

# Improving Internet-wide routing protocols convergence with MRPC timers

Paper #200 — 14 pages

## ABSTRACT

The behavior of routing protocols during convergence is critical as it impacts end-to-end performance. Network convergence is particularly important in BGP, the current inter-domain routing protocol. In order to decrease the amount of exchanged routing messages and transient routes, BGP routers rely on MRAI timers [35] and route flap damping [40]. These timers are intended to limit the exchange of transient routing messages. In practice, these timers have been shown to be partly ineffective at improving convergence, making it even slower in some situations [5, 26, 27].

In this paper, we propose to add timers to routing protocols that enforce an ordering of the routing messages such that path exploration is drastically reduced while controlling convergence time. Our timers, called MRPC (metrics and routing policies compliant), are set independently by each router and depend only on the metrics of the routes received by the router as well as the routing policies of the router. No sharing of information about routing policies between neighboring ASs is required by our solution.

We first show and prove under which conditions MRPC timers can be computed. Our MRPC timers apply to a large class of routing protocols, including all the routing protocols currently in use in the Internet. We rely on the RAML framework to show how to compute MRPC timers for the particular case of BGP. To illustrate the benefit of MRPC timers, we compare them to MRAI on a recent topology of the Internet. Our simulations show that MRPC has far better properties than MRAI in terms of the number of transient messages exchanged, while allowing to control the convergence time of the protocol.

## 1. INTRODUCTION

Convergence problems have been identified in BGP since almost a decade [18, 23]. Two delay mechanisms have been introduced in BGP to reduce the number of routing messages exchanged between routers: MRAI and route flap damping. *MRAI* (Min Route Advertisement Interval Timer) [35] is a timer that determines the amount of time that must elapse between consecutive announcements from a BGP router to a neighbor. As a router expects learning an improved route before its MRAI timer expires, MRAI supposedly addresses

path exploration. Route flap damping [40] addresses a different problem: dynamics generated by faulty and flapping hardware. Route flap damping filters the propagation of consecutive routing updates belonging to a prefix that are following each other over too small time periods, which is a sign of instability. Finding the right values of these timers that fit most situations appears impossible [5, 9, 16, 26, 27], as routing dynamics depends on the type of failures, their location, and on the network topology. The main issue with MRAI is that it slows down at the same time the propagation of obsolete and valid paths. Route flap damping may also slow down the propagation of messages not rooted from instabilities but from normal convergence, making the problem of slow convergence in the Internet even more acute in some situations [5, 27].

In this paper, we focus on improving routing protocols behavior during convergence, by imposing a proper ordering of the routing messages during propagation through carefully designed timers. This ordering of the routing messages reduces the number of messages exchanged by routers, while at the same time not slowing down convergence. Moreover, no inconsistent states may occur during convergence. Our timers, called MRPC, stand for *metrics and routing policies compliant*, because they are set independently by each router and depend on the metrics of the routes received by the router, and/or the routing policies of the router. Our solution does not require that neighboring ASs share information about their routing policies.

Besides our contribution related to the design of MRPC timers to improve convergence, we also complete the RAML framework [17], by adding a scheduling mechanism to tackle the exchange of messages.

After providing some background information on routing protocols in Section 2, we illustrate with a simple and intuitive example our timers in Section 3.1, with a routing protocol based on the usual shortest path algebra. We give the theoretical building blocks of our approach in Section 3.2. In Section 3.3, we describe in a general setting how to compute our MRPC timers, and prove under which conditions they can be computed. As our timers are aimed at being used in real routing protocols, we apply our framework to specific routing algebras in Section 4. This leads us to consider

MRPC timers for the particular case of BGP in Section 5. To illustrate the interest of our solution, we compare in Section 5 the performance of MRPC timers with the current solution implemented in the Internet, MRAI. As routing protocol convergence concerns topological dynamics, not only new destinations announced to the network, we explain how our timers work under general topological events in Section 6 and in flooding protocols. Related work is presented in Section 7, before concluding in Section 8. Note that the proofs of the propositions and theorems stated in this paper can be found in [39].

## 2. BACKGROUND

### 2.1 Routing protocol basics

Routing protocols aim at building forwarding paths towards any destination of a network. This is achieved by exchanging routing information between neighboring routers through routing adjacencies or sessions. Routers announce to their neighbors routing information related to the destinations they can reach and based on this information, they build a routing table selecting for every destination, the neighbor that should be used to forward traffic. The two main types of routing protocols are distinguished by the topological knowledge routers can have about the network:

- *Flooding*: Routers exchange the state of the adjacencies they share with their neighbors. These *link states* are then flooded to all routers of the network. This enables any router to build a map of the whole network, which is then used to compute shortest paths to any destination. When a router learns new routing information, it recomputes its best paths in case some would have been modified and forwards the received link states to its neighbors. These protocols are called link-state protocols. OSPF and IS-IS are examples of such routing protocols.
- *Incremental*: When a router learns routing information from a neighbor, it applies an *import routing policy* which might filter or modify it. Then, if this information triggers the re-computation of part of its routing table, the router may announce new routing information to its neighbors according to its *export routing policy*. Routing information in the form of distance or path vectors is propagated hop-by-hop across the network, not flooded. Routers do not know the whole network topology. RIP or EIGRP are examples of such protocols. The most widely used is the BGP (Border Gateway Protocol) [35], the current interdomain routing protocol.

Routing information is triggered by topological dynamics: link, router or destination appearance or disappearance, metric change... When a routing protocol converges from a state to another because of some topological modification, two issues may arise:

*Routing loops*. Transient routing loops may appear in flooding and incremental routing protocols. Transient loops appear during the convergence of routing protocols, when routers use a topology or paths that correspond to obsolete network connectivity information. Solutions to ensure loop-free routing have been proposed decades ago [14, 21, 28], as well as more recently [12, 25, 30, 34]. These solutions require that routers ask and wait for clearance from their neighbors before re-routing. The approach we propose in this paper is not addressing the general problem of routing loops. Our solution guarantees that routing loops due to obsolete paths do not occur during convergence. Our solution does not require the exchange of special messages between routers, but is based on distributed computation of timers by each router. Furthermore, we do not require all routers to implement our timers, allowing partial and incremental deployment in the Internet.

*Path exploration*. Path exploration is specific to incremental protocols. Path exploration happens when a router has more than one neighbor announcing a path to a given destination. Depending on the arrival order of these announcements, the router might explore transient paths before learning and converging to its best path. Because routers rely on each other's paths in incremental protocols, a router exploring paths may trigger the exploration of paths by its neighbors. This phenomenon may spread across the network, leading to an explosion of the number of messages exchanged and may dramatically slow down network convergence.

### 2.2 Routing protocol models

A routing protocol can be modeled in a formal way, through four main mechanisms:

1. *Comparison*: Each path metric may be compared and ordered according to a decision process. This decision process can usually be modeled with a total preorder relation (see [17, 20]).
2. *Concatenation*: Each path metric related to an incoming (resp. outgoing) route may be modified according to an import (resp. export) policy.
3. *Diffusion*: Each router may accept a subset of the incoming routes and forwards them to a subset of the neighboring routers.
4. *Scheduling*: Each router may delay routing messages.

Metarouting [17, 20] proposes to model routing protocols using algebraic structures, in order to prove mathematically whether a routing protocol converges or not. This framework, called RAML (Routing Algebra Meta Language), focuses on the final state reached by the network protocol, e.g., does the routing protocol converge towards a local or a global optimum. For this, RAML relies on the comparison and concatenation operators (see Figure 1). However,

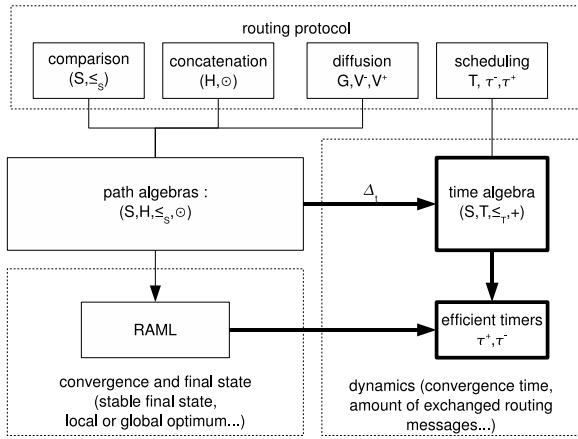


Figure 1: A routing protocol model.

RAML does not address network transient states during convergence, e.g. the amount of exchanged messages, routing consistency, network convergence time, etc.

In this paper, we are rather interested in the behavior of routing protocols during convergence, in terms of messages exchanged and total convergence time, as well as forwarding consistency. For this, we need to add a scheduling mechanism to the RAML framework, as shown in Figure 1. The novelty of this paper is in bold on Figure 1, with the time algebra and the efficient timers. The MRPC timers we build in this paper are one instance of efficient timers.

### 3. MRPC TIMERS

#### 3.1 Overview

We first concentrate on path exploration avoidance in the case of a router announcing a destination to the whole network. Flooding protocols do not suffer from path exploration, as their routing messages describe link state used to first build the network topology, based on which paths are computed. In incremental protocols on the other hand, routers exchange path or distance vectors, that may trigger path exploration until they learn their best possible path. Path exploration can be avoided if each router is able to learn its best possible route first, i.e. if the arrival order of the routes is ideal. In this case, the number of messages exchanged can be significantly decreased. The natural way to impose a particular arrival order of the routing messages consists in delaying routing information propagation. Each router delays some messages by installing timers, in order to favor a particular arrival order of the messages.

