Enabling Policy Innovation in Interdomain Routing: A Software-Defined Approach

Anduo Wang*, Zhijia Chen*, Tony Yang†, and Minlan Yu‡
* Temple University, † Johns Hopkins University, ‡ Harvard University

ABSTRACT

BGP is known to restrict policy expressiveness and induce uncontrolled policy interactions that are hard to understand, reuse, and evolve. We argue that the use of a path vector system as the carrier of interdomain policies is the root cause of these limitations. To this end, we propose an alternative policy scheme built in a software-defined controller to decouple policy making from the path vector system. Rather than treating policies as hardwired attributes of a route, that are configured and consumed as the route goes through the path vector decision process, we let policies flow, interact, and combine to influence end to end routes. This new software-defined scheme creates new space for policy language, route decision, and conflict resolution design, towards more flexible policies, cleaner policy enforcement, and controlled policy interaction. As a realization of our vision, we present an implementation that uses data integrity constraints for representing and reasoning about routing policies, addressing unique challenges in the decentralized interdomain environment.

CCS CONCEPTS

• **Networks** **Network protocol design**; *Programming interfaces*; *Network manageability*;

KEYWORDS

BGP, Path-vector system, Exchangeable logic policy, SDN

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SOSR '19, April 3–4, 2019, San Jose, CA, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6710-3/19/04…$15.00
https://doi.org/10.1145/3314148.3314359

ACM Reference Format:
Anduo Wang*, Zhijia Chen*, Tony Yang†, and Minlan Yu‡. 2019. Enabling Policy Innovation in Interdomain Routing: A Software-Defined Approach. In *Symposium on SDN Research (SOSR '19), April 3–4, 2019, San Jose, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3314148.3314359

1 INTRODUCTION

Border gateway protocol (BGP) [24] is the de facto interdomain routing protocol, partly due to its unique ability to admit routing policies of the individual domains, also called autonomous systems, or ASes. In contrast to a pure distance-vector protocol that only allows shortest (distance) path policy, BGP adds attributes and steps to the path-vector decision process as a means to AS policies — an AS can configure the route attributes based on its own concerns and select paths based on those attributes. While the choice for the path vector system — due to its simplicity, scalability, and more — might have contributed to the popularity of BGP, it also limits BGP's policy expression.

One well known limitation is that BGP only allows policies influencing an upstream provider (farther away from the destination), but not the downstream. Another long standing limitation is about coordinating policies in a decentralized environment: policies set by multiple ASes not aware of each other can interact in a remote AS. Both limitations are rooted in the path vector system. The difficulty with controlling policies in a downstream is due to the unidirectional flow of policy information — carried by the route attributes — from the downstream to the upstream. The lack of policy coordination is because the path vector protocol was originally designed for intradomain under a universally agreed preference for shorter path, but BGP supports autonomy — independent ASes can set arbitrary route selection criteria that interact in unexpected ways.

These limitations impede policy innovation in BGP, making it hard to introduce new policies or predicate their interactions. Many extensions to BGP [9, 21, 22, 33, 34] were proposed to address these limitations via more flexible platforms — MIRO [33] can influence path selection in the downstream neighbors by inter-AS negotiation, Wiser [22] enables joint traffic engineering (TE) between ASes. Unfortunately, most

BGP extensions are incremental fixes to the path vector system. They are ad hoc and hardwired by nature, making them hard, if not impossible, to extend: For example, Wiser adds the `Wiser cost` attribute to be compared after `local preference` during route decision, forcing any non-TE policies to be expressed in `local preference`.

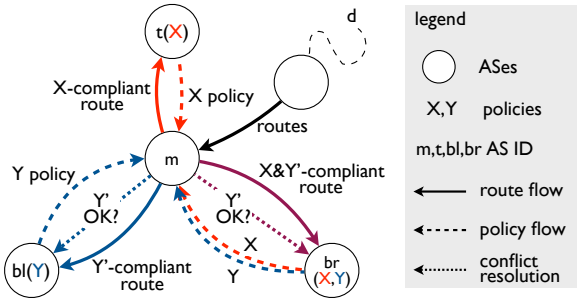

Figure 1: Interdomain policy system for innovation.

