ACADÉMIE DE MONTPELLIER

UNIVERSITÉ **M**ONTPELLIER **II**

— **S**CIENCES ET **T**ECHNIQUES DU **L**ANGUEDOC —

MÉMOIRE DE D.E.A.

présenté à l'Université des Sciences et Techniques du Languedoc
pour l'obtention du diplôme d'études approfondies

SPÉCIALITÉ	: **INFORMATIQUE**
Formation Doctorale	: **Informatique**
École Doctorale	: **Information, Structures, Systèmes**

Instabilités du protocole BGP

par

Clément SAAD

Soutenu le 17 juin 2005 devant le jury composé de :

M. Ehoud AHRONOVITZ, Maître de conférences, LIRMM, Montpellier . . . . . encadrant
M. Jean-Claude KÖNIG, Professeur, LIRMM, Montpellier . . . . . encadrant
Mme. Jocelyne NANARD, Professeur, LIRMM, Montpellier . . . . . directrice

Table des matières

Table des figures

Remerciements

Je tiens sincèrement à remercier mes deux tuteurs Ehoud AHRONOVITZ et Jean-Claude KÖNIG pour leurs remarques pertinentes et leurs conseils précieux. Leur disponibilité et leur enthousiasme dans le travail ne peuvent que vous pousser à vous impliquer d'avantage dans vos travaux. Je n'oublie pas non plus les qualités humaines dont ils ont fait preuve. Pour tout cela je leur suis reconnaissant.

Je tiens également à remercier tous les membres de l'équipe APR, où mon intégration c'est faite de la façon la plus naturelle possible, pour la considération qu'ils ont porté à mes travaux.

Je n'oublie pas Ken SCHUMACHER, qui a "débroussaillé" tout le fonctionnement de BGP, ce qui n'était pas une mince affaire.

Je tiens à saluer tous les occupants de la halle Gorithme pour la bonne ambiance qu'ils y font régner et spécialement les membres du bureau Asimov : Mohamed BOUKLIT, Francois BOUTIN, Olivier CARLONI et Maxime COLLOMB, qui m'ont acceuilli ces quatre derniers mois.

Enfin, je remercie Eugène BOTELLA et Nicolas TERRAPON, à qui je dois mes rares moments de détente durant ce DEA. Heinrich HOERDEGEN et Guillaume BAGAN, mes compagnons de galères ; ainsi que Michel HABIB et Vincent LIMOUZY pour m'avoir initié au plaisir du bateau.

Je remercie Caroline pour son soutien permanent.

Chapitre 1

Introduction

BGP (Border Gateway Protocol) est le protocole de routage externe utilisé dans l'Internet. Ce protocole permet aux systèmes autonomes (par la suite nous utiliserons l'abbréviation AS pour Autonomous System) de s'échanger des données. Un AS est un ensemble de réseaux et de routeurs sous une administration unique. Chaque AS a un protocole de routage interne (RIP, OSPF,...) et une politique de routage externe. Cette politique peut être définie selon des critères commerciaux, de performance, de sécurité, ... BGP a été conçu pour laisser libre choix aux AS d'appliquer leur propre politique. Cependant, ce libre choix peut entraîner globalement des incohérences entre ces politiques laissant apparaître des oscillations de routes. Dans ce cas, nous sommes en présence d'instabilités inter AS. Ce mémoire a pour but de proposer une solution pour résoudre ces oscillations.

Varadhan et al. ont montré dans [1] que les politiques internes aux AS pouvaient entraîner des inconsistances globales. Lobavitz et al. ont étudié dans [2][3] les origines des instabilités du protocole BGP.

Durant ces six dernières années, des solutions statiques et dynamiques ont été proposées. Dans le cas des solutions statiques, la détection des oscillations se fait avant l'activation du routage nécessitant la connaissance intégrale des politiques des AS. Pour les solutions dynamiques, la détection des oscillations se fait pendant le routage. Govindan et al. ont proposé dans [4] une solution statique impliquant toutes les politiques mais Griffin et Wilfong ont montré que le temps de calcul pourrait être exponentiel. Griffin, qui a beaucoup travaillé sur ce sujet, a proposé une solution dynamique en utilisant un historique. Plus de détails sont donnés dans [5][6][7][8][9]. Gao et Rexford ont établi dans [10] quelques conditions suffisantes pour éviter des oscillations. Leur étude est basée sur les relations existantes entre clients et fournisseurs. Récemment, Yilmaz et Matta ont proposé dans [11] une solution dynamique en développant un algorithme randomisé modifiant l'ordre de préférence des chemins pour les AS.

Toutes ces méthodes présentent des inconvénients majeurs. Concernant les solutions statiques, la quantité de données en entrée conduit à un temps de calcul important et nécessiterait d'avoir des informations concernant les politiques risquant de porter atteinte au principe de confidentialité imposé par le protocole BGP. Les deux méthodes dynamiques proposées ne sont pas non plus applicables. Dans le cas de Griffin, le volume de données transitant sur le réseaux est considérable puisqu'un AS joint son historique à chaque annonce de nouveau chemin. La gestion d'un historique nécessite des ressources non négligeable pour un AS. De plus l'historique fournit des informations globales sur les AS, entravant également le principe de confidentialité des politiques. Celle de Yilmaz et Matta n'a pas ces défauts mais leur mé-

thode ne respecte pas le principe de BGP qui doit tenter de satisfaire au mieux les exigences des politiques dictées par les AS et non de les modifier.

Dans ce mémoire nous proposerons une méthode dynamique respectant les contraintes imposées par le protocole BGP, permettant de détecter et résoudre “des oscillations”. Cette méthode est basée sur l’utilisation d’un jeton. Lorsqu’un AS détecte une oscillation, il génére un jeton associé à une route et l’expédie à ses voisins. Les AS le recevant ont le choix de le faire suivre ou de le détruire. Si le jeton revient à son générateur, celui-ci interdit la route associée, resolvant ainsi l’oscillation. Pour être cohérent avec la gestion des pannes et des apparitions de liens de BGP nous proposerons une adaptation de notre méthode en réhabilitant certaines routes préalablement interdites. Pour finir, nous définirons la notion de politiques cohérentes entre elles.

Chaque information échangée entre les AS, tel que le jeton, ne révéle en rien la politique adoptée par son expéditeur. Cette remarque est primordiale puisqu’elle respecte le principe de confidentialité. Notre méthode prend également en compte la durée de convergence importante de BGP, **en essayant de ne pas ajouter de messages supplémentaires** mais plutôt de compléter les messages actuellement échangés dans BGP.

Dans le chapitre 2 nous ferons une description succincte du protocole BGP. Dans le chapitre 3 nous verrons comment formaliser le problème d’oscillations du protocole. Dans le chapitre 4, nous expliquerons le graphe de conflit de Griffin permettant d’étudier les cas d’oscillation. Dans le chapitre 5, nous rappelerons la solution dynamique de Griffin. Dans les chapitres 6 et 7 nous caractériserons les cas d’oscillation et nous proposerons notre méthode de détection et de résolution des oscillations. Enfin, dans le chapitre 8 nous éclaircirons la notion de politiques cohérentes entre elles.

Une partie de ce mémoire a fait l’objet d’un article soumi à la conférence SSS2005 (Symposium on Self Stabilization).

Chapitre 2

Le protocole BGP

BGP (Border Gateway Protocol) est le protocole de routage utilisé dans l'Internet. Nous allons faire une description rapide de ce protocole. Pour plus de détails voir [12][13].

2.1 Principe général

BGP a été introduit dans le but de permettre aux systèmes autonomes de s'échanger des informations à travers l'Internet. Un sytème autonome est un ensemble de réseaux et de routeurs sous une administration unique. Chaque AS décide de son protocole de routage interne (RIP, OSPF,...) et de sa politique de routage externe. BGP doit répondre à plusieurs attentes. La première consiste à essayer de satisfaire au mieux les exigences imposées par les politiques. Les politiques pouvant être d'origine sécuritaire, économique, ou autre, une seconde attente consiste à respecter le principe de confidentialité de ces politiques. Les routeurs BGP ne doivent pas communiquer des informations permettant à des AS de reconstruire la politique d'un autre AS.

2.2 Modélisation des AS

Chaque AS ne contient pas forcément un unique routeur BGP. Tous ces routeurs appliquent la même politique de routage externe définie par l'AS (ie. pour une destination donnée, les routeurs utilisent les mêmes chemins externes). De l'extérieur, tous les routeurs BGP de l'AS apparaîtront comme un seul routeur. Ainsi, un réseau d'AS est modélisé par un graphe où les sommets et les arêtes représenteront respectivement les AS et les liens BGP. Il s'agit plutôt d'un multi-graphe puisqu'il peut y avoir plusieurs liens BGP entre deux sommets. La figure 2.1 représente un réseau et la figure 2.2 représente sa modélisation.

2.3 Protocole à vecteur de chemins

BGP est un protocole à vecteur de chemins. A la différence d'un protocole à vecteur de distance tel que RIP, les routeurs impliqués dans un protocole à vecteur de chemins communiquent à tous leurs voisins les destinations qu'ils peuvent atteindre en associant non pas seulement la distance qui les sépare d'elles mais le chemin intégral pour atteindre ces destinations. Ainsi, la gestion des boucles devient facile puisqu'il suffit à un routeur BGP recevant une route de vérifier s'il n'apparait pas déjà dans cette route ; le cas échéant, il déduit qu'il n'y a pas de boucle sinon il ignore la route.

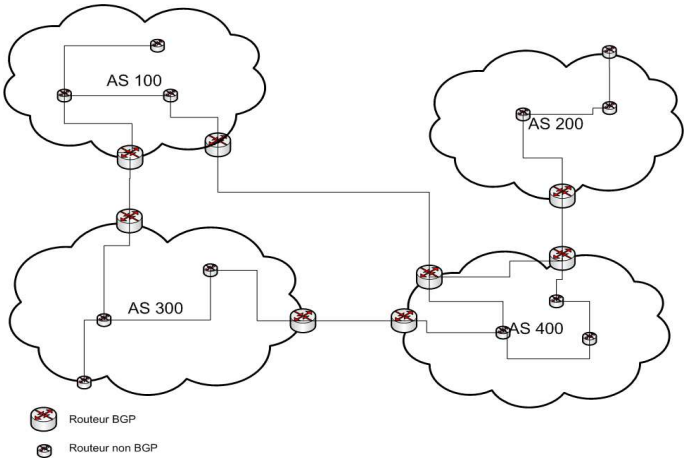

FIG. 2.1 – Exemple de réseau

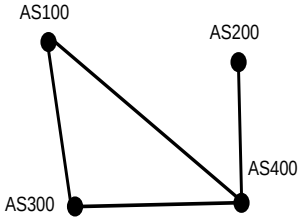

FIG. 2.2 – modélisation du réseau

2.4 Mécanisme d'annonce des routes

Un réseau désirant se faire connaître informe ses voisins BGP de son existence. Les voisins peuvent ignorer ou prendre en compte cette information. Dans le cas où un voisin la prend en compte, il l'annoncera à tous ses voisins. Si un AS annonce une route, il s'engage à accepter le trafic vers cette destination. Sinon il ne l'annonce pas.

