module de routage, *rtModule/BGP*, est implémenté en Tcl dans *ns-2*, et fournit une interface d'enregistrement pour les nœuds. Cet enregistrement, lors de la création du nœud, permet le remplacement des objets classificaters dans le nœud.
- **rtProtoBGP** Cette classe est implémentée en tant que classe dual de *ns-2*. Chaque instance de cette classe dans un nœud, y implémente BGP-4. Cette classe effectue la plupart des opérations BGP du nœud : l'établissement des sessions BGP (internes et externes), la gestion de l'automate à états finis de BGP, la réception des routes de la part des peers (internes et externes) et le choix des meilleurs routes pour chaque destination, ainsi que le stoquage des ces routes dans la table de routage IP (utilisable par *IPv4Classifier*).

4.6.2 Validation du simulateur

Griffin et Premore ont élaboré une série de tests, visant à prouver la validité du modèle de simulation de SSF.NET.BGP4 et sa conformité à spécification du protocole. La plupart de ces tests de validité ont été implémentés dans *ns-2*, en utilisant les mêmes topologies de tests que celles utilisées pour le modèle de SSFNet. En plus de ces tests, les concepteurs de ns-BGP ont introduit un nouveau test de validité pour la réflexion de routes, consistant en une topologie de trois ASes, dont l'un contient plusieurs clusters (chaque cluster contenant un ou plusieurs réflecteurs de routes, et les clients de ces réflecteurs). Le but de ce test était de valider, plus particulièrement, le comportement de plusieurs réflecteurs de routes appartenant au même cluster.

4.7 Simulation d'infection de Ver sur Internet

Élaboré par Premore et Nicol et présentés dans [LYPN02], ce simulateur a été conçu dans le but d'obtenir un modèle de simulation permettant de visualiser les effets de la propagation des vers sur le routage inter-domaine. En effet, des études d'observation de l'état du routage inter-domaine durant la propagation des vers *Nimda* et *CodeRed II* ont montré une corrélation entre les périodes de fortes activité des vers, et les instants d'instabilité du routage inter-domaine. Grâce à la modélisation, les concepteurs espèrent avoir une idée sur les menaces apportées par certains scénarios de propagation, sur les échelles de temps d'apparition de certains évènements, sur l'efficacité de certaines contre-mesures et sur les liens possibles avec d'autres phénomènes observés. Cependant, la relative grande échelle de propagation de ces vers a posé un nombre de défis pour toute tentative de modélisation en détail de ces phénomènes. Les auteurs ont, donc, tenté une nouvelle approche : utiliser une abstraction sélective de la propagation des vers, par l'utilisation de modèles épidemiologiques de propagation de maladies biologiques, en l'associant à des modèles détaillés des protocoles.

Afin de pouvoir simuler les effets de la propagation de vers sur le routage inter-domaine, il faut modéliser ces trois élèments cruciaux :

- un modèle de la propagation du ver et d'infestation des hôtes par le ver.
- un modèle de traffic pour les *scans* émis par le ver.
- un modèle de la manière dont les scans du ver induisent une surcharge sur (certains) routeurs, et la façon dont cette surcharge provoque du traffic de message de routage.

La simulation a été réalisée sur le simulateur SSFNet, à cause des possibilités d'élargissement des topologies de simulation par l'utilisation d'executions parallèles ou distribuées. Cependant, la parallélisation des simulations ne peut constituer une solution pour tous les challenges de la modélisation. Il a fallut combiner les abstractions les plus à même de fournir des résultats, avec la puissance offerte par les techniques de simulation parallèles ou distribuées.

4.7.1 Modèle de simulation

Le but des concepteurs de ce simulateur était de conserver le comportement de routage de BGP, aussi fidèle au modèle réel, qu'il était possible. A cet effet, il ont choisit de travailler avec une forme d'abstraction sélective : les

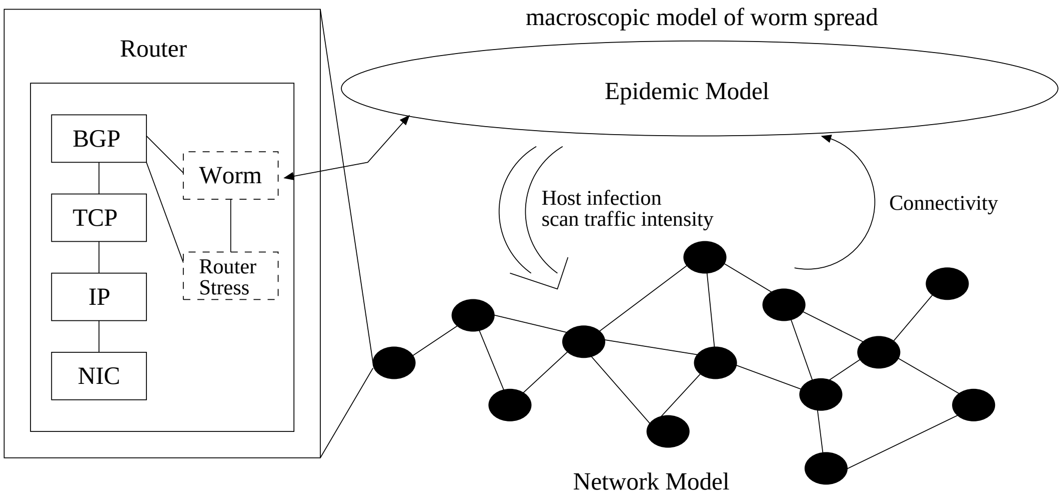

FIG. 4.7 – Modélisation macroscopique de la propagation de ver par un modèle épidemique. Le comportement du routeur BGP est modélisé en détaille.

modèles épidemiologiques, empruntés au domaine des maladies biologiques, constituent un cadre d'abstraction qui convient parfaitement à la modèlisation de la propagation de vers sur Internet.

Il a fallut, alors, combiner ce modèle épidemique avec un modèle détaillé de l'infrastructure du réseau.Le modèle épidemique décrit la propagation du ver tandis qu'il infecte des hôtes de l'Internet, soit en utilisant un modèle stochastique discret (par étapes temporelles), soit en utilisant un modèle déterministe (par l'utilisation d'équations différentielles). Quand au modèle du réseau, il comprend des modèles précis des routeurs et des protocoles. Comme le montre la figure 4.7, chaque routeur du réseau execute une pile de protocoles incluant TCP/IP et BGP. Le lien entre le modèle du réseau, et le modèle épidemique se fait par l'intermédiaire d'un pseudo-protocole, *Worm*. Un autre pseudo-protocole, *Router Stress*, implémente le modèle de surcharge de traffic induite par les scans et les pannes déclenchées dans les routeurs.

Modèle d'infection de Ver

Le modèle épidemique général considère une population fixe de taille N, dans laquelle chaque individus prendre l'un des trois états : il peut être sensible à la maladie (S, *Susceptible*), il peut être infecté par la maladie(I, *Infected*) et il peut être éliminé (R, *Removed*). L'élimination peut signifier plusieurs choses : la guerison si la victime devient immunisée à la maladie, l'isolation de la victime ou la mort de celle-ci. La progression normale pour un individus de la population est S→I→R, notée par le modèle SIR. Mais il peut y avoir d'autres scénarios, par exemple, si la victime qui guérit de la maladie n'aquire pas l'immunitée. La progression devient alors S→I→S.

En considérant le modèle SIR, et en notant par $s(t)$, $i(t)$ et $r(t)$ le nombres d'individus se trouvant dans l'état S, I et R à l'instant t on obtient $s(t) + i(t) + r(t) = N \forall t \leq 0$. L'évolution du système peut être modélisée de manière stochastique en considérant les probabilités, pour les individus, de devenir inefectés en interagissant avec les indivudus déja infectés ; et les probabilités d'élimination des individus infectés.

Kermack et McKendrick ont obtenus, sous la forme d'un système d'équations, un modèle déterministe à temps continu et à états continus fournissant une approximation du modèle stochastique, à la seule condition que la taille de la population (nombre d'individus) soit suffisament grande. Ce système peut être écris sous le forme :

$$\frac{ds(t)}{dt} = -\beta s(t) i(t) \tag{4.1}$$

$$\frac{di(t)}{dt} = \beta s(t) i(t) - \gamma i(t) \tag{4.2}$$

$$\frac{dr(t)}{dt} = \gamma i(t) \tag{4.3}$$

où β représente le *paramètre d'infection*, et γ est la valeur du *paramètre d'élimination.*

Le modèle déterministe tente de capturer le comportement apparent dans les grandes populations. Le modèle stochastique est utilisable dans les populations plus réduites, mais peut êre approprié à d'autres utilisation dépendament de l'utilisation du modèle. Enfin, un modèle stochastique à temps discret peut être construit à partir du modèle continu, en considérant la probabilité d'infection pour remplacer la valeur de β, et la probabilité d'élimination comme remplacement de γ. Grâce à cela, le nombre de nouvelles infections et le nombre d'éliminations à chaque étape suivra une distribution binomiale. Dans les simulations, les concepteurs ont inclus les deux types de modèles : le modèle stochastique, et le modèle déterministe.

Après avoir présenté le modèle général de simulation, on passe à l'exposé des méthodes d'extractions de données, utilisées par les concepteurs pour déduire les paramètres de leur modèle à partir des données réelles. En effet, nombre des paramètres déterminant le comportement du modèle possèdent des valeurs impossibles à extrapoler autrement que par l'étude du phénomène réel. Parmi cela, on sitera :

- les paramètres de propagation du ver sur Internet C'est à dire les valeurs réelles de β et γ dans le cas du modèle déterministe, et les valeurs des probabilités d'infection et d'élimination dans le cas du modèle stochastique.
- les paramètres du modèle de propagation du ver à travers un modèle stratifié du réseau (Décomposition du réseau en unités indépendantes, et la manière dont l'infection se répend à travers ce modèle).
- la relation entre distribution de l'espace IP parmi les ASs et le nombre d'hôtes potentiels dans ces ASs
- la topologie des interconnexions entre ASs et les intreconnexions au sein des ASs
- les paramètres du traffic généré par les *scans* émis par les vers
- les paramètres déterminant la charge du routeur en focntion de l'activité des Vers

4.7.2 Paramètres de la propagation global du Ver

En utilisant un modèle d'infection SI (c'est à dire sans possibilité d'élimination), pour décrire le nombre *cumulé* d'infections durant le temps, les concepteurs du simulateur ont pu inferer les valeurs des paramètres de l'infection. Mais il leur a fallu d'abord, modifier leur modèle de façon à prendre en compte l'heure d'arrêt des scans du VER CRv2 (heure ou il a commencé sa phase d'attaques DDoS), et à considérer les machines *patchées* et éteintes comme éliminées. Ils ont conçu un modèle d'élimination des hôtes, approximativement linéaire (assez proche de l'observation des données), mais qui ne convient pas au modèle SIR d'infection (l'infection est exponentielle, ainsi que l'élimination). Il fallu,donc, modifier leur modèle pour y inclure une fonction de désactivation des hôtes $\gamma(t)$:

$$\frac{ds(t)}{dt} = -\beta s(t)i(t) \tag{4.4}$$

$$\frac{di(t)}{dt} = \beta s(t)i(t) - \gamma(t) \tag{4.5}$$

$$\frac{dr(t)}{dt} = \gamma(t) \tag{4.6}$$

ou : $\gamma(t) = 0$ pour $t < 15:00$ et $\gamma(t) = 18750 hotes/heure$ pour $15:00 < t < 23:00$ (15 :00 et 23 :00 sont les heures de début de désactivation et de fin de désactivation des hôtes, et le taux à été ainsi obtenu, car à la fin de la période de désactivation, 150000 hôtes étaient désactivés). Il de plus utilisé le résultat obtenu, lors d'un précédante étude du Ver, selon lequel $\beta = 1.6/N$, $N = 360000$ (nombre d'hôtes infestés) et $T = 16 heures$ (durée totale de l'attaque). En résolvant ce système, on observe que le nombre d'infections simultanées augmente jusqu'à 300.000 hôtes avant commencer à décliner.

4.7.3 Distribution des infections dans l'espace des hôtes

A partir des données du Projet Route Views [Rou], ont déduit (avec précautions) la distribution des adresses IP parmi les ASs de l'Internet. L'étude des données de l'infection du Ver CodeRed II (ver différent de CRv2, mais utilisant le même type de vulnérabilité), à montré un petit facteur de correlation (r=0.265), entre le nombre d'adresses annoncées par un AS, et le nombre d'hôtes infestés dans cet AS. Les données furent décorrelées en considérant la fraction d'adresses IP infestées dans chaque AS. Ensuite, chaque AS se voit attribué un nombre d'hôtes, et en utilisant la distribution des fractions, on lui attribue un nombre d'hôtes susceptibles d'être infectés.