In this paper, rather than look for yet another path vector extension, we seek a policy system — via an SDN controller that oversees the entire AS — to completely separate policy from the path vector system. The key design choice is about the AS policies: in contrast to a path vector system where policies are hardwired route attributes configured and consumed as the route goes through the decision process, we make policies logic statements and allow them to be freely exchanged between the ASes via their SDN controllers.

Exchangeable logic policy permits more flexible policies, creating a new space for policy and route decision innovations. First, it immediately *eliminates the BGP restriction on control in the downstream ASes.* As shown in Figure 1, `AS t` (top) can simply expose to its downstream neighbor `AS m` (middle) a policy to influence the route selection at `m`. More importantly, logic is universally interpretable — the impact of a logic statement can be accurately derived. This *allows an AS to infer precisely how a policy influences its route selection, and to predicate how a policy interacts with its other policies.* Continuing with the above example, the controller `m` can now act upon policy `X` learned from `t` by calculating `X`'s impact on its routes; likewise, `m` can resolve conflicting policies `X,Y` — these policies can be independently set by two neighbors (`bl` (bottom left), `br` (bottom right)), or by the same neighbor with limited visibility into `m` — by computing the impact of `X` on `Y`.

But what is the "impact" of a policy? A plausible definition is the extra logic condition entailed by this policy if it takes effect. To illustrate the intuition, consider how policy `P` affects other policies like Q_1,Q_2 in the following. Also suppose that a route cannot be simultaneously `safe` and `fast`:

```
% P1: customer routes must be safe
cust(r) ⟹ safe(r)
% Q1: a customer routes must be fast
cust(r) ⟹ fast(r)
% Q2: admin routes must be fast
admin(r) ⟹ fast(r)
```

The impact of `P` is that `safe` must be included whenever `cust` applies. This impact, when taken into consideration for Q_1, will transform it to `cust(r)`$\wedge$ `safe(r)` $\Longrightarrow$ `fast(r)`, resulting in the logical contradiction of `safe(r)` $\Longrightarrow$ `fast(r)`, meaning that the success of `P` is guaranteed to produce the failure of Q_1. On the other hand, `P` does not cause any change to Q_2 — the two policies are independent. More formally, for a policy given by a logic statement, we define the impact of the policy as a fragment of the logic statement that interacts with the network state, the syntactic fragment that contains all the relevant information needed to process that policy. As will become clear in § 3.2, this notion of policy impact corresponds to the residue computed by the subsumption algorithm — a theorem proving technique for relating logic statements.

In the rest of the paper, we present *boléro*, a realization of our vision built on top of the Ravel controller [31], a controller that uses database relations as the network abstraction. *Boléro* relies on Ravel for efficient network control, but introduces database integrity constraints (ICs) [3, 4, 8] as the logic language for representing AS policies, and leverages semantic transformation — an application of subsumption that extracts and applies all the useful information in a policy — as the reasoning tool to coordinate policies. We also present preliminary evaluation of our *boléro* prototype, showing that the overhead in terms of database delay is small and scales well.

2 BARRIERS TO POLICY INNOVATION

We discuss two fundamental limitations to the interdomain routing system — BGP and its extensions — that hinder policy innovation. We use a toy (admittedly contrived) BGP network with MIRO [33] and Wiser[22] as illustrative examples. We also highlight the shortcomings of existing approaches to address those limitations.

2.1 BGP review

BGP is a direct extension of a path vector routing system to accommodate AS-defined policies. Figure 2 shows a simple BGP system: As routes — a path together with a list of configurable attributes — flow in the opposite direction of data traffic, from the downstream receivers (`AS1`, `AS2`) to the upstream senders (`AS4`, `AS5`, multi-homed with source `src`), routing policies are independently set and applied within every AS — when route attributes are set and compared. Take `AS3` as an example, after learning multiple routes from `AS1`,`AS2`, `AS3`

freely applies local policies — e.g., hot-potato routing policy that picks a "shortest" path with least local cost (highlighted in black) — to select and announce to AS4, AS5 one best path.