Afin de diminuer la taille des tables de routage, BGP a la possibilité d'agréger des routes lors de l'annonce d'une route. Par exemple, si un AS est en charge de tous les réseaux ayant les préfixes 200.100.0 jusqu'à 200.100.255, il peut les agréger et annoncer seulement 200.100. Une annonce d'une route est composée d'attributs :

- *AS_PATH* : indique la route suivant une liste ordonnée.
- *NEXT_HOP* : indique l'adresse IP du prochain routeur BGP.
- *LOCAL_PREF* : attribut interne à l'AS utilisé pour le routage externe. Il n'est pas communiqué aux autres AS. Il permet d'accorder un degré de préférence à chaque route.
- *ATOMIC_AGGREGATE* : indique s'il y a eu agrégation de routes.
- *AGGREGATOR* : indique l'AS ainsi que l'IP du routeur qui a effectué l'agrégation.
- *MULTI_EXIT_DISC (MED)* : permet, lorsque deux AS sont interconnectés à l'aide de plusieurs liens, d'en discriminer en associant à chaque lien un degré de préférence. Cet attribut est la cause d'oscillations internes aux AS (plus de détails sont donnés dans [14][15]).

2.5 Processus de sélection

Le protocole BGP est composé de deux sous parties. Chaque AS ne contient pas un unique routeur BGP. La gestion des communications entre les routeurs internes à l'AS se fait par I-BGP. Les communications entre les routeurs externes des différents AS sont gérées par E-BGP.

Un AS peut avoir plusieurs routes pour une même destination. Le processus de sélection permet de faire le meilleur choix parmi ces chemins en fonction de la politique de l'AS. Le processus se déroule de la façon suivante et s'arrête dès qu'il ne reste plus qu'un chemin :

1. Choisir les routes ayant le plus grand *LOCAL_PREF*.
2. Choisir les routes traversant le moins d'AS.
3. Pour chaque voisin, sélectionner les routes qui ont le plus petit *MED*.
4. S'il reste au moins une route E-BGP, c'est à dire annoncée par un AS voisin et non un routeur du même AS, éliminer toutes les routes qui passent au travers de l'AS (route annoncée en I-BGP). Cette étape permet d'éviter la surcharge de son propre réseau.
5. Utiliser une information déterministe (par exemple : choisir la route qui a la plus grande adresse IP dans *NEXT_HOP*).

2.6 Politique de filtrage des annonces

BGP contient trois tables appelées RIB (Routing Information Base) :

Adj-RIBs-In contient les informations de routage apprises par les routeurs BGP voisins.

Loc-RIB contient les informations de routage que le routeur BGP a sélectionné suivant des politiques locales parmi les informations contenues dans Adj-RIBs-In.

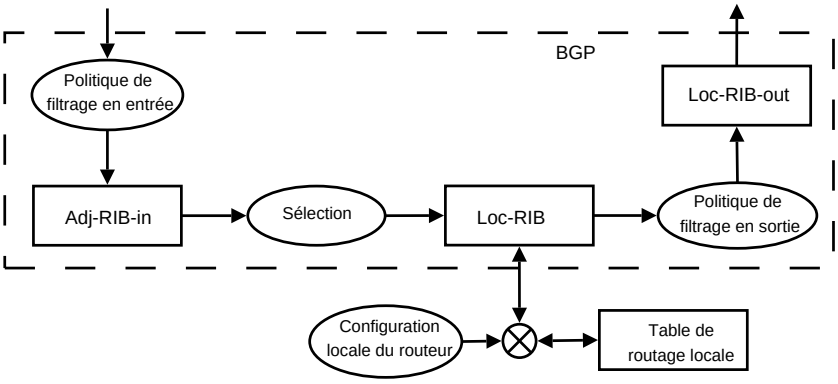

FIG. 2.3 – Politiques de filtrage

Adj-RIBs-Out contient les informations de routage qui peuvent être communiquées à un autre routeur BGP spécifique.

Les annonces de routes reçues par un routeur sont filtrées. Seules les annonces respectant la politique de l'AS seront placées dans la table Adj-RIBs-In. Le processus de sélection est alors exécuté avec en entrée les données de la table Adj-RIBs-In. Les nouveaux chemins sont mis dans la table Loc-RIB. Ensuite, ces nouveaux chemins subiront un nouveau filtrage pour savoir lesquels d'entre eux vont être réannoncés aux voisins. Ce filtrage s'effectue une nouvelle fois en fonction de la politique de l'AS et les chemins sélectionnés seront placés dans la table Adj-RIBs-Out pour être réannoncés (le schéma 2.3 résume ce processus).

Maintenant que nous avons une idée du fonctionnement de BGP nous allons voir comment formaliser notre problème d'oscillations dues à des instabilités entre AS.

Chapitre 3

SPP (Stable Paths Problem)

Le SPP nous donne une vision simplifiée des problèmes d'instabilité des chemins dans des protocoles de routage tels que BGP. Introduit par Griffin et al. dans [7], il permet de se focaliser sur les points essentiels causant les instabilités et de séparer les fonctionnalités de BGP qui ne jouent aucun rôle dans ce problème (par exemple le MED, l'agrégation de routes,...).

3.1 Construction

Soit $G = (V, E)$ un graphe tel que les éléments de V et les éléments de E représentent respectivement les systèmes autonomes et les liens BGP. Chaque AS détermine une liste de chemins ordonnée par ordre de préférence. Dans l'exemple de la figure 3.1, l'AS 1 définit la liste de chemins (130, 10) avec 130 préféré au chemin 10. Le chemin 130 signie que l'AS 1 passe par l'AS 3 pour atteindre la destination 0. Les AS chercheront à atteindre et conserver le chemin ayant un rang de préférence le plus élevé possible. Un AS peut choisir un chemin uniquement si tous les AS qu'il traverse ont choisi le sous chemin correspondant. Par exemple dans la figure 3.1, l'AS 1 peut choisir le chemin 130 uniquement si l'AS 3 a choisi le chemin 30.

Chaque AS définit un ensemble *choix* contenant l'ensemble des chemins possibles pour une destination. La fonction *best* retournera la route possédant la plus haute préférence possible. $best(u) = max(choix(u))$ (u étant un AS). Une solution à SPP est un assignement d'un chemin pour chaque AS, tel que le rang de préférence de ces chemins soit le plus haut possible. Griffin et al. ont montré dans [8] que SPP est un problème NP-complet en utilisant une réduction à 3-SAT.

La figure 3.1 est une instance du problème SPP. Il s'agit d'un exemple classique d'oscillation appelé BAD GADGET très largement diffusé. Nous supposerons, dans tout ce qui suit, que les AS cherchent à choisir un chemin allant à la destination 0. Voici une simulation possible du protocole BGP à partir de la figure 3.1 : au départ, tous les AS choisissent soit un chemin direct, soit le chemin vide noté ϵ. L'AS 1 ne connaît pas le choix des chemins de ses voisins. Il choisit le chemin 10 et le fait savoir à ses voisins. L'AS 2 prend connaissance de cette information et choisit 210 ; lui aussi fait suivre l'information à l'AS 3. La route 20 n'étant pas disponible, l'AS 3 choisit la route 30. Lorsque l'AS 1 en sera informé, il pourra choisir la route 130 qu'il préfère provoquant la perte de 210,... Le processus ne s'arrêtera jamais. Cet exemple est une illustration d'une oscillation ou d'une instabilité du protocole BGP.

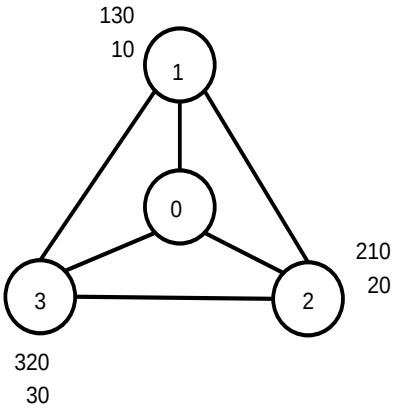

FIG. 3.1 – BAD GADGET - Problème sans solution stable

Chapitre 4

Graphe de conflit

Griffin et al.[6] construisent un graphe de conflit à partir d'une instance de SPP. Le graphe de conflit met en relation les chemins déduits des différentes politiques des AS. Ces chemins sont les sommets de ce graphe et les arcs représentent les compatibilités ou les conflits entre les chemins. Nous allons étudier quelques propriétés de ce graphe.

4.1 Construction

Soit $G = (V, E)$ un graphe. Chaque nœud représente un chemin. Il existe deux sortes d'arcs entre les nœuds :

- Les arcs de transmission : soit u, v étant deux AS et uvP, vP deux chemins appartenant respectivement aux AS u et v. P est un chemin. Si v adopte le chemin vP alors u peut adopter le chemin uvP. Dans ce cas, il y a un arc de vP vers uvP appelé arc de transmission. Il est représenté par un arc en pointillé.
- Les arcs de conflits (figure 4.1) : soit vQ et vP deux chemins appartenant à l'AS v, avec vQ préféré à vP. Soit uvP et uvQ deux chemins appartenant à l'AS u, avec uvP préféré à uvQ. Si v adopte le chemin vQ (préféré à vP) alors u ne peut adopter le chemin uvP puisque vP est indisponible. Ainsi, u choisit un autre chemin avec une préférence inférieur au chemin uvP. Dans ce cas, il y a un arc de vQ vers uvP appelé arc de conflit. Il est représenté par un arc en trait plein.

La figure 4.2 représente le graphe de conflit de BAD GADGET de la figure 3.1.

Prenons l'exemple de l'arc de transmission du chemin 30 vers le chemin 130. Si 30 est adopté alors le chemin 130 peut l'être aussi. L'arc de conflit de 130 vers 210 s'explique de la façon suivante : si 130 est adopté, le chemin 10 ne peut être choisit et par conséquent le

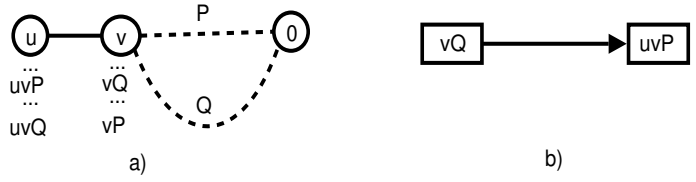

FIG. 4.1 – Condition pour l'arc de conflit $vQ- > uvP$

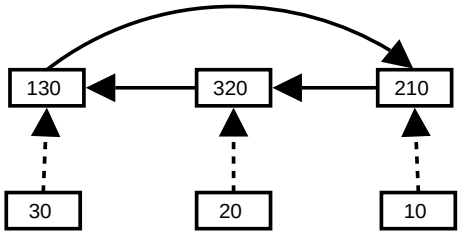

FIG. 4.2 – Graphe de conflit de BAD GADGET avec trois AS

chemin 210 non plus.

4.2 Circuit de conflit

Les circuits de conflit mettent en évidence les inconsistances des politiques.

Définition 1. *Un circuit de conflit est un circuit dans le graphe de conflit qui doit contenir au moins deux arcs de conflits.*

Griffin démontre le théorème suivant dans [6] :

Théorème 1. *Si le graphe de conflit ne contient pas de circuit alors le problème SPP contient une solution stable.*

Ces circuits de conflit sont la source des problèmes d'oscillations de BGP. Notre solution consistera à interdire un chemin appartenant à un circuit afin de le rompre et se ramener, comme nous le confirme le théorème précédent, à un état stable.

Chapitre 5

SPVP (Simple Path Vector Protocol)

Griffin a développé le SPVP qui est une simulation abstraite du comportement de BGP. Trois versions de SPVP ont été proposées. La première, $SPVP_1$, représente le fonctionnement actuel du protocole BGP. La deuxième, $SPVP_2$, est une amélioration de la version précédente. Elle propose une gestion dynamique des messages grâce à un historique de chemins décrit par une structure *path history* utilisée pour détecter des cycles. Mais ces deux versions ne sont pas fiables dans le sens où elles ne permettent pas de résoudre les instabilités. La troisième version, $SPVP_3$, est une extension fiable de $SPVP_2$.

