A Distributed Method for Dynamic Resolution of BGP Oscillations

Ehoud Ahronovitz[1], Jean-Claude König[1], Clément Saad[1]

[1]Université Montpellier 2 - LIRMM
161 Rue Ada
F-34392 Montpellier Cedex 5, France
{aro,konig,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 the 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 inter-domain 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 external routing policies allow the definition of preferences in choosing routes.

As policies may be based on commercial, performance, security criterions, etc. BGP was designed to let ASes freely choose their own policies. Thus, ASes define an ordered list of preferred paths according to the adopted policy and will try to maintain the best path as long as possible for each destination. Unfortunately, locally preferred paths may lead to global inconsistencies expressed by oscillations of routes.

Varadhan & al [9] have already shown that ASes private policies may lead to global inconsistencies. Lobavitz & al [8, 7] have studied the origins of routing instability.

ASes involved in oscillations of routes will exchange successively and repeatedly BGP routing messages, and never converge to a set of stable routes. Such a divergent behavior would degrade the routing performance of the Internet.

Many suggestions were made to solve instability. In particular Griffin suggested a dynamic solution [5, 3, 6, 4] called $SPVP_3$. This method consits in managing a history related to the choice of routes. Thanks to history, an oscillating route can be detected and lead to consider a path as being "bad". Unfortunately, exchanging history messages imply a large amount of updates and do not garantee policy confidentiality.

Gao & Rexford [2, 1] estabished conditions to avoid oscillations, using the commercial relationship between ASes (peer-peer, or provider-customer).

A pair of ASes have a provider-customer relationship if one offers Internet connectivity to the other. They have a peer-peer relationship if they are mutually providing connectivity to their customers. To ensure the stability of the global BGP routing system, each AS is supposed to follow policy configuration guidelines which suggest preferring customers announced routes to providers or peers announced routes. While this

1-4244-0054-6/06/$20.00

solution guarantees stability, it may unnecessarily disallow the use of many routes.

Recently, Yilmaz & Matta [10] developed a dynamic solution using a randomized algorithm to reduce local preference of paths. If a path is adopted and later on abandoned n times by an AS, then it is invalidated and put in the set of bad paths as soon as n is bigger than a predefined *threshold*. But in the worst case of this algorithm, each AS is forced to add a bad path to the set of bad paths.

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 depending on BGP local updates. 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 Stable Path Problem, described in section 2. We show that managing histories is a lot resource 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.

Instance 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. For example, in figure 1, AS1 has two different paths to the destination AS0. Path 130 means the path that goes from node 1 to node 0 via node 3, wherease path 10 is the direct path. AS1 prefers path 130 to 10.

Each AS will try to find and maintain 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 1a, represents the BAD GADGET instance of SPP. This example is frequently used in [3] to show the divergent behavior. 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 receives this information, it can select path 210 and inform AS3. As path 20 is not available, AS3 maintains 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. show in [4] that the detection of oscillations in a SPP instance is NP-Complete, through a reduction to 3-SAT.

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 preference list, while arcs represent either compatibilities or conflicts between paths.

Construction. Let $G = (V, E)$ be a dispute graph, each node representing a path (figure 1b). 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, represented by a dotted line (figure 1b). In this figure, there is a transmission arc from node 30 to 130: if AS3 adopts path 30 then AS1 may adopt 130.

- Dispute arc: consider figure 2. Let vQ and vP be two possible paths for AS v, vQ being preferred

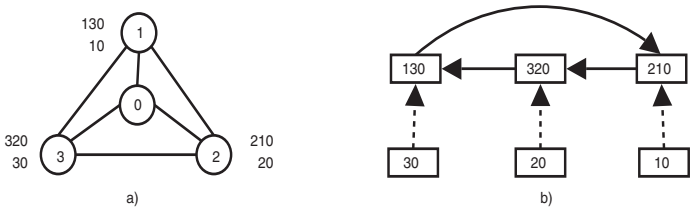

Figure 1. BAD GADGET - A problem with no stable solution and its corresponding dispute digraph

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, represented by a full line (figure 1b). In this figure, there is a dispute arc from 210 to 320: the adoption of path 210 prohibits the choice of 320.

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

Dispute cycle. A dispute cycle in the dispute digraph represents conflicting routing paths.