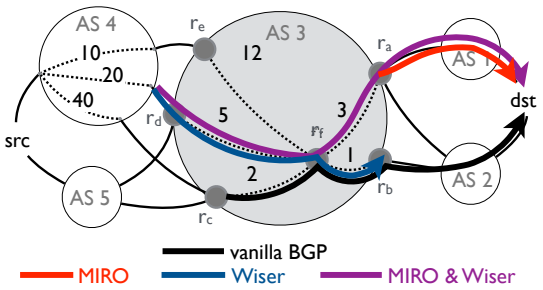

Figure 2: End to end paths determined by local policies of individual ASes: the lines highlighted in color depict the route selected by the corresponding policies. The vanilla BGP policy is hot-potato routing within AS3

BGP, as a path vector policy system, carries policy control attributes (e.g., local AS-defined policies) in the routes and compare them in a prefixed order for path selection. This scheme prevents the upstreams to influence routing in the downstreams, due to the unidirectional policy (route) flow. Besides, path vector system provides little leverage for understanding or coordinating the interactions among locally set AS policies, leading to increasingly complex monolithic policies.

We argue that the BGP policy scheme that couples policy making with the path vector system is fundamentally flawed, it restricts policy expressiveness and induces monolithic policies with uncontrolled interactions.

2.2 Controlling policy in downstream ASes

Some policies require upstream ASes to control downstream ASes. Consider AS4 (or AS5) and its downstream neighbor AS3. AS4 may want to influence route selection at AS3 to get access to alternative routes because the default path (in black) announced by AS3 is not compliant with its local concerns: AS4 may favor a route announcement (in red) that avoids a certain AS2 that it considers harmful, or prefer another alternative (in blue) that improves overall path quality between AS4 and itself (the numbers inside AS 3 and AS 4 represent the internal costs of carrying traffic along the subpaths).

Some BGP extensions can enable AS4 to address the concerns discussed above. For example, MIRO allows AS3 to negotiate a path bypassing AS2 by exchanging with AS4 additional MIRO control messages, which can also carry (optional) path properties that guide route selection at AS3. On the other hand, Wiser extends BGP with an additional normalized path cost attribute that enables AS4 to share with AS3 the sensitive information of Internet path cost, which guides coordinated route decision at AS3. While both extensions introduce advanced route state exchanges beyond vanilla BGP, they are ad hoc solutions for specific problems.

Question 1. Can we build a policy scheme that allows for controlling routing in the downstream ASes?

2.3 Coordinating monolithic policies with partial knowledge

While BGP and its many extensions were primarily designed to promote autonomous policy making within ASes, to fully realize this potential, it is sometimes beneficial and even indispensable to coordinate policies across AS borders. Suppose AS4 wants to access a path from AS3 that is *simultaneously* secure — avoiding AS2, and optimal — minimizing cost between itself and AS3. Can the operator of AS4 simplify the local policy configuration by combining two arguably simpler MIRO and Wiser policies as discussed in the preceding section? As a second example, suppose it is AS4 that employs Wiser for path quality, and AS5 that uses MIRO to avoid AS2. With AS4 and AS5 multi-homed with src, the MIRO and Wiser policies — both affecting route selection at AS3 — will interact and may run into conflict. Can AS5 and AS4 predicate how this interaction might affect their local policy? Can they detect and resolve potential conflicts?

Coordination is at odds with interdomain systems by nature: On one hand, in a decentralized environment, policies created at one or multiple ASes can take effect and interact at a different location; On the other, the decentralized environment causes the policy makers to minimize the local information they reveal. The lack of visibility into neighbor ASes makes it inherently hard to understand and anticipate the outcome of policies, let alone to properly coordinate them.

