Flexible Routing with Policy Exchange

Bin Gui, Fangping Lan, Anduo Wang
Temple University
Philadelphia, USA

ABSTRACT

BGP and its alternatives alike, struggle with distributed policy making in the absence of a central authority: BGP prioritizes independence of the participating networks (e.g., ASes), imposes zero coordination, but has to tolerate inflexible policies each network can express. On the other hand, BGP alternatives (source routing, for example), through coordination, trade independence for flexibility, but only achieve flexibility partially. This paper asks, to achieve flexible routing, what is the fitting adjustment between network independence and coordination? To answer this question, we propose a simple principle that the sole end to interfere with the flexibility of a participating network is to prevent harms — decreasing the level of flexibility — to others. As an instantiation of this principle, we introduce the concept of policy exchange that dynamically adjusts independently set policies on the fly, and develop a preliminary implementation with conditional table, a strong knowledge representation system that allows us to distribute and manipulate policies with the usual SQL-like operators. Our preliminary experiments on realistic network topology and synthetic policies are encouraging.

CCS CONCEPTS

• **Networks** → **Routing protocols**; *Network management*; • **Computing methodologies** → *Reasoning about belief and knowledge.*

KEYWORDS

Policy exchange, interdomain routing, conditional tables, knowledge representation

ACM Reference Format:
Bin Gui, Fangping Lan, Anduo Wang. 2021. Flexible Routing with Policy Exchange. In *5th Asia-Pacific Workshop on Networking (APNet 2021) (APNet 2021), June 24–25, 2021, Shenzhen, China, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3469393.3469395

APNet 2021, June 24–25, 2021, Shenzhen, China, China

ACM ISBN 978-1-4503-8587-9/21/06…$15.00
https://doi.org/10.1145/3469393.3469395

> …how to make the fitting adjustment between individual independence and social control … That principle is, that the sole end … in interfere with the liberty of [any of their member], is self protection. That the only purpose is … to prevent harms to others.
>
> *John Stuart Mill*, On Liberty

1 INTRODUCTION

The Internet consists of independently operated networks (i.e. autonomous systems, or ASes) that use dissimilar policies to collectively drive global routing. It is probably the most important instance of — in the absence of a central authority — distributed policy making [1, 2, 8, 12, 33, 35, 36]. And Border Gateway Protocol (BGP) [34] is the one and only routing protocol that supports such policies.

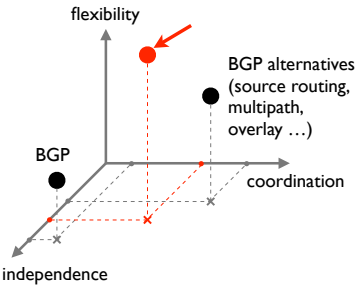

Figure 1: To achieve flexibility, what is the fitting adjustment between network independence and coordination?

One reason that contributes to BGP's unique role is the extreme position taken by BGP that aligns itself with the network owners: Route preference can be arbitrary and set solely based on local metrics; only the best route used for packet forwarding is re-districted to selected neighbors from which packets are allowed; only the route itself is exposed while the policies governing the routing process are sensitive information, thus are hidden from the external Internet. Together, these designs make the operation of each AS independent from the rest of the Internet, and the AS policies are affected only by direct neighbors; But the same design also makes distributed policies inflexible: the network edge and the transit ASes alike have little control over traffic path, their influences are often limited to the first hop. As summarized in Figure 1, BGP prioritizes individual independence, tolerates routing inflexibility, and refrains from remote coordination.

In contrast, many attempts to improve and/or replace BGP strive for more flexible policies, and do not shy away from coordination. Source routing [6, 16, 39, 40] allows the source AS to control the entire path, multi-path routing [14, 21, 38] improves transit AS's visibility to available paths via multiple routes computation, overlay routing [5, 21] combined

with source and/or multi-path routing harnesses the flexibility of those alternatives on selected participants. These flexibility enablers, however, are all at odds with AS independence in some sense [1]: Source routing forces the transit ASes to expose their policies in a universally identical manner (differentiating policies for different sources are impossible), multi-path routing burdens the transit nodes to maintain routes that are not their first choice; and overlay routing imposes a form of virtual circuit service model into the participants. Besides, most of these schemes improve flexibility only at selected participants with best effort: Source routing enhances the sources with more choices at the cost of the transits; multipath routing improves path diversity at the upstream nodes at the cost of the downstream; and overlay links still rely on the underlying routing (potentially over BGP). In sum, these BGP alternatives, through coordination, trade independence for flexibility, but only achieve flexibility partially.

