Optimizing MRAI on Large Scale BGP Networks: An Emulation-Based Approach

Mattia Milani[a,b], Michele Segata[c], Luca Baldesi[d], Marco Nesler[e], Renato Lo Cigno[f], Leonardo Maccari[g]

[a]*Standards, Nokia S&T, München, Germany*
[b]*Department of Telematic Engineering, Universidad Carlos III Madrid, Leganés, Spain*
[c]*University of Trento and CNIT, Trento, Italy*
[d]*University of California, Irvine, USA*
[e]*NTS Italy, Bolzano, Italy*
[f]*University of Brescia and CNIT, Brescia, Italy*
[g]*University of Venice, Venice, Italy*

Abstract

Modifying protocols that pertain to global Internet control is extremely challenging, because experimentation is almost impossible and both analytic and simulation models are not detailed and accurate enough to guarantee that changes will not affect negatively the Internet. Federated testbeds like the ones offered by the Fed4FIRE+ project offer a different solution: off-line Internet-scale experiments with thousands of Autonomous Systems (ASes). This work exploits Fed4FIRE+ for a large-scale experimental analysis of Border Gateway Protocol (BGP) convergence time under different hypotheses of Minimum Route Advertisement Interval (MRAI) setting, including an original proposal to improve its management by dynamically setting MRAI based on the topological position of the ASes in relation to the specific route being advertised with the `UPDATE` messages. MRAI is a timer that regulates the frequency of successive `UPDATE` messages sent by a BGP router to a specific peer for a given destination. Its large default value significantly slows down convergence after path changes, but its uncoordinated reduction can trigger storms of `UPDATE` messages, and set off unstable behaviors known as route flapping. The work is based on standard-compliant modifications of the BIRD BGP daemon and shows the tradeoffs between convergence time and signaling overhead with different management techniques.

Keywords: Internet Global Routing, BGP Convergence, Route Flapping, Testbed Experiments, Network Emulation, Topology Generation, MRAI Setting

1. Introduction

The Border Gateway Protocol (BGP) is the only protocol available to implement the routing function at the Autonomous System (AS) scale. The border routers of ASes use BGP to export their prefixes and to propagate other prefixes, and its path-vector approach allows the Internet to function in a stable way. While BGP evolved in some of its features, (notably, the security aspects) the way it reaches convergence is essentially the same as in its original design despite its known limitations. The reason for this *ossification* is twofold, the first is that operators are very cautious when considering changes to this component that is critical for the whole Internet, the second is that we cannot experiment at scale with BGP, and so it is hard to test the effects of new proposals. As a result, BGP evolves slowly.

One explanatory example is the notorious slow convergence of BGP, due to the presence of the Minimum Route Advertisement Interval (MRAI) timer. When there is a change in the Internet topology, BGP will propagate the information on the new topology using `UPDATE` messages. Without MRAI this may generate a storm of `UPDATE` messages in a very short time, each triggering the recomputation of routing tables and ultimately clogging the routers. MRAI prevents this flood by allowing the propagation of only one `UPDATE` message for every MRAI interval, with the obvious side-effect of slowing down the convergence.

For this reason the value of MRAI has been a

DOI: https://doi.org/10.1016/j.comcom.2024.107940 accepted to Computer Communications *October 29, 2024*

subject of discussion in the literature. As we show in the review of the state of the art (Section 2) its default value was initially set to 30 s and then, in absence of an agreement, it was left to the decision of the network operator[1]. Finding an optimal value for MRAI would have a huge positive impact on the performance of BGP; however, Fabrikant, Rexford et al. [2] showed that modifying MRAI in an uncoordinated way, can lead in specific cases to the exponential growth of the number of `UPDATE` messages required to achieve a new stability point after a route change, which is the opposite of what MRAI should achieve. We detail this problem (Section 3) to show that more than ten years later we are still in this deadlock: we know that MRAI needs to be optimized, but we also know that changing it without coordination may make things worse, and of course, there is no way to coordinate all the network administrators of all ASes to take a certain decision. This example shows the inherent hardness of researching on BGP, it is way too complex to attempt an overall theoretical modeling, simulation approaches are limited because they capture only a part of the complexity of the system, and emulation approaches, so far, could not reach the necessary scale to validate proposals.

Our work has the ambition to show that it is possible to experiment with a real implementation of BGP, namely BGP Internet Routing Daemon (BIRD)[2], reproducing with a real implementation and measures the results from Fabrikant et al. [2]. We also propose a strategy that allows the improvement of the convergence of BGP without hampering its stability (Section 4), and we evaluate it on emulated networks made of tens of thousands of BGP routers using a fully reproducible approach (Section 5). This paper extends and completes a previous work by the same authors [3] using data we could obtain in the frame of the Fed4FIRE+ project, which supported our work. In short, the three contributions of our paper are the following ones.

Confirm Fabrikant Results. We use our emulation framework to confirm Fabrikant's results, which is key to understand the risks associated to dynamically setting MRAI. We reproduce and confirm Fabrikant observation in Section 6.1;

Propose and Test a dynamic MRAI configuration strategy. We provide the initial design of a strategy to dynamically set MRAI. We show experimentally that our approach improves the performance of the standard BGP configuration in terms of convergence time, with a limited penalty in the number of generated `UPDATE` messages (Section 6.2). We achieve such results applying a custom propagation timer that can be computed by each node cooperatively with the others, without requiring global coordination. We compare this solution with the standard 30s MRAI timer and, when scale allows, with BGP without MRAI.

Provide a scalable and repeatable BGP experimentation framework. This paper uses an experimental approach that enables the emulation of networks made of tens of thousands of routers to compare different approaches on real code. In our previous work [3] we already provided the details to reproduce our results. In this work we increase the size of the emulated networks by a factor of 3, and we add the description to the two open source components and one protocol extension we developed to make our experiments possible (Section 7), namely:

- We provide the implementation of MRAI in the open source BIRD daemon. The source code is available in the open-sourced project repository and it will be forwarded to the BIRD community for consideration.
- We present the implementation of an Internet-like topology generator that creates graphs that respect the characteristics of the Internet, including commercial relationships between ASes. Our code is realized in the Python language and has become an integral part of the well known NetworkX Python library.
- Since our proposal includes a modification to the BGP protocol using a custom extension, we also document its design and show that it can be deployed incrementally on the Internet, which is a key requirement for a realistic solution.

Overall, our work showcases how the availability of a large scale testbed not only leads to viable proposals for long-standing research issues, but it also enables the development of open source code that ultimately benefits the community of the researchers and practitioners.

[1] As a relevant example, Cisco suggests 30 s MRAI for any regular Exterior BGP (eBGP) peer [1].

[2] BIRD is one of the most used and well maintained BGP open source implementations, used in many real world scenarios see `https://bird.nic.cz/en/case-studies/`

2. State of the Art

Albeit their extreme importance, global routing and BGP have never been a ‘hot topic’, often perceived more as a management issue relevant only for operators, even if the slow convergence of BGP is a problem that may affect the entire Internet, and consequently the ‘digital life’ of people. Clearly there exist several papers dealing with BGP and its characteristics and features (see [4] and citations therein or [5] to name a few), but the core itself of BGP, i.e., the Path Vector descriptors exchanged by eBGP routers, and the policy-based routing algorithms, do not lend themselves to rigorous theoretical analysis and modeling, thus preventing elegant and sound theoretical results similar to those available for link state and distance vector routing protocols. Indeed, the seminal works by Labovitz et al. at the beginning of this century [6, 7] clearly identified BGP as a major source of Internet failures and disruptions, and also discussed some possible modifications to BGP to improve it. In the following years other works [8, 9] measured and quantified the phenomenon in the Internet, proving that it was real and also potentially very penalizing. After more than 20 years, however, most of the issues are still there.

BGP research has traditionally been based on simulations [10, 11, 12] and/or small-scale testbeds [13, 14], with a few works, like the already cited paper by Fabrikant et al. [2], addressing specific issues with a theoretical analysis based on heuristics considerations on BGP dynamics. Other significant works on the subject are based on measures, like the analysis in [15], that clearly identifies the key role of BGP in properties and performance of the global Internet, or [16, 17] analyzing prefix scaling and router ownership boundaries, but they normally document the behavior or the properties and consequences of BGP rather than exploring possible improvements. The PEERING testbed [18] offers an infrastructure to experiment with BGP, but does not seem prone to explore performance or major protocol modifications. A work published at the time of writing this paper proposes BGPEval [19] that uses an approach similar to ours. The main difference is that we don’t make any assumption on the underlying hardware or software architecture. As better described in Section 5.1, we rely on kernel name spaces on the GNU/Linux OS and we do not assume that a single hypervisor is under the control of the experimenter.