In the above example, what makes it hard for AS4 to combine MIRO and Wiser is that the two policies will interact at AS3 — simply concatenating the MIRO subpath and Wiser subpath known to AS4 will not give the right composition. A plausible composite path (highlighted in purple) depends on the understanding that the MIRO and Wiser policies' subpaths must interconnect at the same egress router of AS3 (r_a), that the two policies conflict — the subpath favored by MIRO (in red) is less preferred by Wiser. But such information is not available to AS4. For the second example, it is even more challenging as AS4 is not even aware of AS5 (and vice versa).

Question 2. Can we create a coordinated policy environment that promotes the composition of independently set BGP policies.

3 *BOLÉRO* ARCHITECTURE

Motivated with the analysis in § 2, we propose *boléro*, an alternative policy scheme that completely separates AS policies from the path vector system. *Boléro* makes AS policies

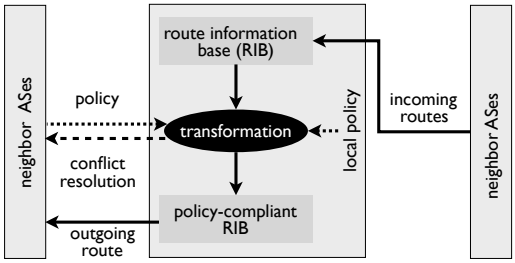

Figure 3: *Boléro* Architecture.

exchangeable logic statements. An important consequence is that the impact of a policy — how the policy influences the route decision process or another policy — can be *accurately computed*, via a semantic transformation process in which all the relevant information of a policy is extracted and processed. *Boléro* makes use of such relevant fragment of a policy in two ways: when used to transform routes, *boléro* gives a generic policy-compliant route selection process that admits disparate policies — addressing *Question 1.(§ 2)*; when applied to transform a second policy, *boléro* offers a conflict resolution mechanism that allows the two policies to co-exist — addressing *Question 2. (§ 2)*.

More concretely, as shown in Figure 3, central to *boléro* is the semantic transformation process that, taking as inputs routes — stored in the route information base (RIB), and policies — either local or learned from a neighbor, checks for policy conflicts. In the presence of a conflict between two policies, the human operator is involved to provide the policy priority, and *boléro* is invoked to transform the low-priority policy to be compliant with the high-priority one. The resulting compatible policies, through similar semantic transforms of the routes, are used to generate a set of policy-compliant RIB (Routing information based).

As a realization of this idea, we implement *boléro* as a policy layer atop the Ravel [31] controller which oversees the entire AS, learns external routes, and disseminates route decisions. Ravel represents the entire network state as logic facts, making it particularly convenient for implementing *boléro* policies. In the rest of the section, we highlight the policy features that *boléro* enhances Ravel.

3.1 Representation

This section presents a database representation for the routing state — routing state as factual data (tables or tables) and routing policies as non-factual data (semantic information or integrity constraints).

Routing state as queryable tables. Following the Ravel convention, routing state is organized into *queryable tables*. Suppose `AS3` (in Figure 2) maintains a `route(destination, next hop, AS path)` table to represent the AS-level routes. When exposed to neighbors, a neighbor like `AS4` can "pull" routes to a particular destination `d` by a query, either in the form of a SQL statement or, equivalently, a datalog rule:

```
--- SQL form
SELECT * FROM route, WHERE route.prefix='d'.
--- rule form
route('d',N,P) :- route(D,N,P) ∧ D='d'.
```

We adopt the rule form for its concise syntax and clear logical meaning. A rule can be read as a logical implication — the body predicates on the right hand side of `:-` implies the head predicate on the left.

