Toward Stable Interdomain Network-Application Integration

Qiao Xiang
Xiamen University

Franck Le
IBM Watson Center

Jingxuan Zhang
Tongji University

Y. Richard Yang
Yale University

ABSTRACT

As an exemplary paradigm of network-application integration (NAI), flexible interdomain routing control, such as SDX and SDI, provides programmable interfaces for applications to specify end-to-end interdomain routes that span across multiple autonomous systems (ASes). Not only do they provide opportunities for applications to optimize their route control, they also allow network service providers to increase their business offerings. However, in an interdomain network, providing these frameworks to applications while running BGP, the de facto interdomain routing protocol, may introduce new stability issues. In this paper, we identify two such stability issues that can happen even if an application only wants to enforce very simple route control rules. To cope with these issues and ensure stable interdomain NAI, we develop a series of stability mechanisms to prevent them from happening, while maintaining the use-announcement consistency among ASes. We use real Internet topology and traffic traces to demonstrate the effectiveness of the proposed mechanisms.

CCS CONCEPTS

• **Networks** → **Programming interfaces**; **Routing protocols**; **Programmable networks**;

KEYWORDS

Network-Application Integration, Interdomain Routing, Programmable Networks

ACM Reference Format:
Qiao Xiang, Franck Le, Jingxuan Zhang, and Y. Richard Yang. 2021. Toward Stable Interdomain Network-Application Integration. In *ACM SIGCOMM 2021 Workshop on Network-Application Integration (NAI '21), August 27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3472727.3472804

1 INTRODUCTION

Flexible interdomain route control is an emerging and representative paradigm of NAI. It draws interests from both academia and industry [1, 2, 10, 11, 13–15, 22–24] because it can provide substantial benefits to both applications and networks. One one hand, it provides programmable interfaces for applications to specify end-to-end routes spanning multiple ASes that satisfy applications' performance requirements (*e.g.*, bandwidth and latency). On the other hand, it provides network service provides new business opportunities.

Programming model. Figure 1 gives an overview of the programmable model of representative flexible interdomain route control frameworks (*e.g.*, SDX [10, 11] and SDI [22, 23]). Essentially, each AS in the framework provides a virtual switch abstraction with a pipeline of match-action tables. Given a request from the application, the AS returns the virtual switch abstraction with the currently selected interdomain route to the destination specified by the application, as well as a number of available alternate routes to the destination and its price. In each match-action table, the `match` fields allow applications to specify policies matching on packet header fields (*e.g.*, TCP/IP 5-tuple) and metadata (*e.g.*, counter) [8]. Compared with the standard SDN programming model, the match-action table in flexible interdomain route control frameworks can be extended to match on network state, including state of available routes and next hops.

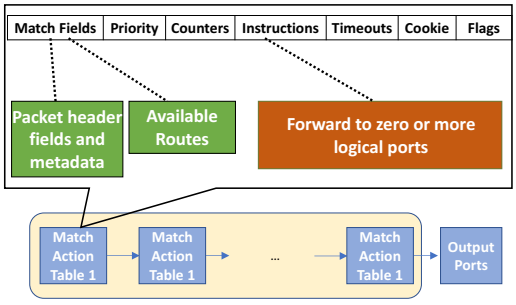


Figure 1: Programming model of flexible interdomain route control frameworks: a virtual switch abstraction with a pipeline of match-action tables.

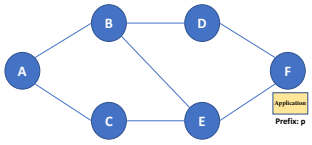
Workflow. The basic workflow for applications to specify end-to-end interdomain route can be illustrated using the example in Figure 2(a). Assume that an application wants to enforce a flexible route policy for traffic destined to IP prefix p at AS F, and that AS B provides the virtual switch abstraction for the application.

First, the application sends a request for a virtual switch abstraction to AS B, expressing that it is interested in the routing information for destination prefix p. Upon receiving the request, B verifies the credentials of the application, and then builds the virtual switch abstraction for p, depicted in Figure 2(b). The virtual switch abstraction indicates that B has two routes $B-D-F$ and $B-E-F$, to p, with the former currently being the selected one.

