Dynamic Detection and Resolution of BGP Oscillations

Ehoud Ahronovitz, Jean-Claude König, Clément Saad

Dynamic detection and resolution of BGP oscillations

Ehoud AHRONOVITZ, Jean-Claude KÖNIG, Clément SAAD

Université Montpellier 2 - LIRMM
161 Rue Ada
F-34392 Montpellier Cedex 5, France

aro@lirmm.fr, konig@lirmm.fr, saad@lirmm.fr

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. Finally, we express and define what coherence between routing policies means.

1 Introduction

Border Gateway Protocol (BGP) is the only interdomain routing protocol used on the Internet. It allows Autonomous Systems (hereafter denoted AS) to exchange routing data. An AS is a set of networks and routers managed by a unique administration. Each AS uses an internal routing protocol such as RIP, OSPF,... and defines its own external routing policy for BGP. These policies may be based on commercial, performance, security criterions, etc. BGP was designed to let ASes freely choose their own policies. Unfortunaly, this free choice may lead to global inconsistencies expressed by oscillations of routes.

Varadhan & al [1] have already shown that ASes private policies may lead to global inconsistencies. Lobavitz & al [2][3] have studied the origins of routing instability.

Many suggestions were made to solve instability. In particular Griffin suggested static and dynamic solutions [4][5][6][7]. Gao & Rexford [8][9] estabished conditions to avoid oscillations, using the relationship customer-provider. Recently, Yilmaz & Matta [10] developed a dynamic solution using a randomized algorithm to reorder preferred AS paths.

However all these solutions have important drawbacks. Indeed, static solutions do not only suppose that the network is stable, but require huge amount of data, e.g. the graph of AS routers, and may involve knowledge of policy data, thus conflicting with privacy. Dynamic solutions such as those suggested by Griffin have much the same drawbacks : they use histories that can grow very long, reveal private policy information and as histories are to be transmitted over the network, they lead to traffic overload.

In this paper we propose a dynamic method for detecting and solving oscillations. Each AS maintains local stateful information on routes in order to detect any oscillation and the route involved in. Then it generates a token linked to the oscillating route. The token is sent to the neighbouring AS routers. Routers that receive the token have to decide whether to forward or drop the token. If the generator receives back the token, we conclude that it is responsible for solving locally the problem. The solution consists in marking the associated route as *barred*. We show that if more tokens are generated by other routers for the same oscillation, then only one route will be marked.

Our work relies on Griffin's model called Stabel Path Problem, described in section 2. We show that managing histories is a lot ressource consuming.

In section 3 we explain the principles of our method. Contrary to Griffin's method, we maintain local stateful information without having to transmit it. We study the properties, advantages and constraints of such a method.

Section 4 deals with the token solution for oscillations. We show why it works then we study some extensions as well as limitations. Finally, in section 5 we give a characterisation of coherent routing policies. We do that because when routing policies are coherent, BGP is self-stabilizing. So we try to connect the divergent behavior with uncoherent routing policies.

2 SPP and dispute digraph

2.1 The Stable Path Problem (SPP)

The Stable Path Problem (SPP) proposed by Griffin and Wilfong in [6] is a modelisation giving a simple view on routing instabilities. It allows to focus on the origins of instabilities. SPP consists of an undirected graph with a single destination. All functionalities and attributes of BGP which are not involved with instabilities, such as MED, aggregation of paths,..., are not considered in SPP.

2.1.1 Construction

Let $G = (V, E)$ be a graph such that the vertices (elements of V) and edges (element of E) represent respectively the autonomous systems and the BGP links. Each AS defines a list of paths ordered with a ranking function from the most to the least preferred path. ASes will try to find and preserve the best ranking path for each destination. An AS can choose a path, only if all ASes belonging to this path adopt the corresponding sub-path. For example, in figure 1 if AS3 adopts path 30 then AS1 can select path 130.