Our MRPC timers are designed to delay routing messages according to the path metric carried in the routing message and/or according to the routing policy configured on the crossed arc. If the routing protocol does not have a notion of routing policies (e.g., RIP), then routers will only rely on the path metric. If the routing protocol allows to express routing policies (e.g., BGP), then MRPC timers can be based

on both the path metric and its routing policies. When policies are used, each router configures one timer per incoming arc, depending on the router's import policy, and one timer per outgoing arc, depending on its export policy. An import (export) policy is the set of rules that modify a learned (advertised resp.) path. Note that a router is not obliged to rely on the route metric at all to build its MRPC timers. It may rely exclusively on its routing policies as in the example of Figure 2.

Let us first illustrate the basic idea with an example. We consider a routing protocol using the usual shortest path algebra, where computing the path length consists in summing the weights of the crossed arcs, and where shortest paths are preferred. Router  $a$  announces to the whole network a reachable destination (see Figure 2). We assume that a router only forwards its best path to its neighbors and only knows its own routing policy, i.e. the weight installed on its inbound arcs.

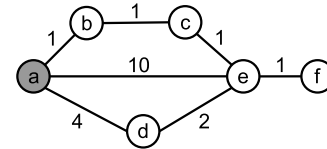
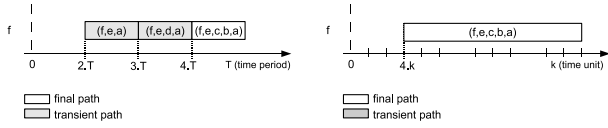


Figure 2: Router  $e$  prefers path  $(e, c, b, a)$  over path  $(e, d, a)$  over path  $(e, a)$  to reach router  $a$ .

Under these assumptions, router  $e$  prefers path  $(e, c, b, a)$  over  $(e, d, a)$  over  $(e, a)$  to reach router  $a$ . We therefore want to make sure that router  $e$  will first learn path  $(e, c, b, a)$  so that its neighbors (router  $f$  especially) will not explore nor even hear about  $e$ 's transient paths. Clearly, if each router delays updates in the same way for any neighbor (as the current MRAI does), then the ideal scheduling will not be achieved. This behavior is illustrated on Figure 3: each routers waits for a constant period  $T$  before updating its neighbors. As a consequence, router  $e$ 's shortest path is the last to be learned as it is the longest in terms of hops and router  $f$  explores transient paths before converging too. Intuitively, if each router delays routing updates on a per-neighbor basis such that the induced delays are proportional to the weights installed on its inbound arcs (see Figure 4), router  $e$  would learn path  $(e, c, b, a)$  after  $3 \times k$  time units, path  $(e, d, a)$  after  $6 \times k$  time units and path  $(e, a)$  after  $10 \times k$  time units (with  $k$  a common scaling factor). Therefore, router  $e$  would immediately converge to its best path, other paths would then be learned but not re-announced to neighbors and router  $f$  would not learn about them nor explore them. Scheduling would then be ideal as updates are propagated in the same order as routers would have been examined in Dijkstra's algorithm. The corresponding timers comply with the metrics and routing policies of the protocol. In the next section, we explain the properties a routing protocol must satisfy for MRPC timers to be computed as well as explain how to compute them.



**Figure 3: Constant timers:** router  $f$  hears about router  $e$ 's transient paths and explores them before converging. **Figure 4: MRPC timers:** router  $f$  converges immediately without hearing about router  $e$ 's transient paths.

### 3.2 Generalized path algorithms

We start from the formalism of Gondran and Minoux [15] and introduce several concepts necessary to understand endomorphism semi-rings. Endomorphisms semi-rings are algebraic structures which allow generalizations of path algorithms. Let  $G(V, E)$  be a directed graph representing the network topology, and  $S$  the metric set related to a path in  $G$ .

*Comparison.* Let  $\preceq_S$  be a total preorder<sup>1</sup> over  $S$ . We define the following binary relations:

$$\forall s, s' \in S, s \approx_S s' \Leftrightarrow (s \preceq_S s') \wedge (s' \preceq_S s)$$

$$\forall s, s' \in S, s \prec_S s' \Leftrightarrow (s \preceq_S s') \wedge \neg(s \approx_S s)$$

We denote by  $\infty_S \in S$  the *infinite metric*. This particular metric verifies

$$\forall s \in S, s \preceq_S \infty_S \text{ and } s \approx_S \infty_S \Rightarrow s = \infty_S.$$

*Concatenation.* Let  $H$  be the subset of endomorphisms over  $(S, \preceq_S)$  defined by  $H = \{h : S \rightarrow S, [\forall s, s' \in S, s \preceq_S s' \Rightarrow h(s) \preceq_S h(s')] \wedge [h(\infty_S) = \infty_S] \wedge [\forall s \in S, s \preceq_S h(s)]\}$ . We distinguish in this set two particular endomorphisms:

$\infty_H : s \mapsto \infty_S$  the function that always returns  $\infty_S$ ,

$0_H : s \mapsto s$  the identity function of  $S$ .

Let  $\preceq_H$  be the preorder over  $H$  such that:

$$h \preceq_H h' \Leftrightarrow \forall s \in S, h(s) \preceq_S h'(s).$$

$\preceq_H$  is induced by  $\preceq_S$  over  $H$  [15]. In the sequel, we call  $H$  the set of *concatenation functions* of  $S$ . Concatenation functions model how the metric associated with a route is changed when the route crosses an arc, as illustrated on Figure 5.

We install on each arc  $(u, v) \in E$  a function  $h_{uv} \in H$ . We denote by  $\odot$  the reversed composition operator. If  $\circ$  denotes the usual function composition operator, we have

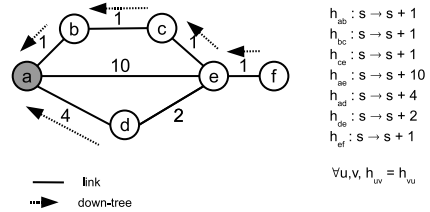
$$\forall h, h' \in H, h \odot h' = h' \circ h.$$

<sup>1</sup>A preorder is a reflexive and transitive binary relation. For more details see [15].

*Generalized path algorithms.* If the previously mentioned properties are satisfied,  $\mathcal{A} = (H, S, \preceq_S, \odot)$  is said to be an *endomorphism semi-ring*. Gondran and Minoux [15] have proposed two generalizations of Dijkstra's algorithm:

- *Direct generalization:* this algorithm requires  $\preceq_S$  to be a total order. With this algorithm, each vertex is examined only once, in the same order as Dijkstra's algorithm is visiting vertices, inducing no path exploration.
- *Greedy algorithm:* this algorithm only requires  $\preceq_S$  to be a preorder, not a total preorder. In this case, vertices may be examined several times, inducing path exploration.

Gondran and Minoux proved that these algorithms converge to an optimal state, leading to a shortest paths tree if  $\mathcal{A}$  is an endomorphism semi-ring. If the generalized Dijkstra algorithm is applicable, then we can apply our MRPC timers and each router will converge directly to its final state (without path exploration). In our context, the source router in Dijkstra's algorithm corresponds to the sink (i.e. the origin of the prefix), and each arc concatenation function  $h_{uv}$  corresponds to the way  $v$  perceives the path metric announced by  $u$ . We call *down-tree* the union of the shortest paths to the sink (see Figure 5) and *up-tree* the reversed tree. If the previous algorithms can be applied, i.e. if the routing protocol can be modeled as an endomorphism semi-ring  $\mathcal{A}$ , we have the guarantee that a down-tree to the sink exists and that the routing protocol converges to a global optimum. If the algebraic properties of the routing protocol are not strong enough to apply the direct generalization of Dijkstra's algorithm, as in the case of the greedy algorithm, then path exploration may occur.



**Figure 5: A small graph instance. Router  $a$  announces a network destination to the whole network.**

To avoid path exploration, each router has to learn its preferred route before the other routes. This can be achieved by favoring message propagation along the up-tree. More precisely, we have to verify the following rule: *the more a routing policy increases the path metric, the more the message should be delayed*. If  $\mathcal{A}$  is an endomorphism semi-ring, enforcing this rule consists in enforcing the same arrival order as the vertex examination in Dijkstra's algorithm. Note that the rule we chose is stronger than forcing routers to learn their best route first. We chose this stronger rule due to its

better convergence properties during topological events (see Section 6).