One of the prominent themes in this body of literature is the trade-off between the generated overhead and the convergence speed of BGP after a reconfiguration event. BGP is known to be subject to *path exploration*, a transitory phenomenon that happens when a router adopts and publishes a sequence of non-optimal paths for a destination before reaching a stable state. Path exploration can generate thousands of update messages in networks made of as few as tens of nodes [20]. In order to reduce the message overhead, BGP uses MRAI, the minimum time between two consecutive `UPDATE` for the same destination sent to the same neighbor, which is set by default to 30 s [21]. MRAI reduces the overhead but strongly impacts BGP convergence [22] and its default value was often discussed [23].

Difficulties and a negative result, however, cannot be a reason to stop seeking solutions. As noted in [24], which studies the effect of disabling the use of MRAI in part of the ASes of the Internet, in the past 10 years the computing capability of routers increased 8-fold, giving routers the possibility of improve processing, but MRAI default timer has not been changed. The authors of this work notice that it is possible to disable MRAI in nearly 90% of ASes without incurring in instabilities, thus improving the convergence speed of BGP at the cost of additional processing of routing messages (they must be processed faster and in bursts), but without a large increases in the number of messages exchanged. Also this study is based on simulations.

Some older studies like [25, 26] suggested that MRAI could be disabled altogether, but later works have clearly shown that this may lead to instabilities. The authors of [27], for instance, acknowledge that fiddling with MRAI can cause routing message bursts and propose to disperse routing messages in case of bursts to distribute the management load among different routers and ASes. Also [28] propose heuristics to adapt MRAI dynamically and shows via simulations, as the previous ones, that convergence can be improved while keeping the number or messages similar to BGP with constant 30 s MRAI.

Several other works including [29, 30, 31] make various proposals to improve BGP performance, reduce convergence times, or reduce route flapping.

Finally, the authors of [32] attempted a general modeling of Path Vector routing protocols analyzing what are the key parameters and the influence of network delays on the convergence speed and properties. This study confirms all the hints and indications of other works.

Symbol	Meaning
$\mathcal{V}$	set of all the nodes of the BGP graph, i.e., ASes
$\mathcal{E}$	set of the logical links between BGP nodes
$G(\mathcal{V}, \mathcal{E})$	BGP network graph
$e_{i,j}$	edge, i.e., logical link between nodes i and j
d, D	destinations (prefixes), set of all destinations
$\mathcal{D}$	set of ASes that export at least one destination d
d_r	set of ASes that, generating an `UPDATE`, cause maximal route reconfiguration
$\mathcal{N}_i$	set of node i neighbors or *peers*
$\tau(j, d)$	value of MRAI used by a node for destination d and neighbor j
$G_A(\mathcal{V}^{\mathcal{G}_\mathcal{A}}, \mathcal{E}^{\mathcal{G}_\mathcal{A}})$	the 'ascending' portion of $G(\mathcal{V}, \mathcal{E})$
$\mathcal{V}^{\mathcal{G}_\mathcal{A}}$	set of nodes in the 'ascending' portion of $G(\mathcal{V}, \mathcal{E})$
$\mathcal{E}^{\mathcal{G}_\mathcal{A}}$	set of edges in the 'ascending' portion of $G(\mathcal{V}, \mathcal{E})$
$G_T(\mathcal{V}^{\mathcal{G}_\mathcal{T}}, \mathcal{E}^{\mathcal{G}_\mathcal{T}})$	the fully connected sub-graph of Tier-1 ASes in $G(\mathcal{V}, \mathcal{E})$
$G_D(\mathcal{V}^{\mathcal{G}_\mathcal{D}}, \mathcal{E}^{\mathcal{G}_\mathcal{D}})$	the 'descending' portion of $G(\mathcal{V}, \mathcal{E})$
$\Delta(v)$	normalized Destination Partial Centrality (DPC) of node v

Table 1: Main symbols and notation used in the paper.

3. The Problem of MRAI Configuration

This section summarises the BGP protocol and provides an elementary description to understand the problem of signaling storm addressed by MRAI. For an in-dept description the reader can refer to the work by Fabrikant et al. [2].

We model a BGP network as a connected, directed graph $G(\mathcal{V}, \mathcal{E})$; $i \in \mathcal{V}$ are BGP instances (also routers for simplicity – see Table 1 for the notation used in the paper). The edges $e \in \mathcal{E}$ are the BGP relationships between ASes, thus the logical links between BGP instances. An edge $e_{i,j}$ can be, for instance, a free peering relation or a customer-provider one. Regardless the commercial relation, if $e_{i,j}$ exists, then i and j are called *peers* and N_i is the set of BGP peers of router i.

Routers export network prefixes to other routers with `UPDATE` messages and cancel them with `WITHDRAWAL` messages. Prefixes populate the Routing Information Base (RIB) of BGP instances and are properly propagated to all routers in the ASes in order to deliver the packets with a certain destination IP, and this justify the simplification of collapsing an entire AS into a single node i in $G(\mathcal{V}, \mathcal{E})$. As a destination IP address is always mapped onto a prefix, we use the terms prefix and destination interchangeably. Every `UPDATE` contains information about one or more destinations. Considering a set D that represents the prefixes that can be advertised on the Internet, then the `UPDATE` describes a route r as the combination of destination and attributes: $r = (\xi, d)$, where the last element of this tuple is a destination network $d \in D$, while ξ represents the attribute list: the path (a sequence of ASes), and a unique ID of the originator router that we skip in our notation for the sake of simplicity.

A router i uses a function $\Gamma(r)$ that outputs a score for the route received by peer j based on a local policy. Given two routes r_1 and r_2, r_1 is preferable if $\Gamma(r_1) > \Gamma(r_2)$ and r_2 otherwise. When $\Gamma(r_1) = \Gamma(r_2)$ BGP provides a tie-breaking procedure described in [21]. The simplest form of policy, that we use in our emulation, takes into consideration the commercial relationships between i and j and the (possibly weighted) length of the AS path indicated in the attributes ξ.

When a router updates its routing table, it also schedules a new `UPDATE` message to propagate the information. However, when the Internet topology changes a router will receive `UPDATE` messages through multiple paths, and may update its routing table several times before it converges to the route r with the highest score. In principle, every modification may generate a new `UPDATE` possibly containing transient information that will be superseded by the next `UPDATE`. Such a storm of `UPDATE` messages may overload the processing capability of the routers.

BGP takles this problem with MRAI, that limits the rate of outgoing `UPDATE` messages from a router. MRAI is generally set to 30 seconds (plus a small random value to avoid synchronizations), as described in the RFC [21] and can be implemented per peer j or per peer j and per destination d. We consider the second, more fine-grained approach, which is also the suggested solution in the RFC: a router i uses a separate timer for each peer $j \in \mathcal{N}_i$ and for each destination d, initialized to $\tau(j, d)$. Consider the case in which router i receives an `UPDATE` including a route r for destination d. If the new route changes the routing table of i and triggers the generation of a new UPDATE towards one of its peers j, the UPDATE is sent immediately, and the MRAI timer is started. During this period UPDATE messages from any peer k may change again the routing table of i and may trigger the generation of another UPDATE towards j, however, the updates are not sent. When the MRAI timer fires, if the routing table was affected with respect to the last update sent to j, the node advertises the new best option. The routes that have been received but not chosen as the best ones, are kept in a queue and can be used, for example, as backups in case of changes in the network. Since UPDATE messages generally arrive in bursts, MRAI does not affect the generation of the first UPDATE, but slows down the generation of the following ones, avoiding UPDATE flooding and possible route flapping. The larger is MRAI the less is the number of UPDATE generated, but also the slower is the convergence of the entire network.

3.1. Exponential Path Exploration

The management of the MRAI timer does not come without controversies. In 2008, thanks to different studies that take into consideration the dimension of the topology and the latency [33], there has been an Internet Draft proposal to reduce its default value to 5 s [34]. In 2011, a follow-up Internet Draft [23] proposed to let operators choose an arbitrary MRAI value for UPDATE messages, while on the other hand WITHDRAWAL could completely ignore it. None of these Drafts was approved in the end.

The idea that each operator can decide how responsive to changes its system is was appealing, but the study presented by Fabrikant et al. showed also how this can be harmful for the overall network [2]: this study presents a theoretical result where, with particular configurations of MRAI timers and realistic topologies, there is an exponential path exploration behavior. This translates into very long convergence time, but most of all, many messages wasting the computational power of routers with a risk of collapse.

Figure 1 shows an example of such particular graphs. This issue happens when there are multiple paths leading to the destination and a specific sequence of timers $\tau(j, d)$ in the propagation path.

