Making BGP-4 More Efficient and Oscillation-Free

Lucas Immanuel Nickel, Sebastian Rieger
Applied Computer Science Department
University of Applied Sciences Fulda, Germany
{lucas-immanuel.nickel, sebastian.rieger}@cs.hs-fulda.de

Morteza Moghaddassian, J.J. Garcia-Luna-Aceves
Electrical and Computer Engineering Department
University of Toronto, Ontario, Canada (M5S 3G4)
{m.moghaddassian, jj.garcialunaaceves}@utoronto.ca

***Abstract*—The Border Gateway Protocol (BGP) suffers from persistent route oscillations in cyclic policy settings, leading to severe control-plane instability. Although the application of the OPERA routing algebra to BGP (OBGP) offers theoretical oscillation freedom, it lacked validation in widely used routing daemons. This paper presents the first implementation of OBGP within the GoBGP daemon, achieved by integrating ordering constraints into the route import and export pipeline while retaining the native finite state machine and preserving operator control over forwarding decisions such as Local Preference. Our experimental evaluation demonstrates that OBGP provides a viable path to eliminate oscillation and reduce control-plane resource consumption. Specifically, our results highlight three key advantages: practical stability is attained in cyclic settings where conventional BGP oscillates indefinitely, Loc-RIB sizes are significantly reduced with particularly pronounced reductions in peak table sizes, and the largest convergence gains occur in regimes where standard BGP is known to suffer from path hunting. These improvements are achieved without modifying BGP signaling, ensuring both stability and practical deployability.**

***Index Terms*—BGP, inter-domain routing, oscillation-free routing, routing convergence, routing stability**

I. Introduction

The Border Gateway Protocol (BGP) is used for routing across Autonomous Systems (ASes) and is therefore a core component of the Internet. However, BGP's path-vector dynamics, combined with conflicting policies, can lead to persistent oscillations. In operational networks, these effects are often mitigated rather than eliminated, typically through suppression mechanisms. While such measures can reduce route churn, they can also delay convergence. As routing tables and route churn grow, this overhead becomes increasingly costly in terms of memory and processing resources.

Recent work on the BGP Vortex [1] further shows that these instabilities can be weaponized. In particular, transitive communities allow an adversary to trigger remote policy changes several hops away. By forcing distant routers to lower their Local Preference or prepend AS paths, an attacker can induce preference cycles and persistent oscillations.

Prior work [2]–[4] has shown that BGP can, in theory, be made stable and loop-free without changing BGP signaling by imposing suitable ordering constraints on route import, selection, and export. This is done using the Ordered Path Etiquette for Routing Algebra (OPERA), and the resulting design is referred to as OPERA-based BGP (OBGP). To the best of our knowledge, no prior work has implemented and experimentally validated OBGP in a routing daemon.

ISBN 978-3-903176-82-9 © 2026 IFIP

The three main contributions of this paper are:

- The first implementation of OBGP in the widely used open-source GoBGP daemon.
- A Kubernetes-based emulation environment and experimental tooling for fill and drain experiments.
- An experimental comparison between OBGP and unmodified GoBGP on `germany50`, `BAD GADGET`, and an adapted `noble-eu` topology, reporting convergence and `Loc-RIB` characteristics.

Section II reviews related work and positions this paper with respect to prior work. Section III outlines the theoretical basis of OBGP. Section IV describes the implementation of OBGP in GoBGP. Section V presents the emulation environment and methodology. Section VI reports the experimental results. Section VII discusses the implications and limitations, and Section VIII concludes.

II. Related Work

This section reviews prior work on BGP oscillations, operational mitigations, and design-oriented approaches to stable inter-domain routing. Existing work either mitigates oscillation, extends route dissemination, or studies conditions for stability. None, however, provides a daemon-level implementation and experimental validation of convergence without changing BGP signaling.

A. The Growing Problem with BGP Oscillations

Because inter-domain routing (IDR) policies are configured independently, even small inconsistencies in common attributes such as Local Preference and the Multi-Exit Discriminator (MED) can induce preference cycles across neighboring Autonomous Systems (ASes). These cycles can in turn lead to persistent oscillations and slow convergence. With the ever-increasing size of BGP routing tables, this has become a major problem for the Internet at large and has been identified as one of the most pressing BGP problems [5].

Conventional BGP propagates only the currently selected best route, making route availability at an Autonomous System (AS) strictly dependent on upstream selections. In a policy cycle, these dependencies create a feedback loop where a policy-driven reselection at one AS changes the exported route, triggering cascading reselections at other ASes [6].

This oscillation problem, often referred to as route flapping in practice, has been known since 1999. Griffin *et al.* formalized these dynamics using the Stable Paths Problem (SPP) and showed that such preference cycles can yield persistent oscillations under path-vector dynamics [7], [8]. The `BAD GADGET` is the minimal configuration used to illustrate this mechanism and is known to exhibit persistent oscillations [8].

BGP oscillations increase convergence time and can leave ASes with inconsistent views of the Internet. Furthermore, recent work [9] has shown that the frequency of BGP route flapping is proportional to the level of congestion, and that route flaps propagate across the entire Internet. This work also provides a summary of the extensive prior literature on the study and classification of BGP anomalies, and it is worth noting that it analyzed BGP route flapping using a large-scale testbed of commercial equipment.