### 3.3 Foundations of MRPC timers

We want to build a timer  $\tau_{uv} \in T$  for each arc  $(u, v)$ , which depends on the path metric of the routes sent by  $u$  to  $v$ , where

$$T = \{\tau : S \rightarrow \mathbb{R}^+ \cup \{+\infty\}\}.$$

We distinguish  $0_T$ , the null-timer, and  $\infty_T$ , the infinite timer, two particular timers such that:

$$\forall s \in S, 0_T(s) = 0;$$

$$\forall s \in S, \infty_T(s) = +\infty.$$

Moreover, we define the operator  $+$  and the binary relation  $\prec_T$  as follows:

$$\forall \tau, \tau', \tau'' \in T, \tau'' = \tau + \tau' \Leftrightarrow \forall s \in S, \tau''(s) = \tau(s) + \tau'(s);$$

$$\forall \tau, \tau' \in T, \tau \prec_T \tau' \Leftrightarrow \forall s \in S, \tau(s) < \tau'(s).$$

We are looking for a transformation which maps a concatenation function  $h_{uv}$  onto a timer function  $\tau_{uv}$ . We denote by  $\Delta_t : H \rightarrow T$  this transformation.

Contrary to MRAI timers, MRPC timers can delay update messages according to the path metric  $s$  carried in the routing message. Note that if some routers do not implement MRPC timers, the only drawback is the possible exploration of some transient paths before convergence is reached. This property of being incrementally deployable is natural for MRPC timers, it does not imply any constraint on their design. We now present the different properties  $\Delta_t$  must satisfy. These properties lead to the functional system, summarized in Section 3.4, used to compute the MRPC timers.

For the interested reader, the proofs of all propositions and theorems stated in this section can be found in [39]. They are skipped in this paper due to space limitations, as well as not to burden readers uninterested in the mathematical details.

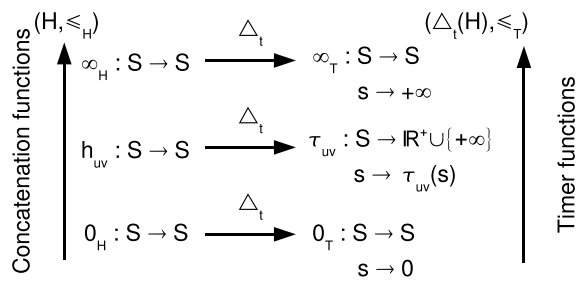


Figure 6:  $\Delta_t$  preserves  $\preceq_H$  ordering.

#### Arrival time ordering

PROPOSITION 1. *Routing messages must reach a given router in same order as its path preferences. The router thus converges directly towards its final state.*

This proposition states that path exploration is totally avoided if the right arrival ordering can be enforced for any router of the network.

THEOREM 1. *Proposition 1 is equivalent to:*

$$\forall h, h' \in H, h \prec_H h' \Rightarrow \Delta_t(h) \prec_T \Delta_t(h').$$

The property derived in Theorem 1 makes the first equation of our functional system. We can interpret this theorem as follows:  $\Delta_t$  is a morphism from  $(H, \prec_H)$  to  $(\Delta_t(H), \prec_T)$  (see figure 6). In other words, any concatenation function can be transformed into a positive delaying function while preserving the  $\prec_H$  ordering. For instance, the delay induced by  $0_H$  (no increase in the path metric) is null, whereas the delay induced by  $\infty_H$  (path filtering) is infinite. Moreover, we can deduce from this theorem that messages following different paths of equal metrics are equally delayed (see corollary 1).

COROLLARY 1. *The messages following different paths of equal metrics are equally delayed. If we denote by  $h$  and  $h'$  the concatenation functions related to those two paths (obtained by composing the functions of each of their arcs), we have:*

$$h \approx_H h' \Rightarrow \Delta_t(h) = \Delta_t(h').$$

#### Timer summability

PROPOSITION 2. *Timers summability*  
The propagation time along a path must be equal to the sum of the propagation times along each arc of the path.

We can prove that Proposition 2 is equivalent to:

$$\forall h, h' \in H, \Delta_t(h \odot h') = \Delta_t(h) + \Delta_t(h').$$

This property leads to the second equation of our functional system. Additionally, we obtain two particular bounds from this property:

$$\Delta_t(0_H) = 0_T \text{ and } \Delta_t(\infty_H) = +\infty_T.$$

These bounds are then used to solve our functional system (third equation). Note that if  $(H, S, \preceq_H, \odot)$  is an endomorphism semi-ring, we can prove that the delays induced by our timers are always positive.

THEOREM 2. *Proposition 2 is equivalent to:*

$$\forall h, h' \in H, \Delta_t(h \odot h') = \Delta_t(h) + \Delta_t(h')$$

**COROLLARY 2. Particular bounds**  
 $\Delta_t(0_H) = 0_T$  and  $\Delta_t(\infty_H) = +\infty_T$ .

**COROLLARY 3. Timers existence condition**  
*If  $(H, S, \preceq_S, \odot)$  is an endomorphism semi-ring, the timers constructed by  $\Delta_t$  always return positive delays,  $\forall s$ .*

Note that corollary 3 provides a condition for the existence of MRPC timers, as implementable timers must be positive. Going back in time is not allowed.

### Confidentiality

**PROPOSITION 3. Confidentiality constraint**  
*A router must be allowed to define MRPC timers only based on its own routing policy and/or the metric related to the path of the received route.*

The name *confidentiality constraint* stems from its purpose: not to reveal the routing policies of neighboring ASs. A router delays routing messages according to its own routing policies and/or the path metric of the received route, without revealing its policies to its neighbors nor requiring to know anything about the routing policies of its neighbors. This constraint is important, especially in today's Internet where routing policies play such an important role and must be kept confidential.

**THEOREM 3.** *If it is possible to define MRPC timers on a per in-arc basis, then it is possible to define timers on a per-routing policy basis.*

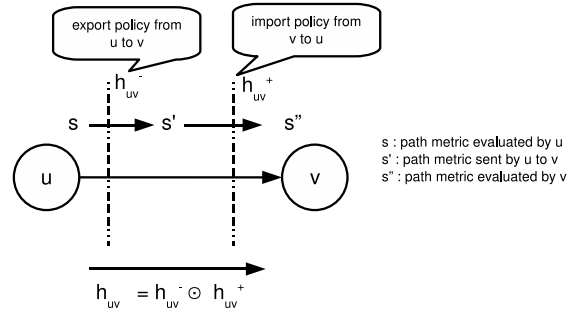
If the computation of MRPC timers was to involve the sharing of routing policies between neighboring ASs, that would constitute a major impediment on the deployment of our timers. However, as Theorem 3 shows, it is possible to satisfy the confidentiality constraint by splitting the timer  $\tau_{uv}$  into two separate timers (see Figure 7):

1. An outgoing timer  $\tau_{uv}^+ = \Delta_t(h_{uv}^+)$  applied by  $u$  to messages sent to  $v$ ,
2. An incoming timer  $\tau_{uv}^- = \Delta_t(h_{uv}^-)$  applied by  $v$  to messages learned by  $u$ .

If routers  $u$  and  $v$  belong to different ASs, they do not have to share their routing policies. They simply apply their own.

### 3.4 Summary

Let  $(H, S, \preceq_S, \odot)$  be an endomorphism semi-ring describing a routing algebra in a graph  $G(V, E)$ . Each router installs a timer per incoming and per outgoing arc, according to the transformation  $\Delta_t$ .  $\Delta_t$  is used by all routers of the network,



**Figure 7: The concatenation function installed on an arc can be split into two successive concatenations: the one due to the export policy of  $u$  ( $\tau_{uv}^+$ ), and the one due to the import policy of  $v$  ( $\tau_{uv}^-$ ).**

and it transforms a routing policy (modeled by a concatenation function) into a timer, i.e. a function that maps a route to a delay. The transformation  $\Delta_t$  generates MRPC timers if and only if it verifies the following properties:

1. *Arrival ordering:*  $h \prec_H h' \Rightarrow \Delta_t(h) \prec_T \Delta_t(h')$ .
2. *Summability:*  $\Delta_t(h \odot h') = \Delta_t(h) + \Delta_t(h')$ .
3. *Bounds:*  $\Delta_t(0_H) = 0_T$  and  $\Delta_t(\infty_H) = +\infty_T$ .

This set of equations allows us to find the transformation  $\Delta_t$  for a given instance of path algebra that models a given routing protocol. Section 4 describes how to build MRPC timers for several basic path algebras, as well as more complex ones.

## 4. MRPC TIMERS AND PATH ALGEBRAS

### 4.1 MRPC timers computation

Let be  $G(V, E)$  a weighted graph whose metrics belong to  $S$ , totally ordered by  $\preceq_S$ . We restrict  $H$  to  $\{h_\lambda : s \mapsto s \otimes_S \lambda, \lambda \in S\}$  where  $\otimes_S$  denotes a global concatenation operator. If  $(S, \otimes)$  is a monoid having  $0_S$  as identity element and  $\infty_S$  as absorbing element, we have a particular case of our formalism.  $(S, \preceq_S, \otimes_S)$  is called a *base algebra* [17]. We now show how to find MRPC timers for the main base algebras. We denote by  $\tau_\lambda$  the MRPC timer corresponding to  $h_\lambda \in H$ .

#### 4.1.1 Some base algebras in Metarouting [17]

- $(\mathbb{R}^+ \cup \{+\infty\}, \leq, +)$  is the *usual* (additive) shortest path algebra.
- $([1, +\infty], \leq, \times)$  is the *multiplicative* algebra. We multiply metrics instead of adding metrics to compute path lengths. Packet losses on a path are the product of the losses on each arc of the path.