5.1 $SPVP_1$

Puisqu'il s'agit d'un algorithme distribué, nous ne décrirons la procédure que pour un seul système autonome appellé *u*. Lorsque *u* reçoit un chemin P d'un de ses voisins, il ajoute P à la liste des chemins entrants. Il compare P avec son *best(u)*. Si le résultat est meilleur (par rapport à la liste définie par l'AS) alors *best(u)* prend P et *u* envoie cette information à tous ses voisins.

Cette procédure ne détecte pas les instabilités et donc, en prenant BAD GADGET, ne convergera jamais vers un état stable.

5.2 $SPVP_2$

Dans cette version, les systèmes autonomes ne se communiquent plus les chemins mais des paires $m = \{P, h\}$ où P est un chemin et h un historique (la construction de cet historique est décrite dans la section suivante). Les fonctions *path(m)* et *hist(m)* retournent respectivement le chemin et l'historique de m.

La procédure précédente est modifiée de sorte à prendre en compte ce changement.

Lorsque *u* reçoit une paire m d'un de ses voisins, il ajoute m à la liste des paires entrantes et compare *path(m)* (ie. P) avec son *best(u)*. Si le résultat est meilleur alors il met à jour le nouveau chemin et l'historique et il diffuse l'information à ses voisins. Nous verrons par la suite comment cet historique est géré.

La présence d'un cycle dans son historique permet à un AS de détecter une oscillation. Toutefois, cette procédure n'est pas fiable puisque l'oscillation est détectée mais pas éliminée.

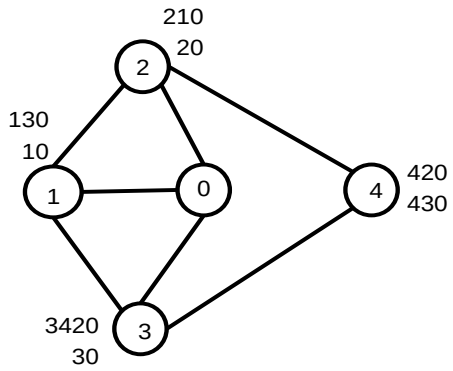

FIG. 5.1 – BAD GADGET avec cinq AS

5.3 $SPVP_3$

Cette troisième version est une version fiable de SPVP. Lorsqu'une instabilité est détectée, $SPVP_3$ tente d'interdire des chemins sélectionnés. Une nouvelle structure de données ***B(u)***, contenant un ensemble de chemins "interdits", est introduite. La fonction ***best(u)*** devient $best(u) = max(choix(u) - B(u))$.

La procédure reste la même que précédement mais, avant de diffuser à tous ses voisins, l'AS vérifie que l'historique ne contient pas de cycle. Si un cycle est détecté, le chemin choisi est ajouté dans ***B(u)*** et la procédure de choix de la meilleure route est relancée.

5.3.1 Construction de l'historique

Chaque AS gère un historique conservant les évènements marquants subis par ses chemins et leurs causes. A chaque annonce de chemin, l'historique associé est joint au message. Lorsqu'un AS reçoit, par un de ses voisins, un chemin et l'historique associé h, si ce chemin entraîne une modification pour l'AS (par exemple le choix du chemin X à la place de Y), les données suivantes sont ajoutées à l'historique de l'AS :

- h
- (+X) si X est préféré à Y
- (-Y) si Y est préféré à X

Regardons le passage de l'étape 2 à 3 dans le tableau 5.1, qui est une exécution de $SPVP_3$ sur le BAD GADGET à cinq sommets de la figure 5.1 . L'AS 4 va annoncer à ses voisins, sa nouvelle route ϵ avec l'historique associé (-420)(+210). L'AS 3 qui avait comme chemin 3420 (notre Y) va devoir choisir le chemin 30 (notre X). Donc il ajoute à son historique l'historique qu'il vient de recevoir (ie. (-420)(+210)) et (- 3420) puisque 3420 est préféré à 30. A l'étape 6 l'AS 2 détecte le cycle ((-210) (+130) (-3420) (-420) (+210)) et $SPVP_3$ interdira le chemin 20.

5.3.2 Inconvénients de $SPVP_3$

Nous devons d'abord souligner que l'historique grandit très rapidement. Lorsqu'un AS expédie une route, il annonce également l'historique associé à cette route. A la réception, les

step	u	best(u)	hist(u)
0	1	(10)	*
	2	(20)	*
	3	(3420)	*
	4	(420)	*
1	1	(10)	*
	2	(210)	(+210)
	3	(3420)	*
	4	(420)	*
2	1	(10)	*
	2	(210)	(+210)
	3	(3420)	*
	4	(ϵ)	(-420) (+210)
3	1	(10)	*
	2	(210)	(+210)
	3	(30)	(-3420) (-420) (+210)
	4	(ϵ)	(-420) (+210)
4	1	(10)	*
	2	(210)	(+210)
	3	(30)	(-3420) (-420) (+210)
	4	(430)	(+430) (-3420) (-420) (+210)
5	1	(130)	(+130) (-3420) (-420) (+210)
	2	(210)	(+210)
	3	(30)	(-3420) (-420) (+210)
	4	(430)	(+430) (-3420) (-420) (+210)
6	1	(130)	(+130) (-3420) (-420) (+210)
	2	(20)	**(-210) (+130) (-3420) (-420) (+210)**
	3	(30)	(-3420) (-420) (+210)
	4	(430)	(+430) (-3420) (-420) (+210)

TAB. 5.1 – Exemple de gestion de l'historique de Griffin pour le BAD GADGET de la figure 5.1. En souligné, les chemins traités à chaque étape. En gras, détection d'un cycle.

AS destinataires doivent mettre à jour leur propre historique. Toutes ces opérations nécessitent beaucoup d'espace mémoire et perturbent les performances réseaux. De plus, le fait de communiquer l'historique à chaque annonce de chemins (permettant à tous les AS d'obtenir des informations globales sur les autres AS) donne la possibilité aux AS de reconstruire la politique de leurs voisins. La contrainte de confidentialité des politiques n'est pas respectée. La solution que nous proposons ne devait pas utiliser un historique.

Chapitre 6

Maintien de l'état des chemins

Nous allons proposer une méthode de détection des oscillations en maintenant l'état des chemins, puis une façon de les résoudre. La gestion d'un historique nécessitant des ressources importantes pour un AS, notre solution doit éviter son utilisation et s'appuyer uniquement sur des informations locales.

6.1 Différences avec la méthode de Griffin

$SPVP_3$ utilise un historique pour détecter les cycles et supprimer des chemins qu'il considère mauvais. Notre modification intervient sur l'identification de ces chemins. Dans $SPVP_3$, lorsqu'un AS détecte un cycle qui correspond à un circuit dans le graphe de conflit, il considère que le chemin qui devait être choisi est un mauvais chemin. Par exemple, lorsque le cycle (+210 -420 -3420 -130 +210) est détecté lors du choix du chemin 20 dans le tableau 5.1, c'est ce chemin 20 qui est considéré comme "mauvais". Nous pensons qu'il s'agit plutôt de 210 étant donné que son interdiction permettrait de rompre le circuit dans le graphe de conflit.

La figure 5.1 illustre l'utilisation de $SPVP_3$ sur un BAD GADGET composé de cinq AS. Avec la méthode de Griffin, il y a convergence vers l'attribution des chemins (130), ϵ, (30), (430) respectivement pour les AS 1, 2, 3, 4. Notre modification amène à l'attribution suivante : ((10),(20),(3420),(420)).

Nous allons voir maintenant notre méthode qui n'utilise plus un historique mais maintient un état des chemins localement pour chaque AS.

6.2 Caractérisations des oscillations

Nous allons faire ici quelques remarques concernant l'historique de Griffin. Nous verrons dans la section suivante que ces remarques nous permettrons de nous passer d'un historique en maintenant uniquement l'état des chemins.

6.2.1 Propriétés générales

En regardant plus attentivement un historique (cf. tableau 5.1), par exemple (-210 +130 -3420 -420 +210), nous pouvons constater que le cycle a été détecté parce qu'un chemin (ici 210) est apparu deux fois : une première fois avec le signe **+** puis une seconde fois avec le signe **-**.

Définition 2. *Nous appelons état d'un chemin le signe (+ ou -) qui lui est associé.*

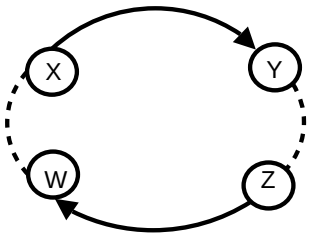

FIG. 6.1 – Circuit dans le graphe de conflit

Généralisons ainsi :

Théorème 2. *En cas d'oscillation, tous les chemins du circuit impliqué dans le graphe de conflit subiront un changement d'état (par la suite nous dirons qu'un tel circuit est un circuit oscillant).*

Preuve : Soit le circuit oscillant C. Si C oscille alors forcément au moins un de ses chemins subit un changement d'état sinon il serait stable. Or, d'après la définition des arcs de transmission et de conflit, si deux chemins X et Y sont reliés par un arc cela signifie que Y dépend de X. Donc si X subit un changement d'état, Y en subira un aussi. On en déduit que tous les chemins du circuit subiront un changement d'état.

□

Remarques

- Le passage d'un état + à un état + (resp. d'un état - à un état -) nécessite forcément le passage par un état - (resp. par un état +) entre ces deux états.
- Au démarrage du protocole, les AS, n'ayant aucune connaissance des chemins de leurs voisins, choisissent comme chemin soit un chemin direct vers la destination, soit ϵ. De ce fait, chaque AS commencera par attribuer un + à un de ses chemins.

Théorème 3. *Si nous sommes en présence d'un circuit oscillant, au premier parcours de ce circuit, au moins un des chemins impliqués passera d'un état + à un état -.*

Preuve : Si un circuit est détecté dès le démarrage du protocole ce théorème est justifié par la remarque précédente.

Supposons que tous les chemins soient passés d'un état - à un état +. Un circuit dans le graphe de conflit contient au moins deux arcs de conflit. Soit X et Y deux chemins impliqués dans le circuit tel qu'il existe un arc de conflit de X vers Y (cf. figure 6.1). L'arc signifie que si le chemin X est choisit, le chemin Y ne peut être adopté. Donc si X est passé d'un état - à un état +, cela signifie que X vient d'être adopté. Par conséquent, Y, d'après la définition de l'arc de conflit, ne peut plus être choisi et de ce fait, ne peut pas passer dans un état +. Nous pouvons en déduire qu'il ne peut pas y avoir uniquement des passages d'un état - à un état +. D'après le théorème précédent, tous les chemins subissent une oscillation, donc l'oscillation subit par Y ne pouvant être dû à un passage d'un état - à un état +, est forcément dû à un passage d'un état + à un état -.

□

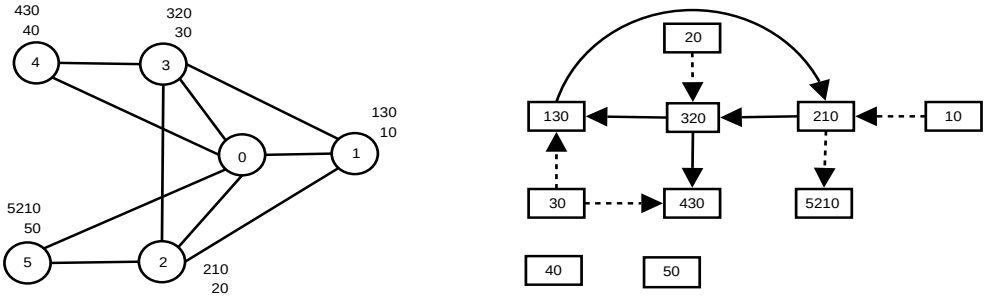

FIG. 6.2 – BAD GADGET avec oscillation du chemin 5210 et son graphe de conflit

6.2.2 Oscillation et circuit

Ces théorèmes nous permettent de considérer uniquement les passages des chemins d'un état + à un état -. On dira qu'il y a une oscillation sur un chemin s'il est passé d'un état + à un état -.