Routing policy as integrity constraint

boléro introduces the use of data integrity constraint (IC) as the unified representation for policies. An IC is a statement about what are the legal data — policy-compliant routing state. We use a particular IC form called *denial rule* which eliminates any routing state that violates the policy. A denial rule is a headless rule — empty head means `False` — meaning that the body predicates cannot be simultaneously true. For example, the MIRO [33] policy of `AS3` is an IC of the following form, saying that a route with a path that contains `'AS2'` results in a violation, thus must be eliminated.

```
:- route(D,N,P), ('AS2' in P).
```

We show the strength of the IC representation with the Wiser policy of `AS3`. This policy selects path that jointly improves cost between `AS3` and `AS4`. Suppose `AS3` maintains a `Wiser(D,R,C)` table to represent the Wiser (normalized) cost `C` via a peering router `R` to destination `D`, `AS4` exposes to `AS3` its Wiser cost by `Advertise(R,C)`, advertising the Wiser cost `C` through a peering router `R`. With `Advertise` and `Wiser`, the Wiser policy only needs to exclude from consideration any path whose total path cost is not the lowest.

```
:- Wiser(D,R,C),Advertise(R,C2),
   Wiser(D,R',C'),Advertise(R',C2'),C+C2>C'+C2'.
```

3.2 Semantic transformation

The basic idea is to use a theorem proving technique called *partial subsumption* to anticipate the impact of a policy, the relevant fragment that actually interacts with routes or another policy. This fragment is called *residue*. When syntactically attached to the query that represents the routes or policy, the residue transforms the query into a policy-compliant one, representing policy-compliant outgoing routes, or conflict-free policies, respectively.

Subsumption. Given two queries `P,Q`, `P` subsumes `Q` if there exists a substitution that makes `P` a subclause of `Q`. When neither of `P,Q` is an IC, subsumption means answers to `P` subsume (is a superset of) the answers to `Q`. Consider R_1 that generates all per-destination (`D`) AS paths (`Y`), R_2 that produces

a restricted query for AS paths (P) through AS2, R_1 subsumes R_2 by the substitution D=I,Y=P.

```
R1 ans1(D,Y) :- route(D,X,Y).
R2 ans(I,P) :- route(I,R,P), ('AS 2' in P).
CM :- route(I,R,P), ('AS 2' in P).
```

When one of the query is a policy, the subsumption of the query by the policy is suggestive of an empty answer. For example, C_M — the MIRO policy of AS3 discussed in § 3.1 — subsumes R_2, meaning that the path selected by the query will always violate the MIRO policy. When both queries are policies, the subsumption of one policy by another implies generalization — the subsuming policy is more general and stronger.

Partial subsumption and residue. What interests us is the *partial subsumption* of a query by a policy, when the query is only subsumed by a subclause of the policy, meaning that the policy will interact with the query, affecting route selection or policy represented by the query. (Partial) subsumption can be determined by the subsumption algorithm developed in [4], producing the corresponding policy residue as a side product. More specifically, we run the subsumption algorithm on the policy IC and the query, attempting to refute the policy by constructing a refutation tree with the policy as the root and elements from the query for resolution. The success of refutation proves subsumption; The failure of refutation signals partial subsumption — at the bottom of the refutation tree, a fragment of the policy must be left. Such fragment is the *residue* we look for.

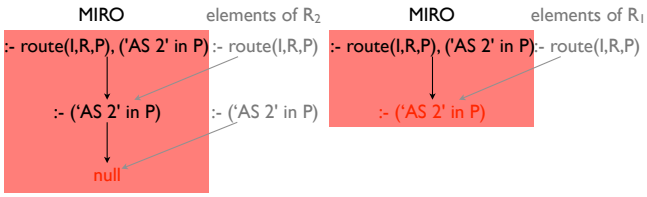

Figure 4: Refutation tree and the residue (in red).

For example, the refutation tree in Figure 4 (left) proves that the MIRO policy subsumes R_2, whereas the refutation tree on the right implies that MIRO partially subsumes R_1. More importantly, the residue `:- ('AS2' in P)` says that the MIRO policy constrains R_1 in the sense that `¬('AS2' in P)` must be taken into account — only routes compliant with this extra condition will be selected.

Route- and policy- transformation

As discussed in the above, running the subsumption algorithm over a policy and a query gives the residue that predicates the impact of the policy on the query. When the query is route selection, the residue prescribes the extra conditions that must be included for picking policy-compliant routes. When the policy is another policy, the residue is the conditions that must be added for the other policy to be compliant. Thus to apply a policy we only need to attach the corresponding residue.