Route flapping is therefore an Internet-wide problem that must be addressed if the Internet is to remain stable as it continues to scale. However, no satisfactory solution currently prevents or fully mitigates its effects.

B. Operational Mitigations and Design-Oriented Approaches

Many approaches have been investigated to either reduce churn or accelerate path exploration (e.g., [10]–[15]). These approaches differ in scope. Some mitigate oscillations operationally, while others aim at stability by construction. The two main operational-mitigation approaches implemented by router vendors are Route Flap Damping (RFD) and the Minimum Route Advertisement Interval (MRAI) [5]. In addition, protocol extensions such as Add-Path improve path visibility and can reduce some transient effects of path exploration, but they do not by themselves guarantee convergence in cyclic-policy settings [16].

RFD penalizes and suppresses routes that exhibit frequent changes, such as repeated withdrawals or attribute modifications [17]. However, aggressive damping can excessively penalize well-behaved prefixes and delay convergence, which motivated later guidance toward more conservative thresholds [18]. More broadly, suppression and timer-based heuristics can reduce update load and make oscillations less visible, but they do not provide stability by design.

MRAI enforces a minimum interval between BGP updates, effectively rate-limiting how quickly route changes are propagated [19]. Routers may then explore alternatives privately before exposing changes, which can reduce churn during path exploration, but this comes at the cost of delaying convergence.

Beyond such operational mitigations and protocol extensions, prior work also established a design-oriented line of research on stable inter-domain routing. Routing-algebra formulations identified structural conditions for convergence, including monotonicity, total ordering, and export discipline [20]–[22]. More recently, stable routing designs have been proposed based on the OPERA routing etiquette to address oscillation at the protocol-design level [2]–[4].

Table I summarizes how prior approaches differ from our validation of OBGP in GoBGP.

III. From OBGP Theory to Implementation

Many approaches have been proposed to address BGP's long-standing stability problems, ranging from protocol replacements to modifications of BGP itself. Given that BGP runs in every AS of the Internet and in many data centers and network overlays, it is impractical to consider approaches that would replace BGP altogether. Changing BGP signaling would also create serious deployment challenges, because some BGP speakers would have to run different protocol variants for some time to ensure compatibility between peers. Finally, it is not possible to dictate how different ASes should set their preferences or to predict what public or private policies will be used in the future.

These considerations suggest that the most deployable solution is one that does not modify BGP signaling and does not constrain how ASes configure their policies. To achieve this, the routing software itself must be designed so that a router cannot trigger route flapping merely by changing its locally preferred route.

Protocols for routing within the same AS such as OSPF [23] and EIGRP [24] assume that all routers in a network select routes based on the same network-wide optimality criteria, such as shortest paths to destination prefixes. By contrast, the use of Local Preference in BGP means that route selection does not adhere to any Internet-wide optimality criterion. In addition, it is well known that AS-path information in BGP can only be used to detect, rather than prevent, temporary routing loops. Oscillation and convergence problems in BGP arise because routers change their preferred routes based on local AS-path information, even though those routes are not necessarily better than the routes selected by routers in other ASes, whether neighboring or distant.

The central design question is therefore whether BGP can remain stable even when route selection and announcement are driven by local AS-path information and local policy, rather than by any Internet-wide optimality criterion.

Prior theoretical work answers this question in the affirmative [2], [3]. We corroborate this result in practice through our implementation of OBGP in GoBGP and our experimental study of its control-plane behavior.

Prior work introduced an *order relation* for local route selection that does not rely on any global optimality criterion [3]. Under this relation, a route is better if it does not contain the router's own AS and is lexicographically smaller when paths are compared first by hop count and then by AS identifiers.

In [3], it was shown that if all routers select loop-free routes using this order relation, then the routing protocol converges. It was also shown that the protocol is loop-free at every instant, that is, it never oscillates, if the order relation is always satisfied whenever a router changes routes locally. OBGP was proposed in [2] based on these results and was shown to converge and be loop-free at all times. However, OBGP has not previously been implemented in a BGP routing daemon or tested to determine the performance benefits of integrating the OBGP order relation into the BGP code that alters, imports, and exports routes.

TABLE I
COMPARISON OF PRIOR APPROACHES RELEVANT TO BGP OSCILLATION AND THIS WORK.

Approach	Category	Prevents oscillations	Implemented in a daemon	Changes signaling	Description
RFD [17], [18]	Mitigation	No	Yes	No	Penalizes and suppresses routes that exhibit frequent changes.
MRAI [5], [19]	Mitigation	No	Yes	No	Enforces a minimum interval between successive BGP updates.
Add-Path [16]	Extension	No	Yes	Yes	Allows a router to advertise multiple paths for the same destination.
OBGP theory [2]–[4]	Design	Yes	No	No	Applies OPERA-based ordering constraints to route import and export.
This work	Validation	Yes	Yes	No	Implements and experimentally validates OBGP in GoBGP.

IV. IMPLEMENTATION OF OBGP IN GOBGP

This section describes how OBGP is integrated into GoBGP by modifying route import and export while leaving session management unchanged. Our implementation is based on GoBGP commit `fc80fdf` and applies OBGP-specific import and export transformations to candidate paths that pass the standard AS-loop check. To preserve interoperability with unmodified BGP speakers, changes are restricted to the `table` and `server` packages, specifically altering the import, cleanup, and export of routes. The native GoBGP finite state machine remains responsible for signaling, session management, and update processing.