Il est important de souligner que le changement de signe pour un même chemin témoigne d'une oscillation mais pas forcément de l'appartenance à un circuit. La figure 6.2 illustre ce phénomène puisque l'AS 5, par exemple, va avoir une oscillation sur son chemin 5210 due au chemin 210 appartenant au circuit de BAD GADGET.

Seule l'interdiction d'un chemin impliqué dans un circuit résoud l'oscillation. Dans notre cas l'interdiction de 130, 210 ou 320 résoudra l'oscillation mais celle de 5210 ne la résoudra pas. Nous montrons, dans la section suivante, comment détecter ce chemin.

6.3 Identification des chemins oscillants

Chaque AS conserve uniquement les changements d'état de ses chemins. De cette manière, on obtient une gestion locale de l'état des chemins et un allègement de l'espace mémoire considérable. De plus, les messages échangés ne sont plus que les chemins. Dans le tableau 6.1, lors du passage de l'étape 5 à l'étape 6, l'AS 2 détecte une oscillation sur le chemin 210. Si ce chemin fait partie d'un circuit, il pourra comme les autres chemins du circuit être considéré comme un "mauvais" chemin.

	AS1			AS2			AS3			AS4		
step	130	10	rib-in	210	20	rib-in	3420	30	rib-in	420	430	rib-in
1	*	*	10	*	*	20	*	*	3420	*	*	420
2	*	*	10	+	*	210	*	*	3420	*	*	420
3	*	*	10	+	*	210	*	*	3420	-	*	ϵ
4	*	*	10	+	*	210	-	*	30	-	*	ϵ
5	+	*	130	+	*	210	-	*	30	-	+	430
6	+	*	130	-	*	20	-	*	30	-	+	430
7	+	*	130	-	*	20	-	*	30	+	+	420
8	+	*	130	-	*	20	+	*	3420	+	+	420
9	-	*	10	-	*	20	+	*	3420	+	+	420

TAB. 6.1 – Exemple de maintien de l'état des chemins pour BAD GADGET

Nous avons vu dans le tableau 6.1 une exécution séquentielle. Cela n'est pas représentatif de la réalité. En effet, l'échange des messages se fait de façon asynchrone. De plus, dans cet exemple, le chemin oscillant appartient à un circuit de conflit mais ce n'est pas toujours le cas. Il faut être capable de différencier ces cas et choisir parmi tous les chemins oscillants appartenant au même circuit, lequel sera interdit.

Nous sommes en présence d'un vrai problème d'algorithmique distribuée.

Chapitre 7

Détection et résolution des oscillations

A ce stade, nous sommes capables de détecter les chemins subissant une oscillation. Nous devons maintenant identifier les chemins appartenant à un circuit et en interdire un pour le rompre. Du fait de la gestion locale de l'historique, chaque AS va réagir en fonction de ses propres observations mais il doit le faire de façon cohérente avec les autres AS. Il y a deux problèmes à régler :

- Le premier concerne la détection d'un circuit. Nous avons vu que si un AS détecte un changement de signe (d'un état + à un état -) sur l'un de ses chemins cela ne signifie pas forcément que ce chemin appartient à un circuit.
- Le deuxième problème se pose lorsqu'un circuit est détecté. Quel chemin doit être interdit parmi tous les chemins impliqués dans le circuit ?

7.1 Détection d'un circuit

Nous allons proposer une méthode basée sur l'utilisation d'un jeton qui déterminera si un chemin, ayant subi une oscillation, appartient à un circuit dans le graphe de conflit. Cette méthode n'ajoute pas de messages complémentaires mais complète les messages initialement prévu par BGP (annonces de nouveau chemin) avec un jeton.

7.1.1 Méthode du jeton

Le principe de cette méthode consiste à ce qu'un AS, ayant détecté une oscillation sur un de ses chemins, génère un jeton associé à ce chemin puis l'expédie. Les AS recevant ce jeton ont le choix de le faire suivre ou de le supprimer. Si le jeton revient à son AS générateur, alors cet AS conclut que le chemin oscillant appartient bien à un circuit dans le graphe de conflit et interdit ce chemin pour rompre ce circuit.

La procédure est la suivante :
Lorsqu'un AS A détecte une oscillation sur un de ses chemins X, il génère un jeton associé à X et envoie son nouveau chemin Y en joignant ce jeton (ie. il envoie le message $(Y, j.h(X))$ où h est une fonction de cryptage et '.' est l'opérateur de concaténation). Si à la réception de ce message, un AS B destinataire ne doit faire aucune modification (suite à l'annonce de Y) alors il détruira le jeton. Si, au contraire, B doit modifier son choix de chemin à cause de Y, il transmettra, comme prévu dans BGP, son nouveau chemin mais il ajoutera en plus le jeton $j.h(X)$, et ainsi de suite...

Dans le cas où A reçoit d'un de ses voisins le message contenant un chemin Z et son jeton $j.h(X)$. Si Z entraîne le choix de X alors A peut conclure que X (qui était à l'origine

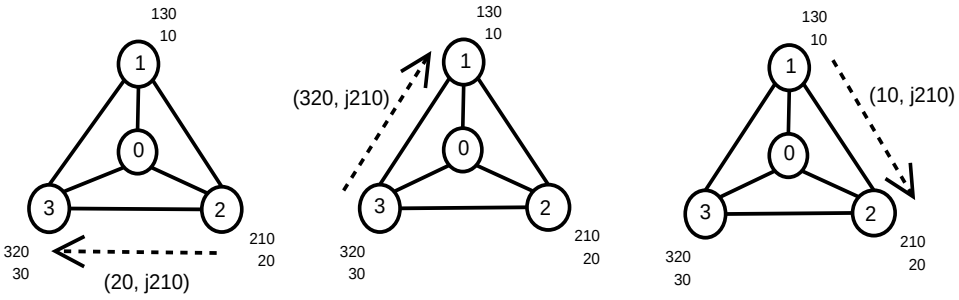

FIG. 7.1 – Echange de jeton dû à une oscillation au chemin 210

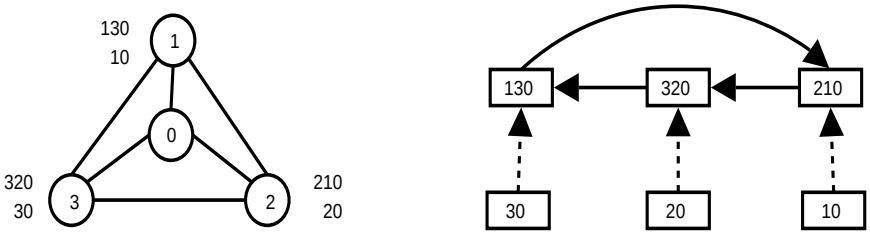

FIG. 7.2 – BAD GADGET et son graphe de conflit

de la génération du jeton $j.h(X)$) appartient à un circuit.

Notons que la valeur du jeton ne dévoile pas le chemin oscillant grâce à la fonction de cryptage qui affectera une valeur au jeton en fonction du chemin oscillant. Lorsque B reçoit l'information $j.h(X)$, il sait qu'il s'agit d'un jeton mais ne sait pas quel chemin oscillant à entrainé sa génération. Seul l'AS A est capable de retrouver le chemin à l'origine de la génération de ce jeton (en l'occurence ici, X).

La figure 7.1 est un exemple d'échange de jeton. L'AS 2 découvre une oscillation sur le chemin 210. Il génère un jeton associé à ce chemin ($j210$) et envoie son nouveau chemin (20,$j210$) en joignant le jeton. A la réception, l'AS 3 modifie son chemin en adoptant 320 donc il fait suivre le jeton avec l'annonce (320,$j210$). L'AS 1 reçoit le message provenant de l'AS 3 et fait suivre le jeton avec sa modification (10,$j210$). L'AS 2 reçoit son jeton et le chemin associé entraîne le choix de 210. Donc 210 est considéré comme un "mauvais chemin". L'interdiction de ce chemin aboutira à une solution.

7.1.2 Pourquoi cette méthode fonctionne ?

Prenons le graphe de conflit de BAD GADGET (figure 7.2).

Nous allons voir qu'il existe une dualité entre le circuit du jeton et le circuit dans le graphe de conflit.

Lorsque l'AS 2 a détecté une oscillation sur 210, c'est à dire que le chemin 210 est passé d'un état + à un état -, il a choisi un autre chemin que 210. Ainsi l'AS 3 pourrait prendre le chemin 320 rendant impossible le choix du chemin 130 pour l'AS 1. De ce fait, l'AS 2 pourrait reprendre le chemin 210 et conclure ainsi que 210 fait bien partie du circuit. Si le circuit oscille, tous les évènements cités de manière hypotétique précédement vont se produire et à chacun de ces évènements le jeton sera généré pour le premier et transmis pour

les autres. Le jeton retrace exactement la signification du circuit du graphe de conflit.

7.1.3 Perte et duplication d'un jeton

Les problèmes de perte ou de duplication d'un jeton sont classiques et doivent être gérés. Nous devons d'abord rappeler que le protocole BGP s'appuie sur le protocole TCP. Ce protocole gérant déjà la perte et la duplication de messages, nous pouvons admettre que ces évènements ne peuvent se produire.

7.2 Comment choisir le chemin à interdire ?

Lors de la détection d'un circuit, tous les AS contenant un chemin impliqué dans ce circuit vont constater une oscillation. Ils vont tous générer un jeton. Si aucune intervention n'est faite, tous les AS retrouveront leurs jetons et donc supprimeront le chemin associé. Même si ce système pourrait permettre d'arriver à un état stable, il n'est pas raisonnable d'interdire tous les chemins impliqués.

Pour résoudre l'oscillation, la suppression d'un seul chemin appartenant au circuit oscillant suffit. Tous les chemins du circuit sont suceptibles d'être interdits. Il nous est impossible de choisir un chemin sous prétexte que l'on aboutirait à une solution meilleure qu'une autre puisque la connaissance de tout le graphe de conflit serait nécessaire. Or dans notre cas, les AS disposent uniquement d'informations locales.

Notre solution consiste à définir un ordre total sur les jetons. Cet ordre permettra, lors de l'échange des jetons, de faire suivre uniquement les jetons prioritaires par rapport à la relation d'ordre jusqu'à ce qu'il n'en reste plus qu'un seul. De cette manière, seul l'AS qui retrouvera son jeton interdira le chemin associé.

On notera $\prec$ cet ordre. Les valeurs des jetons définies en fonction des chemins oscillants doivent respecter cet ordre. Ainsi, lorsqu'un AS générateur d'un jeton $j1$ reçoit un jeton $j2$, il fait le test $j1 \prec j2$. Si c'est vrai, l'AS ne fera pas suivre le jeton $j2$.

Dans l'exemple de la figure 7.3, les trois AS 1, 2, et 3 ont des chemins oscillants qui sont respectivement 130, 210 et 320 (ces chemins appartiennent à un circuit dans le graphe de conflit). Prenons le cas simple où la valeur attribuée au jeton est la lettre "j" concaténée avec le chemin associé. Par exemple l'AS 1 génère le jeton $j130$. Prenons $\prec$ la relation d'ordre lexicographique. Lorsque l'AS 1 recevra le jeton $j320$, il le supprimera. Même constat lorsque l'AS 1 recevra le jeton $j210$. A la fin, seul le jeton $j130$ reviendra à l'AS d'origine et le chemin qui sera supprimé sera 130.