4.7.4 Topologie d'ASs

La topologie utilisée est une topologie déduite à partir des tables BGP publiques. Cette topologie est une bonne approximation de l'Internet à simuler. Elle est peut-être moins précise qu'une topologie de routeurs, mais possède l'avantage d'être facilement obtenue. Le graph obtenu contient 12921 ASs et 25297 liens. Cette topologie a subie une réduction par fusion d'ASs ne possédant pas le mêmes peers, puis un élaguage de liens de sorte que le nombre moyen de liens reste approximativement constant. Cette technique permet de produire des topologies de différentes tailles, permettant différentes applications. Les ASs ont été abstrait à l'extrême : chacun d'eux ne contient qu'un seul routeur BGP ; permettant ainsi une sensibilisation aux crashs dictée par la connectivité du routeur (les plus connecté des routeurs seront moins sensibles que les autres).

4.7.5 Traffic de *scan* des Vers

En effectuant de petites expériences sur une machine dotée du logiciel infecté, les chercheurs ont tenté de déterminer le rythme d'envoi des *scans* par le ver infectant une machine. Ces expériences tendent à prouver que le ver ne peut envoyer plus de 4.75 sondes par seconde. Après exploration des causes possibles de cette limitation, il s'est avéré que ce rythme est borné du fait dûne limitation temporelle présente sur l'implémentation de TCP sur le système.

4.7.6 Charge des routeurs et activité des Vers

Pour l'instant, aucune explication n'a été trouvée pour la correlation existant entre l'augmentation du traffic utilisateur et l'augmentation du traffic BGP sur le réseau. Pour cette raisons, les concepteurs du simulateur ont conçu un modèle d'utilisation du CPU, basé une corrélation découverte entre l'utilisation CPU et le taux d'envoi des scans du Ver. L'utilisation CPU est modélisée en tant que fonction du traffic totale transitant par le routeur (modélisé de façon probabiliste), de la charge du routeur et du nombre de scans émis localement par le ver vers des destination externes. Les scans provenant de l'exterieur d'un AS sont peu nombreux et n'ont pas été pris en considération. Le nombre de cycles requis pour traiter la somme de travail est calculée, et le taux d'utilisation du CPU est calculé à partir de cette valeur. En utilisant des fonction aléatoire, les concepteurs ont aussi dicté le comportement des routeurs face à l'augmentation de la charge : au dessus d'un certain seuil, la probabilité de crash augmente considérablement, jusqu'à atteindre une valeur maximale. Ces fonction aléatoires disposent de plusieurs paramètres permettant de contrôler les circonstances du stress des routeurs.

4.7.7 Expérimentations

Les tests préliminaires sur le simulateur ont montré une extraordinaire consommation de la mémoire, et ce même pour des topologies relativement petites : la topologie à 152 routeurs OSPF et 51 routeurs BGP à requis 8 Moctets de mémoire supplémentaire, lors de l'augmentation du nombre d'hôtes de 329 à 686. Cela suggère, si la progression est linéaire, l'utilisation de 6Goctets de mémoire pour 300000 hôtes, en plus de la mémoire nécessaire à l'établissment des routeurs BGP. Au vu de cette énorme consommation, et l'énorme demande en temps CPU des simulations (30 heures pour les plus complexes), il devient important de ce diriger vers l'exploitation de la simulation parallèle. Mais, les résultats obtenus sont, tout de même, encourageants, en comparaison des résultats de la première méthode de simulation évoquée par les chercheurs (un simulateur complètement dirigé par des échanges de paquets, sans utiliser aucune forme d'abstraction pour la progression du Ver).

4.8 L'émulateur de BGP

Cet "*émulateur*" a été présenté dans l'article intitulé "*A model of BGP Routing for Network Engineering*". Les concepteurs ont décidé de concevoir un émulateur du processus de décision de BGP au sein d'un seul AS. Cette décision est motivée par les exigences des nouveaux environnements : l'efficacité et la fiabilité. Les administrateurs de tels systèmes ne peuvent pas se permettrent de tester des modifications à leurs systèmes, sans posséder l'assurance que ces modifications ne provoqueront pas des effets désastreux. Pour cela, ils doivent posséder un procédé d'expérimentation qui puisse fournir rapidement des prédictions fiable basées sur un modèle précis des protocoles de routage. Le but de ce projet est pouvoir observer l'impact des modifications de configuration de l'AS sur les tables de routage, sans avoir à déployer les nouvelles configurations sur le système réel. Ainsi, cet émulateur peut parfaitement convenir aux tâches de *traffic engineering* : estimer les effets de certaines actions (changement de politique de routage, par exemple) sur le routage au sein de l'AS, évaluer les valeurs de paramètres critiques du système, prédire les réactions du système face à certains évènements (procédures de maintenance, congestion du réseau, dégradation du matériel,...). Dans le passé, de tes outils ont été surtout construits autour des protocoles IGP présents dans l'AS. Mais, pour les grands domaines, appartenant aux ISP par exemple, les flots de traffic au sein de l'AS dépendent, aussi, grandement du routage *inter-domaine*. C'est pour cela, que cet émulateur comprend un modèle qui détermine **le résultat du processus de décision de BGP dans chaque routeur BGP de l'AS**, sans pour autant simuler l'envoi des messages BGP. En effet, l'objectif n'est pas d'observer le comportement dynamique de BGP, mais plutôt d'examiner le résultat final du processus de décision. Il n'est, donc, pas nécessaire de modéliser les échanges (messages) ni la dimension temporelle du protocole (timers, délais, ...).

4.8.1 Modélisation

Comme cité dans la section précédante, l'objectif de l'émulateur est de produire l'état stable des routeurs BGP d'un AS (le contenu de leurs tables de routage) quand des routes sont acquises de l'exterieur de l'AS. Il faut, donc, produire pour chaque point d'entrée, et pour chaque préfixe, le point de sortie sélectionné par le processus BGP. L'émulateur doit fonctionner pour des configurations telles que celle montrée dans la figure 4.8. Il doit alors, prédire précisément la route sélectionné par chaque routeur pour chacun des préfixes vers lequel l'AS reçoit une route. L'émulateur doit produire les informations présentes dans chaque routeur à l'issue de la stabilisation du réseau, sans

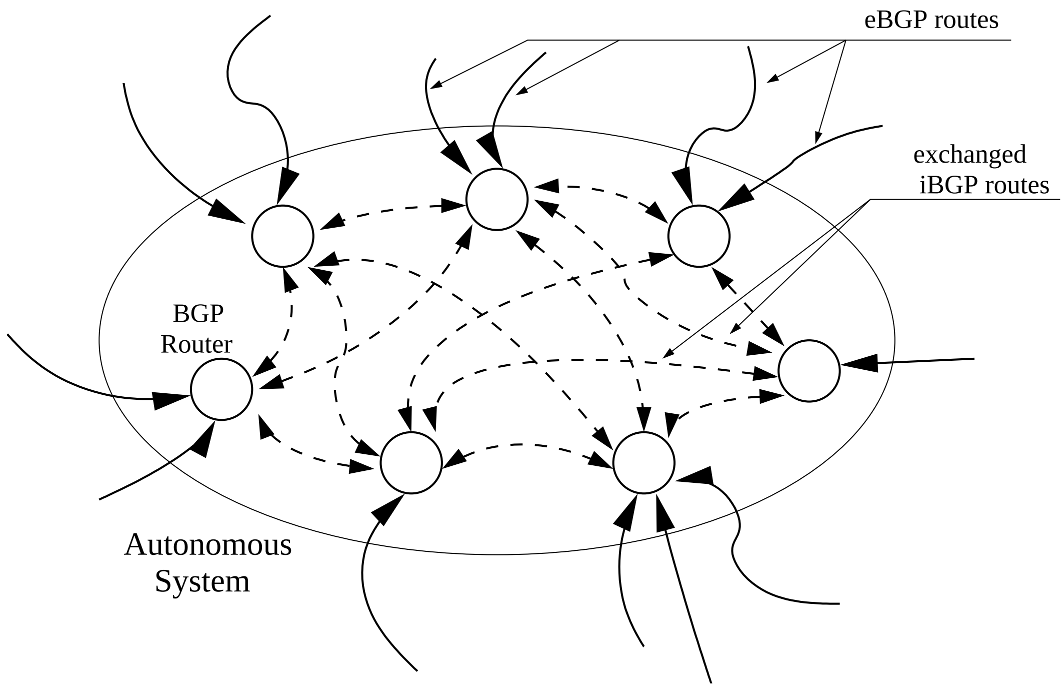

Fig. 4.8 – Système à modéliser dans l'émulateur de BGP

pour cela simuler le mécanisme d'échange de messages de BGP. Il considère pour cela, l'ensemble des routes apprises par des sessions eBGP de l'AS, le graphe iBGP des routeurs au sein de l'AS, et les coûts des chemins entre routeurs, afin de produire les meilleurs routes vers toutes les destinations, et ce pour tous les routeurs. Avant de commencer l'exposé du modèle de l'émulateur, il faut présenter certaines contraintes utilisées lors de l'élaboration de l'émulateur. Ces contraintes permettent de résoudre certains problèmes rencontrées dans la phase de conception de l'émulateur. Les trois contraintes en question sont :

1. *Contrainte 1* : Les routes apprises par eBGP ne doivent changer que lentement par rapport aux décisions de *network engineering*. En effet, si les routes apprises de l'exterieur de l'AS changent trop fréquement, le système de routage interne n'aura pas le temps de propager les informations de routage d'une annonce avant que la suivante n'arrive.

2. *Contrainte 2* : Il doit exister un chemin de signalisation iBGP valide entre chaque paire de routeurs BGP.Dans un AS ou les sessions iBGP forment une clique de routeurs, cette condition est remplie. Dans les ASs disposant de réflecteurs de routes, il est impératif que chaque rou-

teurs puisse être informé des routes apprises des autres routeurs, et ce quelle que soit la position des deux routeurs concernés (clients ou non d'un RR).

3. *Contrainte 3* : Dans les ASs disposant de réflecteurs de routes, les routeurs n'apprenent de routes que de leurs voisins BGP *immédiats*. Ainsi, des différences de choix de routes chez certaines routeurs peuvent causer des oscillations. Pour éviter cela, le réseau doit satisfaire les conditions suivantes : *(a)un routeur doit préferer les routes apprises de ses clients par rapport aux autres routes, (b) le graphe de signalisation ne doit pas contenir de cycle de relations (RR — Client du RR), (c)le plus court chemin IGP entre chaque paire de routeurs est un chemin de signalisation valide.*

Grâce à ces trois contraintes, il devient possible de modéliser la distribution des routes au sein d'un AS. Cependant, le modèle ne peut se contanter de simplement appliquer le processus de décision dans chaque routeur indépendament. En effet, les routes sélectionnées dans certains routeurs affectent les routes disponibles dans d'autres routeurs. Mais grâce aux contraintes 1 et 3 qui assurent l'unicité de l'état stable des routeurs, la solution du système (l'état stable des tables de routage) devient indépendant de l'ordre dans lequel les routeurs s'échangent leurs routes. A partir de cela, on consoit un algorithme de prédiction de route découpé en *trois phases distinctes*, comme le montre la figure 4.9.

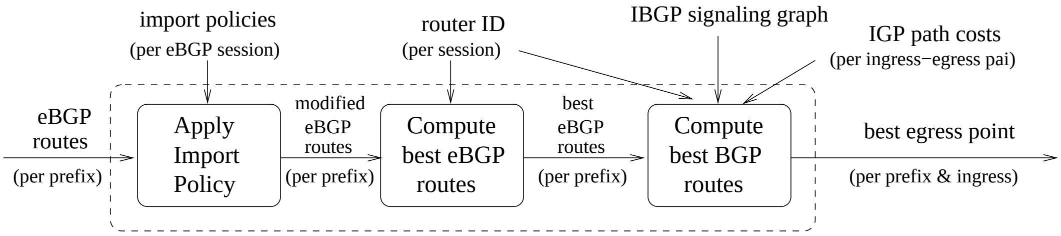

FIG. 4.9 – Décomposition de la selection de route BGP dans tout le réseau, en trois phases

Phase 1 : Application des politiques d'importation

La première phase de l'algorithme suppose que chaque routeur reçoit toutes ses routes eBGP et y applique les politique d'importation, avant tout échange de message. Chaque route apprise par eBGP dispose d'attributs

(AS_PATH, MED, ...) et est associée à une session eBGP. La politique d'importation peut filtrer les routes en fonction de la valeur de certains attributs ou positionner certains de ces attributs. Les routes ainsi filtrées et modifiées forment l'entrée de la seconde phase.

Phase 2 : Calcul des l'ensemble des meilleurs routes eBGP

La seconde phase de l'algorithme débute avec un ensemble complet de routes eBGP et produit l'ensemble des meilleurs routes eBGP candidates ; les meilleurs routes choisies par les routeurs sont des éléments de cet ensemble. Ainsi, la seconde phase de l'algorithme doit satisfaire aux deux conditions :

1. Cette phase doit éliminer toute route apprise par eBGP qui ne peut être une meilleur route dans aucun des routeurs de l'AS.
2. A la fin de cette phase, chaque router BGP ne doit contribuer que d'au plus une route à l'ensemble. Cela est dû à la propriété qu'a BGP de ne propager que la meiller route vers les peers d'un routeur.