A key architectural distinction from conventional BGP is the decoupling of internal Routing Information Base (RIB) sorting from route import and export. While the RIB retains standard BGP tie-breaking to ensure that operators maintain full local authority over forwarding policies and attributes such as Local Preference, route import and export follow OBGP. A runtime feature flag activates this alternative logic, allowing a single binary to serve as both the baseline and the experimental system. This design keeps the baseline and experimental systems directly comparable and supports reproducible evaluation. Because this work focuses exclusively on control-plane behavior, data-plane forwarding, including route installation into the Forwarding Information Base (FIB), is intentionally excluded. These mechanisms are orthogonal to the control-plane properties evaluated in this study. This design therefore enables a direct validation of OBGP's control-plane properties in an operational BGP software stack without confounding changes to BGP signaling.

A. OBGP Import and Export Transformations

Both transformations rely on a strict total order over AS paths to ensure oscillation freedom. The implementation instantiates this order by preferring shorter AS paths, with ties broken deterministically by a lexicographical comparison of the ASN sequence. Algorithm 1 outlines this comparison logic.

Algorithm 1 OBGP Path Comparison Logic

```
function IsBetter(P_new, P_existing)
    if Len(P_new) ≠ Len(P_existing) then
        return Len(P_new) < Len(P_existing)
    end if
    return LexCompare(P_new, P_existing) < 0
end function
```

To ensure a canonical comparison, the current implementation limits support to `AS_SEQUENCE` segments. Furthermore, features that would introduce path ambiguity or violate algebraic consistency, such as multi-pathing or BGP Add-Path, are disabled. This ensures that the system stores at most one candidate path per peer and destination. Consequently, the candidate set size per destination is bounded by $O(\deg(v))$, where $\deg(v)$ denotes the number of peers.

a) Import transformation: The import logic acts as an admission filter within the per-destination calculation routine. In line with standard BGP semantics, an update from a neighbor implicitly withdraws the previously stored path before the new candidate is evaluated. A candidate is considered valid if it has a reachable next hop and passes the standard AS-loop check. Such a valid candidate is admitted into the `Loc-RIB` only if it is strictly better, according to the total order, than the currently worst valid path stored for that destination. If no valid path exists, the first candidate is installed to initialize the destination state and define the admission threshold. Thereafter, the router suppresses any candidate path worse than this threshold, thereby limiting path exploration.

b) Export transformation: The selection algorithm overrides the standard best-path selection hook and explicitly

exports the worst eligible path according to the total order, excluding withdrawals and invalid next hops. Under the stated assumptions, the not-worst import and worst export coupling of OBGP is proven to be oscillation-free and loop-free [3].

B. Superset Pruning

To accelerate convergence during network drains, the implementation includes the superset pruning heuristic proposed in the original OBGP design [3]. Upon processing an explicit or implicit withdrawal, the system scans the known path list for the destination and invalidates any path containing the withdrawn AS sequence as a contiguous subsequence. For example, withdrawing the path (A, B, C) causes the router to prune any longer path that contains (A, B, C) as a contiguous subsequence, such as (X, A, B, C, Y).

As GoBGP processes updates concurrently, this operation modifies shared destination state. As shown in Algorithm 2, a per-prefix mutex ensures consistency during the update and pruning phase. Given the $O(\deg(v))$ bound on the candidate set, lock contention remains negligible.

Algorithm 2 Admission and Pruning of Paths

procedure CALCULATE(P_{new})
  $Mutex.Lock()$
  if P_{new} is Withdrawal **then**
    $P_{removed} \leftarrow ExplicitWithdraw(P_{new})$
    if $P_{removed} \neq \emptyset$ **then**
      $PruneSupersets(ASPath(P_{removed}))$
    end if
  else
    $P_{old} \leftarrow FindExisting(P_{new})$
    if $P_{old} \neq \emptyset$ **then**
      $ImplicitWithdraw(P_{old})$
      $PruneSupersets(ASPath(P_{old}))$
    end if
    $P_{worst} \leftarrow GetWorstValidStoredPath()$
    if $P_{worst} = \emptyset$ **or** $IsBetter(P_{new}, P_{worst})$ **then**
      $InsertSort(P_{new})$
    end if
  end if
  $Mutex.Unlock()$
end procedure

V. EMULATION ENVIRONMENT

This section describes the emulation environment used to compare OBGP and conventional BGP in terms of control-plane stability, convergence behavior, and scalability. The emulation environment is containerized and deployed on Kubernetes to enable declarative deployment of large topologies and to ensure reproducibility across experiments. Because we focus on topology- and policy-induced control-plane dynamics, we use a controlled emulation environment to isolate these effects and treat latency and bandwidth constraints as orthogonal factors. The architecture of the emulation environment is shown in Figure 1.

A. Architecture and Orchestration

All experiments are performed on a Kubernetes 1.32 cluster configured with 20 CPU cores and 14 GB RAM. Topologies are deployed via Helm 3. Each run starts from a fresh deployment. Routers are instantiated as dedicated Kubernetes pods running one container each. The container runs the modified GoBGP daemon, compiled with Go 1.24 and implementing OBGP, together with an HTTP control service implemented in Python 3.12 using FastAPI. The control service communicates with the daemon via gRPC and exposes a REST API for experiment orchestration. Router-specific parameters such as router ID, ASN, neighbor set, and feature flags are passed using environment variables. On startup, the control service resolves peer service names via Kubernetes DNS into IP addresses, renders them into a GoBGP configuration, and starts the daemon once neighbor pods are reachable.