As a first example, consider route transformation that produces policy-compliant outgoing route: To apply the MIRO policy to route selection in R_1, we only need to attach the MIRO condition (in red).

```
R1' ans(D,Y) :- route(D,X,Y), ¬('AS 2' in P).
```

Likewise, to apply the Wiser policy, we obtain the Wiser residue.

```
R1'' ans(D,Y) :- route(D,X,Y), ¬(Wiser(D,X,C)∧Wiser(D,X',C')
    ∧Advertise(X,C2)∧Advertise(X',C2')∧C+C2>C'+C2').
```

Next, consider two policies that conflict, it must be the case that one partially subsumes another, and vice versa. Moreover, provided one with high priority, applying the residue of the higher-ranked policy to the less important will transform the less important to be compliant, thus resolving conflicts. For example, the MIRO policy partially subsumes the Wiser policy with the residue `:- ('AS2' in P)`. Suppose MIRO is valued over Wiser, by annotating the Wiser policy with the MIRO residue condition `¬('AS2' in P)`, we get the MIRO-compliant Wiser policy.

```
-- Wiser policy constrained by MIRO
:- Wiser(D,R,C),Advertise(R,C2),Wiser(D,R',C'),
   Advertise(R',C2'),C+C2>C'+C2',¬('AS2' in P).
```

4 PRELIMINARY EVALUATION

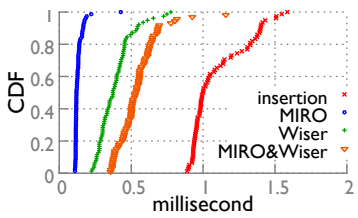

Figure 5: *Boléro* overhead.

Our evaluation result with the *boléro* prototype is promising: the database delay is small (<1millisecond in most cases) and scales well. All experiments were performed on the macOS platform with a 3.4 GHz Intel Core i5 processor and 16 GB 2400 MHz DDR4 RAM.

Experiment setup. We use Routeviews BGP feeds [30] as incoming routes and time the delay to determine the outgoing routes with the residue method under three policy scenarios: the stand-alone MIRO and Wiser policies, and the composition of the two as described in § 3.2. For all policies, we use a common network setup: We embed a Rocketfuel ISP topology annotated with (multiple) peering interconnections and inferred internal costs [28] into a Skitter AS topology . To run the Wiser policy between a pair of ASes, we pick two well-connected ASes that are present in both datasets. For the MIRO policy of the form `avoid an AS X to destination d`,

we randomly select the destination prefixes and ASes in the Skitter topology.

Preliminary result. Figure 5 shows the database delay incurred for *Boléro* route decision. We measured the delay for 10,000 BGP feeds and plotted 100 randomly picked data points. We break the delay into two parts: the `insertion` delay for route table updates and the delay for applying a particular policy — one of `MIRO, Wiser, MIRO&Wiser`. The insertion delay does not vary with different policies, we only plot insertion for MIRO. All operations complete fast — 95% insertion takes <1.424ms, 95% of the policy application finishes in <0.174ms (MIRO), 0.640ms (Wiser) and 0.844ms (MIRO&Wiser) respectively. The dominating delay is insertion, indicating that the pure policy processing delay is small. Among all the policies, consistent with our expectation, the simpler MIRO finishes first, while MIRO&Wiser takes more time than either MIRO or Wiser. All the delays scale well as route table size grows.

5 RELATED WORK

SDN for wide area networks. Several works have proposed to simplify BGP management with SDNs at various attractive points: *Within an AS*, RCP [1, 10, 25] proposes to decouple routing from the routers, leveraging a centralized controller to select the BGP routes for each router; *At a single Internet Exchange point*, SDX [12, 16] proposes more flexible route and traffic control, where it has the central role in interconnecting many networks. *Boléro* leverages SDN for a different purpose, to realize an inter-AS policy system for the entire Internet.