Cette phase fonctionne par *élimination des routes de l'ensemble des routes apprises par eBGP*. Les routes restantes constituront, alors, l'ensemble des meilleurs candidates. Le processus de décision utilisé ici, fonctionne par exclusion des routes de l'ensemble, selon les étapes suivantes :

1. `Plus haute préférence locale.`
2. `Plus court chemin d'ASs.`
3. `Plus petit valeur de l'attribut` *`Origin Type`*`.`
4. `Plus basse valeur du MED (avec le même AS de prochain saut).`
5. `Favoriser les routes apprises par eBGP face aux routes apprises par iBGP.`
6. `Plus petit coût du chemin IGP vers le point de sortie.`
7. `Plus petit ID BGP.`

En fait, dans cette phase, l'élimination des routes utilise tout d'abord les trois premières étapes du processus de décision. L'algorithme applique *globalement* ces trois étapes sur l'ensemble des routes candidates. Il n'est malheureusement pas possible d'appliquer le reste des étapes du processus de décision, ni globalement, ni localement (au niveau de chaque routeur), car l'interaction entre les attributs MED et ID routeur empêchent l'émergence d'un ordre consistent des routes, au niveau de chaque routeur. Afin de gérer ces interactions, l'algorithme détermine les effets des étapes 4 à 7 du processus de décision par l'émulation d'un ordre de propagation de messages particulier. Cette partie de l'algorithme peut être écrite ainsi : `Tant qu'il reste de` `routeurs eBGP avec des routes apprises par eBGP qui ne sont ni éliminés ni marqués comme meilleurs routes :`

1. `Sélectionner n'importe quelle routeur eBGP qui possède au moins une route candidate et considérer la route qui est` *`localement`* `la meilleur route parmi les routes candidates restantes.`
2. `éliminer cette route si elle possède une valeur MED supérieur et le même AS de prochain saut qu'une route localement considéré comme la meilleur dans un autre router. Sinon, éliminer toutes les autres routes dans ce routeur, ainsi que toutes les autres routes possèdant le même AS comme prochain saut et une valeur supérieure du MED. Marquer la route restante dans ce routeur comme étant la meilleur route.`

Phase 3 : Calcul des meilleurs routes pour chaque routeur

Dans cette phase, chaque routeur sélectionne une seule route de l'ensemble des meilleurs routes eBGP. Chaque routeur qui contribue à cette ensemble, sélectionne sa propre route eBGP. Les autres routeurs devront choisir l'une de ces routes propagées via iBGP. Cette sélection est fonction des coûts IGP des chemins entre routeurs et du graphe de signalisation iBGP (graphe tracé par les sessions iBGP établies entre routeurs de l'AS). Dans une configuration ou les sessions iBGP forment une clique, la solution à ce problème est assez simple : pour chaque routeur qui ne possède pas de meilleur route apprise par eBGP, sélectionner le point de sortie le plus proche (selon le graphe IGP), en évitant les égalités par les IDs des routeurs. Cependant, dans les configurations utilisant des réflecteurs de routes, il n'est pas possible de procéder ainsi. En effet, les alternatives présentées à un routeur dépendent de sa position dans la hiérarchie des réflecteurs de routes. Afin de prendre en compte ces dépendances, l'algorithme émule un ordre particulier de transfers des messages BGP. Premièrement, l'algorithme définit au sein du graphe des routeurs reliés par des sessions iBGP, un sens pour les arcs. Ainsi, toute session entre RR définit deux arcs *over*, chacun partant de l'un des RRs pour rejoindre l'autre. De plus, chaque session entre un RR et l'un de ses clients définit deux arcs : un arc *up* allant du client vers le RR, et un arc *down* allant du RR vers le client. À partir de là, l'algorithme opère en définissant deux ordres partiels de visite des routeurs : il faut, tout d'abord, effectuer une visite des routeurs (et une propagation des routes) dans le sens des arcs *up*. Puis la visite des routeurs restants dans les sens *down* permet de s'assurer que chaque routeur possède toutes les routes qu'il lui faut pour procéder à sa sélection de route. Considerer les routeurs dans cet ordre particulier permet de s'assurer qu'aucun routeur ne prendra une décision qui devra changer par la suite. À l'issue de cette phase, l'algorithme aura correctement prédit le routeur de sortie sélectionné par chaque routeur d'entrée, et cela sans proceder à la simulation des échanges de messages BGP.

4.8.2 Implémentation de l'émulateur

Données en entrées de l'émulateur

L'émulateur utilise trois type d'entrées :

- *Les tables de routage BGP* : Les tables de routage des routeurs possédants des sessions eBGP constituent l'entrée de la première phase de l'algorithme. Les données des routeurs à propos des meilleurs routes, ainsi que des préférences locales, sont ignorés.
- *Les fichiers de configurations des routeurs* : Ces fichiers sont utilisés pour déterminer : les politiques d'importation pour chaque session eBGP, le graphe de signalisation iBGP et les coùts IGP entre chaque paire de routeurs.
- *Les informations de voisinage BGP* : Puisque le processus de décision BGP utilise l'ID des routeurs associé avec chaque session annonçant une route, l'algorithme a besoins de connaître ces IDs.

Mise en œuvre

L'émulateur est conçu pour fonctionner, et effectuer directement des opérations sur des données rassemblées sous forme d'une base de données. Cette conception autorise l'accès aux données intermédiaires des opérations, ce qui permet la validation chaque partie de l'algorithme indépendament. Les données en entrées de l'émulateur, ainsi que les résultats intermédiaires de la prédiction sont stoquées dans des tables de la base de données. Ces tables sont :

- la table **Known Routes** contient les routes apprises par les routeurs grâce à des sessions eBGP. Chaque ligne de cette table contient une entrée d'une table de routage d'un routeur possédant des sessions eBGP (préfixe IP, routeur connaissant cette route, l'adresse IP de *next-hop*, AS path,...).
- la table **Import** contient les politiques d'importation des routeurs. Chaque ligne de cette table contient un nom de routeur, l'adresse IP du voisin BGP, une expression régulière définissant des AS path, et des cases correspondants aux autres attributs d'une route BGP (localpref, ...).

- la table **MODIFIED ROUTES** contient les routes modifiées à l'issue de la première phase de l'algorithme.
- la table **Router ID** contient l'ID routeur pour chaque session BGP chez chaque routeur.
- la table **EGRESS POINTS** contient les "meilleurs routes" générées par la phase 2 de l'algorithme.
- la table **RR Clients** represente le graphe de signalisation iBGP (le graphe construit par les routeurs BGP et les sessions iBGP établis entre eux).
- la table **IGP Path Costs** contient le plus court chemin IGP entre chaque paire de routeurs dans l'AS.
- la table **PREDICTED ROUTES** contient le résultat final de l'algorithme : la routes sélectionnée par chaque routeur, ainsi que le routeur de NextHop choisit par chacun de ces routeurs.

L'émulateur fonctionne par l'exécution de trois opérations distinctes sur la base de données, corespondants aux trois phases de l'algorithme :

Calcul des routes modifiées La première opération applique les politiques d'importation aux routes eBGP. Pour chaque ligne de la table **Import**, la politique correspondante est appliquée en sélectionnant toutes les routes apprises par ce routeur sur cette session (ligne de la table **Known Routes**), qui correspondent à l'expression régulière définie dans la politique. Puis Les attributs de ces routes sont positionnés selon les valeurs dans la ligne de la table **Import**. Les routes ainsi modifiée sont mises dans la table **MODIFIED ROUTES**.

Calcul des points(routeurs) de sorties La seconde opération applique la technique de la section 4.1.2 pour générer l'ensemble des meilleurs routes eBGP. Cela est réalisé par l'application successives de `Select` sur la table **MODIFIED ROUTES** pour rafiner l'ensemble des routes candidates. La table **Router ID** est utilisé lors de ces opérations pour rompre les égalités (étape 7 du processus de décision). Les meilleurs routes, ainsi sélectionnées, sont placées dans la table **EGRESS POINTS**.

Calcul des routes prédites La troisième opération utilise les meilleurs routes sélectionnées par les routeurs eBGP (la table **EGRESS POINTS**), le graphe de signalisation iBGP (la table **RR Clients**), les coûts IGP des chemins entre routeurs (la table **IGP Path Costs**) et la technique de la section 4.1.3 pour déterminer la meilleur route BGP pour chaque préfixe dans chaque routeur. De plus, cette opération utilise les ID routeurs de chaque session iBGP (la table **Router ID**) afin d'associer, à partir de ces meilleurs routes, un routeur de nexthop à chaque routeur, pour chaque destination.

4.8.3 Validation de l'émulateur

Afin de vérifier la correction des prédictions produites par le simulateur, il a été nécessaire de comparer les résultats obtenus par l'émulateur aux résultats produits par des routeurs réels dans un réseau opérationnel. Cela a été effectuée en présentant à l'émulateur des *snapshots* individuels des fichiers de configurations des routeurs, des tables de routage BGP et des informations de voisinage BGP. Ensuite, les prédictions de l'émulateur ont été comparées à ce qui a été observé dans la réalité. Afin d'isoler les probables sources d'inexatitude, chaque phase de l'algorithme a été évaluée indépendament, en supposant que les sources des phases précedantes sont correctes. D'un point de vue global, l'émulateur produit des résultats correctes pour plus de 99% des préfixes. La plupart de erreurs peuvent être expliquées par la dynamique des données au niveau des routeurs. Par exemple, si une route est temporairement rejetée au moment, ou la collecte des informations est effectuée, des inconsistences apparaissent entre les prédicatins et les résultats de la configuration réelle.

Chapitre 5

L'Architecture proposée du Simulateur des Instabilités de BGP

Ce chapitre présente la proposition d'un modèle de simulation des instabilités de BGP. A travers une abstraction sélective des éléments du protocole à implémenter et une utilisation d'éléments provenant du système réel, ce modèle a pour ambition de produire, lors de son implémentation, les même types d'instabilités que ceux observés sur Internet, et dans les mêmes proportions. Il faudra en même temps que l'implémentation de BGP soit assez complète de façon à inclure le maximum des éléments actuellements déployés sur les routeurs BGP. En effet, l'étude des instabilités de BGP, puis l'éventuelle implémentation de nouvelles méthodes de correction, ne peut être effectuée que si le modèle de simulation est assez proche du modèle réel pour justifier les essais sur simulateur.

A travers l'étude des instabilités de BGP et des différents simulateurs de BGP construits jusqu'à aujourd'hui, il est apparu qu'un modèle de simulation de BGP ne pouvait reproduire les mêmes comportements que ceux observés sur le réseau réel, qu'à condition de :

1. Disposer d'un modèle de *speaker BGP* le plus fidèle possible aux implémentations BGP réelles.
2. Utiliser un modèle de système autonome plus élaboré que celui largement utilisé dans les simulations : chaque AS ne contenant qu'un unique routeur BGP, ce qui permet d'ignorer les interactions EBGP-IBGP.
3. Employer une topologie de tests la plus proche possible de la topologie réelle de l'Internet.

4. Disposer de politiques de routages s'apparentant au maximum aux politiques actuellement usités entre ISPs sur Internet.

5. Mettre en œuvre des retards dûs aux attentes de CPU et aux traversées des liens, qui soient le plus proche possible des délais réellement imposée aux paquets BGP sur Internet.

Le premier objectif de cette liste peut être atteint par une la réalisation d'une abstraction du protocole BGP, tel qu'il est décrit dans la spécification. Cette abstraction signifira l'élimination ou la simplification des éléments du protocole qui ne sont pas indispensable à la reproduction de l'instabilité du routage inter-domaine. D'autres éléments seront repris tel quel, ou légèrement remaniés pour pouvoir convenir à une l'implémentation du simulateur. La section 5.1 expose le modèle de speakers BGP proposé pour le simulateur.

Les deux points suivants de cette liste sont intimement liés à l'obtention d'une topologie réaliste de l'Internet. La section 5.3 présente une alternative aux traditionnels topologies obtenues par les *logs* BGP. Elle expose aussi une méthode d'obtention de cette topologie. La section 5.2 présente les différentes formes d'ASs présents dans la topologie, qu'il serait intéressant d'implémenter dans le simulateur.

L'obtention de schémas de politiques de routage appliquées sur Internet est malheureusement inconcevable pour la plupart des systèmes autonomes présents sur Internet, du fait de la confidentialité entourant les contrats commercials liant les ISPs. Il a fallu, donc, identifier les *familles* de politiques les plus susceptibles d'exister dans certaines portion de l'Internet. La section 5.4 expose ces politiques, ainsi que le plan prévu pour leur déploiment dans la topologie de simulation.

Le dernier point de la liste ci-haut concerne les délais imposés aux messages BGP du fait des temps de traversée des liens physiques et des attentes d'obtention du CPU dans les routeurs BGP. La section 5.5 expose les procédés de calcul de ces délais pour chaque message, dans le modèle de simulation.

L'utilisation de topologies réalistes implique des simulations fonctionnant sur des ensembles de plusieurs milliers de nœud AS, et peut-être même quelques dizaines de milliers de nœud (speakers BGP). Pour pouvoir effectuer de telles simulations, il faut adopter une gestion des ressources (surtout les ressources mémoire et CPU) particulièrement efficace. Une approche comme celle de [HK03] qui minimise l'utilisation mémoire, ou comme celle de [Quo03] qui considère le temps CPU comme critique, constiturait une bonne tendance.