Il est à noter qu'un choix beaucoup plus fin de la relation d'ordre permettrait d'obtenir plus souvent des résultats se rapprochant de l'optimum.

Remarque

Nous devons souligner qu'il est possible dans certains cas qu'un AS générateur d'un jeton, le retrouve sans que ce chemin n'appartienne à un circuit de conflit. Ce phénomène, illustré par la figure 7.4, est dû aux arcs de transmission. En effet les AS 1 et 2 verront une oscillation sur leur chemin respectif 1240 et 240. Dans le graphe de conflit il existe un arc de transmission de 240 à 1240. Il est clair que si le chemin 240 oscille, le chemin 1240 oscillera aussi puisqu'il dépend entièrement de 240. Donc l'interdiction de 1240 ne résoudra pas l'oscillation et pourtant si la priorité est donnée au plus petit jeton selon l'ordre lexicographique c'est ce chemin qui sera interdit. Une solution consisterait à détecter les cas de "sous

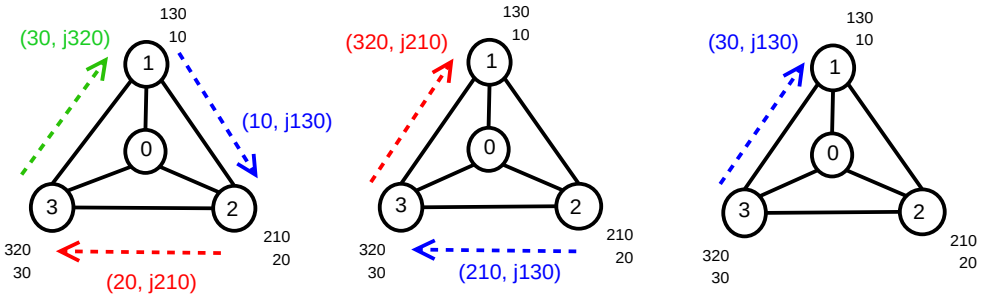

FIG. 7.3 – Simulation de l'utilisation du jeton

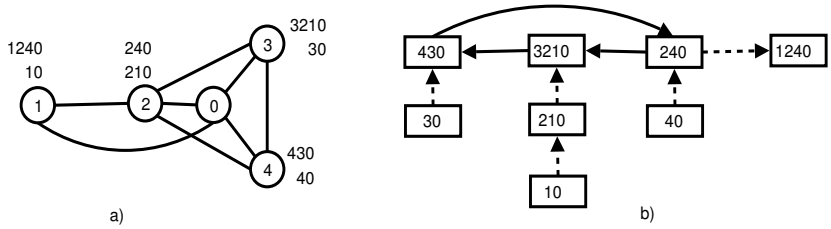

FIG. 7.4 – Variante du BAD GADGET qui peut provoquer une erreur à cause des arcs de transmission

chemin" (ici 240 sous chemin de 1240) à ce moment là l'AS 2 aurait détruit le jeton 1240 et c'est le chemin 240 qui aurait été interdit. Cette solution n'est pas envisageable puisqu'elle oblige l'AS 1 à communiquer l'identité de son chemin oscillant ce qui est contradictoire avec le principe de confidentialité des politiques. L'AS 1 communiquera uniquement la longueur du chemin oscillant c'est à dire 4. Il suffit maintenant de donner la priorité au jeton dont la longueur du chemin associé est la plus courte.

La figure 7.3 illustre la gestion des jetons par les AS.

La procédure en cas de détection d'une oscillation est donnée par l'algorithme 1 et la gestion des jetons est donnée par l'algorithme 2. Ces algorithmes se trouvent en annexe.

Limite de la méthode

Certaines situations peuvent amener notre processus à interdire des chemins inutilement ou interdire plus de chemins qu'il n'aurait fallu. Nous allons voir deux exemples qui illustrent ce phénomène :

- Le premier exemple (figure 7.5) montre une situation dans laquelle le chemin 120 sera interdit alors qu'il existe une configuration stable permettant son choix. Le BAD GADGET présent dans cette figure peut être évité si l'AS 3 choisit le chemin 376540. A ce moment là, l'AS 2 choisira le chemin 20 et l'AS 1 le chemin 120. Or il est possible que l'adoption du chemin 376540 mette du temps à arriver laissant la possibilité au BAD GADGET de faire osciller les chemins 120, 230 et 210 jusqu'à arriver à l'interdiction de 120. Cette interdiction n'est pas légitime. Ce phénomène est équivalent au cas des apparitions de liens. Nous proposons une solution dans une section suivante.
- Le deuxième exemple (figure 7.6) montre que notre méthode aboutira à une solution après avoir interdit deux chemins alors qu'un seul aurait suffit. En utilisant notre mé-

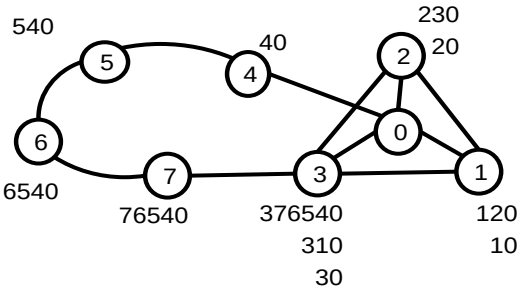

FIG. 7.5 – Système stable aboutissant pourtant à une interdiction

thode sur la figure 7.6, les chemins 130 puis 150 seront interdits, alors que l'interdiction du chemin 320 aurait suffit. Pour résoudre ce problème il faudrait avoir la connaissance du graphe ce qui n'est pas envisageable. Ceci dit, l'interdiction d'un seul chemin au lieu de deux ne garantit pas forcément une meilleure solution. Donc est ce vraiment une limite ?

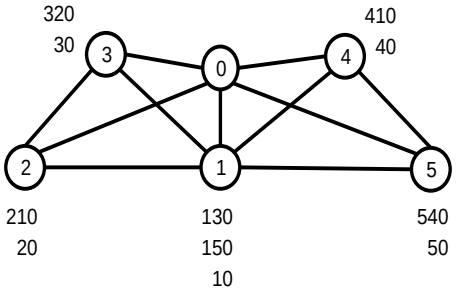

FIG. 7.6 – Deux BAD GADGET imbriqués conduisant à deux interdictions au lieu d'une seule

Une autre question se pose : l'interdiction d'un chemin pour résoudre une oscillation ne risque-t-elle pas de créer une oscillation encore plus grande ?

Théorème 4. *La résolution d'une oscillation n'engendre pas une oscillation plus grande (ie. avec un nombre AS impliqués plus important que dans l'oscillation précédente).*

Preuve : Considérons la figure 7.7. Le système, appelé $S1$, constitué des AS u_1,...,u_n oscille. Nous voulons savoir si la résolution de ce système pourrait entraîner l'oscillation du système en "pointillé", appelé $S2$.

On considère l'AS v, relié avec l'AS u_1. v n'a détecté aucune oscillation. On peut en déduire deux cas :
Soit les chemins de v n'ont aucun lien avec les chemins de u_1. Soit les chemins qui devraient subir une oscillation (par exemple y et z) ne sont jamais atteints parce que l'AS v préfère le chemin x. Si c'est le cas, cela signifie que la configuration d'un de ses voisins lui permet de garder x. Le voisin qui lui permet ceci ne peut pas être oscillant sinon x ne pourrait pas être conservé. On en déduit que quelles que soient les interdictions que les chemins de u_1 pourront subir, elles n'engendreront pas de changement de configuration pour v. Il en est de même pour w et u_2. Donc, en résolvant l'oscillation du système $S1$, la grande oscillation du système $S2$ ne pourra jamais se produire.

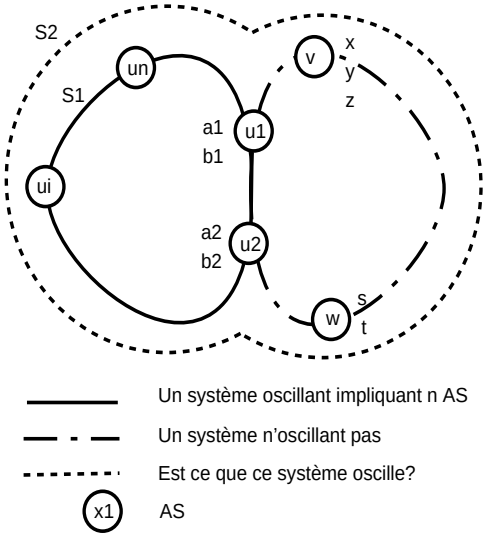

FIG. 7.7 – La résolution du système u_1,...,u_n, n'engendre pas une oscillation sur le système en pointillé

□

Remarque
Intéressons nous à la mise en route du protocole. Pendant cette phase, beaucoup d'oscillations seront détectées aboutissant à des interdictions parfois inutiles dues au démarrage asynchrone du protocole. Une idée consiste, pour chaque AS démarrant, à ne déclencher notre processus qu'après avoir détecté un certain nombre d'oscillations sur le même chemin. Ce nombre pourrait être égal au rapport entre la longueur maximum entre deux AS et la longueur moyenne d'un cycle dans un réseaux. Après cette phase de démarrage, les AS déclencheront le processus dès qu'ils détecteront une oscillation. Ce procédé évite ainsi des interdictions inutiles provoquées par des cas cités précédemment.

7.3 Gestion des pannes et des apparitions de liens

BGP est un protocole autostable, c'est à dire que quelque soit la configuration dans laquelle se trouve le réseau, les AS vont s'échanger leurs chemins jusqu'à arriver à stabilité (si les politiques sont cohérentes entre elles, cf. chapitre suivant).

Nous avons proposé une notion d'interdiction de chemins. Par conséquent, nous devons adapter la gestion des pannes et des apparitions de liens à l'interdiction des chemins.

Il existe deux sortes de panne : les pannes de routeurs et les pannes de liaisons. Nous nous intéressons à la gestion des pannes de liaisons ; elle s'étend facilement pour les pannes des routeurs.

Les apparitions des liens peuvent se produire lorsqu'un AS apparaît pour la première fois dans le réseau ou bien lorsqu'un lien est rétabli à la suite d'une panne.

Lorsqu'une panne de liaison ou une apparition d'un lien se produit, il est possible qu'un chemin préalablement interdit puisse être réhablité.

Une solution simple consisterait à ce que chaque AS, ayant des chemins interdits, déclenche périodiquement un processus de test de réhabilitation, décrit dans une prochaine sec-

tion, sur ses chemins. Nous allons proposer une solution plus fine ciblant les AS concernés par une panne ou une apparition de lien.

7.3.1 Processus d'annonce

Nous allons modifier légèrement le processus de détection d'une oscillation à base d'un jeton. Dans cette solution, chaque AS générant ou faisant suivre un jeton joint son identifiant au message. Lorsqu'un AS générateur d'un jeton Y, reçoit un jeton X prioritaire par rapport à Y, il fait suivre le jeton X en ajoutant son identifiant et l'information "OK" signifiant que, pour lui, le jeton X reste prioritaire. Si X n'est pas prioritaire nous ne détruisons plus le jeton comme nous l'avions fait jusque là, mais nous le faisons suivre avec l'information "NOK" signifiant que le jeton n'est pas prioritaire. Notons que si un AS reçoit un jeton ayant déjà l'information "NOK" alors l'AS fera suivre ce jeton mais ne modifira pas cette information. Ainsi, lorsqu'un AS récupère son jeton, il vérifie s'il porte l'information "OK", auquel cas il interdit le chemin associé. Dans les deux cas ("OK","NOK"), les AS générateurs de jetons, connaissent les identifiants de tous les AS impliqués dans la même oscillation.