	Algebra $(S, \preceq_S, \otimes_S)$	MRPC timer $\tau_\lambda, k > 0$
Shortest paths	$(\mathbb{R}^+ \cup \{+\infty\}, \leq, +)$	$s \mapsto k \cdot \lambda$
Multiplicative algebra	$([1, +\infty], \leq, \times)$	$s \mapsto k \cdot \ln(\lambda)$
Availability	$([0, 1], \geq, \times)$	$s \mapsto -k \cdot \ln(\lambda)$
Bandwidth	$(\mathbb{R}^+ \cup \{+\infty\}, \geq, \min)$	$s \mapsto k \cdot (s - \lambda)$ if $s - \lambda > 0$ , 0 otherwise
Sequence	$(SEQ, \preceq_{SEQ}, \cdot)$	$s \mapsto k \cdot  \lambda $
Constrained local preference	$(\mathbb{N}_k \cup \{+\infty\}, \geq, \otimes_{LP})$	see Section 4.1.2

**Table 1: Some base algebras defined in [17] and their corresponding MRPC timer. We denote by  $\lambda$  the metric installed on the link. We denote by  $k > 0$  a global constant factor used by the all routers of the network.**

- $([0, 1], \geq, \times)$  is the usual *availability* algebra. A path is better if its availability is higher ( $\geq$ ), and the availability of a path is the product ( $\times$ ) of the availabilities of its arcs.
- $(\mathbb{R}^+ \cup \{+\infty\}, \geq, \min)$  is the *bandwidth* algebra. A path is better if it has more bandwidth ( $\geq$ ). The bandwidth of a path depends on the bandwidth of the bottleneck link (*min*).
- $(SEQ, \preceq_{SEQ}, \cdot)$  is the *sequence* algebra. A path is described by a sequence, e.g., the identifiers of the routers crossed on the path to the destination. A path is shorter if its corresponding sequence length (obtained through  $|\cdot|$ ) is shorter. The operator  $\cdot$  consists in concatenating sequences carried by the path metric and the metric installed on the concatenated link.  $\cdot$  may also return  $\infty_S$  if a loop is detected.

Table 1 presents the corresponding MRPC timers. These timers have been found thanks to the functional system mentioned in Section 3.4.

#### 4.1.2 Constrained local preference algebra applied to BGP

In the Internet, neighboring ASs are connected according to one of the following economical relationships:

- *Customer to provider (c2p)*: the customer pays its provider for both its incoming and outgoing traffic.
- *Peer (peer)*: the two connected ASs exchange for free the traffic having their customers as destination.
- *Provider to customer (p2c)*: the provider is paid for the traffic it sends or receives from its customer.

Therefore, an AS prefers sending its traffic to its customers because this traffic brings money, then to its peers as it does not cost a thing and finally to its providers because it has to pay for it. A consequence of this behavior is that inter-domain paths usually verify the following regular expression first presented in [13]:  $(c2p)^*(peer)^?(p2c)^*$  (i.e. a path can be made of 0 or more *c2p* relationships, followed by 0 or 1 *peer* relationship, followed by 0 or more *p2c* relationships). Note that the previous AS relationships are a simplified view of real policies implemented in the Internet [6].

MRPC timers are not bound to those particular AS relationships types. We use them to illustrate what can be achieved in a common setting of routing policies.

Business relationships are typically configured in BGP routers thanks to an attribute of the routes, local preference (local-pref), whose value reflects the interest for an AS to use a neighboring AS to send traffic. An AS usually sets a high local-pref value to routes learned from customers (for instance 150), an intermediate value to routes learned from peers (for instance 100), and a small value to routes learned from providers (for instance 50). The actual local-pref values of the routes do not matter for our formalism, only the ordering of the routes that follows from those values<sup>2</sup>.

Based on the previous regular expression, inbound-outbound arc pairs on which it is allowed to propagate a BGP route are:

$$\{(c2p, c2p), (c2p, peer), (c2p, p2c), (peer, p2c), (p2c, p2c)\}$$

We now present an algebra that can be applied to BGP to model local preferences. We define  $S = \{1, 2, 3\}$ , corresponding to the 3 values of local-pref corresponding of those simplified import routing policies (respectively  $\{p2c, peer, c2p\}$ ). The previous set of valid inter-domain inbound-outbound arc pairs becomes therefore:

$$P = \{(3, 3), (3, 2), (3, 1), (2, 1), (1, 1)\}.$$

We define  $\preceq_{LP}$  and  $\otimes_{LP}$  two operators on  $S$  as follows:

$$\forall i, j \in S, i \preceq_{LP} j \Rightarrow i \geq j$$

$$\forall i, j \in S, i \otimes_{LP} j = \begin{cases} j & \text{if } (i, j) \in P, \\ \infty_S & \text{otherwise.} \end{cases}$$

$\geq$  states that a router always prefers the path with the higher local preference, while  $\otimes_{LP}$  states that a router delays an announcement based on the pair of local preferences set on its in and out arcs. The induced timers  $\tau_{ij}$  are defined by

$$\forall s, \tau_{ij} \mapsto \begin{cases} k \cdot (i - j) & \text{if } (i, j) \in P \\ +\infty & \text{otherwise} \end{cases}$$

<sup>2</sup>For simplicity, we assume that each AS equally prefers all neighbors from a given type: customers, peers, and providers. If an AS originates a prefix, we do as if it had received this route from a customer.

with  $k > 0$  positive.

The values of the timers are summarized in Table 2. We notice that when the local preferences on its inbound and outbound arcs are both  $c2p$  (resp.  $p2c$ ), then an AS immediately forwards routing announcements. On the contrary, the larger the difference between the local preference values, the more the routing messages are delayed. For instance, a message is delayed more if it crosses a  $(c2p, p2c)$  arc pair than a  $(c2p, peer)$  or  $(peer, p2c)$  pair. Finally, routing messages are not propagated when local preferences pairs correspond to AS relationships combinations that violate the routing policies.

In practice, we obtain that if an AS is able to learn a destination from a customer (resp. a peer; resp. a provider), then it receives the corresponding message after 0 (resp.  $k$ , resp.  $2.k$ ) time units.

Such timers can be implemented using BGP communities [7], which are already widely used for similar purposes [10]. It consists in tagging incoming  $i$  (resp. outgoing  $j$ ) eBGP routes according to the corresponding peering agreement. A router forwarding a route through an eBGP session would then examine the couple  $(i, j)$  and delay the BGP message according to the timers defined above. Note that this tagging based on communities does not require to change the local-pref values used by routers.

	$c2p$	$peer$	$p2c$
$c2p$	0	$k$	$2.k$
$peer$	$+\infty$	$+\infty$	$k$
$p2c$	$+\infty$	$+\infty$	0

**Table 2: Example delay generated by an AS. This duration depends on the in-agreement (line) and on the out-agreement (column).**

We remind the reader that MRPC timers do not require any particular peering relationship pattern to be enforced on the propagation path. Our approach works with arbitrary path patterns and local preference values as long as local preferences decrease during the propagation of any valid path, as is required by all semi-ring endomorphism. Recall from Section 3.2 that for a routing protocol to be guaranteed to converge, it must be an endomorphism semi-ring. If routing policies that violate the endomorphism semi-ring properties are implemented, then the protocol may not converge and some path exploration may happen even with MRPC.

## 4.2 MRPC timers for BGP

We considered base algebras  $(S, \preceq_S, \otimes_S)$  where the set  $S$  was defined over simple metric spaces like  $\mathbb{R}^+$  or  $\mathbb{N}$ . Some routing protocols, e.g. BGP, use multiple route metrics. Route metrics are then ordered by importance and evaluated through a *decision process* according to a lexicographic comparison: the first attribute is used to rank the routes, then if several routes have the best metric value, the second attribute

is used, and so on until a single best route remains. Previous works [17, 20] have studied what properties are required to ensure either local or global optimal routing solutions when the algebra is built from lexicographic products. In this section, we aim at understanding under which conditions MRPC timers can be found when complex algebras, as introduced in [17, 20], are used. For the interested reader, additional proofs related to this section can be found in [39].

### 4.2.1 Definition

Let  $\mathcal{A}_i = (H_i, S_i, \preceq_{S_i}, \odot_i)$  and  $\mathcal{A}_j = (H_j, S_j, \preceq_{S_j}, \odot_j)$  be two endomorphism semi-rings. Let  $\mathcal{A} = (H, S, \preceq, \odot) = \mathcal{A}_i \overrightarrow{\times} \mathcal{A}_j$  be the lexicographic product of the following algebras [17, 20].