5.1 Modélisation des *speakers* BGP

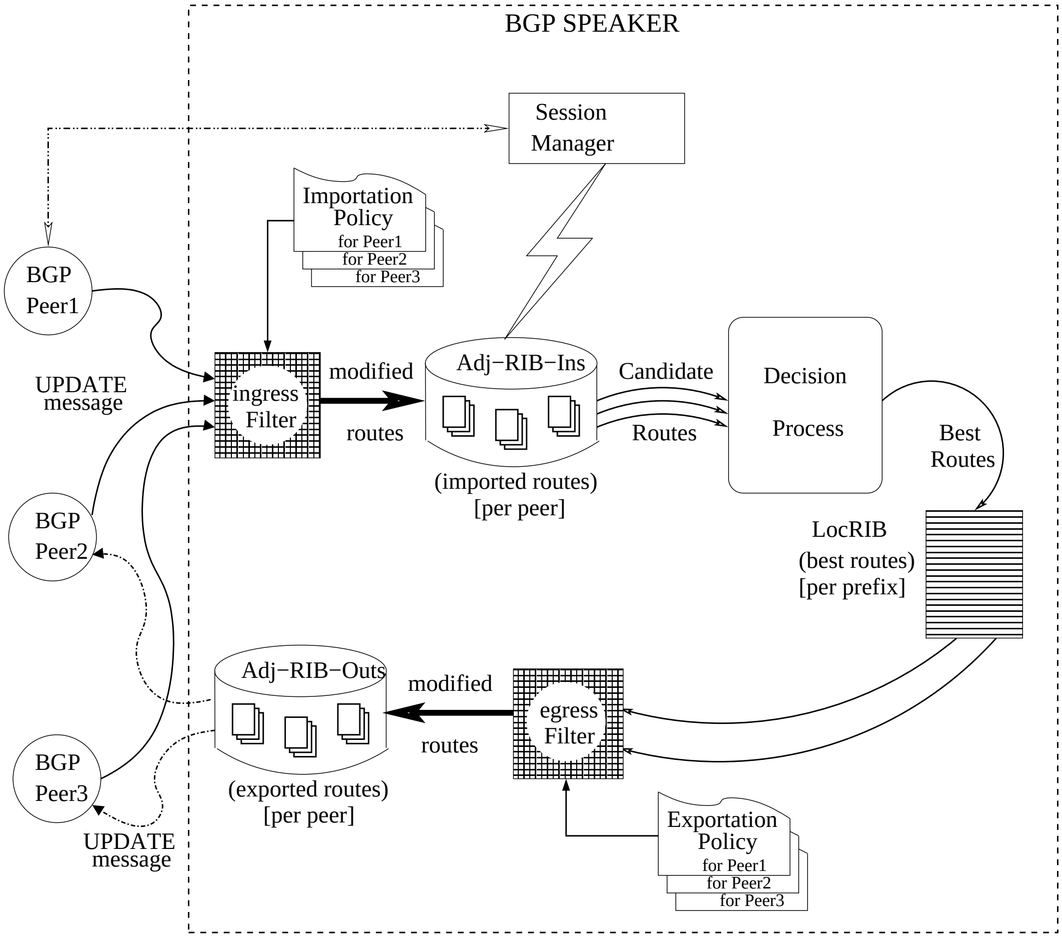

FIG. 5.1 – Schéma de conception d'un *speaker* BGP

Le premier élèment à considérer pour la conception d'un simulateur du protocole BGP, est l'architecture et le fonctionnement du nœud élémentaires des topologies de simulation : le *speaker* BGP. Le modèle du speaker BGP que nous avons adopté est assez similaire aux architectures de nombre des simulateurs étudiés. Mais il contient quelques éléments, qui sont extrêmement simplifiés ou simplement supprimés de certains simulateurs, et qui sont modélisés de manière presque intégral dans ce simulateur. La gestion des sessions BGP, en particulier, adopte dans ce simulateur un mode de fonctionnement moins complexe que celui dicté par la spécification, mais permet néanmoins de simuler les effets des ruptures de liens et des pannes de routeurs.

L'architecture proposée du speaker est montrée dans figure 5.1. Les éléments actifs composant cette architecture sont :

- Le filtre *ingress* : Ce composant est chargé de la réception des messages UPDATE provenant des peers du speaker. Il doit aussi appliquer, pour chaque message reçu d'un peer X, la politiques d'importation correspondant à ce peer. Les messages peuvent, ainsi, être rejetés, acceptés ou modifiés puis acceptés. Les messages qui ne sont pas rejetés par la politique de routage sont, ensuite, placés dans la table Adj-RIB-In correspondant à l'émetteur du message.
- Le Processus de décision : A chaque réception d'un nouveau message, le processus de décision est appelé pour déterminer la meilleur route vers la destination cité dans le message reçu. Ce processus a pour entrées les routes désignant la même destination, présents dans toutes les tables Adj-RIB-In. Ce processus et la méthode qu'il utilise pour choisir ses routes sont détaillés dans la sous-section qui suit.
- Le filtre *egress* : Ceci est le même composant que le filtre *ingrees*, mais il applique les politiques d'exporation aux routes choisies par le processus de décision et placées dans le Loc-RIB. Ainsi, il devra pour chaque nouvelle route installée dans le Loc-RIB, appliquer les politiques correspondant aux peers du speaker. Pour une certaine destination et pour un peer particulier, cette politique permet d'apporter des modifications à la route à annoncer ou d'empecher l'annonce de la route. Les routes destinées à l'exporation vers un peer, sont placées par le filtre dans l'Adj-RIB-Out correspondant au peer.
- Le gestionnaire de sessions : Ce composant s'occupe de la gestion de la liaison avec les peers du speaker. Ainsi, ce gestionnaire conserve, entre autres, pour chacun des peers connectés, un **état** de la session établie avec ce peer. La gestion de cet état n'utilise pas l'automate à états finis de BGP, mais emploi une conduite plus simple. Le fonctionnement de ce composant est expliqué plus en détail dans la sous-section 5.1.2.

Les composants du speaker communiquent par échanges de données, et utilisent aussi des informations configurées avant le début de la simulation. Ces données sont stockées dans des structures particulières :

1. Les tables Adj-RIB-In : Il doit exister une table Adj-RIB-In par peer du speaker. Ces tables servent à conserver les routes reçus des peers du speaker.
2. La table Loc-RIB : Chaque speaker dispose d'un seul Loc-RIB, qui conserve les routes choisis par le processus de décision.
3. Les Tables Adj-RIB-Out : Comme pour les Adj-RIB-Ins, à chaque peer du speaker doit correspondre une table Adj-RIB-Out, contenant les

routes à émettre vers ce peers, sous forme de messages BGP.

4. Les tables de politiques de routage : Les politiques de routage sont une information de configuration spécifiques à chaque speaker. Les tables servant à les stocker, chez un speaker, permettent l'application rapide de ces politiques aux route (importées ou exportées).

Pour tenter de simplifier le modèle, tout en conservant les éléments indispensables, ces disposition ont été décidées :

- L'utilisation d'un seul type de messages BGP, à savoir les messages UPDATE. Attendu que le gestionnaire de session (sous-section 5.1.2) administre les connexions avec les peers d'un speaker sans avoir à utiliser les messages OPEN, KEEPALIVE et NOTIFICATION, leur utilisation a été proscrite.
- L'utilisation des seuls attributs : LOCAL_PREF, AS_PATH, MED, ORIGINATOR_ID et CLUSTER_LIST. En effet :
  - Lors des simulations, toutes les création de nouvelle destination se fera par l'un des routeurs, comme si il venait de l'apprendre par un IGP. L'attribut ORIGINE n'est, donc, pas nécessaire
  - De même, les topologies de simulations ne contiendrons que les routeurs BGP ; Ils devront, eux mêmes, servir de routeurs de bordure en annonçant leur propre adresse comme NextHop. L'attribut NEXT_HOP n'est, donc, pas nécessaire.
  - Les routeurs BGP de la simulation seront dépourvus de capacités d'aggregation, du fait du nombre relativement réduit de destinations annoncées durant les simulations. Les attributs AGGREGATOR et ATOMIC_AGGREGATE sont, donc, inutiles.
  - Les confédérations d'ASs sont relativement peu utilisées par rapport à la réflexion de route. C'est pour cela, que nous avons décidé d'implémenter la réflexion de route sans implémenter les confédérations.
- L'aggregatin de routes n'étant pas utilisée, il n'est pas indispensable de disposer d'une structure complexe de l'attribut AS_PATH. En effet, il n'y a plus de besoins d'AS_SETs dans cet attribut. L'AS_PATH contient, donc, **une séquence d'ASN**.
- L'utilisation des timers MRAI (*Minimum Route Advertisment Interval*) lors des annonces de routes entre ASs. La valeur de ce timer devra être configurable et non figée dans les simulations. De plus, Les valeurs utilisées par les timers MRAI, seront variées par la multiplication de la valeur configurée par une valeur uniformement choisie dans l'interval $]0.75, 1]$.
- La spécification du protocole conseille l'utilisation d'un timer MRAI par destination exportée, pour chaque peer d'un speaker. Mais la trop

grande consommation que provoquerait une telle implémentation (dans la réalité, il y'a jusqu'à 100,000 entrées par table de routage BGP, et chaque speaker possède un nombre potentiellement grand de peers) serait trop contraignante. Pour cela, la spécification de BGP a autorisé l'utilisation d'un **seul timer MRAI par peer**. Toutes les annonces de routes destinées à un peer, quelque soit le préfixe qu'elles déclarent, seront contraintes d'attendre l'expiration du même timer. Ainsi, si le timer est positionné à cause de l'émission d'un message concernant une certaine destination, et que l'annonce d'une autre destination doit être émise, cette dernière doit attendre l'expiration du timer. Dans le simulateur proposé, l'approche utilisant un timer par peer sera celle utilisée, car c'est l'approche adoptée dans la réalité.

- L'utilisation de la limitation du rythme d'envoi des rejets de route (*Withdrawals Rate Limiting*), de la même manière que pour les annonces de routes. Cette disposition, bien qu'interdite par la spécification, est présente sur de nombreuses implémentations du protocole.
- La mise en œuvre de l'implémentation particulière : *Stateless BGP*. Cette implémentation, dont le comportement s'est révelé dommageable sur Internet, est une des sources de la multiplication du nombre de messages UPDATE circulant sur Internet. Son implémentation dans le simulateur s'effectue par la désactivation de la sauvergarde du dernier message émis vers chaque peer, pour une certaine destination.
- L'utilisation de configurations multiples des speakers BGP. Ainsi, différents speakers pourront avoir différents comportements (utilisation du *StateLess BGP*, du WRL, différentes politiques de routage, ...). Cette mesure a pour but de pouvoir observer l'impact de certains détails d'implémentations sur les routeurs que n'implémentent pas, eux mêmes, ces méthodes.

Le reste des détailles de modélisation du speaker BGP concernent le processus de décision et la manière dont il est appliqué ; et la gestion des sessions et des connexions dans le simulateur.

5.1.1 Processus de décision

Lors des simulations, les speakers s'échangeront des messages UPDATE contenant :

1. l'identité BGP de l'émetteur. Cela correspond à l'adresse IP du routeur, et tient, donc, sur 4 octets.
2. le numéro d'AS de l'émetteur (2 octets). Ceci identifira de manière unique le domaine auquel appartient l'émetteur du message.

3. un **champs de routes rejtées** correspondant à un ensemble de préfixe rejetés par l'émetteur. Cet ensemble sera sous la forme

   $$< LongeurChamps, Prefixe1, Prefixe2, \cdots >$$

   ou LongeurChamps représente la longeur en octet du champs de routes rejetées, et ou les préfixes sont sous la forme $< LongeurPrefixe, Prefixe >$. Le préfixe doit contenir assez de bits à 0 à la fin pour former un nombre entier d'octets.
4. un attribut **LOCAL_PREF**, si ce message est reçu d'un peer appartenant au même AS que le récepteur.
5. un attribut **AS_PATH** pour chaque message d'annonce de route. Sans l'aggrégation de routes, la création d'AS_SETs est impossible. Cet attribut ne contiendra qu'une suite de numéros d'AS. Cette suite sera précédée par un sous-champs contenant le nombre d'ASs contenus dans l'AS_PATH.
6. un attribut**MED** contenant sur 4 octets une valeur de différentiation pour les différentes sessions reliant un même AS à l'AS du récepteur du message.
7. un attribut**ORIGINATOR_ID** si ce message circule entre les speakers d'un même AS, et que cet AS contient des réflecteurs de routes. Cet attribut contient l'identité du speaker qui a introduit l'information de routage dans l'AS.
8. un attribut**CLUSTER_ID** si ce message circule entre les speakers d'un même AS, et que cet AS contient des réflecteurs de routes. Cet attribut contient une liste des numéros de clusters traversés par le message, afin d'éviter de le faire boucler dans l'AS.
9. Un champ de routes annoncées, correspondant au NLRI des messages UPDATE. Ce champ doit pouvoir contenir plusieurs préfixes annoncés, et aura donc, une structure similaire à celle du champs de routes rejetées.

Pour plus de simplicité, il serait intéressant de considérer toutes les informations du message comme des attributs de la route. L'identité du messager constitura l'attribut PEER_ID, son numéro d'AS sera le PEER_ASN, et les destinations annoncées formeront l'attribut NLRI.