Database usage in networking. Declarative networking [5, 6, 17–20, 23] and modern SDN control platforms [2, 7, 13–15] demonstrated the many benefits of using database techniques — declarative specification with database languages, strong consistency with transaction processing, and more — for managing *factual* network data. By contrast, *boléro* uses database for semantic network data — policies.

Interdomain routing policies. Many BGP-like protocols [9, 11, 21, 22, 29, 32–34] were proposed in the past to enable more flexible policies. More recently, D-BGP [26, 27] examined the simultaneous partial deployments of these protocols, uncovering the architectural features needed to allow their co-existence on the global Internet. *Boléro* shares with these efforts the goal of diversifying policies, but uses an SDN-based policy system to enable policies previously supported by multiple routing platforms to co-exist within an AS.

6 DISCUSSION

Deployment. Many proposals for changing interdomain routing fail because in order to get any benefit for the change, all or almost all ASes must adopt the new technology. Because of this, there is no particular incentive for anyone to go first and we remain stuck at the status quo. In contrast, *boléro* can offer immediate advantages to any two adjacent peers who agree to adopt the system. *Boléro* makes informed neighboring negotiations, enabling mutually beneficial route exchange — the route sending neighbor gets new revenue for better serving the route receiver. Such bilateral agreements are much easier to reach and do not require large groups of ASes to agree simultaneously on a new scheme and its implementation.

Benefits. By admitting policies that are only previously supported on many BGP-like protocols, *boléro* can benefit and attract users with similar concerns. The cost incurred by *boléro*, however, is higher: *boléro* is a clean slate design that completely discards the path-vector system, whereas most BGP-like proposals that value backward compatibility tend to introduce new policies via minimal modification to the path-vector system whenever possible. But *boléro* also provides unique benefits: the logic-based policy abstraction serves as a common currency for policy interoperation with automated-reasoning support.

Limitations. *Boléro* will suffer from any limitation inherent to the policies it support — if a policy requires the negotiation beyond immediate neighbors so does *boléro*. In addition, we note a significant challenge introduced by *boléro* itself: BGP and its enhancements took great labor to anonymize policies which are considered sensitive properties, but *boléro* requires explicit exposure of AS policies. To this end, we plan to explore the generalization of logical clauses — analogous to aggregation — as a means to policy opaqueness.

7 CONCLUSION

This paper presents *boléro*, a software-defined policy system for interdomain routing that separates routing policies from the path vector system. Rather than burying policies in the route attributes that are manipulated by the prefixed path vector decision process, *boléro* uses the Ravel controller for efficient network control, introduces logic integrity constraints [3, 4, 8] as a unifying abstraction for representing AS policies, and makes use of a generic logical transformation process to manage the logic policies, addressing unique requirements for the interdomain.

Acknowledgments. We thank the SOSR reviewers and our shepherd David Walker for their valuable feedback, Rui Miao and Ramesh Govindan for initial discussion of the ideas in this paper. This work is supported in part by the National Science Foundation Awards CNS-1657285 and CNS-1413972.

REFERENCES

[1] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *NSDI*, 2005.

[2] M. Casado, N. Foster, and A. Guha. Abstractions for software-defined networks. *Commun. ACM*, 57(10):86–95, Sept. 2014.

[3] U. S. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst.*, 15(2):162–207, June 1990.

[4] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 1997.

[5] X. Chen, Z. M. Mao, and J. van der Merwe. Towards automated network management: Network operations using dynamic views. In *Proceedings of the 2007 SIGCOMM Workshop on Internet Network Management*, INM '07, pages 242–247, New York, NY, USA, 2007. ACM.

[6] T. Condie, J. M. Hellerstein, P. Maniatis, S. Rhea, and T. Roscoe. Finally, a use for componentized transport protocols. In *In HotNets IV*, 2005.

[7] B. Davie, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. Gude, A. Padmanabhan, T. Petty, K. Duda, and A. Chanda. A database approach to sdn control plane design. *SIGCOMM Comput. Commun. Rev.*, 47(1):15–26, Jan. 2017.