B. Network Topologies

The evaluation combines one control topology and two preference-cycle topologies to distinguish baseline convergence behavior from policy-induced instability. `germany50` and `noble-eu` are taken from SNDlib [25]. The canonical `BAD GADGET` construction follows [8].

`germany50` represents the control topology. It models a national backbone with 50 nodes and 88 bidirectional links and is used to measure convergence and scaling behavior in the absence of policy cycles.

`BAD GADGET` consists of four ASes, where one AS is surrounded by three neighboring ASes whose Local Preference settings form a cycle. It represents the minimal configuration known to exhibit persistent oscillations under conventional BGP [7] and serves as a litmus test for control-plane stability.

The `noble-eu` topology extends this concept to a larger multi-AS environment. It contains 28 nodes and 41 bidirectional links. Two large preference cycles are induced: one traversing Frankfurt, Hamburg, Amsterdam, London, Paris, Strasbourg, and back to Frankfurt, and another traversing Vienna, Prague, Budapest, Belgrade, Athens, Rome, Milan, Munich, and back to Vienna. Prefixes are announced at Brussels and Zagreb, which connect into these cycles through three equal-cost links each. This topology tests the behavior of BGP and OBGP under larger cyclic preferences.

C. Measurement Methodology and Scenarios

Experiments are orchestrated by a control script on the host that accesses per-pod REST endpoints. Each configuration is repeated five times, and results are aggregated as minimum, mean, and maximum across repetitions. Two protocol modes are evaluated: OBGP and conventional BGP.

Control-plane load is generated by a route-noise workload that injects and withdraws IPv4 prefixes through the GoBGP gRPC API. Prefixes are sampled from the IPv4 address space using a fixed random seed. For `germany50` and `BAD GADGET`, announcements originate at the Aachen router using one prefix block. For `noble-eu`, announcements originate at Brussels and Zagreb using two disjoint prefix blocks. The

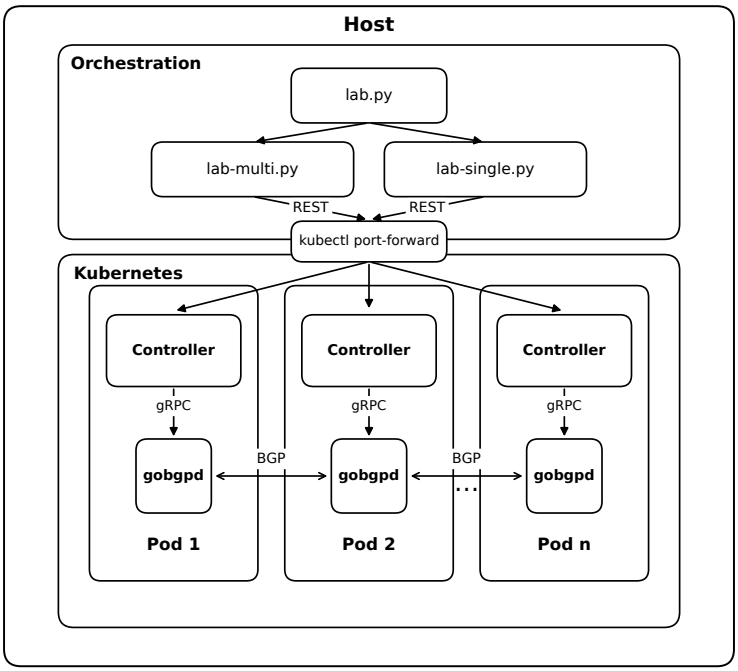

Fig. 1. Architecture of the emulation environment.

workload bounds the number of simultaneously active routes and parameterizes the announcement rate. For `germany50` and `BAD GADGET`, the rates are 1.0, 2.0, and 3.0 prefixes per second at Aachen. For `noble-eu`, the rates are 0.5, 1.0, and 1.5 prefixes per second at each of Brussels and Zagreb. In all cases, prefixes have a nominal lifetime of 60 s with a relative lifetime jitter of 0.5, yielding uniformly distributed lifetimes in the interval from 30 s to 90 s, with at most 90 active prefixes.

Each run begins with a fill phase that establishes a controlled steady state. During 30 s of active noise, the origin nodes inject and withdraw prefixes until a non-empty set of routes is present network-wide. The run then enters a pause phase of 30 s that suppresses further churn while keeping the currently active prefixes installed at the origin.

Drain scenarios start from this frozen prefix set. In full-drain runs, the origin stops the noise generator and withdraws all remaining active prefixes. Convergence time is defined as the first time at which every monitored non-origin router reports zero paths for the injected prefix set over consecutive samples spanning three seconds. In partial-drain runs, the origin withdraws 25%, 50%, or 75% of the currently active prefixes while keeping the remainder, and the post-drain transient is observed without requiring the table to be fully cleared.

Fill runs measure the transient from a clean start. Noise is started at the beginning of the measurement window and paused at its end, with no subsequent drain.