- $S = S_i \overrightarrow{\times} S_j = \{(s_i, s_j) \in (S_i \setminus \{\infty_i\} \times S_j \setminus \{\infty_j\}) \cup (\infty_i, \infty_j)\}$ .  $S$  is the cartesian product of  $S_i$  by  $S_j$ .
- $(s_i, s_j) \preceq_S (s'_i, s'_j) \Rightarrow (s_i \prec_{S_i} s'_i) \vee (s_i \approx_{S_i} s'_i \wedge s_j \prec_{S_j} s'_j)$ . We first compare the metric over  $S_i$ , and if the two metrics are equally preferred, we compare the metric over  $S_j$ .
- $H = H_i \overrightarrow{\times} H_j = \{h_i \overrightarrow{\times} h_j, h_i \in H_i, h_j \in H_j\}$  where  $h_i \overrightarrow{\times} h_j : S \rightarrow S$  is defined by :

$$\forall (s_i, s_j) \in S_i \times S_j, (h_i \overrightarrow{\times} h_j)(s_i, s_j) = (h_i(s_i), h_j(s_j)).$$

### 4.2.2 Lexicographic product of MRPC timers

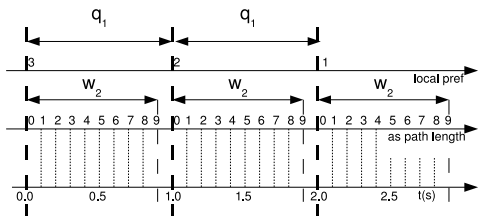
Let  $\mathcal{A}_1 = (H_1, S_1, \preceq_1, \odot_1)$  and  $\mathcal{A}_2 = (H_2, S_2, \preceq_2, \odot_2)$  be two endomorphism semi-rings. Let  $\Delta_1$  and  $\Delta_2$  be two MRPC timer transformations related to  $\mathcal{A}_1$  and  $\mathcal{A}_2$  respectively.  $T_1$  and  $T_2$  are the timer sets related to  $\mathcal{A}_1$  and  $\mathcal{A}_2$  respectively. Let  $k_1, k_2 \in \mathbb{R}_*^+$  be two constants.

[20] studies under which conditions  $\mathcal{A}_1 \overrightarrow{\times} \mathcal{A}_2$  is an endomorphism semi-ring.

Assume that  $S_1$  is a *discrete set* and that  $S'_2 = S_2 \setminus \{\infty_{S_2}\}$  is a *bounded set*. In general,  $\mathcal{A}_1 \overrightarrow{\times} \mathcal{A}_2$  is not an endomorphism semi-ring.

We call *quantum of  $\Delta_1$*  the minimal step between two delays computed by a timer among all the non-null timers over  $T_1$ . We denote this value by  $q_1$ . We call *width of  $\Delta_2$*  the maximal delay computed by any finite timer over  $T_2$ . We denote this value by  $w_2$ .

We now combine timers over  $T_1$  and  $T_2$ . In order to preserve an adequate arrival ordering, we multiply timers over  $T_1$  such that a timer over  $T_2$  always computes a delay negligible compared to a delay computed by a timer over  $T_1$  (see Figure 8). If we consider only the first two steps of the BGP decision process, local-pref and AS path length, we are exactly in this case. The algebra built from peering relationships with local-pref values is discrete. We set an upper bound for the AS path length, for example 9 as on Figure 8. Most AS paths in the Internet are shorter than this bound. The timers for local-pref are set in such a way that their quantum is larger than the maximum delay induced by



**Figure 8:  $\mathcal{A}_1$  is discrete and  $\mathcal{A}_2$  is bounded. The lexical product consists in scaling the  $T_1$  timers to obtain an adequate arrival ordering.**

the AS path length timer. A route that is crossing an arc pair of type  $(c2p, c2p)$  or  $(p2c, p2c)$  will be delayed by 0 time units. A route crossing an arc pair of type  $(c2p, peer)$  or  $(peer, p2c)$  will be delayed by 1 time unit. Finally, a route crossing a  $(c2p, p2c)$  pair will be delayed by 3 time units. If one wants to propagate a routing message on an invalid AS relationship pair, e.g. for backup purposes, the message should be delayed by more than 3 time units to reduce the possibility of path exploration. At the same time, messages are delayed by 0.1 time unit per AS hop in the AS path. Note that if in practice AS paths are longer than the upper bound used to compute the MRPC timers, the only practical consequence will be more path exploration than expected due to a wrong ordering of some of the routing messages. Convergence towards the final state on the other hand will not be compromised.

Let us consider  $h = (h_1, h_2) \in H_1 \vec{\times} H_2$  a concatenation function. If  $h = \infty_H$  we install  $\infty_T$ . Otherwise  $h_1 \neq \infty_{H_1}$  and  $h_2 \neq \infty_{H_2}$  and we install the timer

$$\tau = \Delta_t(h) = k_1 \cdot \Delta_1(h_1) + k_2 \cdot \Delta_2(h_2)$$

where  $k_1 > 0$  and  $0 < k_2 < k_1 \cdot q_1 / w_2$ . Such an approach can easily be extended to lexicographic products of  $n$  algebras instead of 2, as well as to BGP.

### 4.2.3 Dealing with network propagation times

The propagation time of a message between any two routers of the network in terms of medium transmission, queuing and computation is not null. Therefore, MRPC timers must be designed so that their duration is an order of magnitude larger than propagation times inside networks, as is typically the case for default MRAI values [22]. More precisely, for a given network, one first has to estimate  $w_{prop}$ , the worst propagation time between any two routers<sup>3</sup>. One can model propagation times by an algebra  $S_{prop}$  which is bounded by  $w_{prop}$ . As a consequence, as far as the routing protocol used in the network can be modeled by an algebra  $S_1$ , discrete or at least that can be discretized, then by using the lexical product, we can scale timers associated to  $S_1$  in such a way that the arrival order is still ideal despite the propagation times. We model this propagation time across networks

<sup>3</sup>The propagation time contains the router processing time and the transmission time on the link.

in our simulations in Section 5.

The careful reader may have noticed that we did not explicitly model all steps of the BGP decision process in Section 4.2.2, but focused intentionally on interdomain aspects, i.e. local-pref and AS path length. Propagation and convergence inside an AS and across ASs generally take place at different time-scales. Today, MRAI timers for eBGP sessions are an order of magnitude larger than those on iBGP sessions. Convergence inside an AS should happen at the scale of one second or less. However, in large ASs that want to carefully design their timers even with respect to intradomain routing, it can also be done with our formalism, by adding another layer of lexicographic product of MRPC timers for BGP attributes whose scope is limited inside the AS, similarly to [20]. The case of the Multi-Exit Discriminator (MED) attribute is similar to local preference: each router must know all MED value classes used inside the AS in order to be able to properly delay routing messages. For the IGP cost towards the BGP next-hop, the ideal delay would be proportional to the IGP cost of the path followed by the route along the iBGP sessions from the BGP next-hop.

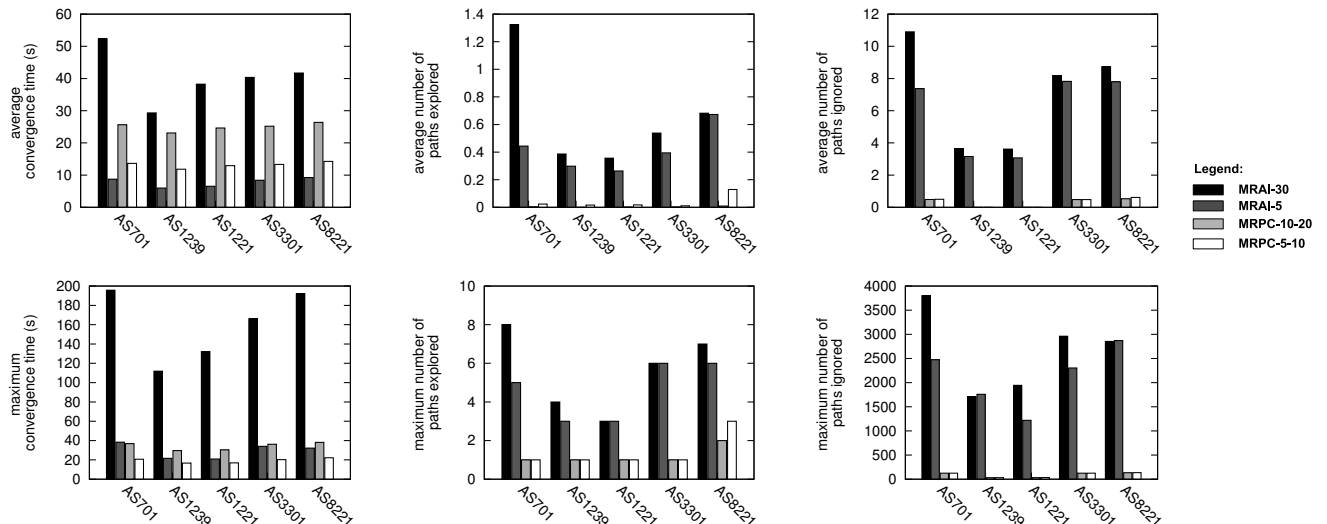
If an AS does not want to include intradomain details in its MRPC timers, it may simply include this part of the MRPC timers within the network propagation time. In such a case, some path exploration may happen *inside* the AS, as it happens today with MRAI, but with properly scaled timers, convergence inside the AS<sup>4</sup> will not trigger routing updates outside the AS before intradomain convergence finished. As already mentioned, if the ordering of the MRPC timers is not ideal, then the only consequence will be potential path exploration. In our simulations of Section 5, we use such an approach: only local-pref and AS path length are explicitly used to compute the MRPC timers. The other steps of the BGP decision process are integrated in the dimensioning of the network propagation time. Such an approach is reasonable if the propagation time inside an AS is small compared to the the timers on eBGP sessions, as should be the case in practice.