Definition 1. *From [6], 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 *if the dispute digraph of S is acyclic then S has a unique solution.*

2.3. A dynamic solution using histories

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

Background. 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. Two functions $path(m)$ and $hist(m)$ are defined to return respectively P and h.

Let $B(u)$ be a set of *bad* 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 $B(u)$.

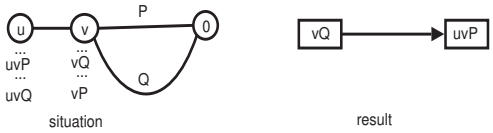

Figure 2. Dispute arc

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 containing subpath P 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).

To illstrate the history management in $SPVP_3$, here is a **sequential** execution of the BAD GADGET presented in figure 3. Table 1 follows this history management.

Initially, ASes 1,2,3 and 4 have adopted respectively paths 10, 20, 3420, 420. All histories are empty. When AS1 announces path 10 to AS2, AS2 selects path 210 and updates its own history adding the event "(+210)" because path 210 is preferred to 20. Then AS2 announces both its new path 210 and the associated history "(+210)". When AS4 receives this information, it deduces that path 20 is no longer available and must choose a path with lower preference. But path 430 cannot be selected, because 30 is not available. So path ϵ is mandatory, and AS4 adds to its own history both the event "(-420)" and the history received from AS2 containing "(+210)". AS4 sends path ϵ and the

newly associated history "(-420 +210)" to AS3. AS3 can no longer maintain path 3420, so chooses path 30 and updates its own history. The new history for AS3 becomes "(-3420 -420 +210)". When AS1 gets this new information from AS3, it selects path 130 as path 30 is available and as path 130 is preferred to path 10. Then AS1 updates its own history to "(+130 -3420 -420 +210)". Finally, when AS2 gets this information, path 10 being no longer available, it chooses path 20 and updates its history to "(-210 +130 -3420 -420 +210)". Hence, an oscillation is detected by AS2 because path 210 changed from state + to -. $SPVP_3$ considers path 20 as "bad".

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)

Table 1. History management for the BAD GADGET example in figure 3. *:empty history. Underlined paths are newly selected paths.

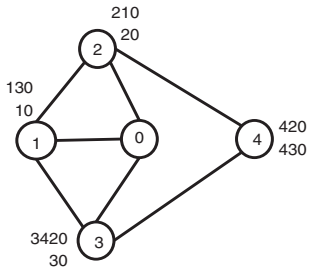

Figure 3. BAD GADGET with five ASes

3. Maintaining path local stateful information (PLSI)

The history management seen above involves a lot of local and network resources. 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. Characterisation of oscillations

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 path 210 changed from state + to state -. We generalize to the following property:

Property 1. *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 elements: Let C be an oscillating cycle. The proof relies on the following: 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's state depends on X's state. So if a state change occurs in X then a state change occurs in Y. Therefore a state change occurs for all paths in cycle. □

Remarks

- If a given path appears twice with state +, then necessarily it appeared with state - between those two.
- At the beginning of the protocol, ASes adopt either the empty path ϵ or a direct path if it exists. Therefore each AS will begin with a + state on one path.

Property 2. *In an oscillating cycle, when we follow the cycle, at least one path changes from state + to state -.*

proof: First, if a cycle occurs at the begining of the protocol then the property is prooved due to the above remarks.

Now, assume that all currently adopted paths changed from state - to state +. We know that an oscillating cycle contains at least two dispute arcs [6]. Let C be an oscillating 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. From our hypothesis, X changed from state - to state +. Consequently, Y cannot change from state - to state +. Thus, it is impossible that all paths in the cycle changed from state - to state +. According to the above remarks, an oscillation occurs in all paths, so the oscillation in Y is necessarily due to a state change from + to -. □

Oscillations and cycles. These remarks allow us to consider only the state change from + to -. So, we can now call *oscillation on a path* the state change from + to -.

Important note: 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 4a: AS1 detects an oscillation on path 1240 caused by a BAD GADGET situation. But 1240 does not belong to the cycle in the dispute digraph figure 4b.

3.2. 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 resources amount: histories are not forwarded. 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*.

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 oscillating path that does not necessarily mean that this path belongs to a cycle. So after detecting an oscillation, we have to search if a cycle exists.
- 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.