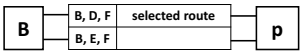
From the received virtual switch abstraction, the application learns of alternate paths, and specifies the flexible routing policy in Figure 2(c). The policy forwards all HTTP traffic destined to p along a primary route $B-D-F$, fast reroutes such traffic to route $B-E-F$ when $B-D-F$ becomes unavailable, and forwards all non-HTTP traffic destined to p along route $B-E-F$.

NAI '21, August 27, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8633-3/21/08…$15.00
https://doi.org/10.1145/3472727.3472804



(a) Network topology: The application wants to enforce flexible routing policy for traffic destined to p at AS B.



(b) The virtual switch abstraction returned AS B.

Match	Action
dstIP = p, dstPort = 80, (B, D, F) = available	fwd(B, D, F)
dstIP = p, dstPort = 80, (B, D, F) = unavailable	fwd (B, E, F)
dstIP = p, dstPort != 80	fwd(B, E, F)

(c) The flexible routing policy specified by the application.

Figure 2: An illustrating example to demonstrate the basic workflow of flexible interdomain route control framework.

When AS B receives the policy from the application, it compiles the policy into device configurations, notifies the application of the result, and sends route updates to its neighbors. In the event that the route $B-D-F$ becomes no longer available, B would notify the application about the modified virtual switch abstraction, and swiftly switch the route for HTTP traffic destined to p to route $B-E-F$, as specified in the flexible routing policy.

Issues. Despite the substantial benefits of flexible interdomain route control, integrating such frameworks with the existing Internet routing system, *i.e.*, the de facto interdomain routing protocol BGP, may introduce various issues. Recent work [2, 6] has revealed that the integration of SDX [11] and BGP can result in data plane issues such as blackhole and persistent forwarding loops. More importantly, new stability issues may arise in the control plane of interdomain networking, causing the whole interdomain network end up in permanent oscillations. These issues have not been fully investiagted and are not yet completely understood.

Contributions of this paper. In this paper, we conduct a study on the stability issues in interdomain networks that simultaneously provide flexible interdomain route control to applications and run BGP among ASes. Specifically, we identify two stability issues that can happen even if an application only wants to enforce very simple route control policies through the programmable interfaces provided by ASes. To cope with these issues, we develop a series of stability mechanisms to prevent such issues from happening, providing stability guarantees for interdomain NAI using flexible interdomain route control. To demonstrate the feasibility and benefits of the proposed mechanisms, we conduct experiments using real Internet topology and traffic traces.

This work does not raise any ethical issues.

2 POLICY SELF-INDUCED INSTABILITY

In Section 1, we provide a simple overview on flexible interdomain route control. Although implementing such control frameworks in an interdomain routing network running BGP may appear simple, it can result in instabilities. In this and the next sections, we illustrate two stability issues identify the two root causes behind them, and present mechanisms to address them. For simplicity, in the remaining of the paper, we refer ASes that provide flexible interdomain route control frameworks to applications as FIRC ASes.

We also assume that all ASes (FIRC ASes or not) support and can communicate only through BGP (without any new extension).

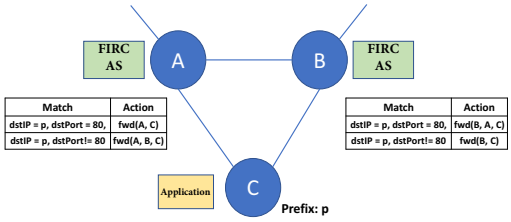


Figure 3: An example of policy self-induced instability.

2.1 Illustration

Interactions between FIRC ASes can result in policy self-induced instabilities. To illustrate them, we consider the simple topology in Figure 3, consisting of one application hosted at AS C, and two FIRC ASes (A, B). The application at AS C wants incoming HTTP traffic to enter C via the link A-C, and all other traffic to enter C via the link B-C. To this end, it specifies a flexible routing policy at AS A to forward HTTP traffic along A-C, and other traffic along A-B-C, and another flexible routing policy at AS B to forward non-HTTP traffic along B-C, and HTTP traffic along B-A-C.