In light of the fundamental tradeoff in the foregoing discussion, this paper asks, *to achieve flexible routing, what is the fitting adjustment between network independence and global coordination? To answer this question, we argue for a simple principle that the sole end to interfering with the flexibility of a network is to prevent harms — decreasing the level of flexibility — to others.*

The principle implies that a flexible routing system should support all legitimate policies — those can be satisfied along some existing paths while not hindering others. But existing routing schemes, to our best knowledge, fail to admit all legitimate policies. As a first step towards realizing our principle, we study and map the failure scenarios into two categories: Depending on the type of (unwanted) policy interactions that leads to the failure, the first category involves an overly strong policy that perfectly matches a node's local concern but unnecessarily excludes policies available to other nodes along the same path affecting overlapping traffic. In the second category, seemingly unrelated policies affecting disjoint traffic, while legitimate on their own, can mutually exclude each other if they cross a common intermediate node. In both cases, the solution is to discover the "right" policy that firmly expresses local interests, but also adapts (generous) to others' needs — including those over overlapping traffic, and those referencing a shared critical node — as much as possible. Our goal is to make such intelligent policy discovery happen by exposing minimal policy information.

Specifically, we introduce policy exchange in which independently set policies are iteratively exchanged between direct neighbors, and adjusted as needed, in a hop by hop manner like classic distance vector algorithm: Each participating network, upon receiving a policy fragment announcement from a direct neighbor — the announcement represents that neighbor's request to protect its internal concern that potentially depends on external decisions, decides whether to honor the request in the fragment: if the current node determines to support the neighbor's concern, but figures that it alone cannot guarantee the fragment within local means (e.g., tweaking local preference or route exporting policies), the current node will then re-distribute the fragment — with transformation to reflect its role as well as concealing the identify of the original sender. The hope is that, by distributing only necessary policy fragments, policy exchange will retain the network independence we hold dear, as well as give sufficient visibility into the rest of the Internet to locate the right policy.

To show that policy exchange is technically feasible, we present a preliminary implementation with conditional tables [3, 4, 20], a knowledge representation system originated in the database community. Conditional table enhances traditional tables with variables and tuple conditions, making it a powerful knowledge representation for policies, allowing policies to be passed around like regular factual data. More importantly, conditional tables allow the usual SQL-like data query and manipulation (select, projection, join etc.), making it a convenient vehicle for rapid prototyping of policy exchange mechanisms (e.g., policy fragment generalization and redistribution). We also note that policy exchange will not magically remove all difficulties with distributed policy making: Policy exchange will not move policy enforcement to the most fitting location hence cannot be used to suppress the impact of finer policies; and it does not detect or resolve conflicts among simultaneously unsatisfiable policies. Nevertheless, our initial implementation is encouraging, and we hope that policy exchange, with some luck, may infect the design of a more flexible future Internet.

2 A SIMPLE PRINCIPLE (EXPLAINED) BY FAILURES

Our main thesis is that a policy should be admitted as long as it does not harm others, that is a path p that satisfies the policy exists and that the selection of p does not eliminate other policies. We say that a policy is legitimate if it is admittable. To guide our design towards a flexible routing system (detailed in § 3) that permits all legitimate policies, we analyze the scenarios when an inflexible routing system fails. We map the failures into two categories depending on the type of interaction that leads to the failure.

Policies affecting shared traffic along a single path.

The purpose of a policy is to codify a node's (whether at the granularity of an AS or a router) own traffic concerns (service requirement or resource restriction). Yet an overly strong policy statement, though perfectly captures the local interest, may rob the choices available to other nodes along the same path. When that is the only path that simultaneously satisfies all the nodes along it, an overly strong policy placed by one node effectively makes it impossible to admit a legitimate

[1] There are other arguments against these alternatives in terms of scalability and dataplane support etc., but this paper focuses on the policy issues.