A la réception d'un message UPDATE, et après que celui-ci soit passé par le filtre *ingress*, il doit être placé dans le Adj-RIB-In correspondant au peer qui a émis ce message. Si une information de routage différente, à propos du même préfixe, est présente dans l'Adj-RIB-In du peer en question, alors l'ancienne information est automatiquement détruite et remplacée par la nouvelle (rejet implicite). Sinon, la nouvelle information est introduite dans l'Adj-RIB-IN. De plus, si l'AS_PATH de la nouvele information contient l'ASN du speaker, alors cette information doit être supprimée (mécanisme d'évitement de cycles).

A la réception d'un message de rejet de route, l'entrée correspondant à ce rejet dans l'Adj-RIB-In du peer est supprimée.

Quand BGP détecte une modification de l'une des tables Adj-RIB-In, il doit appliquer son processus de décision pour choisir la meilleur route vers la destination sur laquelle vient de survenir une information (nouvelle route ou suppression de chemin). Ce processus se présente sous la forme d'un algorithme d'exclusion de routes appliqué sur les routes vers la même destination.

Cet algorithme a, en entrées, les entrées des tables Adj-RIb-In de tous les peers, qui correspondent à une certaine destination (routes candidates). En appliquant la suite de règles suivantes, BGP élimine des routes, jusqu'à ne plus garder qu'une seule route :

1. Éliminer toutes les routes dont la valeur de l'attribut LOCAL_PREF est strictement inférieur à la plus grande des valeur de l'attribut LOCAL_PREF parmi les routes candidates.
2. Si il demeure plus d'une route dans l'ensemble, éliminer toutes les routes dont le nombre d'ASN dans l'attribut AS_PATH est strictement inférieur à la longeur du plus long AS_PATH parmi les routes restantes.
3. Si il reste encore plus d'une route candidate, éliminer pour chaque groupe de messages provenant du même AS, les routes dont la valeur de l'attribut MED est strictement supérieur au minimum des valeurs des MEDs de ces routes - Ne pas comparer entre eux des routes provenant d'ASs différents.
4. Si il existe encore plusieurs route, et certaines de ces routes proviennent de peers internes, ne conserver parmi celles-ci que la route dont la métrique IGP d'accès vers le routeur l'ayant annoncé est le plus petite. Si plusieurs routes ont la même plus basse métrique, il faut toutes les conserver.
5. Si il reste plus d'une route candidate, choisir la route annoncée par le peer **externe** possèdant le plus petit identificateur BGP.
6. Si il reste plus d'une route candidate, choisir la route annoncée par le peer **interne** possèdant le plus petit identificateur BGP.

A l'isse de cette sélection, seule une destination peut subsister. Celle-ci est la meilleur route vers la destination ; elle est, donc, placée dans le LocRIB, et soumise au filtre *egress* pour l'exportation vers les peers.

Avant d'exporter cette route vers un peer, le speaker doit d'abord :

1. Adjoindre à l'attribut AS_PATH de la route, son propre ASN.
2. Si le peer est un peer externe, supprimer l'attribut LOCAL_PATH de la route. Si c'est un peer interne, conserver cet attribut.
3. Si la route provient d'un peer interne, et que le destinataire est interne à l'AS du speaker, celui-ci doit :
   - Si ce speaker est un reflecteur de routes, que l'origine du message est un client, et que le destinataire aussi, alors il ne fait rien.
   - Si ce speaker est un reflecteur de routes, que l'origine du message est un client, mais pas le destinataire, alors le speaker adjoint au message un attribut ORIGINATOR_ID (avec l'ID de l'émetteur) et un attribut CLUSTER_LIST (avec l'identité de son cluster).
   - Si ce speaker est un reflecteur de routes, que l'origine du message n'est pas un client, et le destinataire en est un, alors il ne fait rien.
   - Si ce speaker est un reflecteur de routes, que l'origine du message n'est pas un client, et que le destinataire non plus ne soit pas un de ses clients, alors il doit ajouter l'identité de son cluster à l'attribut CLUSTER_LIST du message.
   - Si ce speaker n'est pas un speaker, alors il ne fait rien.
4. Si la route provient d'un peer interne, et que le destinataire est externe à l'AS du speaker, celui-ci doit supprimer du message les attributs CLUSTER_LIST et ORIGINATOR_ID.

Ensuite le message est émis vers le destinataire, en y apposant l'identité du speaker et le numéro de son AS.

5.1.2 Gestion des sessions BGP

Dans ce modèle de simulation, la session BGP ne possède que deux états : **actif** et **inactif**. Au début de chaque simulation, les états de toutes les sessions sont à **actif**. Puis, l'état peut changer du fait d'évènements externes (provoqués spécifiquement pour l'étude de la stabilité), ou d'évènements internes (surcharge du CPU, augmentation de la charge du lien provoquant l'absence de messages KeepAlive pendant plus de 90 secondes, ce qui provoque la fermeture de la session).

Face à des actions externes de fermetures de session, le gestionnaire provoque la cloture de toutes les sessions du speaker. C'est à dire, que l'intégralité des routes apprises par les peers du speaker sont supprimées des tables Adj-RIB-In, puis le gestionnaire communique aux gestionnaires des peers une information de rupture de session (cela simule les effets d'un crash, ou la mise hors service d'un routeur).

Pour simuler les ruptures de sessions provoquées par une attaque ou une surcharge du réseau, le gestionnaire met en place un timer HoldTimer par session. Pour chaque session, ce timer permet de vérifier la pérennité du lien entre le speaker et son peer. Sa valeur, selon la spécification, doit être de trois fois la valeur du MRAI. Le timer HoldTimer est réinitialisé à chaque fois que le speaker reçoit un message de son peer. Si le speaker ne reçoit aucun message de son peer durant la durée d'attente du timer (qui correspond à la période maximum d'attente de réception des messages KEEPALIVE), puis que le timer expire, le gestionnaire calcul une **probabilité de rupture de la session** en fonction de la charge estimée du routeur et du taux d'occupation du lien. Si cette probabilité est trop élevée, la session est considérée comme perdue (simulation de la non-réception de messages KEEPALIVE durant trois fois la valeur du MRAI). Le gestionnaire transmet cette information au gestionnaire de session du peer afin qu'il désactive lui aussi cette session. Ensuite, le gestionnaire, provoque la suppression de toutes les routes apprises du peer dont la session a été perdue. Si cette suppression affecte le LocRIB (perte d'une route) alors le processus de décision est déclenché pour chaque destination dont la meilleur route a ainsi été invalidée.

Si le speaker est configuré pour rétablir la session en cas de rupture, le gestionnaire de sessions du speaker doit laisser s'écouler une durée de temps spécifiée par la configuration, puis informer le gestionnaire du peer de la reprise de la session, pour qu'ils se communiquent l'un l'autre, l'intégralité des routes contenues dans leurs LocRIB respectifs. Cette opération s'effectue par l'émission de messages UPDATE non retardés par l'application du timer MRAI.

5.2 Modélisation des Systèmes Autonomes

La plupart des simulateurs étudiés adoptent une même structure extrêmement simplifiée de l'intérieur des ASs : Chaque AS est constitué d'un seul speaker (qui identifie cet AS) qui est connecté aux speakers des ASs voisins . Bien que facilement réalisable, cette représentation est loin d'être réaliste, ce qui pourrait amener les simulations l'utilisant à ne pas reproduire certains comportements qu'il serait utile d'étudier (par exemple, les instabilités dues à l'attribution des valeurs des attributs MED en fonction des métriques IGP internes aux ASs). Pour cette raison, le modèle de simulation proposé adopte une architecture complexe de l'intérieur des ASs. Comme l'indique la spécification, les speakers d'un même AS communiquent grâce à des sessions IBGP, et communiquent avec les speakers externes par des sessions EBGP. Et pour augmenter le réalisme des simulations, et tenter de capturer d'autres types d'instabilité, la structure interne des ASs peut être constituée de speakers BGP connectées en clique (approche normale de la connexion IBGP), ou contenir des réflecteurs de routes constituant avec leurs clients des clusters de speakers BGP.

5.2.1 Systèmes Autonomes en *Full Mesh*

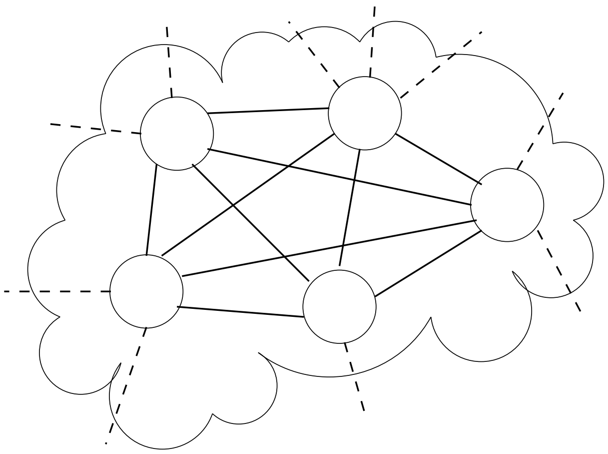

FIG. 5.2 – Les connexions IBGP dans le simulateur

Cette disposition des systèmes autonomes sera utilisée pour les domaines composées d'un *petit* nombre de speakers, pour pouvoir économiser les ressources mémoire. Chaque speaker de l'AS devra établir une session avec *chacun* des autres speakers composant l'AS. Cette disposition permettra aux speakers d'appliquer la règle d'évitement de cycles de routage dans les ASs : "un speakers qui reçoit une route de l'un de ses peers interne, *ne doit pas* la retransmettre vers ses autres peers internes, s'il la choisit comme meilleur route vers la destination spécifiée par la route". Par contre, les routes reçues des peers externes doivent être exportées vers tous les peers internes du speakers, et les routes apprises de l'intérieur de l'AS doivent être exportées vers les peers externes.

A chaque AS doit être associé un graphe de connexions IGP, qui reflète l'organisation interne de l'AS. Ce graphe pourra être utilisé pour l'attribution des valeurs de l'attribut MED des politiques utilisant la stratégie de mapping des métriques IGP à l'attribut MED, et pouriiiii l'application d'une des règles du processus de décision de BGP. Ce graphe devra uniquement spécifier la valeur des métriques IGP reliant chaque paire de routeurs d'un AS.

5.2.2 Systèmes Autonomes avec Réflexion de Routes

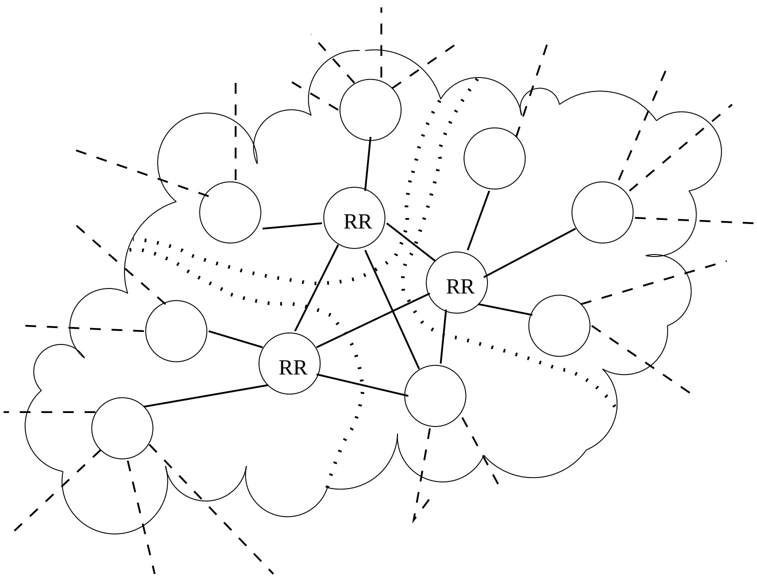

FIG. 5.3 – La réflexion de routes dans le simulateur

Cette disposition des systèmes autonomes sera utilisée pour les domaines au nombre relativement élevé de speakers, ou spécifiquement pour tester l'impact de la stratégie de mapping de MED sur les métriques IGP, sur le routage inter-domaine. Dans ce type d'ASs, les routeurs appartiennent à l'une des deux catégories :

1. la première catégorie comprend les routeur BGP appliquant l'algorithme IBGP décrit ci-haut. Ces routeurs, sont ,pour la plupart, les clients de réflecteurs de route ; mais ils peuvent très bien n'appartenir à aucun cluster. Un client de réflecteur est uniquement connecté à ce réflecteur.
2. La seconde catégorie comprend le réflecteurs de routes. Ces routeurs établissent des sessions IBGP avec leurs clients (routeurs membres de leur cluster), ainsi qu'avec tous les routeurs qui ne sont pas des clients d'autres réflecteurs de routes. Chaque réflecteur de route applique un algorithme de distribution interne des routes un peu particulier :

- quand un réflecteur apprend une route de l'un de ses clients il l'annonce à ces autres clients, ainsi qu'à l'ensemble des routeurs qui ne sont pas ses clients.
- quand un réflecteur apprend une route d'un routeur non clients, il l'annonces à l'ensemble de ses clients.
- quand un réflecteur apprend une route d'un peer externe, il la propage vers tous les peers internes auquels il est connecté.