Figure 1 a), represents the BAD GADGET instance of SPP. We assume that each AS defines its own path to destination AS0. A possible simulation of BGP is: initially, each AS adopts either a direct path to AS0, or the empty path noted ϵ. AS1 does not know the paths of its neighbours. So, it adopts path 10 and broadcasts this to its neighbours. When AS2 gets this piece of information, it can select path 210 and inform AS3. As path 20 is not available, AS3 preserves path 30. When AS1 will receive this information, it will select path 130, thus loosing path 210, and so on ... This process cycles and illustrates a case of BGP oscillation. Griffin & al. showed in [7] that the detection of oscillations in a SPP instance is NP-Complete, through a reduction to 3-SAT.

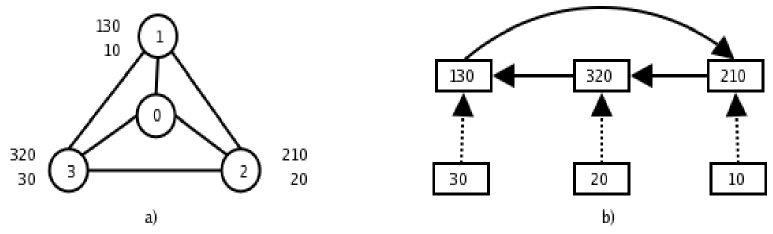

Figure 1: BAD GADGET - A problem with no stable solution and its dispute digraph

2.2 Dispute digraph

Griffin & al. [6], construct a dispute directed graph, deduced from the SPP instance. In this graph, nodes represent paths extracted from ASes lists, while arcs represent either compatibility or conflict between paths.

2.2.1 Construction

Let $G = (V, E)$ be a dispute graph, each node representing a path (figure 1 b)). There are two types of arcs, defined as follows:

- Transmission arc : Let u, v be two ASes, uvP and vP two paths belonging respectively to u and v. If v adopts path vP then u can adopt path uvP. In this case, there is an arc from vP to uvP called transmission arc. It is represented by a dotted line (figure 1 b)).
- Dispute arc (figure 2) : Let vQ and vP be two possible paths for AS v, vQ being preferred to vP. Let uvP and uvQ be two possible paths for AS u, uvP being preferred to uvQ. If v adopts vQ (preferred to vP) then u cannot adopt uvP, since path vP is not available. Thus u will choose another path with a lower rank than uvP. In this case, there is an arc from vQ to uvP called dispute arc. It is represented by a full line.

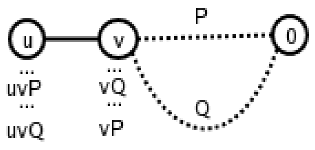

Figure 2: A dispute arc $vQ- > uvP$

Figure 1 b) shows the dispute digraph for BAD GADGET.

2.2.2 Dispute cycle

A dispute cycle in the dispute digraph represents conflicting routing policies.

Definition 1. *A dispute cycle in the dispute digraph is a cycle containing at least two dispute arcs.*

Let S be a SPP instance. Griffin shows in [6] that:

Theorem 1. *If the dispute digraph of S is acyclic then S has a unique solution.*

2.3 A dynamic solution using histories

We explain hereafter the dynamic method, $SPVP_3$, using histories as proposed by Griffin ([6]).

While in BGP ASes exchange paths, in $SPVP_3$, Ases exchange pairs $m = \{P, h\}$ where P is a path and h a history related to P. The functions $path(m)$ and $hist(m)$ are defined to return respectively P and h.

Let $B(u)$ be a set of *barred* paths. Let function $choice(u)$ return the whole paths of AS u. Then $best(u) = max(choice(u) - B(u))$ returns the best choice for AS u among the possible paths.

In $SPVP_3$, When AS u receives a pair m from a neighbour, it compares $path(m)$ to $best(u)$. If the result is better, then it updates its path and the related history. Finally, u sends its updated route and the related history to all its neighbours. With histories, ASes can detect cycles. If an AS detects a cycle, then it adds the current path to $B(u)$.