Let's assume BGP has converged, so that all MRAI timers are off and all the nodes can freely propagate messages. X_0 is the entry point for the destination d that is to be announced through the network. Each node assigns value to a path reading the labels on the edges going from left to right and interpreting them as a binary number. The higher the value of a path, the higher the preference. For example, node X_0 prefers the edge with label 1, and X_3 prefers the path X_3, X_2, X_1, X_0, d over the path $X_3, X_2, Y_2, X_1, X_0, d$ because the first one would be assigned a value of $1111_b = 15_d$, while the second $1101_b = 13_d$. At a certain instant t_0 the node X_0 changes the route towards d selecting the edge with label 0 (for instance because the other edge is broken) and propagates such UPDATE to both X_1 and Y_1 at the same time. At this point in time node X_1 knows the new path through X_0 and the old one passing through Y_1. This old path is not available anymore, but X_1 is not aware of it, so it decides to reach the destination d using the outdated path through its neighbor $Y1$, as that path has a higher score (10_b) than the newly received one going directly through X_0 (01_b). X_1 will then propagate at time t_1 the change to its neighbors with an UPDATE and initialize two MRAI timers to $\tau(X_2, d)$ and $\tau(Y_2, d)$. Roughly at the same time also Y_1 receives the UPDATE from X_0, updates its table and distributes its UPDATE to X_1. This operation introduces a small delay ϵ, so X_1 receives this advertisement at time $t_1 + \epsilon < t_1 + \tau(X_2, d)$ so X_1 cannot further update X_2 because the MRAI timers was just activated (and the same happens with Y_2).

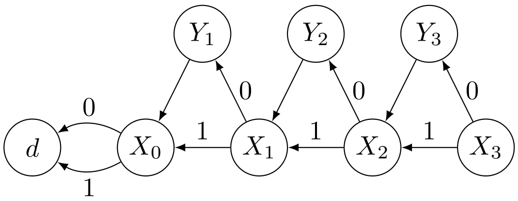


Figure 1: Gadget topology derived from [2] with 3 rings

This behavior makes X_1 send two UPDATE for the same destination d, the first of which was computed on outdated information. Worse than that, the correct UPDATE cannot be propagated to the neighbors before MRAI expires.

The same behavior could happen in the next ring of the chain. The worst case happens if the timers $\tau(X_3, d), \tau(Y_3, d)$ of X_2 are lower than $\tau(X_2, d)$ of X_1: They will fire before X_1 propagates the correct update, so X_2 is going to propagate 2 outdated UPDATE messaged for every message it receives from X_1. So the loophole would repeat twice at this level and would exponentially explode further down the chain.

This is a gadget topology identified by Fabrikant et al. to showcase the effect, but the original paper describes other realistic situations in which the presence of multi-paths and a decreasing value of MRAI on the path produces an exponential number of `UPDATE` messages and route oscillation.

4. Centrality-based MRAI Configuration

A simple solution to this problem would be to always use an increasing MRAI compared to the previous hop. This solution has two drawbacks: first, routers would need to implement some form of dynamic coordination per each propagation path; second, increasing MRAI will slow down convergence.

A better policy would be to have MRAI increase in the initial phase of the propagation, close to the AS k that generate the first `UPDATE` for route r. Then, when the routers around k stabilize, the new stable topology can be quickly propagated to the rest of the Internet. To verify the validity of this intuition we set-up a strategy that exploits the knowledge of the network graph together with the concept of Destination Partial Centrality (DPC).

4.1. DPC - Destination Partial Centrality

DPC is a variant of the so-called load centrality which is defined in its general form as follows [35]. Consider a graph $G(\mathcal{V}, \mathcal{E})$ and an algorithm to identify the (potentially multiple) minimum weight path(s) between any pair of vertices (i, j). Let $\theta_{i,j}$ be a quantity of a generic commodity that is sent from vertex i to vertex j. We assume the commodity is always passed to the next hop following the minimum weight paths, but any routing metric is valid. In case of multiple next hops, the commodity is divided equally among them. We call $\theta_{i,j}(v)$ the amount of commodity forwarded by vertex $v \in \mathcal{V}$ with respect to the vertices i and j. The *load centrality* of v is then given by:

$$LC(v) = \sum_{i,j \in \mathcal{V}} \theta_{i,j}(v) \qquad (1)$$

DPC adapts load centrality to represent the propagation of routes in an IP network. In DPC the *commodity* corresponds to the number of networks that a BGP node exports, so only nodes that are directly connected to destinations generate the *commodity*. We call $\mathcal{D} \subseteq \mathcal{V}$ the set of nodes that export at least one prefix, and M_i, M_j the number of networks that are exported by node i and j, respectively, then $\theta_{i,j} = \frac{M_i + M_j}{2}$. Considering a router v, $\theta_{i,j}(v) = \theta_{i,j}$ if v is in the path from i to j or zero otherwise. In all our experiments we assign one destination per node, so that $\theta_{i,j}$ is always unitary but this is an arbitrary choice that can be replaced with any other suitable one. Considering only the case when nodes in $\mathcal{D}$ export only one prefix, the normalized value of DPC of any vertex $v \in \mathcal{V}$ takes the following form:

$$\Delta(v) = \frac{1}{|\mathcal{D}| \times (|\mathcal{D}| - 1)} \sum_{i,j \in \mathcal{D}} \theta_{i,j}(v) \qquad (2)$$

DPC does not express load in terms of traffic, but it captures the impact of an AS in terms of routing updates it may generate. If one AS exports many prefixes and it changes its local topology (i.e., adds or removes a peering edge) this information needs to be propagated on the Internet to all routers, regardless of the fact the networks actually generate or receive traffic. DPC also models the fact that some ASes do not export network addresses so they do not generate *load*, but still their centrality can be larger than zero. In a previous work we have shown that load centrality can be computed in a distributed way with minimal modifications to a Distance-Vector routing protocol [36]. Section 7.3 describes how DPC can be computed in a distributed manner with an extension to BGP. Our solution can be incrementally deployed on the Internet without requiring any global coordination. Further theoretical details are outside of the scope of this paper, but principles of centrality-based routing can be found in [36, 37, 38].

4.2. Tuning MRAI with DPC

Our proposal configures MRAI as a function of DPC with the following model. We observe that in general, given a route change for destination d, the relative `UPDATE` message propagates in the network in three separated propagation graphs that we formally define and comment in Section 4.3:

- ***Ascending graph*** $G_A(\mathcal{V}^{\mathcal{G_A}}, \mathcal{E}^{\mathcal{G_A}})$;
- ***Tier one graph*** $G_T(\mathcal{V}^{\mathcal{G_T}}, \mathcal{E}^{\mathcal{G_T}})$;
- ***Descending graph*** $G_D(\mathcal{V}^{\mathcal{G_D}}, \mathcal{E}^{\mathcal{G_D}})$.

The intuition we follow is that there is an *horizon effect*, meaning that a router j close to the node k that triggers the update has higher chances of modifying its routing choice than a router i that is far away from k. This is intuitive if we look at Fig. 1 in which X_0 has a choice to make between one path and another, but once this choice is made, all the other routers should just keep using the same next hop they used before the change. Moreover, the BGP network is hierarchical, we know that there are ASes of strategic importance (so-called Tier-1 ASes) that represent a narrow waist of the topology. Once the `UPDATE` reaches a Tier-1 AS it is less likely it will trigger any reconfiguration anymore, so updates can propagate at a lower pace without hampering global routing; however, `UPDATE` must travel to all BGP routers, because if the path changes, every router needs to update its knowledge of the path, so the propagation cannot be stopped.

It must be noted that only G_T is the same for all destinations, while G_A and G_D depends on the destination d. *Ascending* and *Descending* refer to position of nodes in relation to d and G_T: G_A includes all nodes whose path toward d does not include any Tier-1 AS and G_D all the others. In presence of peering relations it is possible that all nodes in a path are in G_A.

The MRAI timer is set following Eq. (3). Considering a graph-wide maximum timer $T = 30\,\text{s}$ and DPC $\Delta(i) \in [0, 1]$ for node i, DPC-based MRAI $\tau(j, d)$ used by node i with neighbor j for destination d is set as follows:

$$\tau(j,d) = \begin{cases} \frac{T}{2}\Delta(i) & \forall i \in \mathcal{V}^{\mathcal{G}_{\mathcal{A}}} \\ \frac{T}{2} & \forall i \in \mathcal{V}^{\mathcal{G}_{\mathcal{T}}} \\ \frac{T\cdot(1-\Delta(i))}{2} + \frac{T}{2} & \forall i \in \mathcal{V}^{\mathcal{G}_{\mathcal{D}}} \end{cases} \qquad (3)$$

In practice, we increase MRAI in the neighborhood of the node triggering the update to avoid the effect identified by Fabrikant, following the natural increase of DPC as nodes are more central in the global topology. If paths are all contained in G_A, for some of the nodes farther away from d $\tau(j, d)$ will decrease, but it is very unlikely that this pattern triggers route flapping[3]. Instead, after the `UPDATE` propagation reach and cross G_T, MRAI increases as DPC decreases, because the `UPDATE` is moving toward less central nodes, where it is less and less likely to trigger any route change, as paths always follow customer-provider relations all leading to the same Tier-1 AS.