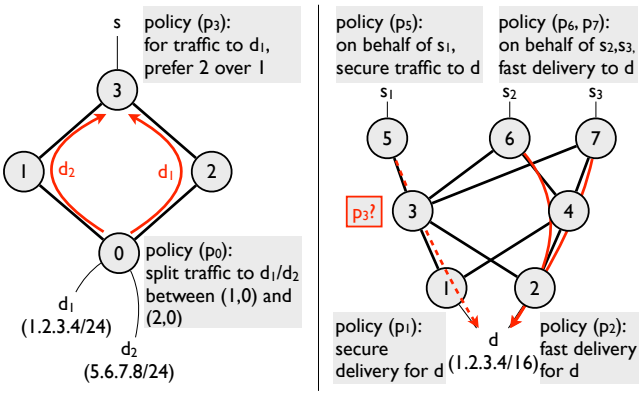

Figure 2: Inflexibility scenarios mapped into two categories: (left) overly strong policy (p_0) unawares of others eliminate otherwise legitimate policies (p_3) along the same path over shared traffic, (right) multiple policies that are seemingly independent can be mutually exclusive if they are realized on paths that cross a common "critical" node.

policy of another node. This reduces the "overall" level of flexibility of the entire routing system.

For example, Figure 2 (left) depicts four ASes (nodes `0,1,2,3`) that compute paths to carry traffic between a (source) host `s` and two (destination) prefixes d_1, d_2. `AS3`'s policy p_3 (on behalf of `s`) desires traffic destined to `d1` to go through the more preferred provider `AS2`. p_3 is satisfiable: there exists a path `[320]` that not only complies with this restriction, but is also compatible with `AS0` whose policy p_0 balances traffic to d_1, d_2. Whether p_3 can actually be enforced or not, however, depends on how p_0 is implemented — to split incoming traffic `AS0` can either instruct d_1 traffic to use link `(0,2)` (d_2 traffic on `(0,1)`), or instruct d_1 to use `(0,1)` (d_2 on `(0,2)`). Among the two options, only the former will expose path `[20]` to `AS3`, which in turn selects `[320]` and implements p_0. Lacking visibility into of the upstream policy p_3, `AS0` may blindly pick the "wrong" implementation, announcing d_1 to `AS1` (d_2 to `AS2`), and excluding the otherwise p_3-compliant path `[320]`.

BGP is known to suffer from such inflexibility, and source routing only address this problem partially: while source routing allows `A0` to expose p_0 to the source — `AS3` — where route selection is made, it does not prevent p_3 from making the overly strong policy that uses (0,2) for d_1 (d_2 traffic on (0,1)). That is, source routing allows route decision at the source, but does not facilitate the downstream ASes to arrive at the right policy that increases choices at the source. *To accommodate every legitimate policy, the routing system needs to facilitate individual AS to arrive at "informed" policies that not only express its own concern, but are also aware of others — permitting as wide a latitude as possible in the construction and enforcement of policies at other ASes.*

Policies affecting disjoint traffic crossing a common node.

Policies for disjoint traffic along separate paths can still inadvertently interfere with each other. When the policy compliant paths cross a common node, policy made at that node to accommodate one flow of traffic may lead to the rejection of another flow. Hence, the nodes at the potential "joint" must carefully craft their policies to simultaneously admit all the flows (and the policies that defined them).

Consider the three source nodes (`s1,s2,s3`) and a destination prefix (d) in Figure 2 (right): Policy p_5 demands secure traffic delivery via some downstream `AS1` (a firewall is probably deployed at `AS1`), p_6, p_7 require faster delivery via `AS2`. The network can simultaneously accommodate all these policies with the routes highlighted — the dashed and solid arrows depict routes selected for traffic from `s1` and `s2/s3`, respectively. This route selection, however, requires `AS3` to make a proper decision p_3, taking into account the traffic requests from all the hosts (`s1,s2,s3`), that is, honoring p_5 but discarding `p6` and `p7`. `AS3` can arrive at this intelligent decision `p3` only if `AS3` has the knowledge that `p5` can be implemented only if it is accepted at `AS3` — perhaps by charging `AS5` with an extra fee, and that p_6, p_7 can still be satisfied even if they are rejected at `AS3`.