Routers are polled at 1 Hz to collect RIB counters (destination and path counts) and AS-path lengths (minimum, mean, maximum). Only non-origin routers are monitored, excluding injection nodes (Aachen for `germany50` and `BAD GADGET`, Brussels and Zagreb for `noble-eu`).

VI. Evaluation

The evaluation compares the proposed OBGP implementation with the conventional BGP implementation of GoBGP with respect to oscillation freedom, convergence behavior, and RIB characteristics. Figure 2 provides a consolidated overview of the main results across all fill, partial-drain, and full-drain scenarios, which are discussed in detail below. Results are reported for the acyclic control topology `germany50` as well as the cyclic topologies `BAD GADGET` and `noble-eu`.

A. Oscillation Freedom

We first evaluate whether the ordering constraints of OBGP eliminate persistent oscillations in practice.

1) Measurement Methodology: Oscillation freedom is assessed using two indicators. The first is the age of `Loc-RIB` entries, measured as the time since the last installation of a route entry in the `Loc-RIB`. The second indicator is the stability of installed route counts, measured as the total number of installed routes in the `Loc-RIB` averaged over all routers.

Detecting oscillations is inherently challenging using external measurements alone. Route flapping is a high-frequency control-plane phenomenon, and install–withdraw cycles may occur faster than the 1 Hz sampling interval used in our experiments. As a result, short-lived flaps may not be distinguishable from stable states in externally observed RIB snapshots.

Therefore, oscillation freedom cannot be reliably inferred from external measurements alone and requires on-device inspection. In practice, persistent flapping is identified by examining the age of RIB entries, since each route reinstallation resets its age counter. Repeated age resets under a constant prefix set therefore indicate ongoing oscillation.

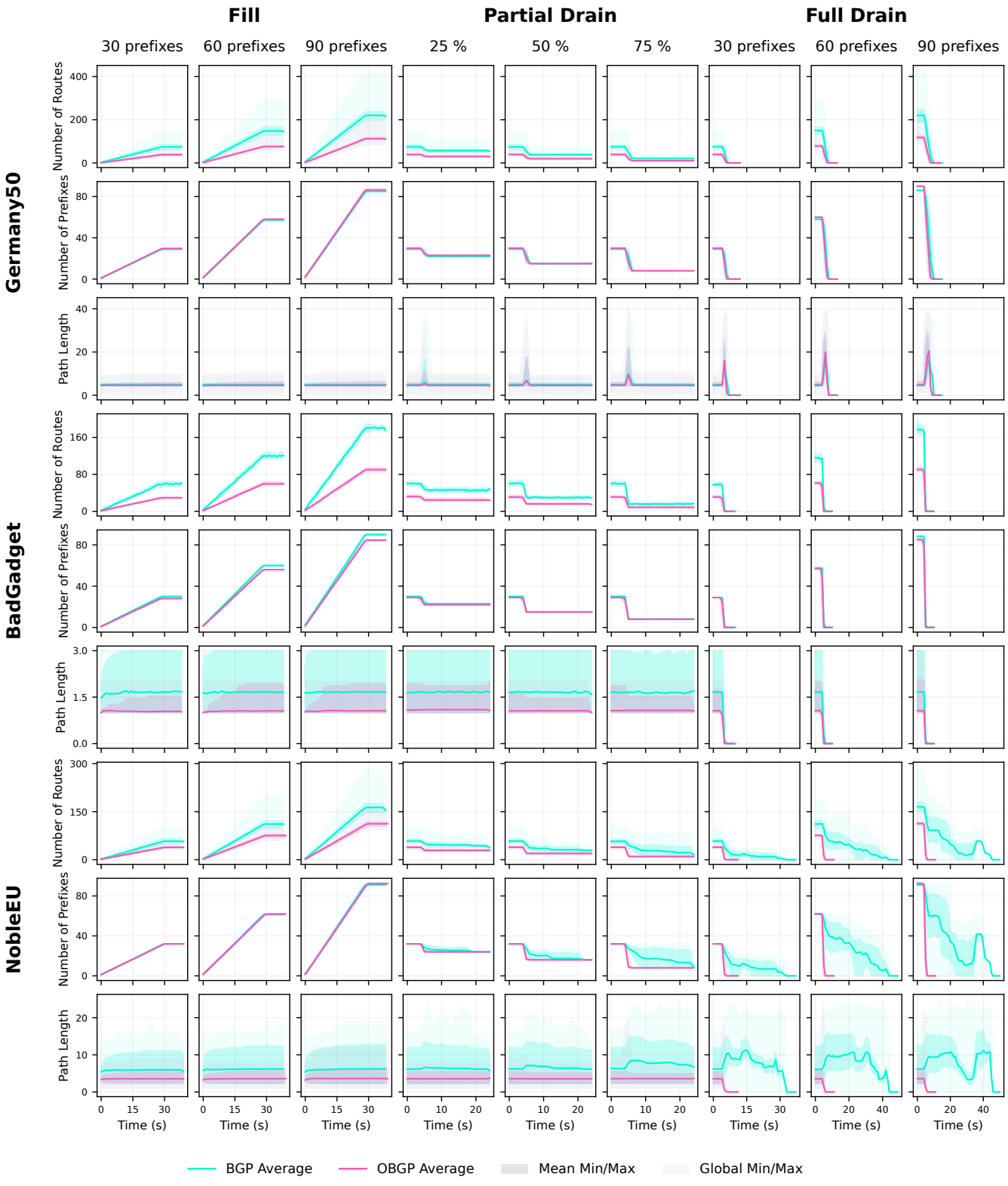

Fig. 2. Summary of results for the fill/partial-drain/full-drain scenarios.