De même que pour les AS en *Full Mesh*, chaque AS utilisant la réflexion de routes doit disposer d'un graph IGP reliant les routeurs BGP, spécifiant les couts IGP de communication entre chaque couple de routeurs.

5.3 Topologies de sessions BGP

5.3.1 Topologies existantes et Topologies requises

La production de simulations reproduisant, au plus près, le comportement du routage inter-domaine réel, ne peu se faire, sans l'utilisation de topologies aussi réalistes que possible. L'obtention de telles topologies passe, souvent, par le traitement de données obtenues de routeurs sur Internet, ou par des méthodes de mesures actives. Et parce que, aucune méhode actuelle, ne peut produire des topologies montrant exactement les speakers BGP de l'Internet et leurs sessions, nous allons utiliser une méthode d'inférence pour produire un **graph de sessions BGP** à partir de **graph d'ASs** et de **graphs de routeurs**.

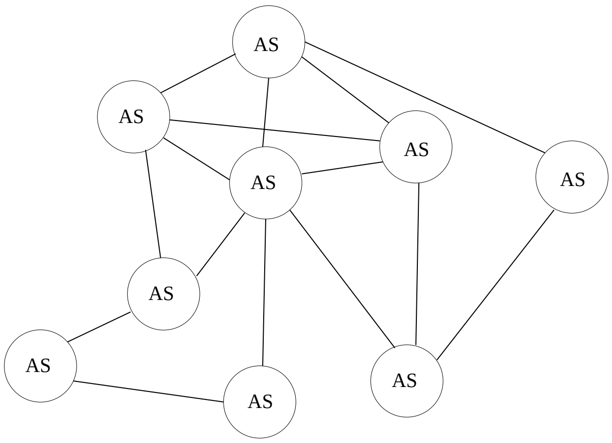

FIG. 5.4 – Schéma de Topologie d'ASs extraite de *logs* BGP

La figure 5.4 montre un exemple de **graph d'ASs**. Ce type de topologies est obtenu par l'analyse de *logs* de routeurs BGP disponibles sur le Web [Rou]. Par exemple, d'un fragment de *log* comme celui-ci :

```
   Network  Next Hop         Metric LocPrf Weight Path
*> 4.0.0.0  134.24.127.3                        0 1740 1 i
*           194.68.130.254                      0 5459 5413 1 i
*           158.43.133.48                       0 1849 702 701 1 i
```

On pourra déduire le présence d'au moins un lien BGP entre les membres de chacun des couples (5459, 5413), (1849, 702),(5413, 1), ... L'analyse de *logs* BGP, obtenus de plusieurs point de l'Internet, fournira une carte presque complète des liens entre ASs ([JSW] recense plusieurs de ces *logs* BGP, ainsi que les topologies déduites de ces *logs*).

Si le simulateur était conçu pour fonctionner avec des ASs *simplifiées* (un seul routeur BGP par AS), ce type de topologie conviendrait parfaitement. Il suffirait, en effet, de connecter le speaker de chaque AS avec les speakers des ASs qui sont liés à lui. Mais, notre modèle de simulation comprend des architectures complexes de l'intérieur des ASs, ce qui permet des combinaisons bien plus complexes. Par exemple, deux ASs pourrons posséder plus d'une session BGP entre eux, du moment que les couples définissant ces sessions soient différents. Ce type de configurations est répandu sur l'Internet, mais aucune simulation à large échelle ne l'a pris en compte (à notre connaissance).

Les topologies les plus réalistes, à produire seraient, donc, des **topologies de sessions BGP** (voir la figure 5.5). Ce type de topologies prendraient en compte la présence de multiples liens entres les ASs, et permetterai la simulation d'un certain type d'instabilité (instabilité produite par les politiques des mapping des MED à partir des métriques IGP internes aux ASs).

Pour produire ce type de topologies, il faut :

- posséder une information sur la connectivité, au niveau routeur, des ASs de l'Internet. Cette information sera utilisée pour attribuer des poids aux liens de la topologie d'ASs.
- ou, assigner aléatoirement des poids aux liens entre ASs dans une topologie d'ASs, pour pouvoir ensuite construire une topologie de sessions.

La figure 5.6 montre un exemple du type de topologies qui peut être obtenus de l'Internet, et qui fournirait l'information voulue sur la connectivité au niveau routeur. Cette topologie est produite par une série de mesures sur Internet. Grâce à elles, on peut déduire la structure des routeurs existants. La méthode utilisée peut, par exemple, effectuer des *traceroute* vers des adresses de destination aléatoirement choisies. Chaque *traceroute* fournit une information, sous la forme des adresses des routeurs successivement parcourus par le paquet de *traceroute*. La compilation des données obtenues de

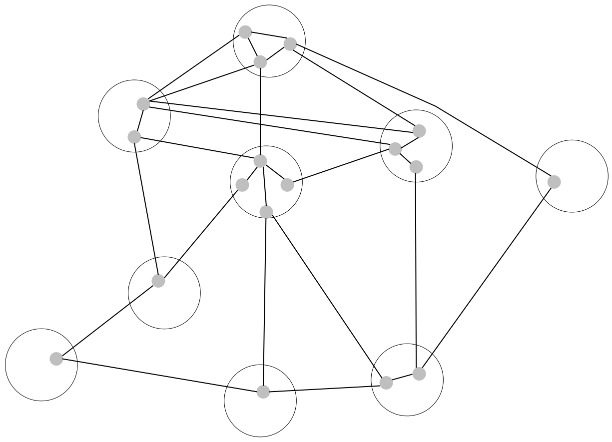

FIG. 5.5 – Schéma de Topologie de Sessions BGP

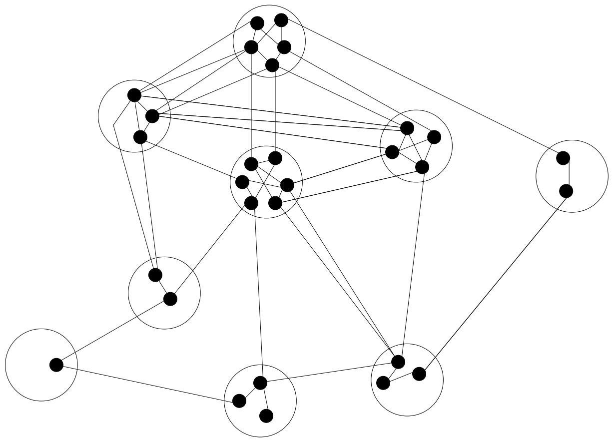

FIG. 5.6 – Schéma de Topologie de routeurs obtenue par des *traceroute*

plusieurs milliers de *traceroute* fournit le type de topologies exposé dans la figure5.6. [CB] et [fIDAC] présentent deux projets de *mapping* de l'Internet au niveau routeur. A partir d'une telle topologie, il faut d'abord aquerir une carte des routeurs, ou chaque route est assigné à son AS. Une méthode pour obtenir ce type de mapping a été produite dans [CJW01]. On supposera, donc, que nous possédons cette carte de routeurs associés à leurs ASs. A partir de cette carte, on assigne un poids à chaque lien de la topologie d'ASs. Ce poids correspondra, pour l'instant, au nombre de liens dans la topologie de routeurs, qui existent entre chaque couple d'ASs. On obtient le type de topologies dépeinte dans la figure 5.7.

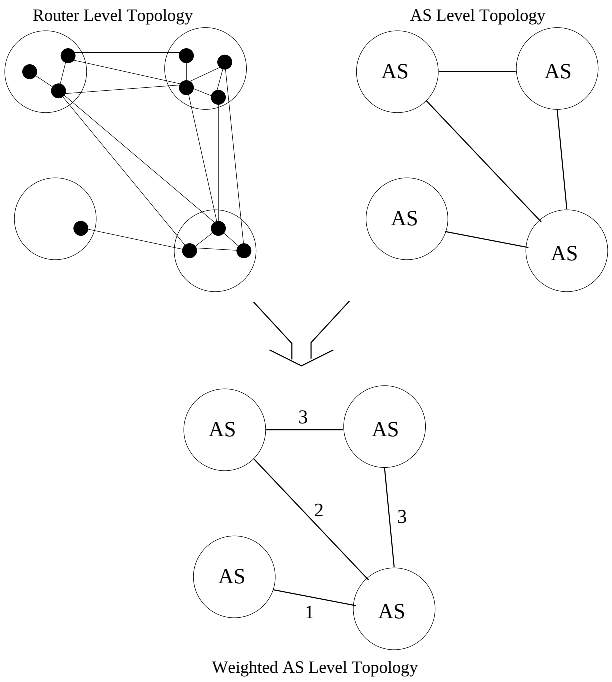

FIG. 5.7 – Obtention d'une topologie d'ASs pondérée

Si il est impossible d'aquerir une topologie de niveau routeur, il suffit d'attribuer aléatoirement des poids aux liens de la topologie d'ASs. Bien que les poids,ainsi attribué, n'ai pas de justification, il suffiront à produire une topologie de sessions BGP.

5.3.2 Méthode d'inférence des topologies de sessions BGP

A partir de la topologie pondérée des ASs, l'algorithme de construction de la topologie de sessions BGP peut être exprimé comme suit :

1. **réduire les poids des liens en les divisant par un facteur commun :** en effet, s'il existe quatre liens de niveau routeur entre deux ASs, cela ne signifie pas qu'il existe quatre sessions BGP entre les speakers de ces ASs. Pour tenter d'estimer le nombre de sessions BGP existantes entre chaque paire d'AS, on divise chaque poids de le topologie par un facteur $factor$ calculé comme suit :

$$factor = \frac{\sum_i PoidsLien_i}{NbrLiens \div (1 - PourcentageLiensManquants)}$$

où, le numérateur de la fraction représente la somme des poids des liens du graph, NbrLiens contient le nombre de liens dans le graph des ASs et $PourcentageLiensManquant$ représente la fraction de liens BGP qui n'apparaissent pas dans le graph des ASs (une études a estimé que cette fraction se siturait dans l'interval $[20\%, 50\%]$). Si la division produit des nombres non entiers, il faut les arrondir à l'entier, supérieur à 0, le plus proche.

2. Créer, dans chaque AS, un nombre de speakers égale au poids du lien sortant de cet AS, de poids maximum. Et associer à chaque lien de l'AS, un compteur initialié avec la valeur du poids du lien.
3. Sélectionner dans l'AS, un speaker n'ayant aucune session. Lui associer une session pour chaque compteur dont la valeur est supérieure à 0, et décrementer ce compteur. Chaque session sera, aussi, associée à l'un des speakers de l'AS voisin, qui ne possèdent pas, déja, de sessions avec un speakers de l'AS courant. Le compteur correspondant dans l'AS voisin sera aussi décrementé.
4. Lorsque tous les compteurs de sessions sont à 0, passer à un AS qui possède au moins un compteur non nul.
5. Lorsque toutes les sessions sont construites (plus de compteurs positifs dans toute la topologie), créer les sessions IBGP dans les ASs. Pour les ASs en *Full Mesh*, établir entre chaque couple de speakers, une session IBGP. Pour les ASs avec réflecteurs de routes, il n'existe, malheureusement, pas de méthode de construction avérée et largement utilisée. Il faudra, donc, choisir des réflecteurs de routes au sein des speakers d'un AS, de manière aléatoire, et leur attribuer des clients choisis au hasard. Ce découpage fait, il ne restera qu'à établir des sessions IBGP entre chaque réflecteur et ses clients ; puis établir une clique de sessions IBGP, dont les nœuds sont les réflecteurs de routes et les speakers n'appartenant à aucun cluster.