BGP was not designed for joint policy routing as discussed above; Source routing only improves the level of flexibility at the sources, and multi-path routing allows multiple policies at a single AS (e.g.,`AS3`), selecting different routes for each upstream neighbor (e.g.,`AS5`, `AS6`), but neither addresses policy making at the potential joints because policy routes are still computed in isolation. *To accommodate multiple legitimate policies that are related because they rely on paths that cross a common node, the routing system should facilitate that common node to arrive at an "intelligent" decision that jointly allocates routing resource — accommodating as many legitimate policies as possible.*

3 A CASE OF POLICY EXCHANGE

This section develops policy exchange which instantiates the do-no-harm principle in § 1, and addresses the failures analyzed in § 2. In policy exchange, policies are no longer fixed term embedded in the route computation process, rather, they become dynamic entities that can be adjusted to fine tune their impacts on others. To capture the "policy impacts", we introduce the notion of policy fragment, a portion of an AS policy that cannot be guaranteed within the local AS. A policy fragment codifies what makes an AS vulnerable to the decision by others, it is thus propagated hop by hop until it is either (accepted and) incorporated into some remote AS's policy decision, or is explicitly rejected. The hope is that, by providing an exchange platform that gives individual AS better visibility into how their local decisions affect others, they will make better informed policies that raise flexibility for everyone.

3.1 A Policy Exchange Protocol

Like routing information exchange, policy exchange is implemented at each policy-based routing entity. Each node

step	policy exchange	p_1	p_2	p_3	p_4	p_5	p_6	p_7
0	5→3: secure	secure	fast	⊤	⊤	secure	fast	fast
1	6(7)→3: fast; 6(7)→4: fast	secure	fast	secure	⊤	secure	fast	fast
2	3→5: secure(2); 3→6: fast(1); 3→7: fast(1)	secure	fast	secure(2), fast(1)	fast	secure	fast	fast
3	6→3: ¬fast(1); 6→4: ¬fast(1)	secure	fast	secure(2), fast(1)	fast	secure(2)	fast@4, ¬fast(1)@3	fast@4, ¬fast(1)@3
4	7→3: ¬fast(1); 7→4: ¬fast(1)	secure	fast	secure(2), fast(1)	fast	secure(2)	fast	fast@4, ¬fast(1)@3
5		secure	fast	secure(2)	fast	secure(2)	fast	fast

Table 1: One possible trace of policy exchange for the network in Figure 2 (right).

Algorithm 1 Policy Exchange at i

1: **if** $e_i(p_i) == false$ **then** // p_i can be enforced within i
2: **for** each neighbor k **do**
3: $p_i^k \leftarrow f_k(p_i)$; // policy fragment relevant to k
4: send p_i^k to k;
5: **loop** (**wait** until i receive p_k^i from some neighbor k)
6: **for** each p_k^i **do**
7: $p_i \leftarrow m_i(p_i, p_k^i)$; // merges p_k^i with local policy
8: **if** $e_i(p_i) == false$ **then**
9: $p_i^j \leftarrow f_k(p_i)$ for every neighbor j;
10: send p_i^j to j;

receives policy fragments from one or more of its directly attached neighbors, performs a calculation, and then distributes the results of the calculation back to its neighbors: the policy fragment specifies a service request (or a resource restriction) placed on the receiving node by the sending neighbor; the calculation estimates the impact of the request (resource), the results of which represent what cannot be realized within the receiving node — requiring support (compliance) from other neighbors, and hence must be further distributed.

Specifically, as depicted in Algorithm 1: In line 1-4, node i initializes its policy fragments — local request (or restriction) — that need to be exposed for realization; The fragment p_i^k is calculated by f_k specific to each neighbor k, meaning that the impact of the same local concern needs to be instantiated in the context of each neighbor. In line 5-10, node i iteratively exchanges policy fragments with all the neighbors: each fragment received is first merged with the local policy via m_i, a merging function that adjusts i's policy as follows: simply combine p_k^i and p_i if the two are compatible with each other; override p_i with p_k^i if conflict arises and i decides to yield to neighbor k. Note that our protocol only develops a policy exchange platform, where the specific implementation of e, f, m (and their semantics) is unspecified and left to the participating ASes. We will show possible instantiations of these functions by examples in § 3.2.