2) Results: Inspection of the on-device Loc-RIB entry age counters showed that OBGP did not exhibit any age resets after the initial installation of routes. In contrast, BGP continuously reset age counters in the cyclic topologies.

Route-count stability followed the same pattern. In BAD GADGET, BGP continued to oscillate even after announcements had ceased, while OBGP stabilized at a constant route count. In noble-eu, BGP also exhibited continuous oscillations, whereas OBGP reached a stable number of routes.

B. Convergence Behavior

The ability of the control plane to adapt to a sudden loss of prefixes was evaluated using partial and full drains.

1) Measurement Methodology: Convergence time is defined as the interval between the first withdrawal event and the point at which the control plane reaches a quiescent state.

2) Partial Drains: In germany50, both protocols converged within about 2 seconds. A similar behavior was observed in BAD GADGET, where the partial drain also converged within about 2 seconds. However, BGP oscillated outside the drain phase, whereas OBGP remained stable.

In noble-eu, partial drains revealed substantial differences between the protocols. For a 25% drain, BGP required about 15 seconds to withdraw all invalid prefixes. For a 50% drain, BGP eventually cleared the withdrawn prefixes after about 18 seconds. For a 75% drain, BGP did not withdraw the predefined fraction of prefixes within the 20-second observation window. In contrast, OBGP converged within about 1 to 2 seconds in all partial-drain scenarios.

3) Full Drains: In germany50, BGP converged within 3 to 7 seconds, while OBGP converged within 3 to 5 seconds. During convergence, the global maximum path length peaked at about 25 hops for BGP and about 30 hops for OBGP. In BAD GADGET, both protocols successfully processed full drains and showed similar convergence trajectories. BGP lagged behind OBGP by roughly 1 second.

In noble-eu, BGP did not reach a stable state and exhibited persistent oscillations, while OBGP converged and remained stable. Full-drain experiments revealed severe control-plane instability and practical non-convergence of BGP caused by extensive path hunting. OBGP drained all routes within about 3 seconds. BGP convergence times increased to 29 seconds for 30 prefixes, 40 seconds for 60 prefixes, and 42 seconds for 90 prefixes. Many BGP runs exceeded the 60-second timeout and had to be repeated.

C. RIB Characteristics and Path Quality

This section evaluates steady-state Loc-RIB properties, focusing on RIB size, path coverage, and path length.

1) RIB Size: RIB size is measured as the number of installed routes in the Loc-RIB averaged over all routers. Table II reports approximate average and peak Loc-RIB sizes under BGP and OBGP. Across all topologies and prefix loads, OBGP reduced average RIB size by approximately 27–51% and peak RIB size by about 43–60% compared to BGP.

2) Path Coverage: Path coverage is the number of injected prefixes that appear in the Loc-RIB at non-origin routers.

In germany50, both protocols should reach 100% coverage because the topology contains no policy cycles. The small dip visible for BGP in some full-drain plots is therefore most likely a measurement artifact.

In BAD GADGET, OBGP converges, although a marginal, constant offset of one to two prefixes in path coverage occurred. Because this offset was uniform across all non-origin routers, we attribute it to a prefix injection failure in the gRPC controller at the origin node. This localized ingress artifact is independent of the proposed approach, as confirmed by the full coverage achieved in the larger noble-eu topology. More importantly, while conventional BGP exhibits persistent oscillations in these cyclic settings, OBGP consistently converges to a stable state.

3) Path Length: Path length is measured as the AS-hop count of each route in the Loc-RIB, averaged over all routes and all routers.

In germany50, both protocols produced comparable path lengths. OBGP achieved average path lengths of about 4.5 hops, while BGP averaged about 5 hops. Maximum path lengths were similar and peaked at about 6 hops for OBGP and about 5.7 hops for BGP.

In BAD GADGET, both protocols achieved a minimum path length of one hop. OBGP remained stable with an average of about 1.1 hops, while maximum path lengths approached 2 hops as the number of prefixes increased. BGP exhibited higher average path lengths between about 1.6 and 1.75 hops, reached maxima of up to 3 hops, and showed noticeable temporal fluctuations caused by oscillations.

In noble-eu, the minimum path lengths converged toward 2 hops for both protocols, although observed global minima ranged from 1 to 6 hops across all routers. OBGP maintained average path lengths slightly below four hops and maximum path lengths around six hops. BGP averaged about 6 hops, reached maxima of up to 11 hops, and exhibited strong temporal fluctuations due to oscillations.

VII. Discussion

The main value of OBGP is not in improving cases that are already well behaved, but in changing control-plane behavior exactly where conventional path-vector dynamics fail. In the acyclic control topology, both protocols behave similarly. In cyclic settings, however, the difference is qualitative rather than incremental. Conventional BGP continues to alternate among policy-induced choices, whereas OBGP converges to a stable set of retained and exported routes. That distinction matters. The contribution is not a better damping rule, a better timer, or a better path-hunting heuristic. It is a control-plane discipline that preserves BGP signaling while making convergence predictable.