At first glance, these flexible policies seem to satisfy the objectives of AS C. However, in reality, they will result in persistent route oscillations, as illustrated by the following sequence of events:

- **Step 1**. We assume the application has requested both A and B to implement the flexible routing policies depicted in Figure 3.
- **Step 2**. Because B is forwarding its HTTP traffic to A, B must withdraw its route B-C from A. Otherwise, A may select B as its next-hop for some of its traffic including HTTP traffic, causing a forwarding loop between A and B. As such, B sends a BGP UPDATE WITHDRAW withdrawing the B-C route from A. Similarly, A sends a BGP UPDATE WITHDRAW withdrawing A-C from B.
- **Step 3**. Since B has withdrawn the B-C route from A in Step 2, the only available route at A to C is A-C. A selects that route and advertises it to B. Similarly, B selects B-C and advertises it to A.
- **Step 4**. B receiving the route A-C, installs flexible-match rules to forward HTTP traffic destined to C to A. A receiving the route B-C, installs flexible-match rules to forward non HTTP traffic destined to C to B.
- **Step 5**. B and A perform the same actions as those described in Step 2, resulting in a persistent route oscillation.

2.2 Root Cause

While the previous section illustrated that simple network topologies and flexible routing policies could result in persistent route oscillations, this section identifies the root cause behind the policy self-induced instability.

Fundamentally, the root cause of such instability is the granularity mismatch between the flexible-match FIRC policy and the coarse-grained routing information exchange using BGP. This mismatch increases the possibility of forming a dispute wheel [9].

Specifically, in the example above, If BGP was extended to support flexible-match updates, there would exist no dispute wheel for HTTP traffic destined to C or non-HTTP traffic destined to C. However, because BGP cannot differentiate traffic based on destination port, B must withdraw the complete route B-C from A to prevent A from sending any HTTP traffic to B, and so does A. In this way, a dispute wheel is created. In other words, the instability in Figure 3 is caused by use-announcement granularity mismatch.

2.3 Solution

Strawman approaches: A strawman to avoid the policy self-induced instability is to let the application play the role of a centralized controller to ensure the correctness of flexible routing policies at different FIRC ASes, and break the use-announcement semantics of BGP. For example, in Figure 3, using this strawman approach, FIRC AS A forces the corresponding BGP speaker to not withdraw route AC from AS B, and vice versa. In this way, A is given the illusion that B is using BC, B is given the illusion that A is AC, while each of them is using fine-grained policy to forward HTTP traffic and non-HTTP traffic along different routes. Although this design seems to work, it requires all FIRC ASes to have a strong faith that the application can ensure the correctness of flexible routing policies. More importantly, this design violates use-announcement consistency, which can lead to black holes and forwarding loops.

One may think of another strawman approach, which is called the *single-route* guideline. This guideline states that: when the application specifies a set of routes R toward the same destination in the flexible routing policies, if the application ensures that given any AS X occurs in more than one route in R, the sub-route from X to the destination in these routes must be the same, then the flexible routing policies will not lead to policy self-induced instability. Consider the example in Figure 3: if the application only selects AC and BC as the routes in its policies at AS A and B, respectively, no policy self-induced instability will happen.

However, this guideline is too restrictive and limits the flexibility of the application to assign routes on a fine-grained granularity, *e.g.*, in Figure 3, the set of routes ABC, AC and BC violate the single-route guideline, but in fact, using them simultaneously in the flexible routing policies will not cause policy self-induced instability.

Policy self-conflicting graph: To eliminate policy self-induced instability while still ensuring the flexibility of applications to use different routes when specifying flexible routing policies, we develop a novel data structure, *policy self-conflicting graph*, which enables a less restrictive guideline for applications. Specifically, such a graph is constructed as follows. Assume the application specifies a set of routes R toward the same destination in the fine-grained policies. For each AS i shown in R, a node v_i is created. For any link $i - j$ shown in R, a directed link $v_i \rightarrow v_j$ is created. Using such a graph, we develop the following guideline to eliminate policy self-induced instability.