Lorsqu'une panne ou une apparition de lien se produit, le ou les AS impliqués (par exemple, dans le cas d'une pannes de liaison les deux AS se trouvant aux extrêmités) peuvent avertir, grâce aux identifiants, les AS concernés. Si un AS possédant un chemin interdit, reçoit cette information, il lance le processus de test de réhabilitation sur ce chemin afin de savoir s'il peut être réhabilité ou non.

7.3.2 Processus de test de réhabilitation

Ce processus à pour but de tester si un chemin peut être réhabilité. L'AS possédant un tel chemin, simule sa réhabilitation à l'aide d'un jeton blanc (ou jeton de simulation). Les voisins le font suivre si la réhabilitation de ce chemin leurs fait modifier leur choix sinon, ils envoient un accusé de réception. S'il récupère son jeton alors l'AS générateur ne réhabilitera pas le chemin puisqu'il génèrerait une oscillation. Sinon, s'il reçoit des accusés de réception de la part de tous ses voisins il réhabilitera le chemin.

Le principe est le suivant :
Considérons un chemin X interdit par un AS A. Nous voulons savoir si ce chemin peut être réhabilité. A génère un jeton blanc en fonction du chemin X et il conserve l'information (NIL, jBX, l). Le premier champ correspond à l'identifiant de l'AS qui a annoncé à A le jeton blanc (en l'occurence ici, vu que A est le générateur, ce champ est à NIL). Le deuxième correspond au jeton blanc et le troisième au nombre de liaisons sur lesquelles A a envoyé le jeton blanc. Il expédie à ses l voisins le jeton et le chemin X (X, jBX). Ce jeton simule le cas où la route X serait réhabilitée. Chaque AS recevant ce jeton ne modifiera pas son chemin mais indiquera le chemin qu'il choisirait en fonction du chemin joint au jeton. Si le chemin joint n'entraîne aucun changement de chemin pour un AS alors il envoie un accusé de réception à son AS annonceur qui décrémentera de un le compteur du nombre de messages envoyés. Lorsqu'un AS possède un compteur égal à zéro, il envoie un accusé de réception à son AS annonceur et ainsi de suite. Sinon, il fait suivre le jeton blanc avec le chemin qu'il aurait choisi et conserve l'information $(idAnnonceur, jBX, l')$. Si le jeton revient à A et le chemin associé entraîne l'impossibilité de choisir X alors le test a échoué et le chemin X ne peut pas être rétabli. Par contre, s'il reçoit l accusés de réception alors il peut réhabiliter le chemin puisqu'il n'engendre plus d'oscillation.

Complexité :

Si on considère le sous graphe constitué des n AS concernés par la réhabilitation du chemin X, ce processus tracera un arbre recouvrant sur ce sous graphe. Cet arbre est enraciné par l'AS A et les feuilles sont les AS ne pouvant plus faire suivre le jeton blanc. Donc en $O(h)$ étapes (h étant la hauteur de l'arbre) l'AS A sait s'il doit réhabiliter ou non le chemin X. Chaque arête de l'arbre transmet au plus deux informations : le jeton blanc et l'accusé de réception. Si m est le nombre d'arêtes de l'arbre, la complexité de cette procédure en nombre de messages est de $2m$.

Un exemple est donné plus loin.

Le processus de test de réhabilitation est décrit par l'algorithme 3. La réception d'un jeton blanc est décrite par l'algorithme 4. Ces algorithmes se trouvent en annexe.

Remarque
Le jeton blanc ne contient aucune information permettant aux AS le recevant de connaître quel chemin pourrait être réhabilité. Lorsque l'AS B reçoit de l'AS A le message (X,jBX), il ne sait pas si A veut réhabiliter X ou s'il est en train de faire suivre un jeton avec son propre choix liée à une réhabilitation. Par conséquent, le processus de test de réhabilitation respecte également le principe de confidentialité des politiques.

7.3.3 Panne de liaison

Lorsqu'une panne de liaison se produit, les deux AS à l'extrémité de cette liaison vont s'en rendre compte. Si ces AS avaient généré un jeton suite à une oscillation, ils savent que l'interdiction du chemin qui avait eu lieu pour résoudre cette oscillation est maintenant illégitime. Malheureusement, BGP n'impose aucune annonce si aucun changement de route n'a lieu. Donc ces AS se doivent d'en informer les autres AS afin de réhabiliter éventuellement ce chemin. C'est l'AS possédant le plus petit identifiant entre les deux qui se chargera d'annoncer cette information (ce choix est totalement arbitraire, il permet de transmettre une seule fois l'information). Grâce aux identifiants joints avec le jeton, il connaît tous les AS impliqués et les prévient en leur envoyant un message les informant de la panne. Les AS, possédant des chemins interdits, vont lancer le processus de test de réhabilitation sur ces chemins et réhabiliteront ceux ayant réussi le test.

Prenons l'exemple de la figure 7.8 a). Il s'agit d'un BAD GADGET avec six AS où notre méthode a abouti à l'interdiction du chemin 130. Supposons que le lien reliant les AS 4 et 5 tombe en panne (cf. figure 7.8 b)).

Lorsque les AS 4 et 5 prennent connaissance de la panne, l'AS 4 avait généré un jeton dû à l'oscillation du chemin 420. Il annonce l'information de la panne à tous les AS qui étaient impliqués dans cette oscillation (cf. figure 7.8 c)). Les AS 2 et 3 ne possèdent pas de chemin interdit. Par contre l'AS 1 lance le processus de test de réhabilition sur le chemin 130 : l'AS 1 génère un jeton blanc associé à 130 ($jB130$), envoie ce jeton avec le chemin 130 (130,$jB130$) (cf. figure 7.8 d)) et conserve l'information $(NIL, jB130, 2)$. L'AS 2 et 3 reçoivent cette information, se rendent compte qu'il s'agit d'une simulation grâce au jeton blanc et regardent si le chemin 130 les feraient changer de chemin. L'AS 3 n'aurait aucune modification à faire et par conséquent, envoie un accusé de réception à l'AS 1 qui décrémente son compteur de messages envoyés. Par contre, l'AS 2 a adopté le chemin 210 donc il modifierait son chemin en choisissant 20. Il fait suivre le jeton blanc avec le chemin 20 (20,$jB130$) (cf. figure 7.8

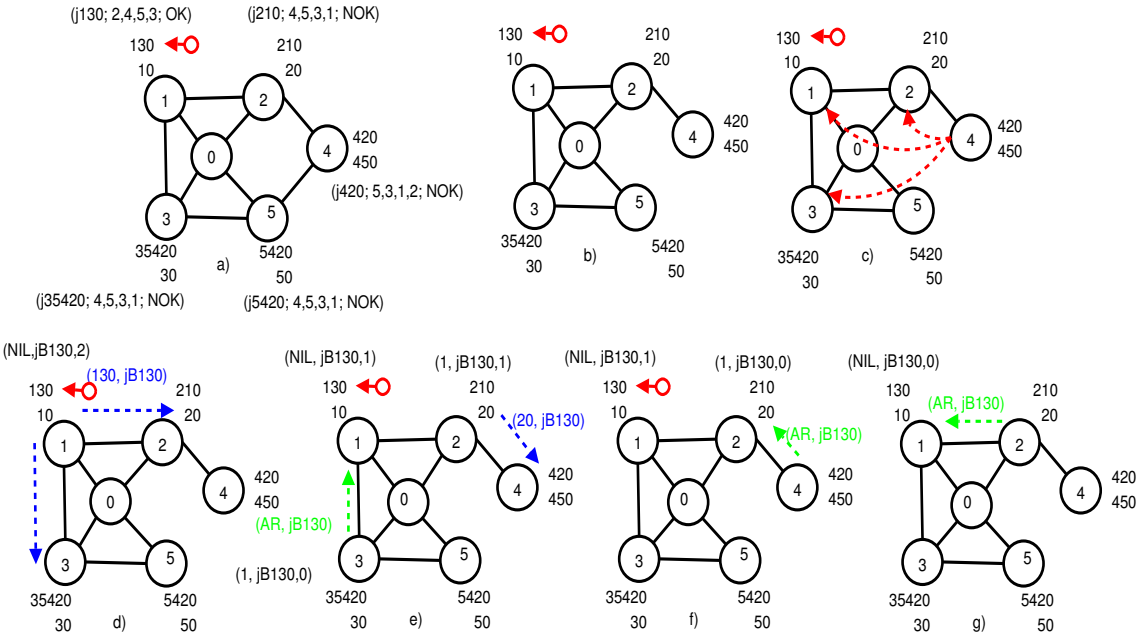

FIG. 7.8 – Illustration d'une panne dans un BAD GADGET à 6 sommets

e)) et conserve l'information $(1, jB130, 1)$. L'AS 4 reçoit cette information et pourrait ainsi prendre 420 mais il n'a plus de voisin donc il envoie un accusé de réception à l'AS 2 qui décrémente son compteur valant maintenant zéro ((cf. figure 7.8 f)). Lui aussi envoie un accusé de réception à l'AS 1 qui décrémente son compteur et réhabilite la route 130 puisque le compteur vaut zéro. De cette manière le système, aboutira aux choix des chemins suivants : (130 20 30 420 50).

7.3.4 Apparitions ou rétablissements de liens

Intéressons nous maintenant aux cas des apparitions de liens. Si un AS A, ayant généré un jeton pour un chemin X lors d'une oscillation précédente, est ammené grâce à l'apparition d'un nouveau lien, à choisir un chemin Y. Si Y est préféré à X, il sait que l'interdiction qui s'était produite pour résoudre cette oscillation n'est plus légitime. Il utilise le processus d'annonce décrit précédemment pour avertir tous les AS concernés par l'oscillation.

Prenons l'exemple illustré par la figure 7.9 a). Avant l'apparition de l'AS 6 (cf. 7.9 b)), nous avions un BAD GADGET ayant conduit à l'interdiction du chemin 130. L'AS 2 avait récupéré son jeton avec les identifiants des AS impliqués dans l'oscillation (3, 1, 4) et l'information "NOK". Lorsque l'AS 6 apparaît, l'AS 2 améliore son chemin en choisissant le chemin 260 meilleur que 2410. Il avertit les AS qui étaient impliqués dans l'oscillation pour qu'ils puissent réhabiliter les chemins illégitiment interdits (cf. 7.9 c)). Dans cet exemple, les identifiants des AS impliqués sont 2, 3, 1 et 4. Les AS 3 et 4 n'avaient aucun chemin interdit, donc ils ignorent l'information. Par contre l'AS 1 lance le processus de test de réhabilitation sur le chemin 130 et le réhabilitera (cf. 7.9 d) , e) et f)). On aboutira à la configuration (130, 260, 30, 4130).