The token method. Assume AS A detects an oscillation on path X. Therefore, AS A generates a token related to X and sends the new chosen path with the token in the classical BGP announce. If an AS B has not to update its path when receiving this message then it drops the token. But, if B has to update its selection, then it forwards the token with its own newly selected path to all its neighbours, and so on ... If A receives back its own token with path Y and if Y implies the choice of X then A concludes that X belongs to an oscillating cycle and marks X *barred*.

Note that the token value does not reveal the oscillating path (e.g. 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.

Why does it works ? Assume the dispute digraph of BAD GADGET (figure 1b). 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 downgraded and 20 is selected. Thus 320 may be adopted by AS3 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, the above listed events did happen and 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.

Which path should be *barred*? When an oscillating cycle is detected, all ASes having a path involved in this 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

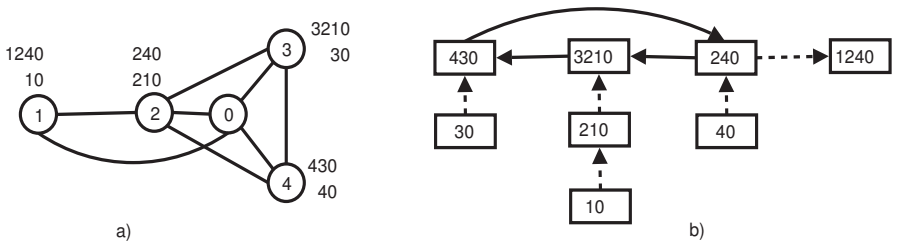

Figure 4. Path 1240 does not belong to a cycle

	AS1			AS2			AS3			AS4		
step	130	10	SP	210	20	SP	3420	30	SP	420	430	SP
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 figure 3. SP: currently selected path. *: empty entry. + or -: path state

in a cycle since marking only one is sufficient to break the cycle. But, ASes manage only local information. Any total order relation on tokens allows to solve this problem. In fact, a total order allows to always forward only the highest priority token. So, only one token will be retreived by its generator. Let $\prec$ be this order relation. When the AS who generated token $j1$ receives a token $j2$ it checks if $j1 \prec j2$. If this is true, j2 is not forwarded.

Conditions on the order relation. The order relation has to take into account the oscillating path length. In our method, the lower path length, the higher priority. This phenomenon is caused by transmission arc: when a path P oscillates all paths Q containing P also detect an oscillation.

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.

Hereafter, we define $\prec$ as follows: Let j_1 and j_2 be two tokens, $|j_X|$ be the length of token j_X and $<_L$ be the lexicographical order. Then,

$j_1 \prec j_2 \Leftrightarrow$ if $|j_1| = |j_2|$ then $j_1 <_L j_2$ else $|j_1| < |j_2|$.

For example, in figure 6, ASes 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. AS1 generates token "j130". When AS1 receives token "j320" it drops 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 a different solution, sometimes better, sometimes worse.

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

4.2. Discussion on barred paths

May solving one oscillation generate another one? We claim that the resolution of an oscillation never generates a bigger oscillation.

proof: Consider figure 7. The system made of 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 and (x, y, z) its prefered path list. v does not detect any oscillation. We deduce two possible cases:

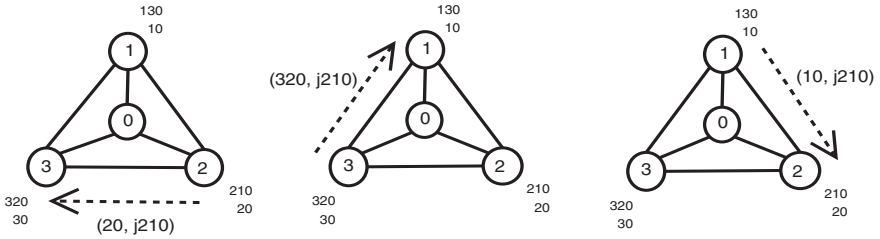

Figure 5. Token flow after an oscillation on path 210

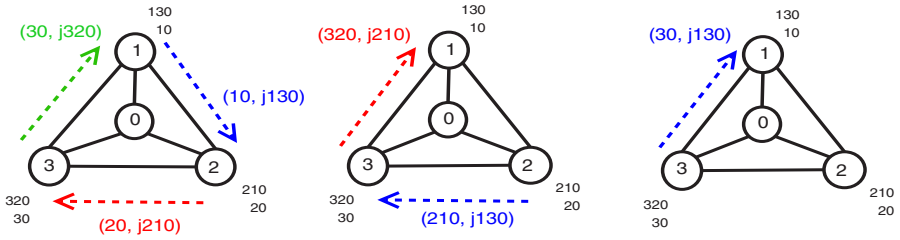

Figure 6. Token ordering and unicity

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 with table T of paths' states */
/* *creatToken*(op) generates a token related to oscillating path op */
/* *bestPath*() returns the best path related to u's current policy */
/* *path* the currently selected path */
/* *generator* is true if u generated a token */