The reduction in control-plane state is likewise a direct consequence of the design rather than an incidental implementation artifact. The central mechanism is the interaction between import and export. Import limits which paths are retained

TABLE II
AVERAGE AND PEAK Loc-RIB SIZE UNDER BGP AND OBGP. $\Delta[\%] = \frac{\text{BGP}-\text{OBGP}}{\text{BGP}} \cdot 100$.

Topology	Prefixes	Avg BGP	Avg OBGP	Avg Δ [%]	Peak BGP	Peak OBGP	Peak Δ [%]
germany50	30	75	45	40	150	60	60
	60	150	75	50	300	125	58
	90	225	110	51	420	200	52
BAD GADGET	30	60	30	50	65	33	49
	60	120	60	50	125	63	50
	90	180	90	50	185	93	50
noble-eu	30	55	40	27	100	40	60
	60	110	80	27	200	100	50
	90	160	115	28	280	160	43

in the Loc-RIB. Export prevents policy-driven alternation from propagating to neighbors. Together, these mechanisms explain why the largest gains appear in cyclic topologies and during drains, where conventional BGP is known to suffer from extended path exploration. Put differently, OBGP does not merely converge faster once instability appears. It reduces the amount of unstable state that can arise in the first place.

The same interaction also explains the stability of route export over time. By discarding dominated paths at import and deterministically exporting the worst admissible path, OBGP establishes an implicit threshold on subsequent route choices. Additional admissible paths may still enter the Loc-RIB, but export remains stable unless the threshold path itself changes.

Superset pruning serves a different role. It does not create stability by itself. Instead, it accelerates convergence during withdrawals by removing paths that depend on a withdrawn AS sequence. This shortens the lifetime of transient state and explains why the largest gains are visible in drain scenarios.

A further implication is architectural. OBGP separates *local forwarding* from *route propagation*. Local policy remains intact because the native BGP tie-breaking logic still determines the route used locally, including attributes such as Local Preference. What changes is which routes are admitted and which of them may be exported. This distinction matters because oscillation freedom does not require removing policy. It requires limiting how policy-driven best-path decisions feed back into the network.

This separation also has security implications. In attacks such as the BGP Vortex [1], an adversary can manipulate attributes such that remote routers repeatedly change their preferred routes and induce persistent oscillation. OBGP constrains this mechanism because export follows a strict total order rather than policy-driven best-path decisions. Adversarial inputs may still affect local preference and forwarding, but one source of control-plane instability is constrained by design.

The evaluation is deliberately scoped to topology- and policy-driven control-plane dynamics. It excludes AS-path constructs that complicate a strict total order, in particular AS_SET, AS_CONFED_SEQUENCE, and AS_CONFED_SET. It also excludes dissemination features such as Add-Path and multi-path. We likewise do not model heterogeneous link delays or bandwidth bottlenecks. These factors affect deployment realism and transient timing. They do not change the main result. In the evaluated setting, enforcing OBGP's ordering constraints in a production-ready daemon eliminates persistent oscillation and reduces transient control-plane state.

These limitations define the boundary of the current validation. The theory underlying OBGP does not depend on one particular lexicographic order over one particular path representation. It depends on the existence of a strict total order that is respected by import and export. The implementation in this paper instantiates that principle for ordinary AS_SEQUENCE paths. The next step is to extend the same idea to richer path objects and richer dissemination semantics.

VIII. CONCLUSION AND FUTURE WORK

This paper shows that OBGP can be integrated into a widely used BGP daemon without changing BGP signaling, and that the resulting system behaves as predicted by prior theory. In cyclic-policy settings where conventional BGP exhibits persistent oscillation, the GoBGP-based prototype converges to a stable control plane. It also reduces both average and peak Loc-RIB size and substantially shortens drain transients. These results matter not because they make well-behaved cases slightly better, but because they show that oscillation freedom can be ensured in an operational software stack without sacrificing interoperability or local policy control.

More broadly, this paper does not propose another mitigation for route flapping. It validates a different design principle for inter-domain routing. Stability should be built into route import and export, not added afterwards through suppression and timers once instability has already been created. In that sense, this implementation is not merely a prototype. It shows that a real BGP daemon can operate under the constraints imposed by an algebraic discipline such as that imposed by OPERA and still behave as an ordinary interoperable BGP speaker on the wire.

A natural next step is to extend the ordering relation beyond the restricted path model used in this paper. The current prototype assumes AS_SEQUENCE segments because they admit a direct lexicographic order over ASNs. Operational BGP deployments, however, also encounter structured path objects such as AS_SET, AS_CONFED_SEQUENCE, and AS_CONFED_SET. Ordered segments can still be compared positionally, while unordered ones first require a canonical

representation. The same idea extends to hierarchical or recursively structured path objects. Sub-objects can be normalized first, and the resulting typed representation can then be compared. OBGP does not depend on one special ordering rule. It depends on the existence of a strict total order. Once such an order is available, the same import and export discipline can be carried over to richer path objects.

A second major direction concerns multi-path dissemination, especially Add-Path. Here the problem is different. With structured AS paths, the challenge is to order one richer path object. With Add-Path, the challenge is to preserve stability while disseminating multiple admissible paths. In the current prototype, each peer contributes at most one candidate per destination, and export selects one worst admissible path. Add-Path changes the exported object from a single path into an ordered path set. Stability-preserving support for Add-Path therefore requires more than assigning a rank to path identifiers. It requires a discipline for ordered families of paths.