We illustrate the operation of policy exchange by revisiting the example in Figure 2 (right)[2]. One specific execution trace, among many others due to the distributed and asynchronous nature of the policy exchange algorithm 1, is summarized in Table 1: At step 0, all ASes start with their initial policies, furthermore, `AS5` sends its policy request for a secure route to `d` (5→3: secure), since `AS5` depends on `AS3` to get to `d`. This causes `AS3` to change its policy from unconditional traffic delivery (⊤) to secure delivery (`secure`); Likewise, at step 1, `AS6` (`AS7`) sends a fast route request to its two downstream neighbors `AS3`,`AS4`: while this causes `AS4` to adopt fast traffic delivery, the resulting policy of `AS3` is more interesting, to merge the conflicting requests (secure vs. fast), `AS3` can introduce a cost tag (charging 1 unit for fast request, and 2 unit for secure request, as a means to maximize local revenue, denoted by `secure(2), fast(1)`). To realize this charging scheme, `AS3`, as shown at step 2, sends the policy fragment offering the new service (with additional cost) to `AS5`,`AS6`,`AS7`, respectively. After receiving these fragments at step 3, `AS5` — with no other providers available — accepts and changes its policy to `secure(2)`, whereas `AS6` (`AS7`) rejects the new offer — it was already promised a free fast route — and changes its policy to `fast@4, ¬fast(1)@3`. Finally, at step 4, step 5, respectively, `AS6`,`AS7` — to make sure they are not charged — expose the policy fragment `¬fast(1)` to its downstream providers, the result is that `AS3` will converge to `secure(2)`, a policy that can be practically enforced. All the other nodes converge to a locally enforceable policy as well, so the algorithm terminates.

3.2 Preliminary Result

T_p	A	B	
	r	x	
	r	y	
	b	x	l(x)<l(y)

I_1	A	B
	r	[120]
	r	[1340]
	b	[120]

I_2	A	B
	r	[1230]
	r	[130]
	b	[130]

Table 2: T_p represents shortest path policy (p), $I_{1\text{-}2}$ show possible instances of best route selection.

We now sketch ongoing efforts towards building a policy exchange system. The main challenge is that a knowledge representation for policies that allow policies to be distributed, while hiding information about the internal network, is not available. Prior policy aware routing schemes (like BGP and its many alternatives) embed policies in the protocol, relying on network-specific mechanisms to express policies. To this end, we introduce a representation system for policies as the key technique enabler, outline our plan to embed the policy exchange system into the Internet routing infrastructure, and preliminary evaluation on realistic topology.

[2]Due to limited space, we skip the details of the simpler example in Figure 2 (left).

Policy exchange as knowledge exchange. We borrow from the database community a powerful knowledge representation system called conditional table [3], which augments a regular table (or relation) with variables and additional conditions over those variables — a tuple is presented only when the associated condition holds. A conditional table, depending on the evaluation of the variables and conditions, corresponds to a set of concrete instances — regular tables with different instantiations of the variables that satisfy the condition. This makes conditional tables a natural representation for routing policies if we consider a policy one single prescription for all possible route selection outcomes. For example, consider a routing entity `1` with two neighbors from whom routes can be learned to a destination `0`. The shortest path policy (`p`) of `1` can be represented by T_p in Table 2, where $T_{p=A,B}$ has two attributes: attribute `A` shows whether a path is selected or not — `'r'` represents candidate routes and `'b'` denotes best route, and attribute `B` specifies the path itself. T_p contains two candidate routes as represented by variables `x` and `y`, and one best route `x` if its associated condition holds — `x` is shorter than `y` (expressed by a function `l` that returns a path's length). Observe how T_p alone represents all possible route selection outcomes, two such best routes selected are shown in I_1 and I_2.