La figure 5.8 montre un exemple d'application de l'algorithme pour un graph de 4 nœuds. Avant toute chose, il faut disposer d'un graph valué d'ASs (figure 5.8(a)). A partir de là, on estime le facteur de réduction des poids du graph. Cette valeur calculée, on divise chacun des poids du graph par le facteur (figure 5.8(b)). Les nouveaux poids non entiers sont arrondis à l'entier positif, supérieur à 0, le plus proche (figure 5.8(c)). Puis, on introduit, dans chaque AS, un nombre de speakers correspondant au poids le plus haut des liens de chaque AS (figure 5.8(d)). On associe, alors, à chaque speaker, de chaque AS un maximum de sessions BGP, en évitant que deux sessions n'attachent le même couple de speakers (figure 5.8(e), (f), (g) et (h)). La

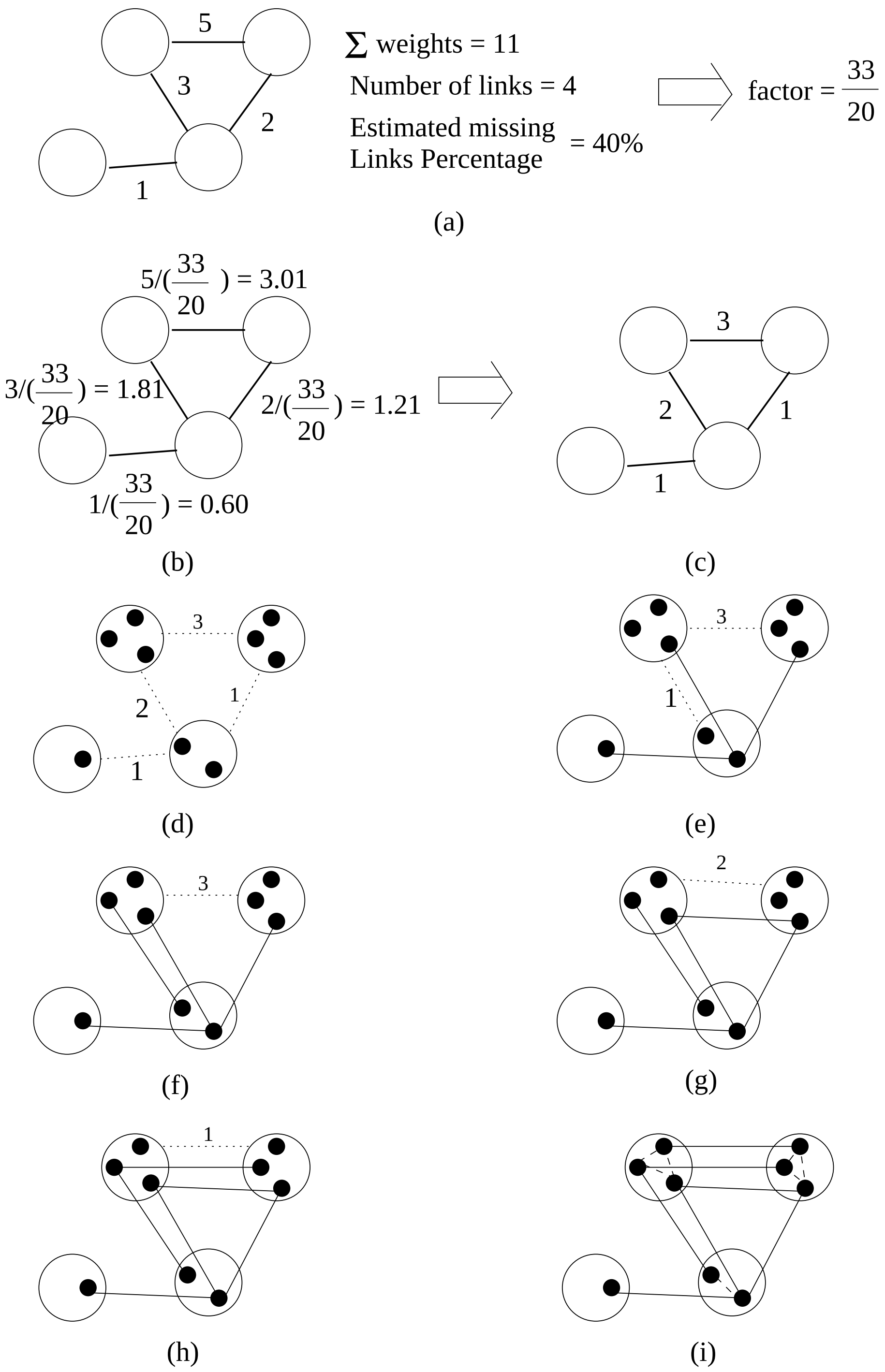

FIG. 5.8 – Exemple de construction de la topology des ASs

figure 5.8(i) montre la topologie résultate après l'établissement des sessions IBGP.

5.4 Politiques de routage et leurs mises en œuvre

Les politiques de routage sont le moyen par lequel les simulations prennent un caractère de réalité. En effet, dans l'Internet, rare sont les ASs qui ne surcharge par le comportement par défaut de BGP de choix de route (moindre nombre d'ASs dans l'AS_PATH) par une politique qui attribue de valeurs de l'attribut LOCAL_PREF aux routes importées, selon leurs sources. Les politiques permettent, en quelque sorte, d'implémenter la statégie de routage des ASs et leurs contrats de service de transit.

Dans le simulateur, les politiques de routage seront implémentées sous forme de **règles de décision**. Ainsi, un speaker disposera d'une politique de routage par peer qu'il perçoit. Chaque politique contiendra une suite ordonnée de règles de décision , sous la forme :

$$< DecisionCriteria => Decision >$$

où :

- *DecisionCriteria* est une combinaison logique de conditions sur les attributs de la route filtrée. La combinaison se fait par les opérateur NOT, AND et OR.
- *Decision* constitue l'action à executer sur cette route.

Les conditions sur les attributs de la route sont exprimé avec différents opérateurs suivant que les attributs sont des valeurs uniques ou des ensemble :

- Les attributs PEER_ID, PEER_ASN, LOCAL_PREF, MED et NLRI constituent des attributs aux valeurs simples. Les conditions devront pouvoir exprimer des comparaison entre les valeurs de ces attributs et des valeurs constantes, par les opérateurs de comparaison $=$, $<$ et $>$.
- L'attribut AS_PATH est un ensemble d'ASN. Les conditions devront, donc, pouvoir exprimer la relation d'appartenance d'un élément à l'ensemble ($\in$) et la relation d'inclusion d'un ensemble dans un autre ($\subset$).
- Les attributs PEER_ID et PEER_ASN doivent pouvoir être recherchés dans une liste de valeurs. Il faut, donc, en plus que ces la relation d'appartenance à un ensemble puisse être appliquée à ces attributs.

Pour ce qui est des actions à effectuer, en plus des actions **REJECT** et **ACCEPT**, les politiques de routage doivent pouvoir attribuer des valeurs aux attributs LOCAL_PREF et MED, mais aussi *ajouter* des ASNs dans les

AS_PATHS.

Quand un des filtres applique une politique sur une route donnée, il doit tester les conditions de chacune des règles de politiques, une par une. Si l'action, associée à une règle dont la condition est vérifiée, spécifie l'acceptation, la route st accéptée et l'application de la politique s'arrête là. Si l'action, associée à une règle dont la condition est vérifiée, spécifie la modification d'un attribut, l'attribut est modifié en conséquence, puis le filtre continue d'appliquer les règles de la politique. Si l'action, associée à une règle dont la condition est vérifiée, spécifie le rejet, la route est immédiatement rejetée, et l'application de la politique s'arrête là. Un cas particulier existe lorsqu'au niveau du filtrage *ingress*, une route est rejetée mais qu'il existe dans l'Adj-RIB-In correspondant au peer émettant cette route, une ancienne route vers la même destination, alors l'ancienne route doit aussi être supprimée (rejet implicite de route, mais pas de remplacement par la nouvelle route). Si le filtre arrive à la fin des règles sans trouver de règle qui accepte ou rejète la route, alors la route est automatiquement acceptée.

Dans [HK03], Hao et Kopol ont identifié trois types de politiques de routage qu'il serait intéressant d'implémenter dans les simulations :

1. Les politiques de transit : permettent aux ASs de spécifier qu'il fournissent un service de transit au traffic issue de certains autres ASs. Ces politiques correspondent aux politiques appliquées par les providers d'accès à l'Internet.
2. Les politiques locales de sélection de route : permettent aux ASs de spécifier qu'ils ne fournissent pas de service de transit au traffic isue de certains ASs. Ces politiques correspondent aux politiques appliquées par les clients des providers.
3. Les politiques d'ajout d'ASs : (*AS prepending* ou *AS padding*) ces politiques sont appliquées par des ASs qui tentent d'influencer les choix d'autres ASs en utilisant un ralongement artificiel des chemins qu'ils annoncent (par l'ajout de leur propre ASN plusieurs fois dans le chemin). Ces politiques ne correspondent pas à une quelconque relation commercial entre ASs, mais certaines recherches ont prouvé leur existance.

Un AS n'appliquant aucune de ces politiques particulière pour une connexion avec l'un de ces peer, aura le même comportement que s'il appliquait une politique de partage totale avec ce peer.

La distribution de ces politiques sur la topologie ne peu se faire avec certitude. Une méthode de distribution consisterait en l'identification préalable,

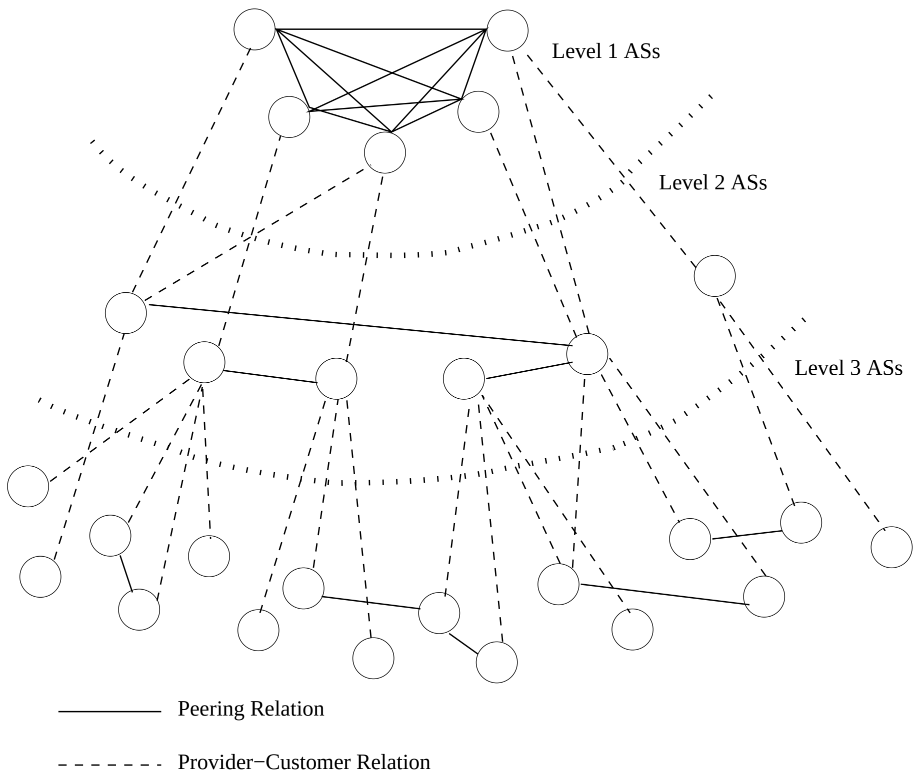

FIG. 5.9 – Niveaux de la hiérarchie Internet

pour chaque AS de la topologie, du niveau de la hiérarchie Internet auquel il appartient (Les topologies générées par l'univeristé de Berkeley [Aga] établissent ce type de discrimination, en distinguant cinq niveaux dans la hiérarchie). La seconde étape consisterait à associer à chaque lien reliant deux ASs, un type de relation : si les deux ASs appartiennent au même niveau de la hiérarchie, on peut considérer qu'ils sont en relation de *peering* (égale à égale) ; et si ils appartiennent à deux niveaux différents, on considérera qu'une relation *provider-customer* (fournisseur-client) les unis (voir la figure 5.9). La dernière étape consiste à associer les politiques au speakers d'un AS selon la relation reliant cet AS à l'AS des peers du speaker. Ainsi :

– Si la relation, unissant l'AS du speaker à l'AS de son peer, est une relation *provider-customer*, et que le speaker appartienne à l'AS le plus haut dans la hierarchie de la topologie, alors il faut que le speaker puisse annoncer au peer les routes qu'il a apprise par ses autres clients, les routes provenant de ses propres providers et les routes provenant de ses peers.Il doit, donc, appliquer une politique de transit à ce peer.
– Si la relation est une relation *provider-customer*, est que le speaker

appartienne à l'AS le plus bas dans la hierarchie de la topologie, alors il faut que le speaker exporte, au peer, ces propres routes et les routes apprises par ses clients uniquement. Il appliquera ,donc , une politique de transit si l'ASN du peer lui ayant communiqué la route à exporter, correspond à l'un de ses clients, sinon il appliquera une politique de non-transit.
- Si la relation est une relation de *peering*, le speaker peut exporter ses propres routes et les routes de ses clients. Donc, il appliquera une politique de transit si la route provient d'un client, sinon il appliquera une politique de non-transit.

Les ASs du niveau le plus haut de la topologie sont sensés offrir la connectivité à tous le monde. Leurs speakers appliqueront donc une politique de transit, quel que soit les peers auquel ils se connectent.

Enfin, il serait utile d'appliquer des politiques d'ajout d'ASs, mais comme il n'existe, pour l'instant, aucune méthode pour apprendre exactement leurs sources, l'application de ces politiques ne peut se faire qu'en choisissant de manière aléatoire les speakers appliquant ce type de politiques.

5.5 Délais de Transit et Délais de Traitements

Dans la réalité, les messages BGP circulant entre speakers BGP prennent un certain temps avant d'être traités par le récepteur du message. Ce retard entre l'émission d'un message et la fin de son traitement par le récepteur, comprend :

1. le délais de traversée du lien reliant les peakers communiqunats.
2. le temps d'attente, au niveau du récepteur, imposé par le partage du CPU entre les différentes tâches du routeur (*forwarding* des données, traitements IGP,...).
3. le temps de traitement du message lui-même.

Dans le modèle de simulation, nous avons imposé deux types de délais, correspondants à ce qui présenté ci-dessus :

Le délais de traversée de lien par un message est imposé à chaque message transmit d'un speaker vers un autre speaker n'appartenant pas au même AS (Les temps de traversée des liens internes sont supposés très faibles).