Centrality is used to capture the position in the BGP graph that an AS occupies. In this work we pre-compute the propagation graphs and the DPC in advance, in order to verify that our intuition is correct, however Section 7.3 explains how to design a BGP extension to compute DPC on-line. Together with the theory provided in [36, 37, 38] this paves the way for a distributed and incrementally deployable solution.

4.3. Propagating `UPDATE` *messages on the BGP graph*

Following a canonical model, the links in the BGP graph $G(\mathcal{V}, \mathcal{E})$ can be of two kinds, either peer-to-peer or customer-provider depending on commercial agreements. We indicate with $\Lambda = \{\pi, c, s\}$ the edge labels indicating peer, customer, or provider relationship on a directed edge. The function $\lambda : \mathcal{E} \to \Lambda$ assigns to the edge (i, j) the role i has with respect to j. Hence, $\lambda(i, j) = \pi \iff \lambda(j, i) = \pi$, while $\lambda(i, j) = c \iff \lambda(j, i) = s$.

We call $C_i = \{j \in \mathcal{N}_i : \lambda(i, j) = s\}$ the set of customers of node i, and define a Tier-1 AS as one AS that exports no prefixes, has no providers, and is in a peering relationship with all the other Tier-1 ASes. We indicate as $G_T(\mathcal{V}^{\mathcal{G}_{\mathcal{T}}}, \mathcal{E}^{\mathcal{G}_{\mathcal{T}}})$ the fully connected sub graph of Tier-1 BGP nodes, such that $\mathcal{V}^{\mathcal{G}_{\mathcal{T}}} \subset \mathcal{V}$, $\mathcal{E}^{\mathcal{G}_{\mathcal{T}}} = \mathcal{V}^{\mathcal{G}_{\mathcal{T}}} \times \mathcal{V}^{\mathcal{G}_{\mathcal{T}}}$. The propagation of a route follows the *no-valley* and *prefer-customer* standard assumption, depicted in Fig. 2 that we reproduce from Elmokashfi et al. [39], which means that routes learned from customers are announced to all neighbors, while routes learned from peers or providers are announced only to customers.

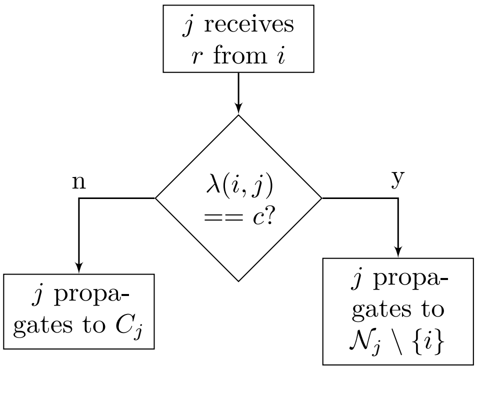

j receives r from i
$\lambda(i,j) == c$?
n
y
j propagates to C_j
j propagates to $\mathcal{N}_j \setminus \{i\}$

Figure 2: Flow chart for `UPDATE` forwarding.

When a node $i \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{G}_{\mathcal{T}}}$ generates an `UPDATE` containing a route r, it is propagated to all nodes using the links of $G(\mathcal{V}, \mathcal{E})$ with a very specific pattern

[3] A path is all within G_A only if there are some peering relations on it.

of route propagation from i to the rest of the network (we assume that the majority of the modifications to the BGP graph come from nodes outside of $\mathcal{V}^{\mathcal{G_T}}$). We model this pattern by sub-dividing the graph G in three components. The subdivision of G depends on the route r, so it should not be seen as a static partition of G, rather as a dynamic sectioning that empowers the proper allocation of MRAI. In the first component nodes i either forward the update to a provider node, or send it to a peer or customer node on paths that include a peering relation. We call $G_A(\mathcal{V}^{\mathcal{G_A}}, \mathcal{E}^{\mathcal{G_A}})$ the graph made of all nodes and edges that are involved in this phase of UPDATE propagation.

More formally let $i \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{G_T}}$ be the originator of the first UPDATE and $j \in \mathcal{V}$ another node in G. Then $j \in G_A(\mathcal{V}^{\mathcal{G_A}}, \mathcal{E}^{\mathcal{G_A}}) \iff \exists\, p_{ij}$ a path in G between i and j, with $p_{ij} = (i, k_0) \dots (k_l, j)$ such that:

1. $k_x \notin \mathcal{V}^{\mathcal{G_T}},\ \forall x = 0, \dots, l$; and
2. any of the following conditions holds:
   (a) $\lambda(u,t) = c,\ \forall (u,t) \in p_{ij}$;
   (b) $\exists z < l : \lambda(u,t) = c,$
   $\forall (u,t) \in p_{ik_z}, \lambda(k_z, k_{z+1}) = \pi,$
   $\lambda(u,t) = s,\ \forall (u,t) \in p_{k_{z+1},j}$.

Condition 1 simply indicates that no nodes in the path is Tier-1. Condition 2(a) describes the case in which the path towards a Tier-1 AS is made of only customer-provider relationships. Condition 2(b) represents the case in which the chain of customer-provider edges is interrupted by a peering edge and followed by a chain made of provider-customer edges. If a route is propagated by a Tier-2 AS to its Tier-3 customers without ever involving a Tier-1 provider, then all the nodes involved belong to the ascending graph.

Routes that are not completely contained in G_A propagate in $G_T(\mathcal{V}^{\mathcal{G_T}}, \mathcal{E}^{\mathcal{G_T}})$, and then in G_D, the graph connecting the nodes in $\mathcal{V}^{\mathcal{G_T}}$ to all the nodes in $\mathcal{V}^{\mathcal{G_D}} = \mathcal{V} \setminus \mathcal{V}^{\mathcal{G_T}} \setminus \mathcal{V}^{\mathcal{G_A}}$. More formally $j \in \mathcal{V}^{\mathcal{G_D}} \iff \exists p_{ij}$ a path in G between $i \in \mathcal{V}^{\mathcal{G_T}}$ and j, with $p_{ij} = (i, k_0) \dots (k_l, j)$ such that:

1. $j, k_x \notin \mathcal{V}^{\mathcal{G_T}} \cup \mathcal{V}^{\mathcal{G_A}},\ \forall x = 0, \dots, l$; and
2. $\lambda(u,t) = s,\ \forall (u,t) \in p_{ij}$.

This simply indicates that the UPDATE propagates along provider-customer relationships.

$\mathcal{V}^{\mathcal{G_T}}$ is fixed and known, and this is a realistic assumption considering that Tier-1 ASes are a few well known ones, while the ascending and descending graphs depend on the originator node; however, a certain node j can ascertain to which graph it belongs just by looking at the AS path specified in the route r, making the implementation of the proposal easy and local, at least once the DPC is known to all nodes.

5. Scenarios

The results we present in Section 6 are obtained in two different scenarios: one for the Fabrikant-gadgets as shown in Fig. 1, and the second one for *Internet-like* topologies that we generate following Elmokashfi et al [39]. Figure 7 shows an example of an Internet-like topology and Section 7.1 describes the generator we implemented.

As suggested by the authors of the original paper, to obtain the route flapping behavior in the chain-gadget topologies we halve the initial value of the MRAI timer at each X_i from left to right (Fig. 1). More formally, the initialization is $\tau(X_{i+1}, d) = \frac{\tau(X_i, d)}{2}$, while the initial timer values used by X_0 is set to 30 s, then timers are decremented independently. Each Y_i node uses the same timer of X_{i-1} to propagate routes towards X_i. For these scenario we consider an increasing number of rings, starting from 2, with 5 nodes, up to 8 with 17 nodes, whereas for the Internet-like topologies we consider here topologies with 12 000 ASes. We use a modified implementation of BIRD on the Fed4FIRE+ testbed to run the experiments. For the evaluation, we exploit a BGP emulator we develop that is openly available online and that we presented in [3]. In Section 5.1 we briefly describe the working principle of the emulator, together with the pointer to the resources that the interested reader can refer to for further information, in particular how to reproduce the results.

In the Internet-like scenario we change the route of the destination d_r, selecting the attachment node from the set of nodes that induce the worst case situation, i.e., the change causes a reconfiguration of the largest possible number of nodes. In both topologies, in order to trigger the change in the network we use a well known technique called *AS_-PATH Prepending*, whereby an arbitrary number of entries in the AS_PATH list is added forcing the generation of a new BGP UPDATE. For the Fabrikant gadget we repeat each experiment 10 times, while for the Internet-like topologies we repeat 10 experiments choosing different d_r to get more variability. In all cases the initial value of MRAI is subject to a random relative jitter $\rho = 0.05$. Every time the timer is reset, a random value x is drawn from the distribution $U[1.0 - \rho, 1.0]$ and MRAI is set to the default value assigned to the node and route multiplied by x.