[8] P. Godfrey, J. Grant, J. Gryz, and J. Minker. Integrity constraints: Semantics and applications. In *Logics for Databases and Information Systems*, 1998.

[9] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. Pathlet routing. In *ACM SIGCOMM*, 2009.

[10] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 2005.

[11] T. G. Griffin, A. D. Jaggard, and V. Ramachandran. Design principles of policy languages for path vector protocols. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 61–72, New York, NY, USA, 2003. ACM.

[12] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett. Sdx: A software defined internet exchange. In *SIGCOMM*, 2014.

[13] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker. Fml: Practical declarative network management. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, pages 1–10, New York, NY, USA, 2009. ACM.

[14] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, N. Gude, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang. Network virtualization in multi-tenant datacenters. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 203–216, Berkeley, CA, USA, 2014. USENIX Association.

[15] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: a distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, 2010.

[16] V. Kotronis, X. Dimitropoulos, R. Klöti, B. Ager, P. Georgopoulos, and S. Schmid. Control exchange points: Providing QoS-enabled end-to-end services via sdn-based inter-domain routing orchestration. In *ONS 2014*.

[17] C. Liu, B. T. Loo, and Y. Mao. Declarative automated cloud resource orchestration. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, pages 26:1–26:8, New York, NY, USA, 2011. ACM.

[18] C. Liu, L. Ren, B. T. Loo, Y. Mao, and P. Basu. Cologne: A declarative distributed constraint optimization platform. *Proc. VLDB Endow.*, 5(8):752–763, Apr. 2012.

[19] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative networking: Language, execution and optimization. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 97–108, New York, NY, USA, 2006. ACM.

[20] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative routing: Extensible routing with declarative queries. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 289–300, New York, NY, USA, 2005. ACM.

[21] R. Mahajan, D. Wetherall, and T. Anderson. Negotiation-based routing between neighboring isps. In *NSDI*, 2005.

[22] R. Mahajan, D. Wetherall, and T. Anderson. Mutually controlled routing with independent ISPs. In *NSDI*, 2007.

[23] Y. Mao, B. T. Loo, Z. Ives, and J. M. Smith. Mosaic: Unified declarative platform for dynamic overlay composition. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 5:1–5:12, New York, NY, USA, 2008. ACM.

[24] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, 2006.

[25] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha de Lucena, and R. Raszuk. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In *HotSDN*, 2012.

[26] R. R. Sambasivan, D. Tran-Lam, A. Akella, and P. Steenkiste. Bootstrapping evolvability for inter-domain routing. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, pages 12:1–12:7, New York, NY, USA, 2015. ACM.

[27] R. R. Sambasivan, D. Tran-Lam, A. Akella, and P. Steenkiste. Bootstrapping evolvability for inter-domain routing with d-bgp. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 474–487, New York, NY, USA, 2017. ACM.

[28] N. Spring, R. Mahajan, and T. Anderson. Rocketfuel: An isp topology mapping engine. http://research.cs.washington.edu/networking/rocketfuel/.

[29] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. Hlp: A next generation inter-domain routing protocol. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 13–24, New York, NY, USA, 2005. ACM.

[30] R. Views. Route views. http://www.routeviews.org/routeviews/.

[31] A. Wang, X. Mei, J. Croft, M. Caesar, and B. Godfrey. Ravel: A database-defined network. In *SOSR*, 2016.

[32] Y. Wang, I. Avramopoulos, and J. Rexford. Design for configurability: Rethinking interdomain routing policies from the ground up. *IEEE J.Sel. A. Commun.*, 27(3):336–348, Apr. 2009.

[33] W. Xu and J. Rexford. MIRO: Multi-path interdomain routing. In *ACM SIGCOMM*, 2006.

[34] X. Yang, D. Clark, and A. W. Berger. Nira: a new inter-domain routing architecture. *IEEE/ACM Trans. Netw.*, 15(4), 2007.