## 5. MRPC VS. MRAI TIMERS

The theoretical sections (Section 3.3 and 4) proved the correctness of our approach, i.e. no path exploration is guaranteed if the ordering imposed by the timers is strictly enforced. As no theory tells us how many messages are exchanged between routers with MRAI, we need to rely on simulations to compare MRPC timers and MRAI. In this section, we evaluate the impact of deploying MRPC timers with BGP, and compare them to MRAI.

The current default values of MRAI in the Internet are defined in RFC4271 [35]. The suggested value for MRAI on eBGP sessions is 30 seconds, while it is 5 seconds on iBGP sessions. These values have been found to be too high, both empirically [26] and based on simulations [16]. Re-

<sup>4</sup>In large ASs relying on route reflection or confederations, path exploration may also occur [19, 33].



**Figure 9: Comparison of convergence properties of MRAI and MRPC timers.**

cently, [22] has suggested the use of MRAI values of 5 seconds or less on eBGP sessions, and 1 second or less on iBGP sessions. As pointed out at the IETF in response to [22], the issue of MRAI is not its value, but the fact that one value does not fit all situations [16]. We will confirm in this section that whatever the actual value of MRAI used, the basis fact remains that MRAI is a mechanism that does not explicitly address path exploration and the number of routing messages exchanged during convergence. A proper ordering of the routing messages during convergence is necessary to keep low the number of messages exchanged in all situations.

We have developed a simulator that computes routing message propagation and that triggers each routing event at a specified time. We have built an Internet AS graph made of 29,146 ASs and 78,934 links, as observed by trace collectors [1], and inferred the AS relationships [2]. We assume that each AS implements a consistent routing policy: each AS installs for each customer (resp. peer, resp. provider) a local preference equal to 3 (resp. 2, 1). Remember that the deployment of MRPC timers in the Internet does not require an AS to modify its local-pref values. We only need a mechanism, e.g. communities, to map the peering agreements on the inbound and outbound arcs crossed by the route with our timer values.

Keep in mind that the sole purpose of our simulations is to compare MRPC timers to MRAI to understand their respective performance. We do not expect to capture with our simulations the full complexity of routing protocols behavior in the Internet. We only use them as a benchmark to compare two mechanisms: MRPC and MRAI timers.

We install a random constant delay belonging to  $[0, 1[$  seconds for each AS router. This random delay models the propagation time required to traverse a given AS. We run four different sets of simulations:

1. *MRAI-30*: At the beginning of the experiment, we set for each AS a random value belonging to  $[0, 30[$  seconds corresponding to the next expiration time of its MRAI timer. Then, every 30 seconds, its MRAI expires and the AS forwards its best route if it has been improved during the last MRAI round.
2. *MRAI-5*: The current recommended value in RFC4271 [35] for MRAI is 30 seconds, so the higher bound of 30 seconds was chosen in the previous setting. The recommended MRAI values have recently been revised in [22] to be 5 seconds or less for eBGP sessions and 1 second or less for iBGP sessions. In this setting we do similarly as in MRAI-30, but the first expiration time of the timer belongs to  $[0, 5[$  instead of  $[0, 30[$ . Then the MRAI timer expires every 5 seconds.
3. *MRPC-10-20* we install on each inter-AS link an MRPC timer. Such a timer is obtained by using the lexical product (see Section 4.2) with the local preference algebra and the AS path algebra (see Section 4). In order to bound the AS path algebra, we decide to set the maximal AS PATH length to 10. In some AS paths are longer than this bound, then the only consequence is that some path exploration may occur. This value and the upper bound of the propagation time between two ASs (set to 0.1 s) therefore gives us the duration of the different timers. First, an AS delays a message 1 second per AS hop in the AS path. Then, depending on its in/out AS relationship pair, an AS will add an extra delay of 10 seconds (community belonging to  $(c2p, peer)$  or  $(peer, p2c)$ ) or 20 seconds (community belonging to  $(c2p, p2c)$ ).
4. *MRPC-5-10*: The MRPC-10-20 setting is actually not comparable in convergence time to the MRAI-5 one

that uses far smaller timers hence leads to smaller convergence time. MRPC timers can be scaled down almost arbitrarily, simply by changing how much delay is added per AS hop and per in/out AS relationship pair. The MRPC-10-20 delays are divided by two in this experiment, with 0.5 second per AS hop and 5 (( $c2p, peer$ ) or ( $peer, p2c$ )) and 10 seconds (( $c2p, p2c$ )) for AS relationship pairs.

We focus on three aspects of convergence: network convergence time, number of explored paths, and number of ignored paths. The network convergence time is the time it takes for all routers in the AS-level topology to converge. The number of explored paths refers to the number of transient paths chosen as best by the routers and thus propagated by routers. The number of ignored paths refers to the number of paths a router has learned but never selected as best, before it converges towards its final best. We rely on those three metrics because the literature has given too much importance to the first aspect, network convergence time. To understand routing protocol convergence, we must go beyond the simplistic view of convergence time, and encompass the routing messages exchanged during convergence, i.e. path exploration. We run our simulations by originating a prefix in several ASs, including tier-1 (such as AS701 and AS1239), tier-2 (such as AS1221 and AS3301), and stub ASs (such as AS8221), and measuring convergence time, explored paths, and ignored paths in the whole AS-level topology.

Figure 9 compares the four sets of simulation, by showing results for prefixes originated in the previously mentioned ASs and measuring the three convergence-related metrics across the whole AS-level topology. The AS numbers of the x-axis of Figure 9 represent the AS from which we originate a prefix. Let us first focus on results for MRAI. The *MRAI-5* setting leads to far smaller convergence times (average and maximum) than *MRAI-30*. We can appreciate the particularly bad convergence times of the *MRAI-30* timers, consistent with observations in the Internet [23]. *MRAI-5* leads to pretty fast network convergence, even smaller than MRPC timers for our settings on average (but not in the worst case). The drawback of MRAI, whatever the value of the timers, can be seen in the number of paths explored as well as the number of paths ignored. MRAI tries to prevent transient paths by waiting before sending messages. The global effect is actually to slow down all messages, both those that will not be selected as best at the end of convergence and those that will be selected as best at the end of convergence. This is a very ineffective strategy, as only the paths that will not be selected as best at the end of convergence should be delayed. With MRAI, the number of paths a router learns before learning its best final path is large, especially in the worst-case (lower right graph of Figure 9).

If we turn to MRPC timers, we observe that they perform systematically better than MRAI in terms of path exploration and ignored paths (middle and right graphs of Figure 9). On

average, a router receives as first message the best route he will ever learn, i.e. without incurring path exploration. Even in the worst-case, most routers will explore only one transient path. However, path exploration occurring with MRPC timers does not have a cascading effect as with MRAI, as such transient paths have almost no chance to propagate very far unless these paths are good enough with respect to the path metrics and the routing policies of the routers that receive them. The number of ignored paths illustrates best this point: with MRAI, exploration triggers the propagation of a large number of paths (sometimes thousands) that will never be selected as best by some routers (right graphs of Figure 9). These paths are typically replaced by better paths within a short period of time, but they still unduly consume resources of the routers. Note that the large number of ignored paths is related to the routing diversity available in the Internet [29, 36, 38].

With respect to convergence time, the performance of MRPC timers depends mostly on the choice of the delays set per AS hop and per policy. The convergence of *MRPC-5-10* is exactly half the convergence time of *MRPC-10-20*. This illustrates the scaling properly of MRPC timers. Note that when scaling down MRPC timers, some path exploration may happen due to the stochastic nature of BGP pass-through times [11]. However, the additional path exploration is very small compared to the improvement in convergence time. One of the interesting properties of MRPC timers is that they allow to trade-off smaller convergence time with slightly increased path exploration and ignored messages.

## 6. DEALING WITH DYNAMICS

### 6.1 Incremental protocols and multiple originators

So far, we have considered the nominal case of a new destination announced to the whole network. By construction, routers were updated in the right order from the sink to the leaves of the up-tree and no loop could appear, a router being updated by an up-to-date router.

The injection of a shorter path in the network (link or router restoration, metric decrease) is equivalent to the nominal case. Indeed it corresponds to the spread of a new valid path from one router and can only lead to some routers electing this new path as shortest. Therefore routers joining the resulting new sub-up-tree are still updated in the right order. As a consequence, in this situation we do not have to modify our mechanism. On the other hand, the deletion of a shortest path from the network (link or router failure, metric increase) must be handled somehow differently.

Let us consider the network topology from Figure 10. Each router has several routes to reach destination router  $a$ , and uses as best the route labeled with '>'. The other routes will be stored as backup paths and will be used only if the best route fails.

Assume now that link failure happens in the network, for

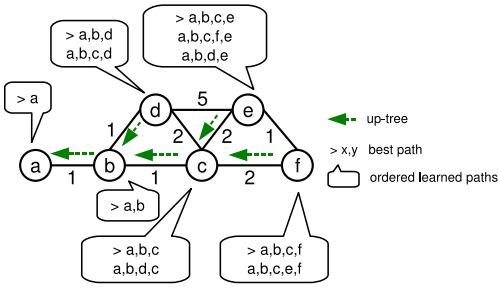


Figure 10: Shortest paths to router *a*.

example link  $(b, c)$  as represented on Figure 11. Given the way they are impacted by this event, routers can be divided into three categories:

- *Safe routers*, the routers which are not impacted by the topological event. These routers are shown in white on Figure 11.
- *Originator routers*, the impacted routers which are still able to compute a valid path on their own. These are the routers which are going to initiate the spread of backup paths into the impacted sub-tree. *Originator routers* are represented in grey on Figure 11.
- *Empty routers*, the impacted routers which have to wait for some new routing information to compute a new valid path. These are represented in black on Figure 11.