More importantly, conditional table [3] can be queried with the usual SQL join, projection, and selection etc. in the same way as a regular table: This allows the policy merger (m) and fragment generator (f) in Algorithm 1 to be formulated in terms of these SQL primitives. Specifically, since the fragment of a policy (T) are tuples in T that, collectively, fail to produce a best route, to compute the policy fragment, we only need to select from T two types of tuples: (1) those tuples where (`A='b'`) and whose condition evaluate to false (the negation of which is satisfiable), and (2) tuples where `A='r'` and whose B attributes occur in those tuples in (1); Similarly, to merge a received policy T_1 with the receiver's current policy T_2, we only need to transform T_1 into T_1' to reflect its propagation from the origin (of T_1) to the receiver, followed by the union of T_1' and T_2. Ultimately, we hope to develop a conditional table based knowledge exchange system as a means to rapid development of policy exchange.

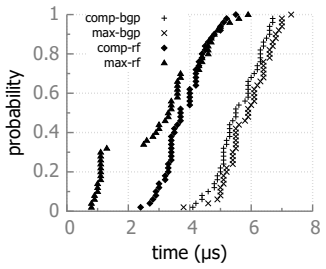

Figure 3: The performance of merging `m` (`comp` and `max`) on two topologies: Rocketfuel topology (rc) and topology generated from BGP traces (bgp).

Integration with the routing infrastructure. In the context of BGP, to properly "install" the adjusted policies (output of the policy exchange protocol) so that policies are enforced during route selection, we only need to refactor those policies into the usual BGP mechanisms, such as the local preference attribute that overrides all other attributes during route selection to enforce preference, or (export) filters to restrict route distribution. This translation can be achieved manually by a human operator who has to configure BGP whether policy exchange is used or not. However, with the introduction of a formal representation system (conditional table) of routing policies [9], we hope to develop tools that can automatically synthesize those BGP configurations, so that human operators only have to understand policies in the conditional table, without worrying about the tedious implementation in the specific BGP mechanism.

Evaluation We implemented the merging function m (Algorithm 1 line 6-7) in policy exchange by two Python functions `comp` and `max`: `comp`(p,i,t) checks whether a set of policies p received from different neighbors are compatible with node iâĂŹs policy in topology t. If conflict arises then i uses `max`(p,i,t) to maximize the total number of admissible policies in p. In the following, we present performance result of merging m, and leave the development of function e and fragment generator f as future work. All experiments were ran on a 64-bit laptop with AMD Ryzen 7 4800H CPU and 15.4G RAM.

We first generate two realistic network topologies: (1) a Rocketfuel topology (AS 7018 with 11292 nodes and 25382 edges); and (2) a topology inferred from the BGP update file (extracted from the AS paths, has 5018 nodes and 8213 links) from the Route View collector route-views2.oregon-ix.net, on February 1, 2021 at 00:00 PST. We then embedded in these two topologies the 7 nodes described in Figure 2 (right) as follows: We randomly pick 7 nodes as `AS1` to `AS7`; `AS3` is the unique provider of `AS5`, it is the node at the "joint" that collects p_5^3, p_6^3 and p_7^3 on destination `d`; `p5` is `secure` policy — requires at least one secure router fragments to the destination d; Each node in the topology is also labeled as a secure router with probability p_s; `p6` and `p7` are `fast` policies that require the selected route to have a length that is below average; When `p5` conflicts with `p6` and `p7`, we assume `AS3` prioritizes `p5`.

p_s	1	2	3
0.01	46.2%	50.3%	0.35%
0.05	41.4%	39.2%	19.4%
0.1	38.7%	30.5%	30.8%
0.5	29.1%	19.9%	51%

Table 3: Probability of satisfying # (1,2,3) announcements under different p_s

Figure 3 plots the processing time of `comp` and `max` on the BGP topology(bgp) and Rocketfuel topology(rf). As expected, the `comp` and `max` delays are negligible, both $\leq 8\mu$s. Table 3 shows the number of satiable policies (as computed by `max`) under different p_s values (1000 iterations for each p_s): as the probability of secure routers increases from .01 to .5, the

probability of simultaneously satisfying all three policies increases from 0.35% to 51%. We can also see that the probability of max = 2 sharply falls from 50.3% to 19.9%, which is faster than max = 1 (from 46.2% to 29.1%). The reason is that, intuitively, the increase of p_s means that AS1 or AS2 is more likely to satisfy p5. This illustrates how our policy merging function can correctly recognize a maximum subset of jointly satisfiable policy fragments.