2.4 History construction

Each AS manages a history tracing all events that happened to its paths as well as their causes. For each announce of a path P, a router joins the associated history h. When an AS u receives the pair $m = \{P, h\}$ from a neighbour, if P implies a modification (i.e. path X replacing Y), then the following is added to the history of u:

- h
- $(+X)$, if X is preferred to Y (u got a better choice),
- $(-Y)$, if Y is preferred to X (u lost a better choice).

This phenomenon is shown in table 1 steps 2 to 3. Table 1 represents an example of history management with the BAD GADGET in figure 3. For example, AS4 announces its new path ϵ with the history (-420)(+210) (h). AS3 looses path 3420 (Y), and chooses path 30 (X). Therefore it adds in its own history h and (-3420) since 3420 is preferred to 30.

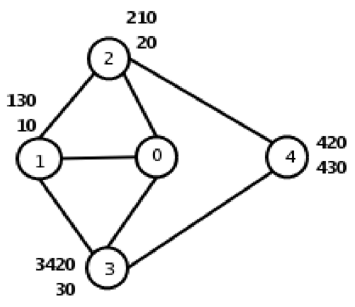

Figure 3: BAD GADGET with five ASes

3 Maintaining path local stateful information (PLSI)

The history management seen above involves a lot of local and network ressources. So we rather propose to maintain local stateful information on routes allowing to detect oscillations. The advantages of this method are described hereafter.

3.1 Differences with $SPVP_3$

$SPVP_3$ uses histories in order to detect cycles. If a cycle is detected, $SPVP_3$ adds a path to $B(u)$. As soon as an AS detects a cycle it considers that the chosen path is wrong and it adds it to $B(u)$. For example, when cycle (+210 -420 -3420 -130 +210) is detected (see table 1), $SPVP_3$ adds 20 to $B(u)$. We prefer considering 210 as wrong because, if we delete 210 we "break" the dispute cycle in the dispute digraph and due to the previous theorem, we obtain a stable solution. In figure 3, $SPVP_3$ converges towards (130),ϵ,(30),(430) respectively for ASes 1,2,3,4. Our solution (PLSI) converges towards (10),(20),(3420),(420).

3.2 Characterisation of oscillations

3.2.1 General properties

Let us pay attention to the history in table 1. We can observe that cycle (-210 +130 -3420 -420 +210) has been detected because a path (here 210) changed from state + to state -.

step	*u*	best(*u*)	hist(*u*)
0	1	(10)	*
	2	(20)	*
	3	(3420)	*
	4	(420)	*
1	1	(10)	*
	2	$\underline{\text{(210)}}$	(+210)
	3	(3420)	*
	4	(420)	*
2	1	(10)	*
	2	(210)	(+210)
	3	(3420)	*
	4	$\underline{(\epsilon)}$	(-420) (+210)
3	1	(10)	*
	2	(210)	(+210)
	3	$\underline{\text{(30)}}$	(-3420) (-420) (+210)
	4	(ϵ)	(-420) (+210)
4	1	(10)	*
	2	(210)	(+210)
	3	(30)	(-3420) (-420) (+210)
	4	$\underline{\text{(430)}}$	(+430) (-3420) (-420) (+210)
5	1	$\underline{\text{(130)}}$	(+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	$\underline{\text{(20)}}$	**(-210) (+130) (-3420) (-420) (+210)**
	3	(30)	(-3420) (-420) (+210)
	4	(430)	(+430) (-3420) (-420) (+210)

Table 1: History management for the BAD GADGET example in figure 3.

Theorem 2. *In the case of an oscillation, a state change occurs in all paths of a cycle in the dispute digraph. We call* oscillating cycle *such a cycle.*

proof: Let C be an oscillating cycle. If C oscillates then a state change occurs for at least one path. However, according to the definitions of transmission and dispute arcs, if there is an arc from X to Y , it means that Y depends on X. So if a state change occurs in X then a state change occurs in Y. Therefore a state change occurs for all paths. □

Note

- A state change from + to + (resp. from - to -) implies a previous move to state - (resp. state +).
- At the initial step, ASes adopt either the empty path ϵ or a direct path. Therefore each AS will begin with a + state on one path.