To speed up the initial convergence of the network we configure only one destination from any node in $\mathcal{D}$, consistently with the definition in Eq. (2). In all experiments we start the network emulation, then wait for the routing tables of all nodes to converge, and finally trigger the change in d_r and measure the effects of the change till convergence. For the performance evaluation, we consider convergence time and the number of `UPDATE` messages generated. We repeat each experiment with the following MRAI strategies:

30 s fixed: the default value according to BGP RFC [21];

No MRAI : `UPDATE` messages are sent immediately without delay, MRAI deactivated.

Fabrikant style: MRAI will be set on each node according to the policies described in Fabrikant work, to reproduce the worst case scenario;

DPC-based: following the centrality computed offline and the formula in Eq. (3).

it is possible to test other values of MRAI and verify if, even with static settings there exists a better trade-off between overhead and convergence time. However, even in case we find a value that performs better than the current one, it is extremely improbable that operators will converge to a different default value. Furthermore, any fixed value we heuristically test is bound to the tested topology, and can become unsuitable when technology evolves or the Internet topology changes its characteristics.

5.1. BGP Emulation Toolchain

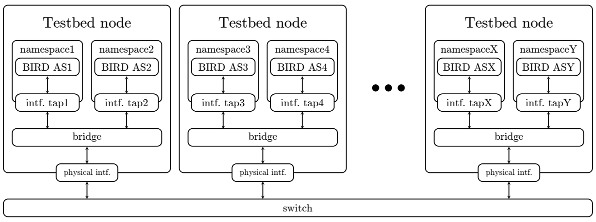


Figure 3: BGP emulation toolchain architecture [3].

Within the *Internet on FIRE* project we developed an emulation toolchain capable of reproducing BGP networks. The main goal of the toolchain is to enable the analysis of BGP improvements in complex, Internet-like networks, as testing new features on the "real world Internet" can be extremely difficult, beside leading to catastrophic consequences in case of mistakes.

The toolchain is developed targeting Fed4FIRE+ federated testbeds, so we refer to that framework for the reservation of resources, however the rest of the methodology is general enough to enable running it on different testbeds. Figure 3 shows the toolchain architecture. In particular, it assumes the presence of computing nodes which are connected via wired interfaces. The toolchain reads the topology from a GraphML file and reserves a certain number of testbed nodes depending on the number of ASes. On each physical node we run multiple instances of the BIRD BGP daemon, and each BIRD instance represents one AS. The number of ASes per node depends on its capabilities. More specifically, the tool tries to equally distribute all the BGP deamons among the reserved physical CPUs, we suggest to remain below a ratio of 8 : 1 BIRD processes for each CPU. This process of resource gathering has been automated through convenience scripts.

Each BIRD daemon runs in an isolated Linux kernel namespace, communicating with the outside through a virtual interface. In turn, each virtual interface is bridged with the physical interface of the node, enabling each BIRD daemon to potentially communicate with any other one. Depending on the topology, the toolchain configures a /30 IPv4 network for each pair of connected ASes, and then set up the BGP TCP session.

The toolchain has, in principle, no limits on scalability, and we tested topologies up to 20 000 ASes. The only limit is the size of the testbed and the time necessary to setup the experiment. Indeed, the toolchain automatically installs the necessary software, configures the nodes, deploys the AS instances, etc. These steps need to be repeated for each experiment, as nodes reserved in Fed4FIRE+ testbeds are "vanilla" Linux systems. In fact, booting and configuring the nodes might take several hours, more than what is necessary to run the experiment.

The interested reader can get more detailed information in the original paper [3], as well as on the resources that are available on the project website[4] and on GitHub[5]. Available resources include a de-

[4] `https://iof.disi.unitn.it`

[5] `https://github.com/internetonfire/`

tailed set of instructions enabling users to reproduce our emulations, as well as a virtual machine that comes with the necessary software pre-installed for simple tests and experiments.

6. Numerical Results

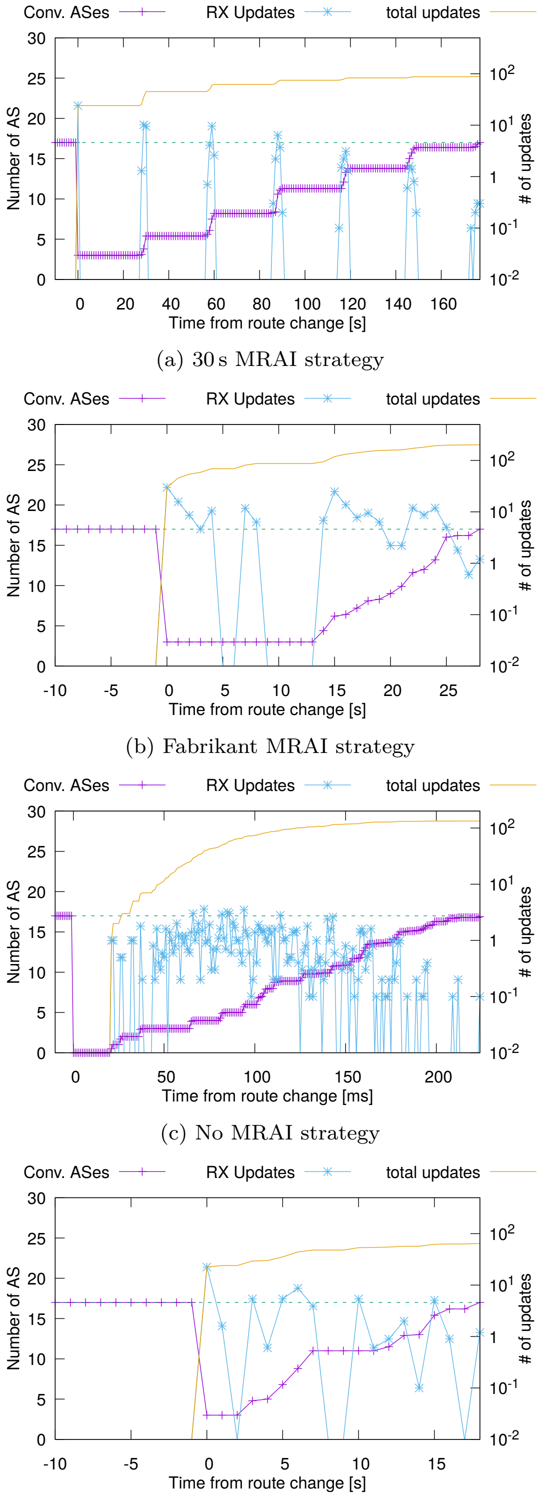
6.1. Results on Chain Gadgets

	Strategy	10^{th}	mean	90^{th}
updates	30 s	81.6	88.5	95.2
	Fabrikant	157.1	201.3	220.8
	No MRAI	131.3	133.0	138.1
	DPC	63.0	64.2	66.0
conv. [s]	30 s	146.58	156.40	177.22
	Fabrikant	18.14	22.98	25.35
	No MRAI	0.19	0.20	0.21
	DPC	15.60	16.83	18.60

Table 2: Statistics on the number of updates and the convergence time for the experiments with a 17 node chain like the one in Fig. 1.

Table 2 and Fig. 4 report the results for a chain topology (Fig. 1) with 17 nodes. Table 2 presents the aggregate results for an easy comparison. It is clear that the standard 30 s MRAI requires a long time (more than 150 s on average) to converge, sending between 80 and 95 `UPDATE` messages. Fabrikant MRAI settings confirm that the predicted explosion of `UPDATE` messages happens also with a real implementation of the protocol. Compared to the no MRAI strategy this also slows down convergence, as convergence is reached after exploring many paths, and MRAI is in use. Simply removing MRAI leads to very fast convergence, but, compared to DPC, doubles the number of `UPDATE` messages sent. This is a very simple topology, and messages are sent in a very short time, hinting that in larger topologies the number and rate of `UPDATE` may clog the system. DPC, the method proposed in this paper, achieves a convergence time roughly 10 times faster than the standard 30 s setting sending the minimum amount of `UPDATE` among the tested solutions. Interestingly, also the variability of sent `UPDATE` is minimal, indicating a very stable behavior.