Proposition 1. *Assume ASes exchange routing information using standard BGP, and an application uses a set of routes $R = \{r_1, \ldots, r_K\}$ in flexible routing policies to forward traffic toward the same destination. If there is no loop in the corresponding policy self-conflicting graph, such policies will not cause any policy self-induced instability.*

Proof. (Sketch) In BGP, if AS A uses B as next-hop toward a given destination D, A will not announce a route to B. Given any route $r \in R : \{N_1 - N_2 - \ldots - D\}$, there exists a path $v_{N_1} \rightarrow v_{N_2} \rightarrow \ldots \rightarrow D$ in the policy self-conflicting graph. When self-induced instability happens in the network, a loop can be found in the self-conflicting graph by following the propagation of BGP withdraw messages. Hence the contraposition of this claim, *i.e.*, Proposition 1, is true. □

Revisiting the example in Figure 3, we can create a policy self-conflicting graph for the set of routes $\{A - C, A - B - C, B - C, B - A - C\}$, and find it has a loop $A - B - A$. As such, by applying Proposition 1, the application will know that using specify these routes simultaneously in the policies sent to FIRC ASes A and B will lead to policy self-induced instability; hence it should not do that.

We develop a generic route enumeration framework to compute the maximum set of routes that will not cause policy self-induced instability. We omit the details of this framework due to space limitation. In the example in Figure 3, by applying the route enumeration framework, the maximal sets of routes the application can use to specify fine-grained policies without causing any policy self-induced instability are $\{A - C, A - B - C, B - C\}$ and $\{A - C, B - A - C, B - C\}$.

3 POLICY OVERWRITING INSTABILITY

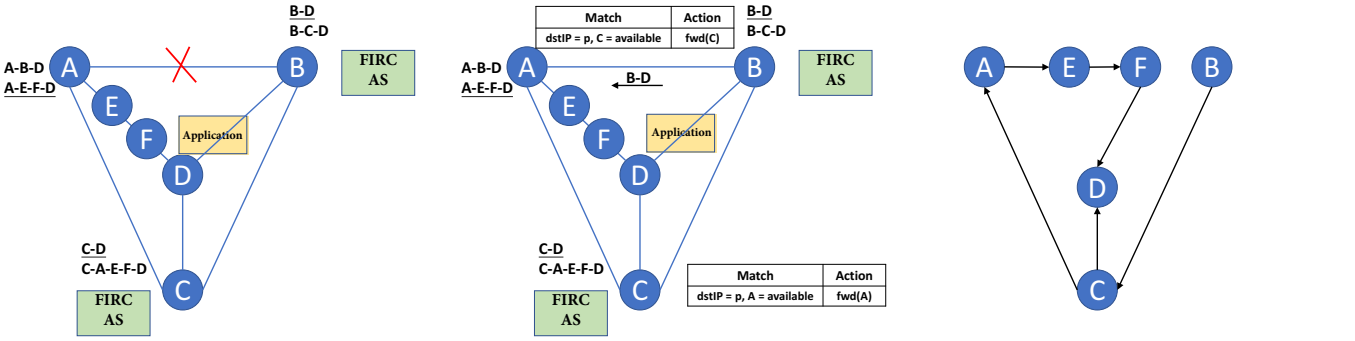
Using the policy self-conflicting graph, an application can eliminate the policy self-induced instability when specifying flexible interdomain route control policies. However, implementing flexible-routing policy at FIRC ASes can still result in other instability issues.

3.1 Illustration

Consider Figure 4(a) as an example. In this network, AS A, B and C are academic BGP ISPs (*e.g.*, Internet2 [12]). B and C are also FIRC ASes. AS D is a university network and hosts an application. D is the BGP customer of B and C. A and C are BGP peers to each other, and BGP providers of B. There also exists a BGP customer route $A - E - F - D$. The route selection preference of A, B, and C are shown in the figure. At the beginning, link $A - B$ is not available. As such, without FIRC flexible routing policy, the network converges to a state where A, B, and C are forwarding traffic to D using $A - E - F - D$, $B - D$, and $C - D$, respectively.