Theorem 3. *In an oscillating cycle, when we follow the cycle for the first time, at least one path in the cycle will change from state + to state -.*

proof: If a cycle is detected at the initial step then the theorem is prooved due to the previous note.

Assume that all paths changed from state - to state +. We know that an oscillating cycle contains at least two dispute arcs. Let C be a cycle and X and Y be two paths in C with a dispute arc from X to Y. This means that if X is adopted then Y cannot be chosen. Therefore, if X changed from state - to state + then X is adopted. Consequently, Y cannot change from state - to state +. Thus, one cannot change only from state - to state +. According to the previous theorem, an oscillation occurs in all paths, so the oscillation in Y is necessarily due to a state change from + to -. □

3.2.2 Oscillations and cycles

These theorems allow us to consider only the state change from + to -. So, we call *oscillation on a path* the state change from + to -.

A state change for a path means that there is an oscillation but we don't know if this path belongs to the cycle or not. This phenomenon is illustrated in figure 4: AS1 detects an oscillation on path 1240 caused by BAD GADGET but 1240 does not belong to the cycle in the dispute digraph.

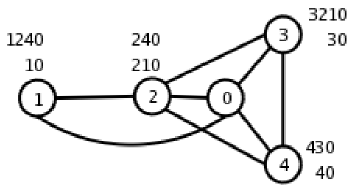

Figure 4: Error caused by a transmission arc

3.3 Identifying oscillating paths

Each AS maintains only the state of its own paths. Thus we have only local management of path states. Moreover, messages exchanged between ASes contain only paths. Thus, we consume only low amount of resources. Table 2 represents this local management. When moving from step 5 to step 6 in table 2, AS2 detects an oscillation on 210. If this path is involved in a dispute cycle then it will be marked *barred*.

	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

Table 2: Local management for BAD GADGET (rib-in is the currently selected path)

4 Detecting and resolving oscillations

As each AS maintains only local information, actions beetween ASes must be coherent. There are two problems :

- The first concerns the detection of cycles. As we have seen, if there is an oscillation on a path that does not mean necessarily that this path belongs to a cycle.
- The second problem is: when a cycle is detected which path among all paths involved in the cycle should be marked?

4.1 Detecting cycles with a token

We describe hereafter how a token allows to detect cycles and why it works.

4.1.1 The token method

As soon as AS A detects an oscillation on path X, it generates a token related to X and sends the new chosen path with the token. If an AS B has not to update its path when receiving this message then it destroys the token. But, if B has to update its path then it forwards the token with its own newly selected path to all its neighbours, and so on... If A receives a path Y with its own token and if Y implies the choice of X then A concludes that X belongs to a cycle and marks X *barred*.

Note that the token value does not reveal the oscillating path (the token value can be assigned with a hashtable). Thus, when B receives the token, it does not know which path oscillates.

Figure 5 is an example of a token flow. AS2 detects an oscillation on 210. It generates a token related to 210 (denoted $j210$) and sends its new path (20) with the token. When AS3 receives this information it chooses path (320) instead of (30). So it forwards token $j210$ with (320). AS1 receives the message from AS3 and modifies its path to (10). It forwards token $j210$ with path (10) and when AS2 receives this message it retreives its token and adopts (210). AS2 concludes that 210 is involved in a cycle. Then AS2 marks *barred* this path and a new stable solution is found.