Figure 4 presents the detailed behavior during the route change episode. Each line reports the average of 10 curves (one per run) measured starting form the time when d_r triggers a reconfiguration. The



(a) 30 s MRAI strategy

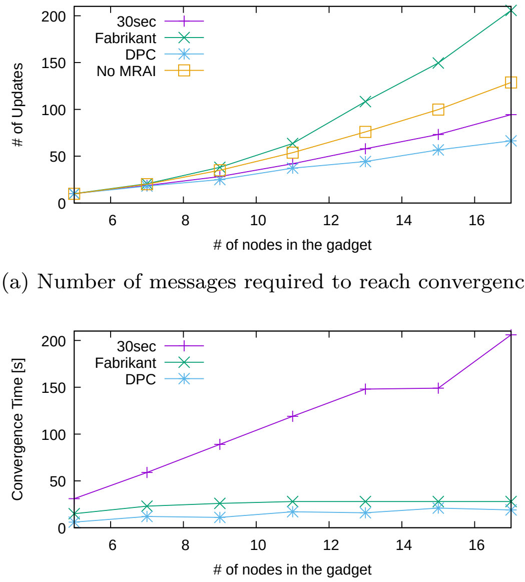
(b) Fabrikant MRAI strategy

(c) No MRAI strategy

(d) DPC-based MRAI strategy

Figure 4: Time evolution of MRAI strategies on a 17 nodes chain like the one in Fig. 1.

time axis spans the entire time needed to reach stability in steps of 1 s, reduced to 1 ms for No MRAI because it converges in less than 1 s. The four strategies have a very different behavior. The 30 s MRAI strategy creates bursts of UPDATE which make nodes converge gradually. The Fabrikant configuration has a long phase in which all the nodes (excluding one-hop neighbors) do not have a valid path. The No MRAI strategy behaves as expected, convergence is almost immediate but path exploration generates more 130 UPDATE messages concentrated in less than 250 ms even if the number of nodes is just 17. The DPC-based strategy does not have a clear pattern of UPDATE generation as its MRAI is different in different nodes, but offers a clear improvement over the previous strategies.



(a) Number of messages required to reach convergence

(b) Time in second required to reach convergence

Figure 5: Convergence time and UPDATE messages with chains of growing length.

Figure 5 presents the trend of the number of UPDATE sent and the convergence time as the number of nodes in the Fabrikant-gadget network (Fig. 1) grows. Figure 5a reports the average number of UPDATE messages on chains of growing length and the exponential growth of the number of UPDATE messages in the Fabrikant configuration is clearly visible. Also No MRAI and 30 s have a super-linear growth, while DPC has a linear growth. Figure 5b reports the convergence time: DPC substantially improves Fabrikant configuration and outperforms the 30 s strategy. No MRAI always converges in less than a second, and thus is not reported.

Since Fabrikant configuration halves the MRAI at every hop we could not test chains longer than 17 nodes as the MRAI value would be negligible, hence "Fabrikant effect" canno tbe properly observed. Nevertheless we can confirm that:

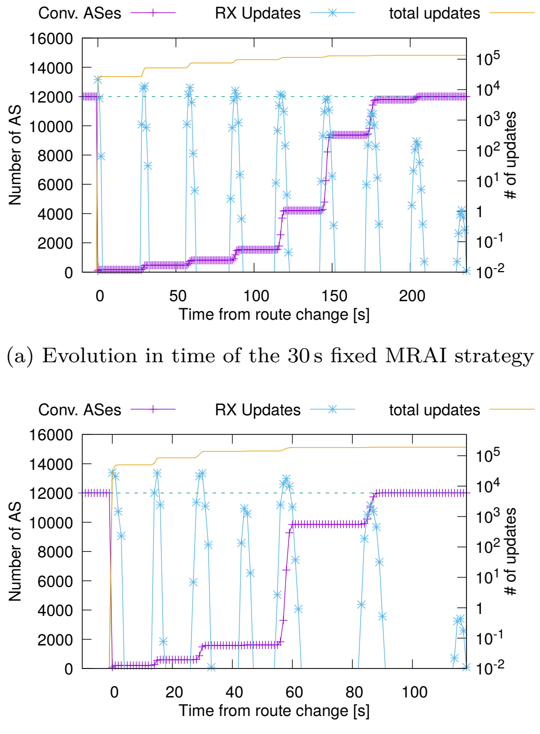
- Fabrikant configuration is systematically outperformed by any other strategy, thus we verified its abnormal trend;
- The No MRAI strategy produces the fastest convergence but the time-density of UPDATE messages is not sustainable since it produces tens of reconfiguration per node in a few hundreds of milliseconds. While this can be handled in a small gadget, the computational overhead needed to update routing tables made of tens of thousands of destinations would not be acceptable;
- The 30 s strategy prevents path exploration, but strongly impacts convergence time; and,
- DPC-based configuration seems to provide the best trade-off between convergence speed and number of UPDATE messages.

6.2. Internet-like Topologies and Scaling

	Strategy	10^{th}	mean	90^{th}
updates	30 s	106 946	136 563	162 169
	DPC	158 237	189 850	210 452
conv. [s]	30 s	205.9	220.5	235.1
	DPC	88.0	89.0	118.1

Table 3: Statistics on the number of updates and the convergence time (in [s]) for an Internet-like topology with 12 000 nodes.

Once verified that in the critical topologies designed by Fabrikant, DPC-based MRAI proves to be a viable solution, we report the results obtained on a 12 000 node Internet-like topology, the largest we could reliably emulate on Fed4FIRE+. We compare only the DPC-based and 30 s MRAI strategies since Fabrikant configuration is not applicable to a



(a) Evolution in time of the 30 s fixed MRAI strategy

(b) Evolution in time of the DPC MRAI strategy

Figure 6: Evolution in time of the two main MRAI strategies on the same Elmokashfi topology of 12 000 nodes, average of 10 repetitions for each strategy.

generic topology, and No MRAI is unfeasible with thousands of nodes. Table 3 reports the main statistics and shows that DPC produces on average more UPDATE messages but converges in just 40% of the time needed by the 30 s strategy.

Figure 6 reports the time evolution of all the experiments for additional insight. The first one is that the number of UPDATE bursts (the blue curve spikes in the figure) is reduced with DPC-based MRAI. This means that the increase in the total number of messages reported in Table 3 is compensated by a smaller number or "rounds" necessary to make BGP converge. DPC-based centrality does not only reduces the interval between bursts, it eventually makes each round or UPDATE exchanges more effective as convergence is faster. The second insight is that with DPC-based MRAI some rounds of UPDATE exchanges have a greater effect than others on the number of nodes that reach convergence; it is an interesting phenomenon that needs further study, as it suggests that there are some nodes that are more important than others, and should converge as early as possible. With centrality-based MRAI tuning we would like to help the convergence of those nodes, and then quickly propagate "good" information to the rest of the network.

7. Open Source Code and Specifications

Emulating BGP involves a number of complex steps that are needed to recreate realistic conditions. These steps require the realization of code and specifications (open source) that are an integral part of the contribution of this paper. This section documents our implementation of a realistic BGP topology generator, the implementation of per neighbor and destination MRAI in the open source BGP daemon BIRD, and the design of a BGP extension to support the distributed computation and propagation of DPC. The software is publicly available both through GitHub[6] and Zenodo[7].

7.1. BGP topology

We implement the model provided by Elmokashfi et al. [39] that generates graphs preserving the structure of the Internet from a stochastic perspective. Graph nodes represent BGP routers (one per AS) that can be of four kinds (i.e., Tier-1, Mid-level, Customer, and Content Provider: {T, M, C, CP}). Their number, interconnection and peering agreement is generated based on the analysis of the Internet topology. Figure 7 reports a generated network with 1000 nodes.

The graph generator is implemented in Python and is now integral part of the well known NetworkX python package, one of the most supported and used libraries for network science[8]. Figure 8 shows the graph generator resource footprint in terms of time and memory needed on both a normal laptop and a server, respectively in purple and green, for increasing number of nodes. The first one uses an Intel I7 7500u cpu (2.7 - 3.5 GHz) and 16 GB of memory, while the second has an Intel Xeon Silver (2.3 GHz) with 503 GB of memory available. Overall the normal laptop requires twice the time and resources when generating bigger graphs, but it is still possible to execute it in a reasonable time. Furthermore the execution time and resources used

[6] https://github.com/internetonfire

[7] https://zenodo.org/doi/10.5281/zenodo.10721373

[8] https://networkx.org/documentation/stable/reference/generated/networkx.generators.internet_as_graphs.random_internet_as_graph.html

on the server are remarkably more constant, so that whiskers representing minimum and maximum are barely visible.

The topology information is stored in a GraphML file, a standard format to represent graphs, the nodes attributes are (we refer the reader to [39] for details and terminology):

- type {T, M, C, CP}: defines the type of AS;
- destinations: a string of comma separated IPv4 network identifiers (prefixes) with the respective netmask, indicating the list of networks the AS exposes.

Given $i, j \in \mathcal{V}$, the attributes for an edge $(i, j) \in \mathcal{E}$ are:

- type {transit, peer}: indicates whether there is a customer-provider or a free peering relationship;
- customer $z \in V$: if the edge (i, j) is of type customer-provider, z identifies which node between i and j is the customer. For peer edges, this attribute is set to "none".