To avoid path exploration in such a situation two main issues must be overcome:

- Obsolete paths must be purged from the network. In incremental protocols, some obsolete paths might still be present in the routers Routing Information Bases<sup>5</sup>. This is achieved by (i) letting routers announce explicit withdraw messages to each other without delaying them (this implies that withdraws are forwarded before they are treated by routers) and (ii) scaling MRPC timers so that they do not expire before all the explicit withdraws have been propagated in the network. The value of the scaling factor is obtained by estimating the propagation time of a withdraw in the network. Depending on the network and routing protocol, this value can be either measured or evaluated. Trade-offs can be chosen between (i) a best case propagation time leading to short timer values but potentially to some path exploration, (ii) an average case leading to larger timer values but less path exploration, or (iii) a worst case leading to potentially large timer values but no path exploration.

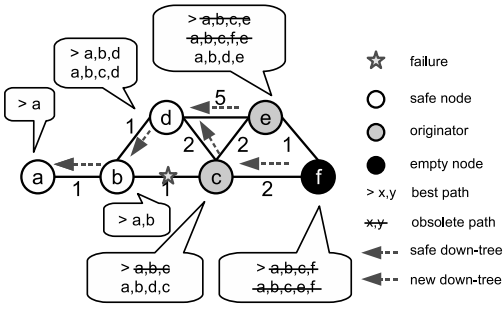
<sup>5</sup>For example, BGP maintains special tables (Adj-RIB-in's) which contain the paths announced by neighbors. A router then elects for each destination the best path and inserts it in its routing table. In the case its best path is withdrawn, it will recompute a new best path based on the routes in its Adj-RIB-in's, which may be obsolete.

- Originators have to be “synchronized” as if the backup paths propagation had been initiated by the sink itself. This is achieved by letting routers wait for an extra delay, called *synchronization delay*. Routers compute it by applying their MRPC timer on the metric of the sub-path from the sink to themselves on their new path to the sink. However, this requires that routers are able to detect that they have switched to a worse path. To do so, we propose that before installing a path to a destination in its RIB<sup>6</sup>, a router compares the path to be installed with the one already in the RIB (this path can be empty, corresponding to an infinite metric). If the new path to be installed is worse, for example in the case of a shortest path deletion somewhere in the network, then the new path is not immediately installed. Instead the router waits for an amount of time equal to the *synchronization delay* before installing the new shortest path in its RIB. However, because obsolete paths are flushed, *empty* routers have no paths for some destinations until they receive new ones from *originators*. During this time *empty* nodes may therefore undergo some traffic loss. To tackle this issue we just need a default value of *synchronization delay* to be defined for the very case of a router having no path to the destination. The value of this *synchronization delay* depends on the network and routing protocol as it is obtained applying a MRPC to the longest path of the network for this protocol (this path can either be measured or evaluated). As a consequence, *empty* routers would keep on forwarding packets on their old best path to the sink *s* until their *synchronization delay*. The key point to understand why traffic loss can be reduced by doing so is that by construction<sup>7</sup> our method ensures that during reconvergence a router in the network can only be in two states: either (1) it is using its old shortest path or (2) it has a new valid path (meaning it has been updated). Therefore, the packets forwarded by an *empty* node *u* on its old shortest path will either only cross *empty* routers and therefore traffic might indeed be lost in case of link failure on this path for instance. Otherwise there is a router *v* on the old shortest path of *u* which have been updated or is at least an *originator* and therefore the traffic will be correctly forwarded from *v* to the sink.

This behavior is illustrated on Figure 11. As previously, we consider a routing protocol using the usual shortest path algebra, where routers only forward their best path to their neighbors. After the failure of link  $(b, c)$ , explicit withdraws are propagated across the network without delays. Once this is done, routers *e* and *c* are *originator nodes* while router *f* is an *empty node*.

<sup>6</sup>A RIB is the set of best routes chosen by the protocol to each each destination.

<sup>7</sup>As far as MRPC timers are large enough compared to the propagation time of withdraws.



**Figure 11: Routers  $c$  and  $e$  originate the backup paths propagation.**

Router  $e$  compares its new shortest path to router  $a$ ,  $(a, b, d, e)$ , whose weight is 6, with the previous shortest path,  $(a, b, c, e)$ , whose weight is 4. The new path is worse than the previous one, hence router  $e$  waits for  $6 \times k$  (the *synchronization delay* computed on the metric associated to the path  $(a, b, d, e)$ ) before installing this new path in its RIB, then it waits  $1 \times k$  time units (as the value of its MRPC timers as the weight on the arc  $(e, f)$  is 1) before announcing the path to  $f$ . The resulting delay induced by router  $e$  is  $7 \times k$  time units. Router  $c$  does the same and obtains a delay of  $6 \times k$  time units. As a consequence, router  $f$  learns its new best path  $(a, b, d, c, f)$  first. What is more as far as  $c$  is up-to-date, then the traffic of  $f$  indeed reaches  $a$  even if  $f$  has not been updated yet.

With this add-on to the mechanism, dynamics is handled in the same way as in the nominal case. Once obsolete paths have been flushed and if timers are large enough compared to the propagation time of withdraws, no path exploration nor loops appear. Traffic loss may also be reduced.

## 6.2 Flooding protocols and FIB updates

So far, in the case of incremental protocols, our MRPC timers were based on the routing policies of each link and the metric of the route received by the router. Each router was delaying routing messages according to its MRPC timers.

With a flooding protocol, routing messages contain information about link states and are not related to a destination that originates the routing message. However, it is possible to reproduce in flooding protocols a behavior similar to the one of incremental protocols. Instead of delaying routing messages, we delay the time when a route is installed in the forwarding information bases<sup>8</sup> (FIB). The routing messages are then forwarded as is usually the case in flooding protocols.

When a path is changed, its installation in the FIB is done according to the MRPC timer related to the total cost of the path from the router to the sink. Its installation in the

<sup>8</sup>A Forwarding Information Base (FIB) in a router is a table that keeps mappings between destinations and an output interfaces where the packet has to be sent.

FIB is independent from the path followed by the routing message. If a router  $u_p$  received a routing message that requires it to use a new shortest path to reach  $u_1$ , denoted by  $\mu = (u_1, u_2, \dots, u_p)$ , then  $u_p$  installs this path in its FIB after an amount of time equal to

$$t_\mu = \sum_{i=1}^{p-1} (\Delta_i(|u_i, u_i + 1|))$$

Example 1: In the IS-IS protocol using the usual shortest path algebra:

$$t_\mu = k \times \sum_{i=1}^{p-1} (|u_i, u_i + 1|)$$

Example 2: In the multiplicative algebra

$$t_\mu = k \times \sum_{i=1}^{p-1} (\ln(|u_i, u_i + 1|))$$

Note that the timers are independent from the topology structure. The behavior is however similar to the one of MRPC timers in incremental protocols. Path exploration however does not happen. Furthermore, the concept of *synchronization delay* or *originator node* have no meaning in this context since routing messages are not related to sinks nor propagation paths.

Nevertheless, by construction of the timers, the more a node will be close to the destination, the closer the corresponding paths will be installed inside the FIB. We can thus prove by recurrence that a packet crossing a router that uses an obsolete path, follows at first an obsolete path. Given that this obsolete path has finite length, the packet will have to cross a router that has already converged or the sink itself. Therefore, traffic is still correctly forwarded towards the sink.

## 7. RELATED WORK

Many works investigate improvements to BGP convergence. Most have focused on obsolete path detection in order to avoid their propagation in the network [3, 8, 31, 32, 41]. This is most of the time achieved using heuristics on the AS path attribute [3, 31, 32]. To overcome lack of information about convergence, many authors also propose modifications to the protocol by adding extra information in BGP updates [8, 31, 32, 41]. For instance, [8] adds information to the AS path attribute to identify dependencies between paths to distinguish between valid and invalid paths. [41] on the contrary tends to prevent convergence from malign instability by rejecting obsolete paths through fault information carried in BGP updates.

Other works have tackled the issue of transient paths exploration. For instance, [4] proposes to delay BGP route propagation according to carried AS path, but the authors only consider the AS path length decision step. [24] presents a dynamic adjustment of the MRPC timers based on the ac-

tivity (number of messages received) related to this destination. In [37], the authors suggest to forward firstly routing messages along the *current* covering shortest path tree. They propose a heuristic computed by each router in order to infer which neighbors cross it to reach the destination. As routing messages are diffused on the obsolete shortest path tree, path exploration cannot be avoided, especially in the case when a new link or a new router appears.

## 8. CONCLUSION

Internet routing convergence has been shown to be problematic since almost a decade [18, 23]. In this paper, we explained how to *improve the convergence* of routing protocols through timers that enforce a proper ordering of the routing messages, therefore limiting path exploration to a minimum. We designed such timers, called MRPC, for *metrics and routing policy compliant*. MRPC timers depend only on the path metrics of the routes received by a router and its routing policies. No knowledge about routing policies from other ASs is assumed in our solution.