$oscillatingPath \leftarrow detectOscillation(T)$;
if $oscillatingPath \neq null$ **then**
  /* Initialisation for the algorithm 2 */
  $generator \leftarrow false$;
  $myToken \leftarrow creatToken(oscillatingPath)$;
  $path \leftarrow bestPath()$;
  for $v \in N(u)$ **do**
    $send(path, myToken, v)$;

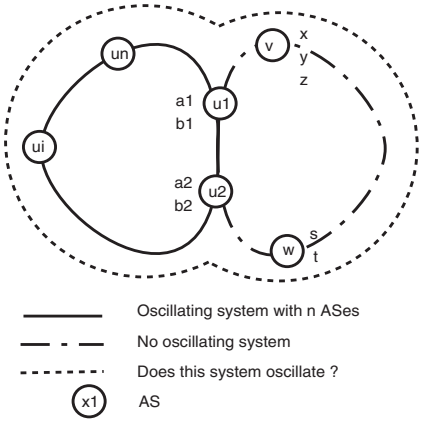

Figure 7. Solving oscillation for system $u_1, ..., u_n$ never implies oscillations of the dotted line system

- either paths of v have no links with 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 's neighbours allows v to maintain x. This neighbour has not detected any oscillation; otherwise

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

```
Data : reception of message < path_v, token, v >
Result: Forward or delete the token received
/* update(p, v) As u take into account the new path p annouced by AS v */
/* bestPath() returns the best path related to u's current policy */
/* markBarred(p) marks barred path p */
/* lgPath(t) returns the length of path related to token t */
/* getGenerator() returns the local path associated to the token */
/* path the currently selected path */
/* generator is true if u generated a token */

update(path_v, v)
newPath ← bestPath();
if path ≠ newPath then
    path ← newPath;
    /* token has ≺ higher priority or length of associated path is less than length of path associated to own token (myToken) */
    if generator = true and
    (token ≺ myToken or lgPath(token) < lgPath(myToken)) then
        generator ← false;
    if generator = true and token = myToken then
        if path = myToken.getGenerator() then
            markBarred(path);
            path ← bestPath();
            for v ∈ N(u) do
                send(path, null, v);
    else
        for w ∈ N(u) − {v} do
            send(path, token, w);
```

x cannot be maintained. So, all the markings which occured on u_1 do not imply any modification on v's configuration. Idem for ASes w and u_2. Therefore, resolving system $u_1, ..., u_n$ cannot imply an oscillation of the bigger dotted line system. □

There is no false positive. Our method always resolves an oscillation: a route will never be barred if there is no oscillation, which means that no false positive detection is possible. In fact, a marking occurs when an AS gets back its token. It is impossible to get back a token erroneously, as the token is forwarded only if an update is done.

But future route updates may lead a mark to become unnecessary. For exemple in figure 8, route 120 is first mark barred. This situation is correct until the reception of a new announce, for example AS3 receiving the announce 76540. Such an announce should lead to rub out the marking for 120.

What happend? Let us recall that ASes do not have a global view of the network. So when AS1 barres path 120, it is due to local oscillations in ASes 1, 2 and 3 system. When AS3 receives later on from AS7 the announce 76540, it has a bigger view of the network, so path 120 may be unmarked. We plan to take into account this problem in a more general case: failure and appearence of links and ASes. This is a big problem, that we are studying currently.

Limits. Our method will not always give an optimal result. Some network configurations (eg. systems including many oscillating cycles) may lead to unecessarily mark barred more than one path though less marks would be sufficient to solve the oscillation. We can say that the method always gives a solution but it does not take into account any optimization criterion. In fact, minimizing the number of marked paths is not necessarily a good optimization criterion. We think that any method cannot take into account an optimization criterion unless it has a global view of the whole BGP routers network.

The following example (figure 9) illustrates a situation of two nested BAD GADGETS. 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.