- **Step 1**. Assume when link $A - B$ becomes available and route $B - D$ is being sent to A, the application at D specifies a policy to let B always prefer C as next hop, and let C always prefer A as next hop, as shown in Figure 4(b). Before A receives the $B - D$ and B and C receive the FIRC policy from D, the routes used by A, B, and C are still $A - E - F - D$, $B - D$, and $C - D$.
- **Step 2**. After A receives B-D from B, and B and C receive the FIRC policy sent from D, the routes used by A, B and C are $A - B - D$, $B - C - D$, and $C - A - E - F - D$, respectively.
- **Step 3**. A, B and C announce their routes counterclockwise, and the updated routes used by A, B and C are $A - B - C - D$, $B - C - A - E - F - D$[1], and $C - A - B - D$.

[1]B announces this route A because this route is specified by the application at D, hence is treated as a customer route



(a) Link $A - B$ is not available. When the application at AS D does not specify any policy, the network converges.

(b) When link $A - B$ becomes available, and $B - D$ is being sent to A, the application hosted at AS D specifies a policy to let B always prefer C as next hop, and let C always prefer A as next hop.

(c) The corresponding policy self-conflicting graph is loop-free, but network may enter persistent oscillation.

Figure 4: An example of policy overwriting stability issue.

- **Step 4**. A, B and C announce their routes counterclockwise, and the updated routes received by A, B and C are $A - B - C - A - E - F - D$, $B - C - A - B - D$, and $C - A - B - C - D$. A, B and C all detect a loop in AS-PATH. Each of these routes is disregarded. The routes used by A, B, and C are $A - E - F - D$, $B - D$, and $C - D$.
- **Step 5**. A, B and C announce their routes counterclockwise, and the updated routes received by A, B and C are $A - B - D$, $B - C - D$, and $C - A - E - F - D$, same as STEP 2, resulting in a persistent route oscillation.

3.2 Root Cause

We illustrated above that even if the policy self-conflicting graph is loop-free and hence the network is free from policy self-induced instability, simple network topologies and flexible routing policies could still result in route oscillations. Next, we identify the root cause behind this instability.

The root cause of this instability is: allowing the application to overwrite the standard BGP route selection policy at FIRC ASes may create dispute wheels; the policy self-conflicting graph does not capture the dynamics of BGP route selection and export; hence fails to capture these dispute wheels.

Specifically, the policy self-conflicting graph is a special variation of the p-graph [18]. A p-graph captures both the order of route rank functions within each AS and the route availability dependence between all available routes in ASes. As such, it has a global view to identify the existence of dispute wheel. In contrast, the self-conflicting graph only captures the route availability dependence between all currently available routes at a small number of ASes (*i.e.*, FIRC ASes). With only a partial view, it cannot identify the dispute wheels caused by application overwriting the route selection function at FIRC ASes. As such, we call this instability the *policy overwriting instability*.

3.3 Solution

Strawman approach: A strawman approach to solving the policy overwriting instability is *limited-exposure*. Specifically, in this approach, a FIRC AS treats the application as a BGP provider, exposes BGP customer routes to the application, and only exposes BGP peer/customer routes when no BGP customer route is available. With this strawman, we have the following proposition:

Proposition 2. *When the underlying BGP routing protocol follows the standard route selection policy (*i.e.*, prefer customer routes over peer/provider routes first, and then prefer shortest AS path), the limited-exposure approach ensures that an interdomain network with FIRC ASes is free from policy overwriting instability.*

The proof of this proposition is straightforward and hence omitted. In Figure 4, we see that by using the limited exposure approach, the application can only use route $B - D$ and $C - D$ at FIRC ASes B and C, respectively, ensuring no policy overwriting instability will happen. But as shown in this example, this approach provides very limited flexibility for applications to pick different routes for different traffic.