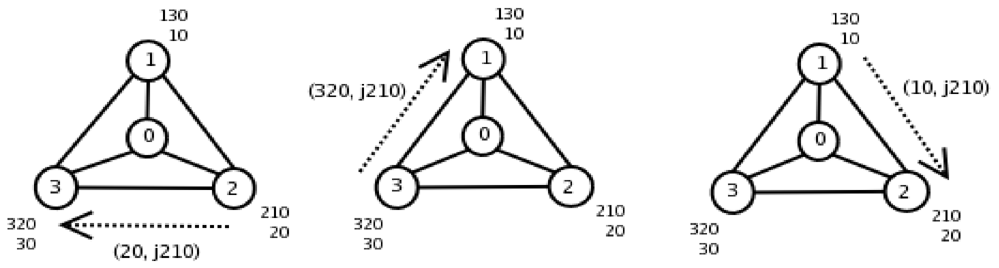

Figure 5: Token flow after oscillation on path 210

4.1.2 Why does it works ?

Assume the dispute digraph of BAD GADGET (figure 1 b)). We state that the token follows the dispute cycle. In fact, as AS2 detects the oscillation on 210, this means that 210 changed from state + to state - and then 210 is not adopted. Thus 320 may be adopted making impossible the choice of 130 for AS1. Then AS2 may adopt 210 and concludes that 210 is involved in a cycle. If the cycle oscillates, all listed previous events did happen and for each cycle a token was generated by the first event. This token was forwarded during the following events. Therefore the token follows exacly the cycle in the dispute digraph.

4.1.3 Which path should be *barred*?

When a cycle is detected, all ASes having a path involved in a cycle will notice oscillation. So all these ASes will generate a token. Each of them will retreive its own token and will mark *barred* the associated path. It is not reasonable to mark *barred* all paths involved in a cycle since marking only one is sufficient to break the cycle. Furthermore, ASes contain only local information. Our solution consists in totally ordering tokens. This order allows to always forward only the highest priority token. Finally, only one token is retreived by its generator. Let $\prec$ be this order relation. When the AS who generated token $j1$ receives a token $j2$ it checks if $j1 \prec j2$. When this is true, j2 is not forwarded.

In figure 6, AS 1, 2, 3, detect an oscillation on respectively paths 130, 210 and 320. Assume that the order relation is the lexicographic order and that the token value is “j” followed by the path value. For example, AS1 generates token “j130”. When AS1 receives token “j320” it destroys it. Idem when AS1 receives “j210”. AS1 will retreive its token “j130” and will mark *barred* path 130.

We must notice that making another choice for the order relation would allow sometimes a better solution.

The process for oscillation detection is given in algorithm 1 and the token management in algorithm 2.

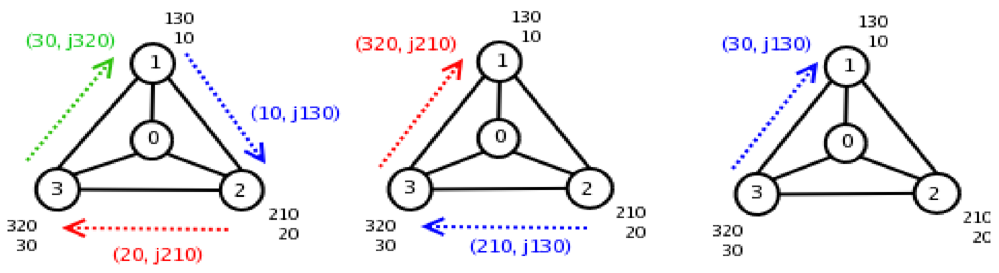

Figure 6: Token unicity

Notes

- It is possible that the AS who generates a token, retreives it, and that the path may not be involved in a cycle. This phenomenon is caused by transmission arcs. Figure 4 illustrates this case. Indeed, path 1240 oscillates because path 240 oscillates. 1240 is not involved in a cycle contrary to 240. It is necessary to mark *barred* path 240 and not 1240. The lower path length, the higher priority.
- We do not have to manage the loss of a token since BGP uses TCP transport and managing duplication of tokens is easy with identifiers.