We also realized a generator of the Fabrikant gadget networks.

7.2. Implementing MRAI on BIRD

As mentioned in Section 1 BIRD does not implement the functionalities necessary to execute experiments with BGP timers: indeed, there is no notion of MRAI inside the deamon and therefore all the `UPDATE` where distributed immediately. We implemented the MRAI timer following both the BIRD guidelines but also the specifics of the original BGP RFC [21], i.e., the initial value of the timer and the usage of a jitter. BIRD provides the possibility to initialize the nodes through configuration files, so we included new parameters to tune the new functionalities for each BGP session:

- `mrai_time`, which can be used to configure the MRAI value in milliseconds;
- `mrai_type`, flag used to distinguish between a peer-based MRAI timer or, by default, peer and destination-based timer;
- `mrai_jitter`, jitter value used to randomize the MRAI value at every iteration.

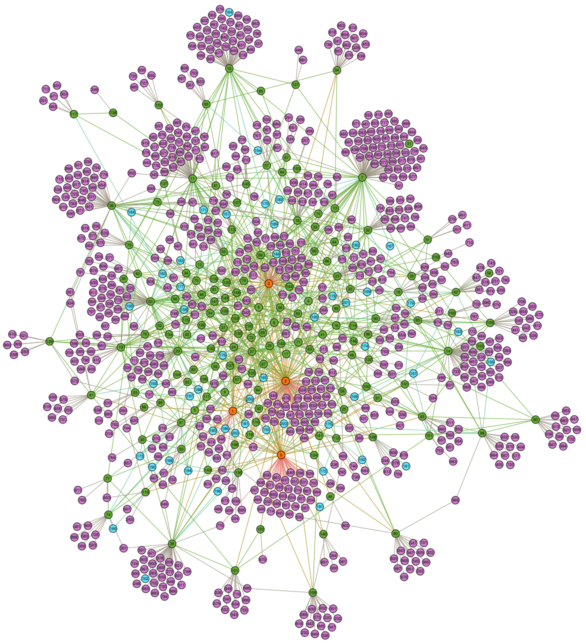


Figure 7: Elmokashfi (Internet-like) topology with 1000 nodes, derived from [39], where Tier-1, Mid-level, Customer, and Content-Provider nodes are represented in orange, green, purple, and cyan, respectively.

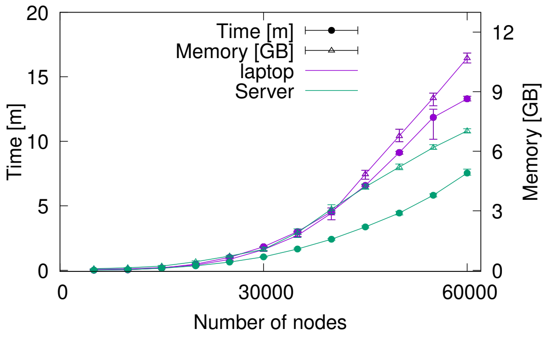


Figure 8: Time (in minutes) and memory required to generate an Internet-like topology with our Python implementation versus the number of AS nodes. Results show the average (the dot), maximum and minimum (whisker) across 10 independent runs on an Intel I7 7500u (2.7 - 3.5 GHz) in purple, and across 50 independent runs on an Intel Xeon Silver (2.3 GHz) with 503 GB of memory in green.

As the original RFC [21] describes, MRAI is a timer defined at the BGP-session level, but affects separately each destination forwarded in that session. We anyhow provide the possibility to apply a single timer to all the destinations that should be distributed towards that neighbor, by setting `mrai_type` equal to 0. We test one destination per run so the value of this flag does not affect our experiments.

We implemented the timer through the introduction of a specific function and the manipulation of the outgoing queues. Following the RFC [21], the timer should not delay the evaluation of new routes in the `Adj-RIBs-In` but only the process that takes the best route from the `Adj-RIBs-In` and places it in the "active" table `LOC-RIB` and further propagate the node decision to the `Adj-RIBs-Out` queues. The procedure is executed after the `Adj-RIBs-in` evaluation and there can be two possible situations:

- **MRAI not active**: the `UPDATE` in the `Adj-RIB-Out` can be propagated and, after that, the timer will be started according to the configured value;
- **MRAI active**: it is not possible to propagate the new information, so the evaluation of the best route will be executed once the timer expires.

The second phase is where MRAI actually affects BGP behavior, stopping potentially flapping routes from propagating further and avoiding their distribution. Once the routing table has been updated it is also mandatory to compute the outgoing queue `Adj-RIBs-Out`, substituting the path in the `UPDATE` message if a new route becomes the preferred one or even not doing anything if there has been an oscillation, i.e., a `WITHDRAWAL` whose effect is cancelled by an `UPDATE`. In the first scenario, after the distribution of the message the timer will become active, with a value defined by the combination of `mrai_time` and `mrai_jitter`. Once this timer expires the function `dest_mrai_timeout` is triggered in order to resume the route comparison process.

7.3. The DPC BGP Extension

Here we describe the extension to the BGP protocol to support a distributed computation of the DPC metric. This proposal requires the introduction of two new parameters that extend the *Path attributes* in the `UPDATE` message, presented in Table 4. These attributes should be flagged as *optional-transitive*, as defined in [21], to ensure that legacy nodes can ignore but still forward them.

The first attribute, NH, is used to identify the list of Next Hops (NHs) that the node can use to propagate the route d contained in the `UPDATE` message. It is not possible to assume there is only one NH due to the multi-path peculiarities of BGP and the capability of the ASes to share aggregated paths. The first element `nNH` encodes the number of elements in the list, and it is a 16 bit object. The second element is a the actual list of ASes identifiers. Each AS identifier is 32 bit in size, for a total size of `nNH`×32 bit.

Attr. Name	Attribute fields	
	[2 Byte]	[Variable size]
NH	nNH	NH_AS
ASLoad	nASL	ASL_List

Table 4: Description of BGP parameters for distributed computation of the DPC metric

The second object identifies the current load (the commodity we use to compute DPC) of known ASes. This is necessary for the receiving AS i to compute its own load and DPC, summing the load provided as input from the other ASes. As the previous attribute, the first item is a 16 bit counter that identifies the number of ASes in the list `ASL_List`. Each object in the list is a tuple (θ, AS_{id}, t) that describes the current load θ associated with AS identifier AS_{id} and a value t to describe when the load has been propagated. This is necessary to identify outdated values that are circulating in the network. The load θ and AS_{id} should be encoded with 32 bit, while t can be stored as *UNIX time* in milliseconds, following the standard dimension of 64 bit. In total `AS_List` has a size equal to `ASLoad`×128bit.

A BGP node should keep a dictionary with the last load values associated with each other AS that shares its metric. In case of changes to its content, the new values should be included in the next `UPDATE` message sent to the neighbors. Each node is responsible to update its own load value once the input ASes metrics changes and then redistribute the changes.

To let a generic node i compute its input load, it has to know if neighbor j has chosen it to forward traffic to d. To achieve this goal we envision three possible methods:

***Periodic Route-Refresh*:** Node i periodically asks node j for its current configuration of best paths. This kind of messages, defined in [40] is usually done only when there has been a change in the policy and a re-evaluation is required, but in this case can be exploited to acquire more knowledge of the decision process of other nodes. This option is fully backward-compatible and incrementally deployable;

***`UPDATE` back-propagation*:** Once node j has decided to use node i as best neighbor to reach d, it includes i into the set of nodes that, according to the policies, should be updated. The message is going to be dumped by i because of the loop-detection mechanism, but the mechanism can still trigger an update of the centrality depending on the attributes in the `UPDATE`;

***Traffic analysis*:** This method requires the cooperation of the BGP daemon and a traffic analyzer. Once the analyzer detects that the neighbor j is forwarding traffic towards i for the destination d it's safe to assume that j chose i as best neighbor and therefore communicate it to the BGP daemon towards a new internal API.

These modifications to BGP will let the network self-compute the load value depending on the number of routes given as input and outputs while still providing enough flexibility for ASes to obfuscate the path through aggregation.

8. Conclusions

Conducting Internet-scale research is extremely challenging for many reasons. One of them is the performance evaluation of the proposals: It cannot be tackled with analytic models because they lack the required detail level, it cannot be easily conducted with (realistic) simulations because of computing resources limitation as well as the lack (again) of realistic models, and it cannot be carried out with simple lab experiments because they lack the scale of the Internet. The use of large-scale testbeds where the proposed solutions are implemented in the protocols under study is thus an almost mandatory tool to make this kind of research credible.