La détection d'une panne est décrite par l'algorithme 5 et celle d'une apparition de lien par l'algorithme 6. La réception d'une annonce est décrite par l'algorithme 7. Ces algorithmes

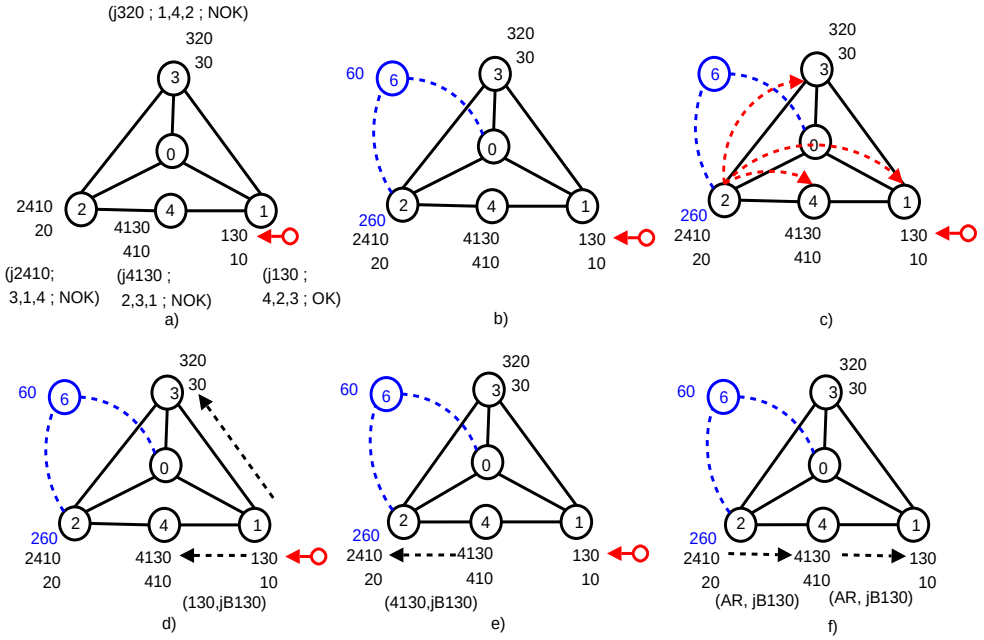

FIG. 7.9 – Le chemin 130 est interdit et l'AS 6 apparaît

se trouvent en annexe.

Généralisation

Cette stratégie est également valable dans le premier exemple cité dans les limites de la méthode du jeton. Supposons qu'un AS, ayant généré un jeton en fonction d'un chemin X suite à une oscillation précédente, puisse désormais adopter un chemin Y. Si Y est préféré à X, il réagit de la même manière que dans le cas des apparitions de liens.

Remarque

On pourrait se demander l'utilité du test de réhabilitation. En effet lorsqu'une panne de liaison de produit, les deux AS aux extrèmités savent si cette panne résoud l'oscillation dans laquelle il étaient impliqués, puisqu'ils ont les identifiants de tous les AS impliqués. Idem dans le cas des apparitions de liens.

Dans les deux cas, l'AS annonceur est sûr que l'interdiction préalable d'un chemin n'est plus légitime. Alors pourquoi tester si ce chemin peut être réhabilité ? Si nous sommes sûrs que la réhabilitation de ce chemin n'engendrera pas la même oscillation que celle ayant conduit à son interdiction, nous ne sommes pas sûrs qu'elle n'en engendre pas une autre. Ce phénomène est illustré par la figure 7.10. Supposons qu'après la résolution du BAD GADGET des sommets 1, 2 et 3, l'AS 4 apparaît en proposant les chemins 410 et 40. L'AS 3 ajoute dans sa liste de chemins le chemin 340. Supposons maintenant que la liaison 23 tombe en panne. Alors le test de réhabilitation échouera puisque la réhabilitation de 130 engendrerait une oscillation sur les AS 1, 3 et 4.

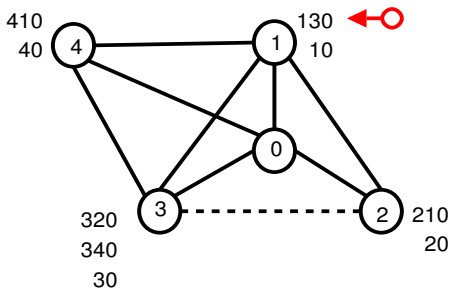

FIG. 7.10 – La réhabilitation de 130 engendre une nouvelle oscillation

Chapitre 8

Cohérence globale des politiques

Nous avons vu que toutes les oscillations étaient dues à une incohérence des politiques entre elles. Nous savons que lorsque ces politiques sont cohérentes, BGP est auto-stable. Une question naturelle se pose : “A quelles conditions des politiques sont cohérentes entre elles ?”.

8.1 Caractérisation

Soit $<_\alpha$ une relation entre les chemins. Nous allons définir cette relation localement, c’est à dire entre les chemins d’un même AS, et globalement, c’est à dire sur les chemins inter AS. Pour fixer les idées nous pouvons interpréter $<_\alpha$ par *“meilleur que”*.

Définition 3. $<_\alpha$ ***locale :*** *Pour un AS A, $\forall\ P, Q$ chemins de A, si P est préféré à Q alors $P <_\alpha Q$.*

Cette définition est cohérente avec les politiques adoptées par les différents AS.

Définition 4. $<_\alpha$ ***globale :*** *$\forall\ P, Q$ chemins appartenant à deux AS différents, si P est un sous chemin de Q alors $P <_\alpha Q$.*

Cette définition permet de garder la cohérence des politiques entre les AS. Il est clair que tous les chemins qui passent par un sous chemin P ne devraient pas être meilleurs que P.

Théorème 5. *Si $<_\alpha$ est une relation d’ordre stricte alors les politiques sont cohérentes entre elles.*

Preuve : Nous allons voir qu’il existe une équivalence entre cette caractérisation de la cohérence des politiques et le graphe de conflit.

- Soit un arc de transmission de P vers Q. D’après la définition de cet arc, P est un sous chemin de Q, donc en utilisant la définition de l’ordre global on a $P <_\alpha Q$.
- Soit un arc de conflit de P vers Q. D’après la définition de cet arc, le choix de P interdit le choix de Q. Il existe un chemin R appartenant au même AS que P, avec une préférence inférieure à ce dernier, qui permettrait le choix de Q. Cela signifie qu’au niveau local, on a $P <_\alpha R$ et, qu’au niveau global, on a $R <_\alpha Q$ puisque R est un sous chemin de Q. Par transitivité on a $P <_\alpha Q$.

Il est clair maintenant que l’absence d’un circuit dans le graphe de conflit permet de conclure que $<_\alpha$ est une relation d’ordre stricte. On sait que l’absence d’un circuit dans le graphe de conflit témoigne de la cohérence des politiques entre elles. Donc si $<_\alpha$ est une relation d’ordre stricte, les politiques sont cohérentes entre elles.

□

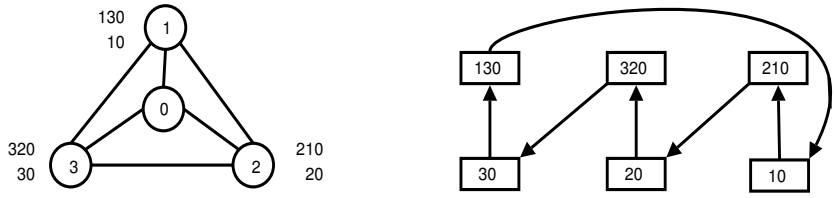

FIG. 8.1 – Nouveau graphe de conflit

Prenons comme exemple BAD BADGET (figure 8.1). D'après la relation d'ordre stricte, on a :

$$130 \underbrace{<_\alpha}_{local} 10 \underbrace{<_\alpha}_{global} 210 \underbrace{<_\alpha}_{local} 20 \underbrace{<_\alpha}_{global} 320 \underbrace{<_\alpha}_{local} 30 \underbrace{<_\alpha}_{global} 130.$$

On aboutit à une contradiction, donc les politiques établies dans BAD GADGET ne sont pas cohérentes entre elles.

8.2 Un nouveau graphe de conflit

Nous pouvons maintenant proposer un nouveau graphe de conflit définit de la façon suivante : Soit P et Q deux chemins, il y a un arc de P vers Q si $P <_\alpha Q$. S'il n'existe pas de circuits dans ce graphe alors il existe une solution stable. Un exemple de ce nouveau graphe de conflit est donné figure 8.1.

Chapitre 9

Conclusion et perspectives

D'après [16], des cas d'oscillations de BGP ont déjà été constatés. BGP étant le protocole de routage inter AS, il est nécessaire de résoudre rapidement ces oscillations, car le fonctionnement d'Internet en dépend.

Dans ce mémoire, nous avons étudié et caractérisé les cas d'oscillations provoqués par l'incohérence globale des politiques adoptées par les AS. Nous avons proposé une méthode dynamique de détection et de résolution des oscillations, basée sur un maintien des états des chemins et le principe du jeton.

Cette méthode respecte les contraintes imposées par le protocole BGP. Les contraintes sur les choix des routes sont très légères et le principe de confidentialité des politiques est conservé. Notre méthode devait s'adapter à la gestion des pannes et d'apparitions de liens de BGP. Nous avons proposé un processus d'annonce des pannes et de nouveaux liens réhabilitant des chemins préalablement interdits.

Nous avons également souligné le temps de convergence important que mettait BGP avant d'arriver à stabilité (dans le cas où c'est possible). Nous devions faire en sorte de ne pas aggraver ce temps de convergence. Seule l'information du jeton est ajoutée aux messages échangés par BGP ce qui semble raisonnable. Une solution compatible avec notre méthode, est proposée dans [17] pour accélérer dans certains cas ce temps de convergence. Enfin, nous avons définie la notion de cohérence des politiques entre elles permettant de conclure si une configuration des réseaux est susceptible d'avoir des cas d'oscillations.

Nous avons développé un simulateur reproduisant la vision abstraite du protocole BGP (ie. SPP). Notre solution se comporte bien sur les échantillons de tests que nous lui avons fait subir. Néanmoins, il serait intéressant de tester cette solution sur un simulateur reproduisant exactement le comportement du protocole BGP (MED, agrégation de route, ...). Une équipe à Versailles, travaillant sur ce projet, développe actuellement un simulateur BGP. Nous avons également récupéré le simulateur C-BGP [18] et la suite de ce projet consistera à implémenter notre solution sur un de ces simulateurs.

Il sera nécessaire d'étudier les comportements byzantins que peuvent adopter les AS. Les comportements égoïste en sont un exemple. Si un AS ne veut pas interdire un chemin lorsqu'il détecte une oscillation, il ne génère pas de jeton et fait suivre tous les autres jetons qu'il reçoit. De cette manière, il aura conservé son chemin puisque ce sera un autre AS qui interdira un chemin. Une solution consisterait à utiliser un système de signature. Chaque AS signe les informations qu'il transmet. Ainsi, le routeur présentant un comportement anormal sera identifié.

Pour généraliser, il faut appliquer un système de protection sur l'ensemble des messages échangés. La couche TCP sur laquelle repose le protocole BGP permet une sécurisation des données échangées (voir [19]). Quelques études ont déjà été faites sur ce sujet dans [20][21]. On pourrait également s'inspirer de ce qui a été fait dans les réseaux peer-to-peer en s'appuyant sur [22][23].

Enfin, il sera primordial de se pencher sur le problème de la connectivité des AS. En effet, aujourd'hui BGP peut converger vers une solution stable en laissant des AS sans chemin pour aller vers une destination. Le but sera de détecter et de résoudre ces cas.