Algorithm 1: Detection of an oscillation by AS u

```
    Data    : Table T of paths' states
    Result  : A token if an oscillation occured, broadcasted to neighbours

    /* detectOscillation(T) returns the oscillating path */
    /* generator is true if $u$ generated a token, false else */
    /* creatToken(op) generates a token related to oscillating path op */
    /* choicePath() returns the best path related to $u$'s policy */

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

Algorithm 2: Processing the reception of a token by AS u

```
Data    : reception of message $< oscillatingPath, token, v >$
Result  : Forwards or deletes the token received
/* $path$ currently selected path */
/* getGenerator() returns the local path associated to the token */
/* markBarred(p) marks *barred* path p */
/* lgPath(t) returns the length of path related to token t */

$newPath \leftarrow choicePath()$;
if $path \neq newPath$ then
    $path \leftarrow newPath$;
    /* token has $\prec$ higher priority or length of associated path is less than length of path associated
    to own token (myToken) */
    if $generator = true$ and
    $(token \prec myToken$ or $lgPath(token) < lgPath(myToken))$ then
        $generator \leftarrow false$;
    end
    if $generator = true$ and $token = myToken$ then
        if $path = myToken.getGenerator()$ then
            $markBarred(path)$;
            $path \leftarrow choicePath()$;
            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
```

Limits Our method may unnecessarily mark *barred* some paths.

- The first example (figure 7) illustrates a situation where path 120 is marked *barred* because AS3 receives path 376540 too late.

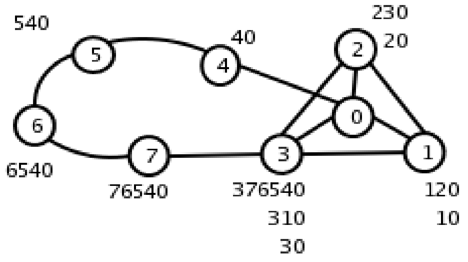

Figure 7: Wrong marking

- The second example (figure 8) illustrates a situation where two paths will be marked *barred* instead of one. In fact, paths 130 and 150 will be marked *barred* whereas marking path 320 would be sufficient enough. Nevertheless, one marking instead of two is not necessarily a better solution.

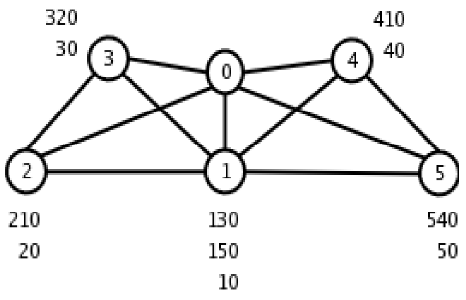

Figure 8: Two paths marking instead of one

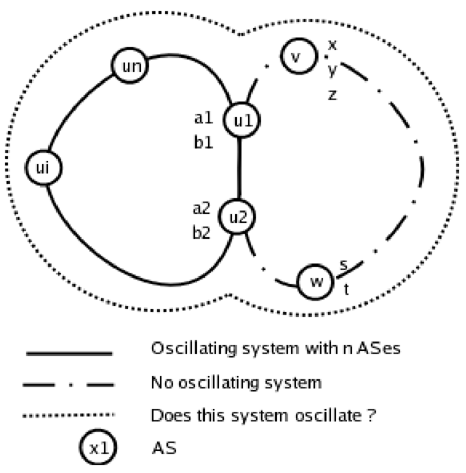

Figure 9: Solving system $u_1, ..., u_n$ never implies oscillations of the dotted line system

Another possible question is: may solving one oscillation generate a bigger one? The answer is in the next theorem.

Theorem 4. *The resolution of an oscillation never generates a bigger oscillation.*

proof: Consider figure 9. The system involving ASes $u_1, ..., u_n$ oscillates. We must check if resolving this system does not generate an oscillation of the dotted line system. Let v be an AS connected to u_1. v does not detect any oscillation. We deduce two cases: either the paths of v have no links with the paths of u_1, or paths which could oscillate in v (for example y and z) are never chosen because v prefers path x. In this latter case, this means that the configuration of one of v neighbours allows v to preserve x. This neighbour has not detected any oscillation; otherwise x cannot be preserved. So, all the markings which occured on u_1 do not imply any modification on v 's configuration. Idem for w with u_2. Therefore, resolving system $u_1, ..., u_n$ implies no oscillations of the dotted line system. □

5 Coherence between routing policies

We saw that inconsistency of policies may induce oscillations. So what are coherent policies?

5.1 Characterisation

Let $<_\alpha$ be an order on paths. We define locally this order, for paths belonging to the same AS, and globally for inter AS paths. We can interpret $<_\alpha$ as "better than".

Definition 2. $<_\alpha$ ***locally :*** *Let A be an AS; $\forall P, Q$ paths of A, if P is preferred to Q then $P <_\alpha Q$.*

This definition is coherent with AS policies.