4 LIMITATIONS

Policy exchange does not magically remove all policy issues. In particular, policy exchange does not detect (resolve) conflicts among local policies, nor does it move policy enforcement to the most fitting location when scalability is concerned.

Conflicting policies and route oscillation. It is well known that routing policies can conflict and cause the routing system to oscillate permanently [7, 13, 15, 17, 18, 30, 37]. For example, three ASes connected in a circle with (clock-wise) cyclic preferences — each AS prefers path via its clock-wise neighbor than its direct path — have conflicting policies that cause permanent oscillation. Policy exchange cannot address such conflicts: the cyclic preference at each AS are perfectly local, none of the ASes has a policy fragment to start an exchange. Moreover, policy exchange only improves visibility into the concerns that cannot be addressed within an AS. Besides, even when a policy fragment received by a routing entity conflicts with its local policy, the receiving entity can still ignore it. Generally, policy exchange neither detects nor resolves conflicts among independently set policies.

Policy holes and routing scalability. Address aggregation in the CIDR routing structure, till today, remains one of the main vehicles to scalability, but policy can punch a hole in the aggregate and drastically drive up the routing table size [19, 32]. For example, a more specific prefix (d_1) of an existing aggregate (d), owned by AS1, can be announced to two different links (l1,l2). This implements a useful traffic engineering policy at AS1 that balances traffic received at l1,l2. At the same time, d_1 will be propagated throughout the Internet, *unnecessarily* creating routing entries for d_1 everywhere. Policy exchange can mitigate this problem, but only partially: Suppose AS1 sends the load balancing restriction that traffic cannot exceed a threshold on each link. This restriction, when received at some upstream AS2 that is on all the paths to d, will be determined to be locally enforceable. Hence it will be kept local at AS2, and suppressed from further populating the rest of the Internet. However, AS1 may not trust AS2, or simply lacks the incentive to start policy exchange with AS2.

While policy exchange does not directly solve the above problems, its building blocks — the new policy representation system, the decoupling of policy from routing — may migrate to new solutions beyond flexible routing, and may clarify and simplify the interaction between policy and other components of the routing system.

5 RELATED WORK

Policy distribution. Existing inter-domain routing schemes often employ some forms of policy distribution: In the MIRO [38] system, a participant can pull neighbors for alternative routes satisfying a particular need. In Wiser [28], neighboring ASes exchange normalized cost to jointly optimize traffic delivery in both ASes. In BGP, restricted form of policy information disclosure is also available with MED [31] attribute and the community attribute [22]. More recently, a BGP extension for policy distribution [23] that enhances community attribute based policy tagging and negotiation was also proposed. Our work attempts to generalize these efforts: policy exchange systematically coordinates policies on the fly, it does not require pre-mediated agreement, and is not restricted to a specific scheme tailored to a particular task.

Declarative networking. In the context of database usages in networking, we are the closest to declarative networking [10, 11, 24–27, 29] which introduces network datalog — a data query language reminiscent of SQL — as a compact and efficient language for expressing networking such as routing protocols and overlay networks. Our work goes beyond programming with factual data. To our best knowledge, we are the first to investigate the use of indefinite data (i.e. conditional tables) to lift policies (intentions) to first order data that can be queried and transformed.

6 CONCLUSION

This paper made the case of policy exchange as a means to accommodate distributed policies in the absence of a central authority. Policy exchange instantiates a simple principle that a policy should be permitted as long as it poses no harm to others, realized in a way that minimizes information disclosure. This makes policy exchange a possible solution to the longstanding policy routing problem in the Internet. We present a preliminary design and ongoing efforts towards a practical policy exchange system. While making any changes to the Internet infrastructure has proven to be extremely difficult, we believe that a deeper understanding of the limit of distributed policy making is needed. We also hope that, with some luck, our knowledge powered technique enablers may infect the design and migrate into the fabric of the future routing system.

Acknowledgments. This work is supported by the National Science Foundation Award CNS-1909450.

REFERENCES

[1] Policy requirements for inter Administrative Domain routing. RFC 1125, Nov. 1989.

[2] Policy routing in Internet protocols. RFC 1102, May 1989.

[3] S. Abiteboul, R. Hull, and V. Vianu, editors. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1995.