4.3. Differences with $SPVP_3$

$SPVP_3$ uses histories in order to detect cycles and has to forward histories between ASes. So policies privacy is not respected and trafic overload is generated.

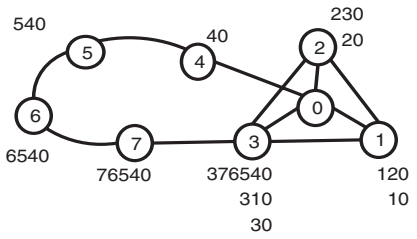

Figure 8. Late update

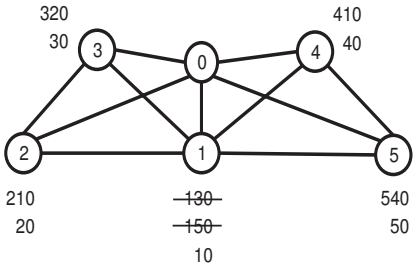

Figure 9. Two paths marked instead of one

Our method makes only local management of paths. Tokens circulate in classical BGP announces, so overload is minimized to the token size in the BGP announce.

Another difference is that, contrary to $SPVP_3$, we prefer marking one of the paths directly involved in a detected oscillation. 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 note, 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).

Note that tokens do not reveal any policy element. So we respect privacy. Also, the data structures manipulated are local and rather lightweight. Constraints imposed to ASes are low: maintenance of paths state and generation of forwarding of tokens in BGP announces.

5. Coherence between routing policies

We saw that inconsistency of policies may induce oscillations. So besides solving oscillations, we try to define what are coherent policies?

5.1. Characterisation

Let $<_\alpha$ be an order on paths. Hereafter, 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 maintain coherence between AS policies. Clearly a path containing P cannot be better than P.

Theorem 1. *If $<_\alpha$ is a strict 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 (path in AS u) to Q (path in AS v). This means that if P is not selected, then there exists a path R in AS u, which allows the selection of Q in AS v. In other words, R is a sub-path of Q. So, locally we have $P <_\alpha R$ and globally we have $R <_\alpha Q$. By transitivity we have $P <_\alpha Q$. Thus, we conclude that the dispute digraph is acyclic if and only if $<_\alpha$ is a strict order relation. We know that policies are coherent (BGP converges) if the dispute digraph is acyclic (section 2.2). So, if $<_\alpha$ is a strict order relation then the policies are coherent between themselves. □

Consider figure 10. With the strict 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.

Note The theorem reciprocal is false: coherent policies do not imply the existence of a strict order relation.

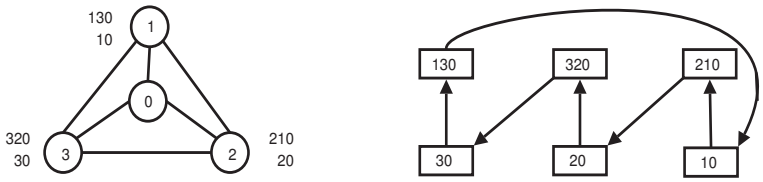

Figure 10. New dispute digraph

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.

6. Conclusion

As BGP is currently the only external protocol in the Internet, instabilities such as oscillations 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 solving 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] L. Gao, T. G. Griffin, and J. Rexford. Inherently safe backup routing with bgp. *in Proc. IEEE INFOCOM*, April 2001.

[2] L. Gao and J. Rexford. Stable internet routing without global coordination. *in Proc. ACM SIGMETRICS*, June 2000.

[3] T. G. Griffin, F. B. Sherpherd, and G. Wilfong. Policy disputes in path-vector protocols. *Proc. 7th Int. Conf. Network Protocols (ICNP'99)*, pages pp. 21–30, 1999.

[4] T. G. Griffin, F. B. Sherpherd, and G. Wilfong. The stable paths problem and interdomain routing. *Proc. IEEE/ACM Transactions on Networking*, 2002.

[5] T. G. Griffin and G. Wilfong. An analysis of bgp convergence properties. *Proc. ACM SIGCOMM'99*, pages pp. 277–288, 1999.

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

[7] C. Lobavitz, G. R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Trans. Networking*, vol. 6:pp 515–528, October 1998.

[8] C. Lobavitz, G. R. Malan, and F. Jahanian. Origins of internet routing instability. *in Proc. IEEE INFOCOM*, vol. 1:pp. 218–226, 1999.

[9] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, pages 32:1–16, 2000.

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