In this work, we have explored the use of the federation of testbeds provided by Fed4FIRE+ to evaluate changes in the management of the MRAI timer of BGP using the BIRD open source implementation properly modified with our proposal and other techniques from the literature. The experimental work has been carefully crafted to make results easily reproducible (given all the software we developed and the scripts we have devised are public and open source), offering the community not only a detailed description of all the experimental machinery we have developed, but also the code developed, the scripts to run the experiments and the post-processing tools to obtain results and graphs.

An experimental setup that can be used to carry out research on BGP to help improve the overall performance of Internet global routing has an intrinsic value; furthermore our contribution also shows that it is indeed possible to improve Internet convergence after a route change by properly managing MRAI, and this without the risk of signaling overhead explosion. We have run experiments with simple topologies to show that analytic results presented in the past with simplified models show up also in real experiments running actual protocols. Next, we have clearly explained the theoretical foundations of our proposal, and validated the results with experiments emulating networks with up to 12 000 ASes, showing that it is possible to modify BGP to improve Internet routing convergence time after changes in topology, a fairly frequent event with the growing number of ASes and prefix destinations.

Acknowledgements

This research was supported by the European Commission, H2020 Programme, Grant Number 732638 Fed4FIRE+ through the Open Call 5 Experiment IoF (Internet on FIRE) when all authors were with the University of Trento, Italy. Mattia Milani is now with Nokia, However, these results were produced before he joined Nokia and therefore do not necessarily reflect Nokia's view on the subject.

References

[1] P. Gill, M. Schapira, S. Goldberg, A survey of interdomain routing policies, SIGCOMM Comput. Commun. Rev. 44 (1) (2014).

[2] A. Fabrikant, U. Syed, J. Rexford, There's something about MRAI: Timing diversity can exponentially worsen BGP convergence, in: IEEE INFOCOM, 2011.

[3] M. Milani, M. Nesler, M. Segata, L. Baldesi, L. Maccari, R. L. Cigno, Improving BGP Convergence with Fed4FIRE+ Experiments, in: Proceedings IEEE INFOCOM (WKSHPS), 2020.

[4] R. B. da Silva, E. Souza Mota, A Survey on Approaches to Reduce BGP Interdomain Routing Convergence Delay on the Internet, IEEE Communications Surveys & Tutorials 19 (4) (2017).

[5] D. Perouli, T. G. Griffin, O. Maennel, S. Fahmy, I. Phillips, C. Pelsser, Detecting the unintended in BGP policies, in: IEEE International Conference on Network Protocols (ICNP), 2012.

[6] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, Delayed internet routing convergence, IEEE/ACM Trans. on Networking 9 (3) (2001).

[7] C. Labovitz, A. Ahuja, R. Wattenhofer, S. Venkatachary, The impact of internet policy and topology on delayed routing convergence, in: IEEE INFOCOM, Vol. 1, 2001.

[8] Z. M. Mao, R. Bush, T. G. Griffin, M. Roughan, Bgp beacons, in: 3rd ACM SIGCOMM Conference on Internet Measurement, 2003.

[9] R. Oliveira, B. Zhang, D. Pei, R. Izhak-Ratzin, L. Zhang, Quantifying path exploration in the internet, IEEE/ACM Trans. Netw. 17 (2) (2006).

[10] D. Pei, X. Zhao, D. Massey, L. Zhang, A study of BGP path vector route looping behavior, in: 24th International Conference on Distributed Computing Systems (ICDCS), 2004.

[11] X. Dimitropoulos, G. Riley, Large-scale simulation models of BGP, in: IEEE 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS), 2004.

[12] B. Quoitin, C. Pelsser, O. Bonaventure, S. Uhlig, A performance evaluation of BGP-based traffic engineering, International Jou. of Network Management 15 (3) (2005).

[13] K. Zhang, S.-T. Teoh, S.-M. Tseng, R. Limprasittipom, K.-L. Ma, S. Wu, C.-N. Chuah, Performing BGP experiments on a semi-realistic Internet testbed environment, in: IEEE 25th International Conference on Distributed Computing Systems (ICDCS) Workshops, 2005.

[14] Y. Song, A. Venkataramani, L. Gao, Identifying and Addressing Reachability and Policy Attacks in Secure BGP, IEEE/ACM Trans. on Networking 24 (5) (2016).

[15] M. Roughan, W. Willinger, O. Maennel, D. Perouli, R. Bush, 10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems, IEEE Jou. on Selected Areas in Communications 29 (9) (2011).

[16] T. Krenc, A. Feldmann, BGP Prefix Delegations: A Deep Dive, in: Proceedings of the Internet Measurement Conference, 2016.

[17] A. Marder, M. Luckie, A. Dhamdhere, B. Huffaker, k. claffy, J. M. Smith, Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale, in: Proceedings of the Internet Measurement Conference, 2018.

[18] B. Schlinker, T. Arnold, I. Cunha, E. Katz-Bassett, PEERING: virtualizing BGP at the edge for research, in: of the 15th CoNEXT, 2019.

[19] N. Rodday, G. D. Rodosek, BGPEval: Automating Large-Scale Testbed Creation, in: 19th International Conference on Network and Service Management (CNSM), 2023.

[20] S. Deshpande, B. Sikdar, On the impact of route processing and MRAI timers on BGP convergence times, in: IEEE GLOBECOM, Vol. 2, 2004.

[21] Y. Rekhter, T. Li, S. Hares, A Border Gateway Protocol 4 (BGP-4), Tech. Rep. 4271, Internet Engineering Task Force (2006).

[22] X. Wang, O. Bonaventure, P. Zhu, Stabilizing BGP routing without harming convergence, in: IEEE INFOCOM (WKSHPS), 2011.

[23] P. Jakma, Revisions to the BGP 'Minimum Route Advertisement Interval', Tech. rep., Internet Engineering Task Force (2011).

[24] A. García-Martínez, P. R. Torres, M. Bagnulo, BGP convergence in an MRAI-free Internet, Computer Networks 240 (2024).

[25] B. Zhang, D. Massey, L. Zhang, Destination reachability and BGP convergence time [border gateway routing protocol], in: IEEE GLOBECOM, Vol. 3, 2004.

[26] A. Sahoo, K. Kant, P. Mohapatra, BGP convergence delay after multiple simultaneous router failures: Characterization and solutions, Computer Communications 32 (7) (2009).

[27] R. Gill, R. Paul, L. Trajkovi, Effect of MRAI timers and routing policies on BGP convergence times, in: IEEE 31st International Performance Computing and Communications Conference (IPCCC), 2012.

[28] N. Laskovic, L. Trajkovic, BGP with an adaptive minimal route advertisement interval, in: IEEE International Performance Computing and Communications Conference (IPCCC), 2006.

[29] A. Bremler-Barr, Y. Afek, S. Schwarz, Improved BGP convergence via ghost flushing, in: IEEE INFOCOM, Vol. 2, 2003.

[30] G. Huston, M. Rossi, G. Armitage, A Technique for Reducing BGP Update Announcements through Path Exploration Damping, IEEE Jou. on Selected Areas in Communications 28 (8) (2010).

[31] W. Sun, Z. M. Mao, K. G. Shin, Differentiated BGP Update Processing for Improved Routing Convergence, in: IEEE International Conference on Network Protocols (ICNP), 2006.

[32] D. Pei, B. Zhang, D. Massey, L. Zhang, An analysis of convergence delay in path vector routing protocols, Computer Networks 50 (3) (2006).

[33] J. Qiu, R. Hao, X. Li, The optimal rate-limiting timer of BGP for routing convergence, IEICE Trans. on Communications E88-B (4) (2005).

[34] P. Jakma, Revised Default Values for the BGP 'Minimum Route Advertisement Interval, Tech. rep., Internet Engineering Task Force (2008).

[35] U. Brandes, On variants of shortest-path betweenness centrality and their generic computation, Social Networks 30 (2) (2008).

[36] L. Maccari, R. Lo Cigno, Improving Routing Convergence With Centrality: Theory and Implementation of Pop-Routing, IEEE/ACM Trans. on Networking 26 (5) (2018).

[37] L. Maccari, L. Ghiro, A. Guerrieri, A. Montresor, R. L. Cigno, On the Distributed Computation of Load Centrality and its Application to DV Routing, in: IEEE INFOCOM, 2018.

[38] L. Maccari, L. Ghiro, A. Guerrieri, A. Montresor, R. L. Cigno, Exact distributed load centrality computation: Algorithms, convergence, and applications to distance vector routing, IEEE Trans. on Parallel and Distributed Systems 31 (7) (2020).

[39] A. Elmokashfi, A. Kvalbein, C. Dovrolis, On the Scalability of BGP: The Role of Topology Growth, IEEE Jou. on Selected Areas in Communications 28 (8) (2010).

[40] E. Chen, RFC2918: Route Refresh Capability for BGP-4, Tech. Rep. 2918, Internet Engineering Task Force (2000).