[4] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, SIGMOD âĂŹ87, page 34âĂŞ48, New York, NY, USA, 1987. Association for Computing Machinery.

[5] S. Agarwal, Chen-Nee Chuah, and R. H. Katz. Opca: robust interdomain policy routing and traffic control. In *2003 IEEE Conference onOpen Architectures and Network Programming.*, pages 55–64, 2003.

[6] K. Argyraki and D. R. Cheriton. Loose source routing as a mechanism for traffic policies. In *Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, FDNA âĂŹ04, page 57âĂŞ64, New York, NY, USA, 2004. Association for Computing Machinery.

[7] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong. Route oscillations in I-BGP with route reflection. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 235–247, New York, NY, USA, 2002. ACM.

[8] H.-W. Braun. Models of policy based routing. RFC 1104, June 1989.

[9] M. Caesar and J. Rexford. Bgp routing policies in isp networks. *Netwrk. Mag. of Global Internetwkg.*, 19(6):5âĂŞ11, Nov. 2005.

[10] 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.

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

[12] D. Estrin and M. Steenstrup. Inter domain policy routing: Overview of architecture and protocols. *SIGCOMM Comput. Commun. Rev.*, 21(1):71âĂŞ78, Jan. 1991.

[13] N. Feamster, R. Johari, and H. Balakrishnan. Implications of autonomy for the expressiveness of policy routing. *IEEE/ACM Trans. Netw.*, 15(6):1266–1279, Dec. 2007.

[14] I. Ganichev, B. Dai, P. B. Godfrey, and S. Shenker. Yamr: Yet another multipath routing protocol. *SIGCOMM Comput. Commun. Rev.*, 40(5):13âĂŞ19, Oct. 2010.

[15] L. Gao and J. Rexford. Stable Internet routing without global coordination. In *ACM SIGMETRICS*, 2000.

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

[17] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE Trans. on Networking*, 10:232–243, 2002.

[18] T. G. Griffin and G. Wilfong. An analysis of BGP convergence properties. In *SIGCOMM*, 1999.

[19] G. Huston. Commentary on Inter-Domain Routing in the Internet. RFC 3221, Dec. 2001.

[20] T. Imieliundefinedski and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761âĂŞ791, Sept. 1984.

[21] H. T. Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi. Bananas: An evolutionary framework for explicit and multipath routing in the internet. In *Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, FDNA âĂŹ03, page 277âĂŞ288, New York, NY, USA, 2003. Association for Computing Machinery.

[22] T. Li, R. Chandra, and P. S. Traina. BGP Communities Attribute. RFC 1997, Aug. 1996.

[23] Z. Li, L. Ou, Y. Luo, S. Lu, H. Chen, S. Zhuang, and H. Wang. BGP Extensions for Routing Policy Distribution (RPD). Internet-Draft draft-ietf-idr-rpd-05, Internet Engineering Task Force, June 2020. Work in Progress.

[24] 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.

[25] 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.

[26] 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.

[27] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative routing: Extensible routing with declarative queries. SIGCOMM '05. ACM, 2005.

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

[29] 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.

[30] D. McPherson, V. Gill, D. Walton, and A. Retana. Border Gateway Protocol (BGP) persistent route oscillation condition. RFC 3345, 2002.

[31] D. R. McPherson and V. Gill. BGP MULTI_EXIT_DISC (MED) Considerations. RFC 4451, Mar. 2006.

[32] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, and S. Uhlig. Interdomain traffic engineering with bgp. *Comm. Mag.*, 41(5):122–128, May 2003.

[33] Y. Rekhter, S. Hotz, and D. D. Estrin. A Unified Approach to Inter-Domain Routing. RFC 1322, May 1992.

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

[35] A. Seehra, J. Naous, M. Walfish, D. MaziÃĺres, A. Nicolosi, and S. Shenker. A policy framework for the future internet. In *In Proc. HotNets*, 2009.

[36] M. E. Steenstrup. An Architecture for Inter-Domain Policy Routing. RFC 1478, June 1993.

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

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

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

[40] D. Zhu, M. Gritter, and D. R. Cheriton. Feedback based routing. *SIGCOMM Comput. Commun. Rev.*, 33(1):71âĂŞ76, Jan. 2003.