Chapitre 10

Annexe

Algorithm 1: Détection d'une oscillation pour un AS u

Data : Tableau T de l'état des chemins
Result : Génère un jeton en cas d'oscillation et le diffuse aux voisins

/* detectOscillation(T) retourne le chemin oscillant */
/* demandeur = vrai si u a généré un jeton, faux sinon */
/* creatToken(op) créer un jeton en fonction du chemin oscillant op */
/* choicePath() retourne le meilleur chemin selon la politique de l'AS */

$demandeur \leftarrow false$;
$oscillantPath \leftarrow detectOscillation(T)$;
if $oscillantPath \neq null$ **then**
    $demandeur \leftarrow true$;
    $token \leftarrow creatToken(oscillantPath)$;
    $newPath \leftarrow choicePath()$;
    for $v \in N(u)$ **do**
        $send(newPath, token, v)$;
    end
end

Algorithm 2: Réception d'un jeton pour un AS u

Data : réception du message <$oscillantPath$, $token$, v>

Result : Fait suivre ou supprime le jeton reçu

/* $path$: chemin actuellement choisi */
/* getGenerator() retourne le chemin à l'origine de la création du jeton */
/* interdire(p) interdit le chemin p */
/* lgPath(t) retourne la longueur du chemin associé au jeton t */

$newPath \leftarrow choicePath()$;
if $path \neq newPath$ **then**
  $path \leftarrow newPath$;
  /* token est prioritaire par rapport à $\prec$ ou la longueur du chemin associé à token est plus petite que la longueur du chemin associé à myToken */
  if $demandeur = vrai$ **and** ($token \prec myToken$ **or** $lgPath(token) < lgPath(myToken)$) **then**
    $demandeur \leftarrow false$;
  end
  if $demandeur = vrai$ **and** $token = myToken$ **then**
    if $path = myToken.getGenerator()$ **then**
      $interdire(path)$;
      $path \leftarrow choicePath()$;
      for $v \in N(u)$ **do**
        $send(path, v)$;
      end
    end
  end
  else
    for $w \in N(u) - \{v\}$ **do**
      $send(newPath, token, w)$;
    end
  end
end

Algorithm 3: Processus de test de réhabilitation

Data : Un chemin $barredPath$
Result : Retourne vrai si $barredPath$ peut être réhabilité, faux sinon
/* creatWhiteToken(op) créer un jeton en fonction du chemin oscillant op */
/* involve(p) retourne le chemin choisi si u connait p */

$whiteToken \leftarrow creatWhiteToken(barredPath)$;
$d \leftarrow |N(u)|$;
for $v \in N(u)$ **do**
  $send(barredPath, whiteToken, v)$;
end
while $d \neq 0$ **do**
  $reception < message >$;
  /*u récupère son jeton et le chemin associé empêche l'adoption de barredPath*/
  if $< message >=< whiteToken, p >$ **and** $involve(p) \neq barredPath$ **then**
    $returnfalse$;
  end
  /*u reçoit un accusé réception*/
  if $< message >=< whiteToken, AR >$ **then**
    $d \leftarrow d - 1$;
  end
end
$returntrue$;

Algorithm 4: Réception du jeton blanc

Data : réception du message <*p*, $whiteToken$, *v*>
Result : Fait suivre ou non le jeton blanc
/* $currentPath$ représente le chemin actuellement choisie */
/* annonceur(jB) retourne l'identifiant de l'annonceur de jB */

$path \leftarrow involve(p)$;
if $path \neq currentPath$ **then**
  $d \leftarrow |N(u) - \{v\}|$;
  for $w \in N(u) - \{v\}$ **do**
    $send(path, whiteToken, w)$;
  end
  while $d \neq 0$ **do**
    $reception < whiteToken, AR >$;
    $d \leftarrow d - 1$;
  end
  $send(whiteToken, AR, annonceur(whiteToken))$;
end
else
  $send(whiteToken, AR, annonceur(whiteToken))$;
end

Algorithm 5: Détection d'une panne pour un AS u

Result : Annonce d'une panne
/* detectPanne() retourne l'AS innaccessible */
/*listId() retourne la liste des identifiants joins avec le dernier jeton reçu */
/*id() retourne l'identifiant de l'AS */

$v \leftarrow detectPanne()$;
/* L'AS v n'est plus joignable et je suis prioritaire pour annoncer la panne */
if $v \neq null$ **and** $v.id() > u.id()$ **then**
    $List \leftarrow listId()$;
    /* On avertit tout les AS concernés*/
    for $id \in List$ **do**
        $send(id, "panne")$;
    end
end

Algorithm 6: Détection d'une apparition de lien pour un AS u

Result : Annonce d'une apparition de lien
/* detectApparition() retourne le chemin accesible grâce à l'apparition d'un lien*/
/* cheminOscillant() retourne le dernier chemin à l'origine de la création d'un jeton*/
/* prefered(p1,p2) retourne vrai si p1 est préféré à p2 */

$path \leftarrow detectApparition()$;
/* Si path est meilleur que le dernier chemin oscillant*/
if $path \neq null$ **and** $prefered(path, cheminOscillant())$ **then**
    $List \leftarrow listId()$;
    /* On avertit tout les AS concernés*/
    for $id \in List$ **do**
        $send(id, "apparition")$;
    end
end

Algorithm 7: Réception d'une annonce de panne ou d'apparition d'un lien pour un AS u

Data : réception du message <"$apparition$"> ou <"$panne$">
Result : Réhabilite les chemins illégitimement interdits
/* gestion des interdictions (avec B ensemble des chemins interdits)*/
/* rehabiliter(p) réhabilite le chemin interdit p */
/* testDeRehabilitation(p) retourne vrai si p peut être réhabilité */

for $p \in B$ **do**
    if $testDeRehabilitation(p) = true$ **then**
        $rehabiliter(p)$;
    end
end

Bibliographie

[1] Kannan Varadhan, Ramesh Govindan, and Deborah Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, pages 32 :1–16, 2000.

[2] Craig Lobavitz, G. Robert Malan, and Farnam Jahanian. Origins of internet routing instability. *in Proc. IEEE INFOCOM*, vol. 1 :pp. 218–226, 1999.

[3] Craig Lobavitz, G. Robert Malan, and Farnam Jahanian. Internet routing instability. *IEEE/ACM Trans. Networking*, vol. 6 :pp 515–528, October 1998.

[4] R. Govindan, C. Alaettinoglu, G. Eddy, D. Kessens, and W.S. Lee. An architecture for stable, analyzable internet routing. *IEEE Networks*, pages 13(1) :29–35, 1999.

[5] Timothy G. Griffin and Gordon Wilfong. An analysis of bgp convergence properties. *Proc. ACM SIGCOMM'99*, pages pp. 277–288, 1999.

[6] Timothy G. Griffin, F. Bruce Sherpherd, and Gordon Wilfong. Policy disputes in path-vector protocols. *Proc. 7th Int. Conf. Network Protocols (ICNP'99)*, pages pp. 21–30, 1999.

[7] Timothy G. Griffin and Gordon Wilfong. A safe path vector protocol. *Proc. IEEE INFOCOM*, vol.2 :pp. 490–499, 2000.

[8] Timothy G. Griffin, F. Bruce Sherpherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *Proc. IEEE/ACM Transactions on Networking*, 2002.

[9] Lixin Gao, Timothy G. Griffin, and Jennifer Rexford. Inherently safe backup routing with bgp. *in Proc. IEEE INFOCOM*, April 2001.

[10] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *in Proc. ACM SIGMETRICS*, June 2000.

[11] Selma Yilmaz and Ibrahim Matta. A randomized solution to bgp divergence. *in Proc. of the 2nd IASTED Int. Conf. on Communication and Computer Networks (CCN'04)*, November 2004.

[12] C. Huitema. Routing in the internet (2nd edition). *Hardcover*, Januar 2000.

[13] Luc Saccavini. Le routage bg-4.

[14] Timothy G. Griffin and Gordon Wilfong. Analysis of the med oscillation problem in bgp. *in Proc. of the 10th IEEE Int. Conf. on Network Protocols*, pages pp. 90–99, 2002.

[15] Ken Schummacher. Stabilite dans bgp. December 2004.

[16] Johan Nykvist and Lenka Carr-Motyčková. Simulating convergence properties of bgp. *in Proc. Computer Communications and Networks*, pages pp. 124–129, October 2002.

[17] Hongwei Zhang, Anish Arora, and Zhijun Liu. A stability-oriented approach to improving bgp convergence. *in 23rd IEEE Int. Symposium on Reliable Distributed Systems (SRDS'04)*, pages pp. 90–99, 2004.

[18] Sebastien Tandel and Cedric de Launois. C-bgp - an efficient bgp simulator. *http ://cbgp.info.ucl.ac.be/*.

[19] A. Heffernan. Protection of bgp sessions via the tcp md5 signature option. *Request for comments : 2385*, August 1998.

[20] Stephen T. Kent. Securing the border gateway protocol : A status update. *in Lecture Notes in Computer Science*, pages pp. 40–53, December 2003.

[21] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure border gateway protocol (s-bgp)- real world performance and deployment issues. *IEEE Journal on Selected Areas in Communications*, 18 :pp. 582–592, 2000.

[22] Yinglian Xie, David R. O'Hallaron, and Michael K. Reiter. A search engine on every desktop : Practical issues for p2p keyword search.

[23] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, pages 21(2) :120–126, 1978.

[24] Sylvie Delaet, Duy-So Nguyen, and Sebastien Tixeuil. Stabilite et auto-stabilisation de bgp. *Algotel*, pages pp. 183–190, 2003.

[25] Mickael Meulle and Quang Nguyen. Un modele de graphe pour le routage bgp. *Algotel*, 2004.

Résumé

Les systèmes autonomes (AS) dans l'Internet utilisent des protocoles différents pour le routage interne et externe. BGP est le seul protocole de routage externe. Il permet aux AS de définir de façon indépendante leur propre politique de routage externe. De nombreux travaux cités dans la bibliographie traitent de comportements divergents dûs à cette flexibilité. Lorsque les politiques ne sont pas en conflits, BGP est auto-stabilisant, ce qui signifie que quelque soit la configuration du réseau, BGP converge vers une solution stable. Malheureusement, comme nous l'a montré Internet, les politiques des AS peuvent être incohérentes et générer ainsi des oscillations. Dans ce mémoire, nous proposons une méthode dynamique distribuée pour détecter et résoudre le problème d'oscillation de BGP. Cette méthode respecte le principe de confidentialité de BGP et impose de faibles contraintes aux AS pour converger vers une solution stable. Pour ce faire, un routeur doit maintenir uniquement des informations locales qui seront les états de ses routes, afin de détecter des oscillations. Dans ce cas, il génère et envoie un jeton lié à une route oscillante. Chaque routeur fait le choix de faire suivre ou non le jeton en fonction de ses données locales et de sa politique. Si le routeur générateur récupère son jeton, alors il interdit la route associée. Néanmoins, certaines routes peuvent être réhabilitées, notament dans le cas des pannes et des apparitions de liens. Nous montrerons comment notre méthode gère ce phénomène. Enfin, nous définirons la notion de politique cohérentes entre elles.
Mots clés : protocole BGP - instabilités - oscillations - cohérence des politiques - résolution dynamique

Abstract

Autonomous Systems (AS) in the Internet use different protocols for internal and external routing. BGP is the only external protocol. It allows ASes to define their own routing policy independently. Many papers cited in reference deal with a divergence behavior due to this flexibility. In fact, when routing policies are not conflicting, BGP is self-stabilising, which means that whatever network configuration, BGP converges to a stable solution. Unfortunately, as experienced on the Internet, AS routing policies may be uncoherent, thus generating oscillations. In this paper we propose a distributed dynamic method for detecting and solving oscillations of BGP. It respects private policy choices and requires only a few low level constraints in order to converge to a stable solution. Essentially, a router has to maintain only local path stateful information to detect instabilities. In this case, it generates and launches a token linked to a route. Each router makes the decision to forward or not the token according to local data and local policy. If the originating router receives back the token, then it marks the route as *barred*. Nevertheless, routes may furtherly be unmarked, in particular when a link breaks or appears. We show how own method manages this phenomenon. Finally, we express and define what coherence between routing policies means.
Keywords : BGP protocol - instabilities - oscillations - coherence of policies - dynamic resolution