We explained and proved under which conditions MRPC timers can be computed, and applied it to the case of BGP. We compared the performance of MRPC timers against the current mechanism that limits the exchange of routing messages during convergence, MRAI. Using low MRAI values has been shown to reduce convergence time, but a one value fits all situations does not exist [16]. We showed that MRPC timers significantly reduce the number of exchanged messages. Furthermore, they can be scaled down to obtain very low convergence time while also keeping low the number of exchanged messages. Also, no major modification to the working of current routing protocols is required. Finally, our approach is generic, in that it applies to a large class of routing protocols, including existing flooding and incremental protocols. Our solution works on the ordering of the of the routing messages exchanged in incremental protocols, while in flooding protocols FIB updates are updated in a specific order.

So far, routing algebras have been largely focused on the comparison and concatenation mechanisms. In this paper, we worked on the scheduling mechanism. The fourth mechanism, diffusion, has still to be studied. Current routing protocols rely on manually configured diffusion topologies to propagate the routes across the Internet. Such diffusion topologies are hard to design as many different considerations are relevant, like correctness, optimality, and scalability. Further work towards establishing a firm basis for diffusion mechanisms that adapt to topological changes and retain certain properties, especially scalability, should lead to routing protocols that have far better properties than the current ones.

## 9. REFERENCES

- [1] University of Oregon Route Views Project.  
<http://www.routeviews.org/>.

- [2] BATTISTA, G. D., ERLEBACH, T., HALL, A., PATRIGNANI, M., PIZZONIA, M., AND SCHANK, T. Computing the types of the relationships between autonomous systems. *IEEE/ACM Trans. Netw.* 15, 2 (2007).
- [3] BREMLER-BARR, A., AFEK, Y., AND SCHWARZ, S. Improved BGP convergence via ghost flushing. In *Proc. of IEEE INFOCOM* (2003).
- [4] BREMLER-BARR, A., CHEN, N., KANGASHARJU, J., MOKRYN, O., AND SHAVITT, Y. Bringing order to BGP: decreasing time and message complexity. In *Proc. of ACM PODC* (2007).
- [5] BUSH, R., GRIFFIN, T., AND MAO, Z. Route flap damping: harmful? NANOG presentation, [www.nanog.org/mtg-0210/ppt/flap.pdf](http://www.nanog.org/mtg-0210/ppt/flap.pdf), October 2002.
- [6] CAESAR, M., AND REXFORD, J. BGP Routing Policies in ISP Networks. *IEEE Network Magazine* (2005).
- [7] CHANDRA, R., TRAINA, P., AND LI, T. BGP Communities Attribute. IETF RFC1997, August 1996.
- [8] CHANDRASHEKAR, J., DUAN, Z., ZHANG, Z., AND KRASKY, J. Limiting path exploration in BGP. In *Proc. of IEEE INFOCOM* (2005).
- [9] DESHPANDE, S., AND SIKDAR, B. On the impact of route processing and MRAI timers on BGP convergence times. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)* (December 2004).
- [10] DONNET, B., AND BONAVENTURE, O. On BGP communities. *SIGCOMM Comput. Commun. Rev.* 38, 2 (2008).
- [11] FELDMAN, A., KONG, H., MAENNEL, O., AND TUDOR, A. Measuring BGP pass-through times. In *Proc. of Passive and Active Measurement conference* (2004).
- [12] FRANCOIS, P., AND BONAVENTURE, O. Avoiding transient loops during the convergence of link-state routing protocols. *IEEE/ACM Trans. Netw.* 15, 6 (2007).
- [13] GAO, L. On inferring Autonomous System relationships in the Internet. *IEEE/ACM Trans. Netw.* 9, 6 (2001), 733–745.
- [14] GARCIA-LUNA-ACEVES, J. J. A unified approach to loop-free routing using distance vectors or link states. *SIGCOMM Comput. Commun. Rev.* 19, 4 (1989).
- [15] GONDRAN, M., AND MINOUX, M. *Graphs, Dioids and Semirings: New Models and Algorithms*. Springer-Verlag, 2008.
- [16] GRIFFIN, T., AND PREMOR, B. An experimental analysis of BGP convergence time. In *Proc. of IEEE ICNP* (November 2001).
- [17] GRIFFIN, T., AND SOBRINHO, J. Metarouting. In *Proc. of ACM SIGCOMM* (2005).
- [18] GRIFFIN, T., AND WILFONG, G. An analysis of BGP convergence properties. In *Proc. of ACM SIGCOMM* (September 1999).
- [19] GRIFFIN, T., AND WILFONG, G. On the Correctness of iBGP Configuration. In *Proceedings of SIGCOMM* (August 2002).
- [20] GURNEY, A., AND GRIFFIN, T. Lexicographic products in metarouting. In *Proc. of IEEE ICNP* (2007).
- [21] JAFFE, J., AND MOSS, F. A responsive distributed routing algorithm for computer networks. *IEEE Transactions on Communications* 30, 7 (Jul 1982).
- [22] JAKMA, P. Revised default values for the BGP 'Minimum Route Advertisement Interval'. Internet draft, draft-jakma-mrai-02.txt, work in progress, November 2008.
- [23] LABOVITZ, C., AHUJA, A., BOSE, A., AND JAHANIAN, F. An experimental study of Internet routing convergence. In *Proc. of ACM SIGCOMM* (August 2000).
- [24] LASKOVIC, N., AND TRAJKOVIC, L. BGP with an adaptive minimal route advertisement interval. In *Proc. of IEEE IPCCC* (2006).
- [25] LEE, S., YINZHE, Y., NELAKUDITI, S., ZHANG, Z.-L., AND CHUAH, C. Proactive vs reactive approaches to failure resilient routing. In *Proc. of IEEE INFOCOM* (March 2004).
- [26] MAENNEL, O., TUDOR, A., FELDMANN, A., AND BÜCKLE, S. Observed properties of bgp convergence. RIPE45 presentation, [www.ripe.net/ripe/meetings/ripe-43/presentations/ripe43-routing-flap.pdf](http://www.ripe.net/ripe/meetings/ripe-43/presentations/ripe43-routing-flap.pdf), May 2003.
- [27] MAO, Z., GOVINDAN, R., VARGHESE, G., AND KATZ, R. Route flap damping exacerbates Internet routing convergence. In *ACM SIGCOMM* (2002), pp. 221–233.
- [28] MERLIN, P., AND SEGALL, A. A failsafe distributed routing

- protocol. *IEEE Transactions on Communications* 27, 9 (1979).
- [29] MÜHLBAUER, W., FELDMANN, A., MAENNEL, O., ROUGHAN, M., AND UHLIG, S. Building an AS-Topology Model that Captures Route Diversity. In *Proc. of ACM SIGCOMM* (2006).
  - [30] NELAKUDITI, S., ZHONG, Z., WANG, J., KERALAPURA, R., AND CHUAH, C.-N. Mitigating transient loops through interface-specific forwarding. *Comput. Netw.* 52, 3 (2008), 593–609.
  - [31] PEI, D., AZUMA, M., MASSEY, D., AND ZHANG, L. BGP-RCN: improving BGP convergence through root cause notification. *Computer Networks* 48, 2 (2005), 175–194.
  - [32] PEI, D., ZHAO, X., WANG, L., MASSEY, D., MANKIN, A., WU, S., AND ZHANG, L. Improving BGP convergence through consistency assertions. In *Proc. of IEEE INFOCOM* (2002).
  - [33] RAWAT, A., AND SHAYMAN, M. A. Preventing persistent oscillations and loops in iBGP configuration with route reflection. *Computer Networks* (December 2006), 3642–3665.
  - [34] RAY, S., GUÉRIN, R., AND SOFIA, R. Distributed path computation without transient loops: An intermediate variables approach. In *Proc. of International Teletraffic Congress* (June 2007).
  - [35] REKHTER, Y., AND LI, T. A border gateway protocol 4 (BGP-4). IETF RFC4271, January 2006.
  - [36] SAVAGE, S., COLLINS, A., HOFFMAN, E., SNELL, J., AND ANDERSON, T. The End-to-End Effects of Internet Path Selection. In *Proc. of ACM SIGCOMM* (1999).
  - [37] SUN, W., MAO, Z., AND SHIN, K. Differentiated BGP update processing for improved routing convergence. In *Proc. of IEEE ICNP* (2006).
  - [38] TEIXEIRA, R., MARZULLO, K., SAVAGE, S., AND VOELKER, G. In search of path diversity in ISP networks. In *Proc. ACM IMC* (2003).
  - [39] THE-MRPC-TEAM. Proofs for *Improving Internet-wide routing protocols convergence with MRPC timers*. <http://sites.google.com/site/mrpcproofs/>.
  - [40] VILLAMIZAR, C., CHANDRA, R., AND GOVINDAN, R. BGP Route Flap Damping. IETF RFC2439, November 1998.
  - [41] ZHANG, H., ARORA, A., AND LIU, Z. G-BGP: Stable and fast convergence in the Border Gateway Protocol. Tech. rep., June 2003. <http://www.cse.ohiostate.edu/~zhangho/publications/G-BGP-TR.pdf>.