Definition 3. $<_\alpha$ ***globally :*** *$\forall P, Q$ paths belonging to two differents Ases, if P is a sub-path of Q then $P <_\alpha Q$.*

This definition allows to preserve coherence between AS policies. Clearly a path containing P cannot be better than P.

Theorem 5. *If $<_\alpha$ is a total order relation then the policies are coherent between themselves.*

proof: There is a connection between these definitions and the dispute digraph.

- Let be a transmission arc from P to Q. this means that P is a sub-path of Q. With the definition of global order we have $P <_\alpha Q$.
- Let be a dispute arc from P to Q. this means that Q cannot be chosen if P is currently selected. There is a path R belonging to the same AS as P, with P preferred to R. If R is adopted then P is not adopted, so Q may be adopted. Locally we have $P <_\alpha R$ and globally we have $R <_\alpha Q$ since R is a sub-path of Q. By transitivity we have $P <_\alpha Q$. Thus, we conclude that the dispute digraph is acyclic if and only if $<_\alpha$ is a total order relation. We know that policies are coherent (BGP converges) if the dispute digraph is acyclic (section 2.2.2). So, if $<_\alpha$ is a total order relation then the policies are coherent between themselves. $\square$

Consider figure 10. With the total order relation we have

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

Due to the contradiction, we deduce that BAD GADGET policies are not coherent between themselves.

5.2 A new dispute digraph

Now we can suggest a new definition for the dispute digraph: Let P and Q be two paths. There is an arc from P to Q if $P <_\alpha Q$. If the digraph is acyclic then there is a stable solution. Figure 10 shows the new dispute digraph for BAD GADGET. In our future works, we intend to develop the properties of this graph.

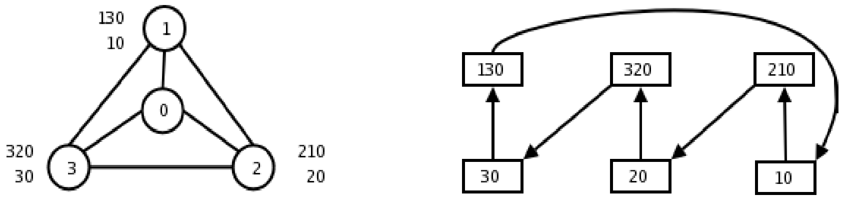

Figure 10: New dispute digraph

6 Conclusion

As BGP is currently the only external protocol in the Internet, instabilities such as ascillations should be resolved as quickly as possible. In this paper, we studied route oscillations caused by globally incoherent routing policies. We proposed a distributed dynamic method to solve the oscillations problem. In fact, it is distributed because nor global data neither a central algorithm are invoked; local data and a circulating ligthweight token are only required; dynamic because the network topology may change without affecting the algorithm. We ended by an attempt to define coherence between routing policies, which may lead to prevent instabilities, rather than detecting and resolving them.

Future work for the short term relies on simulating the behaviour of our algorithms on different network topologies. For the mean term, we think that management of network failures may be treated by such a method, but unlikely, recovery or appearence of links looks more difficult : a BGP router does not forward recovery or appearence information unless it changes its routing tables.

For the long term, we shall have to study some byzantine behaviours. For example, what happens when an AS detects an oscillation, but does not generate a token, as this may lead to suppress a route. Another important problem is AS connectivity: should top priority be given to connectivity or efficiency? In other words, if sovling an oscillation leads to breaking AS connectivity is it worth keeping the oscillation, rather than disconnecting ASes?

References

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

[5] 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.

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

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

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

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

[10] 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.