Le délais de traitement induit par la charge du routeur est un délais d'attente imposé à chaque message BGP entrant dans un routeur. Ce délais comptera pour l'attente induite par le routeur pour les traitements autres que BGP (*forwarding*, IGP, ...).

Pour le délais de traversée des liens, l'approche classique consiste à associer à chaque message émis un délais dont la valeur est aléatoirement choisie dans un certain interval. Cette approche, extrèmement simplifiée, ne permet pas de simuler les congestion des liens de l'Internet, et leurs effets sur le routage inter-domaine. Pour tenter de combler cette faille, nous avons imaginé une méthode d'estimation du délais de transit en fonction de la congestion sur le lien. Cette congestion s'exprime par le biais d'une augmentation du taux de perte des paquets sur le lien et une augmentation du RTT (*Round Trip Time* : temps d'aller et retour des paquets). Nous avons, donc, exprimé le délais de transit d'un paquet en utilisant ces variables. Cela donne :

$$TempsTransit = \frac{TaillePaquet}{DebitLien}$$
$$DebitLien = \frac{TailleMaxSegmentTCP}{RTT \times \sqrt{p}}$$

ou, $TempsTransit$ représente le temps qu'il faut au paquet pour atteindre le récepteur, $Taillepaquet$ représente, en octets, la taille du message BGP, $DbitLien$ représente le débit théorique atteint par TCP sur ce lien. La seconde formule, tirée de [MSMO97] calcule ce débit en fonction de la taille maximale des paquets traversant ce lien (en général, 1460 octets) du RTT du lien (indique en milliseconde, le temps qu'il faut à un paquet pour atteindre le récepteur puis revenir vers son émetteur) et du taux de perte des paquets, p.

Pour pouvoir appliquer cette modélisation, il faut disposer des taux de perte et des RTTs de chaque lien reliant un couple d'ASs. Grâce à cela, il devient possible de simuler une congestion sur un point, ou sur une région de la topologie, et d'observer ses effets sur le routage inter-domaine.

Le second délais modélisé exprime, en quelque sorte, l'occupation du routeur. En effet, le délai induit par la charge du routeur impose l'attente que subit chaque message BGP du fait de l'occupation du CPU du routeur par d'autres traitements (le CPU d'un routeur BGP réel est partagé entre différentes tâches de traitement : traitements IGP, traitements BGP, traitements de *monitoring*, ...).

Dans notre modèle, ce délai est calculé en fonction de deux composantes :

1. une composante représentant le délai induit par les traitements BGP du routeur.
2. une composante représentant le reste des traitements nécessitant le CPU du routeur (*forwarding* des paquets, traitements IGP,...).

La première composante est calculée en dénombrant les messages BGP passant par le routeur durant un labs de temps. La seconde composate est fonction de l'activité du réseau. En effet, plus il y'a de paquets transitant par les routeurs, plus les messages BGP arrivant à ces routeurs sont retardés avant d'être traités. Pour simplifier le traitement, on considérera que cette seconde composante est égale à une valeur fixe (spécifiée par la configuration de la simulation). De cette manière, les simulations pourront varier la charge du réseau en variant simplement ce paramètre, et observer ensuite son impact sur la stabilité inter-domaine.

En conclusion, le délai des messages BGP induits par la charge des CPU des routeurs sera calculé comme suit :

$$D_{CPU} = D_{TraitBGP} + D_{AutrTrait}$$

où,

- D_{CPU} est le délai imposé à un message BGP au niveau d'un routeur BGP,

- $D_{TraitBGP}$ est le délai imposé à un message, induit par le traitements des autres messages,

- $D_{AtrTrai}$ est l'attente imposée à un message dúe aux autres traitements du CPU. Elle possède une valeur fixe.

Pour calculer $D_{TraitBGP}$, il faut mettre en place un dénombrage des messages BGP passant par chaque routeur durant des intervals de temps fixes. Grâce à cela, chaque message BGP subit un délai :

$$D_{TraitBGP} = \gamma \times N \times T_S^2$$

où,

- N est le nombre de messages décompté durant une unité de temps (une seconde),
- T_S est le temps de traitement d'un message BGP par le CPU,
- γ est un paramètres dépendant de la loi d'arrivée des messages. Nous ne connaissons malheureusement pas la distribution de ces arrivées. Il serait , alors, utile (pour plus de simplicité) de prendre une valeur fixe de ce paramètre.

Enfin, et pour reproduire l'irrégularité des arrivées des messages BGP, il faut varier le délai D_{CPU}, par exemple en le multipliant par un facteur aléatoirement choisit pour chaque message.

Chapitre 6

Conclusion

Actuellement, BGP est "le" protocole de routage inter-domaine de l'Internet, et ce, depuis plus d'une dizaine d'années. Malgré les différentes versions par lesquels il a passé, et les relativement différentes implémentations de ce protocole, tous les grands acteurs de la fourniture de service de connectivité se sont accordés sur l'utilisation de BGP pour le routage inter-domaine.

Mais la conception de BGP, certaines de ses implémentations, et certains choix de configuration des ASs sont la source de divers problèmes qui dégradent les performences du routage inter-domaine. Réunis sous le nom d'instabilité, ces divers problèmes (variations rapides des tables de routage, oscillations de route,...) empêchent la convergence de BGP(ou, du moins, la ralentissent fortement) et provoquent un traffic important de messages BGP sur le net.

La résolution de ces problèmes passe par l'étude attentive du protocole BGP, composant principale du routage inter-domaine. Parmi tous les procédés d'étude de BGP existants, notre choix de la simulation se justifie par les immenses possibilités qu'offrirait un modèle de simulation imitant au plus près le comportement d'instabilité de BGP.

Le dernier chapitre de ce rapport était assigné à la présentation d'une proposition de modèle de simulation de BGP. Ce modèle, en plus d'offrir le prototype d'un speaker "réaliste" et de présenter une modélisation complexe de l'intérieur des ASs, expose une proposition pour un nouveau type de topologies de simulation de BGP, les topologies de sessions BGP, ainsi que l'algorithme pour construire ces topologies. Enfin, ce chapitre présente une modélisation des délais imposés aux messages BGP, en fonction d'une représentation de la charge du réseau en terme de traffic utilisateur.

La prochaine étape consistera en l'implémentation du simulateur et sa validation (comparer les comportement produit par le simulateur, au comportemnt attendu, dans certaines situations précises). Dans la suite, le simulateur pourra être utilisé sur des topologies de sessions BGP à l'échelle de l'Internet, pour tenter de reproduire l'instabilité du routage inter-domaine. Il faudra, alors, produire des *scénarios* d'instabilité.

Si le simulateur arrive à générer des instabilités en proportions proches de celles produites sur Internet, on pourra alors penser à implémenter sur le simulateur les méthodes d'amélioration de la convergence proposées par différentes études (SSLD, ou le *Ghost Flushing* par exemple), pour pouvoir évaluer leur efficacité et leurs effets sur l'instabilité. Un point important à considérer, sera la réalisation de simulation avec, une partie seulement des speakers BGP, implémentant une certaines méthode d'amélioration de la convergence ; car, vu la diversité des implémentations de BGP, il n'y a pas de garantie qu'une méthode prouvée comme efficace soit implémentée par tous les constructeurs. Il sera, alors, intéressant d'observer les interactions entre speakers mettant en œuvre un certain certain algorithme et speakers ne possédant pas cette technologie.

Bibliographie

[Aga] S. Agarwal. Internet Topologies by Berkeley. `http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/`.

[BCC00] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. RFC 2976, IETF, Avril 2000.

[Bei02] I. Van Beijnum. *BGP*. O'REILLLY, 2002.

[Bla00] U. Black. *IP Routing Protocols : RIP, OSPF, BGP, PNNI and CISCO Routing Protocols*. Prentice Hall Series, 2000.

[CB] B. Cheswick and H. Burch. The Internet Mapping Project. `http://research.lumeta.com/ches/map/`.

[CJW01] H. Chang, S. Jamin, and W. Willinger. Inferring AS-level Internet topology from router-level path traces. In *Proc. SPIE*, volume 4526, pages 196–207, juillet 2001.

[Dim] X. A. Dimitropoulos. Creating Realistic BGP Models.

[FBT04] T. D. Feng, R. Ballantyne, and L. Trajkovic. Implementation of BGP in a network simulator. In *Applied Telecommunication Symposium, ATS '04*, pages 149–154, Avril 2004.

[fIDAC] Cooperative Association for Internet Data Analysis (CAIDA). The Internet Atlas Project. `http://www.caida.org/projects/trends/data/`.

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

[FWR04] N. Feamster, J. Winick, and J. Rexford. A model of BGP routing for network engineering. *SIGMETRICS Perform. Eval. Rev.*, 32(1) :331–342, 2004.

[GR97] R. Govindan and A. Reddy. An Analysis of Internet Inter-Domain Topology and Route Stability. In *Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE*

Computer and Communications Societies. Driving the Information Revolution, page 850. IEEE Computer Society, 1997.

[GW99] T.G. Griffin and G.T. Wilfong. An Analysis of BGP Convergence Properties. In *Proceedings of SIGCOMM*, pages 277–288, Cambridge, MA, August 1999.

[Hal01] S. Halabi. *Internet Routing Architectures, Second Edition.* Cisco Press, 2001.

[HK03] F. Hao and P. Koppol. An Internet scale simulation setup for BGP. *SIGCOMM Comput. Commun. Rev.*, 33(3) :43–57, 2003.

[JSW] S. Jamin, S. Shenker, and W. Willinger. The UMICH Topology Project. `http://topology.eecs.umich.edu/`.

[LABJ00] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *Proceedings of SIGCOMM*, pages 175–187, September 2000.

[Li03] W. Li. Inter-Domain Routing : Problems and Solutions. `http://citeseer.ist.psu.edu/li03interdomain.html`, 2003.

[LMJ98] C. Labovitz, R. Malan, and F. Jahanian. Internet Routing Instability. In *IEEE/ACM Transactions on Networking*, volume 6, pages 515–528. ACM Press, 1998.

[LMJ99] C. Labovitz, G. R. Malan, and F. Jahanian. Origins of Internet Routing Instability. In *Proceedings of IEEE INFOCOMM '99 Conference*, March 1999.

[LYPN02] M. Liljenstam, Y. Yuan, B. J. Premore, and D. Nicol. A Mixed Abstraction Level Simulation Model of Large-Scale Internet Worm Infestations. In *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*, page 109. IEEE Computer Society, 2002.

[MGeRK02] Z. Mao, R. Govindan, and G. Varghese et R. Katz. Route flap damping exacerbates internet routing convergence, 2002.

[MGWR02] D. McPherson, V. Gill, D. Waltson, and A. Retana. Border Gateway Protocol (BGP) Persistent Route Oscillation Condition. RFC 3345, IETF, Aout 2002.

[Mil84] D.L. Mills. Exterior Gateway Protocol Formal Specification. RFC 904, IETF, Avril 1984.

[MRT] Multi-Threaded Routing Toolkit. `http://www.mrtd.net/`.

[MSMO97] Mathis, Semke, Mahdavi, and Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. In *Computer Communication Review*. ACM SIGCOMM, 1997.

[NCM02] J. Nykvist and L. Carr-Motyckova. Simulating convergence properties of BGP. In *Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, Octobre 2002.

[NS2] The Network Simulator ns-2. `http://www.isi.edu/nsnam/ns/`.

[Pre03] B. J. Premore. *An Analysis of Convergence Properties of BGP using Discrete Events Simulation*. PhD thesis, Dartmouth College, Hanover, New Hamshire, Mai 2003.

[Quo03] B. Quoitin. Synthesis and recommendations for interdomain traffic enginneering, Novembre 2003. Information Society Technologies.

[Quo04] B. Quoitin. *C-BGP User's Guide*, Avril 2004.

[RG95] Y. Rekhter and P. Gross. Application of the Border Gateway Protocol in the Internet. RFC 1772, IETF, Mars 1995.

[RL95] Y. Rekhter and T. Li. A Border Gateway Protocol (BGP-4). RFC 1771, IETF, Mars 1995.

[Rou] Route Views Project. `http://antc.uoregon.edu/route-views/`.

[SSF] SSFNet. `http://www.ssfnet.org/`.

[Ste99] J. W. Stewart. *BGP-4 : InterDomain Routing in the Internet*. Addison-Weasley, 1999.

[TGSE01] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The Impact of Routing Policy on Internet Paths. In *INFOCOM*, pages 736–742, 2001.

[TMS01] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. RFC 3065, IETF, Février 2001.

[Tra95] P. Traina. Experiences with the BGP-4 Protocol. RFC 1773, IETF, Mars 1995.

[VCG98] C. Villamizar, R. Chandra, and R. Govidan. BGP Route Flap Damping. RFC 2439, IETF, Novembre 1998.

[VGE00] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks (Amsterdam, Netherlands : 1999)*, 32(1) :1–16, 2000.

[WZPa] L. Wang, X. Zhao, D. Pei, and al. Observations and Analysis of BGP under Stress. USC/Information Science Institue, Internet Initiative Japan, UCLA.

[ZEB] GNU zebra : free routing software. `http://www.zebra.org/`.