One promising direction is to define disjoint ordering domains rather than treating all advertised paths as part of a single selection process. A router could maintain several path classes, each with its own total order and import/export threshold. These classes could capture different routing goals, e.g., low latency or high bandwidth. Within each class, OBGP's not-worst import and worst export logic would still apply. Across classes, additional constraints would be needed to prevent cross-class feedback from reintroducing oscillation. Under this view, Add-Path and multi-dimensional routing become problems of coordinating several independently ordered channels rather than exporting an unrestricted set of paths.

Overall, this work provides the first daemon-level validation of OPERA-based BGP in a widely used routing daemon without changing BGP signaling. The central result is that a routing design previously established only in theory can be brought into an operational routing system without sacrificing interoperability or local policy control. Our GoBGP implementation shows this concretely: under cyclic policies, it eliminates persistent oscillation, converges, and reduces control-plane state. In this sense, the contribution is not only an implementation of a known design, but a validation that the design holds up in a real routing stack. This gives the field a concrete baseline for future work. The next step is to extend that baseline from plain paths to structured path objects, from single-path export to ordered path sets, and from one-dimensional ranking to richer routing algebras.

To facilitate reproducibility and foster further research, the source code of the modified GoBGP daemon, the emulation environment, and the experimental raw data are available at https://github.com/Stinktopf/gobgp.

Acknowledgment

This work was supported by a fellowship of the German Academic Exchange Service (DAAD) and by the Canada Excellence Research Chair in Intelligent Digital Infrastructures at the University of Toronto, funded by the Tri-agency Institutional Programs Secretariat.

References

[1] F. Stöger, H. Birge-Lee, G. Giuliari, J. Subira-Nieto, and A. Perrig, "BGP vortex: update message floods can create internet instabilities," in *Proceedings of the 34th USENIX Conference on Security Symposium*, ser. SEC '25, 2025.

[2] J. J. Garcia-Luna-Aceves, "Eliminating routing loops and oscillations in BGP using total ordering," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, 2022.

[3] J. J. Garcia-Luna-Aceves, "Stable, loop-free, multi-path inter-domain routing using BGP," in *ICC 2022 - IEEE International Conference on Communications*, 2022.

[4] J. J. Garcia-Luna-Aceves, "Techniques for loop-free multi-path inter-domain routing in communications networks," US Patent 12580854, 2026.

[5] X. Wang, O. Bonaventure, and P. Zhu, "Stabilizing BGP routing without harming convergence," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2011.

[6] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," *Computer Networks*, vol. 32, no. 1, 2000.

[7] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1999.

[8] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, 2002.

[9] K. Cereceda, J. But, and P. Branch, "Experimental observations and analysis of BGP route flapping dynamics," in *2025 International Conference on Information Networking (ICOIN)*, 2025.

[10] J. Luo, J. Xie, R. Hao, and X. Li, "An approach to accelerate convergence for path vector protocol," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 3, 2002.

[11] D. Pei, M. Azuma, D. Massey, and L. Zhang, "BGP-RCN: improving BGP convergence through root cause notification," *Computer Networks*, vol. 48, no. 2, 2005.

[12] J. Chandrashekar, Z. Duan, Z.-L. Zhang, and J. Krasky, "Limiting path exploration in BGP," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 4, 2005.

[13] A. Bremler-Barr, Y. Afek, and S. Schwarz, "Improved BGP convergence via ghost flushing," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, 2003.

[14] W. Sun, Z. M. Mao, and K. G. Shin, "Differentiated BGP update processing for improved routing convergence," in *Proceedings of the 2006 IEEE International Conference on Network Protocols*, 2006.

[15] F. Wang and L. Gao, "Path diversity aware interdomain routing," in *IEEE INFOCOM 2009*, 2009.

[16] D. Walton, A. Retana, E. Chen, and J. Scudder, "Advertisement of multiple paths in BGP," RFC 7911, 2016.

[17] C. Villamizar, R. Chandra, and D. R. Govindan, "BGP route flap damping," RFC 2439, 1998.

[18] C. Pelsser, R. Bush, K. Patel, P. Mohapatra, and O. Maennel, "Making route flap damping usable," RFC 7196, 2014.

[19] A. García-Martínez, P. R. Torres, and M. Bagnulo, "BGP convergence in an MRAI-free Internet," *Computer Networks*, vol. 240, 2024.

[20] J. L. Sobrinho, "Network routing with path vector protocols: theory and applications," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2003, pp. 49–60.

[21] C.-K. Chau, R. J. Gibbens, and T. G. Griffin, "Towards a unified theory of policy-based routing," in *IEEE International Conference on Computer Communications*, 2006.

[22] I. van Beijnum, J. Crowcroft, F. Valera, and M. Bagnulo, "Loop-freeness in multipath BGP through propagating the longest path," in *2009 IEEE International Conference on Communications Workshops*, 2009, pp. 1–6.

[23] J. Moy, "OSPF version 2," RFC 2328, 1998.

[24] R. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle, "EIGRP-a fast routing protocol based on distance vectors," in *Networld/Interop 94*, 1994.

[25] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, 2010.