Full RIB exposure with self-stabilizing filter at FIRC AS: To eliminate policy overwriting instability while still providing applications enough flexibility to select different routes in flexible routing policy, we develop a novel design, which includes two key design points. First, this design supports FIRC ASes treating the application as BGP customer and expose all available routes to the application. Second, a self-stabilizing filter based on SS-BGP [19] is deployed at each FIRC AS X. Specifically, this filter intercepts every BGP route update message at FIRC AS X and compare the new route with the current route. Given a destination, if a new route r learned from a neighbor AS Y has a higher rank than the current route, but the AS-PATH of r contains X, the filter disregards r and blocks all future routes to this destination learned from AS Y for a short period, and notifies the application about this information. Consider the example in Figure 4. Using this design, the FIRC ASes B and C can expose all available routes to the application hosted at D. Even if D specifies the same policies shown in the figure, the self-stabilizing filters deployed at B and C ensure that the network converges to a state where $B - D$ and $C - D$ are used to forward traffic to D.

We prove the following proposition about the full RIB exposure with self-stabilizing filter design.

Proposition 3. *Assume the underlying BGP routing protocol in the signaling plane layer follows the typical route selection policy (*i.e.*, prefer customer routes over peer/provider routes first, and then prefer shortest AS path). The full RIB exposure with self-stabilizing filter design ensures that an interdomain network with FIRC ASes is free from policy overwriting instability.*

Proof. (Sketch) We show that allowing applications to select route from all available routes at each FIRC AS, the route selection functions at all ASes still maintain isotonicity[2]With this property, the correctness of proposition can be proved by applying Theorem 3 in [19]. □

Although this design ensures that an interdomain network with FIRC ASes is free of policy overwrite instability, one concern is that with this design, the routes selected by application may not be used by FIRC ASes in the stable state. And this was also illustrated in the example above. To address this, we give the following guideline for applications to select routes that will be used by FIRC ASes in the stable state.

Proposition 4. *Assume the BGP policies of each AS follows the typical route selection policy and converges. Given a destination, if the specified routes of an application to reach destination AS A at different FIRC ASes only violates the local preference (*i.e.*, prefer customer over peer/provider) of at most one FIRC AS B, FIRC is free from policy overwriting instability, and the application-specified local preference always provides a route in the stable state.*

Proof. (Sketch) In the scenario described above, the only way the self-stabilizing filter is triggered is because B receives a route that contain A, and sends it to A. However, A is using B as next hop. So the route B receives will have contain $A-B$. B will detect the loop, disregard it, and not send to A. □

Revisit the example in Figure 4. Assume D only specifies one policy at FIRC B to choose C as next hop. The network will converge at a state where A, B, and C use $A-B-C-D$, $B-C-D$, $C-D$ to forward traffic to D.

4 EVALUATION

We conduct trace-driven experiments to demonstrate the feasibility and benefits of the proposed stability mechanisms for guaranteeing stable interdomain network-application integration, while providing applications sufficient flexibilities in conducting interdomain route control.

Real AS-level Internet topology. We use the global AS-level Internet topology annotated with the business relationships between neighboring ASes from CAIDA from 2016-01-01 [3] to evaluate the performance of FIRC. The topology includes 63361 ASes and 320978 AS-level connections.

Real Internet-scale traffic traces. We use the traffic traces collected in the CAIDA Anonymized Internet Traces 2016 Dataset [4] to evaluate FIRC with actual Internet traffic. The traffic traces are captured from a 10GE Internet backbone link between Tier 1 ASes in Chicago and Seattle.

[2]Isotonicity means that given two routes r_1 and r_2, if AS A prefers r_1 over r_2, a neighbor AS B does not have the opposite preference after its local processing of the two routes [19]

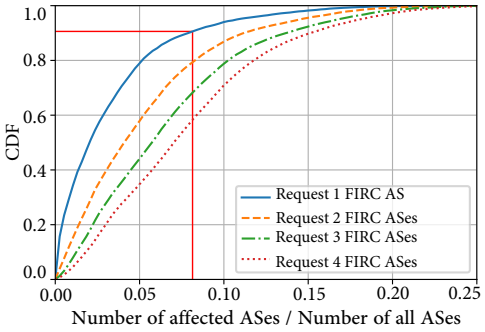


Figure 5: CDF of controlled FIRC ASes.

Scenario: inbound traffic engineering. In our experiments, we consider the scenario where an application located as a multi-homed stub AS wants to balance the incoming traffic along different inbound links of the stub AS. We assume FIRC ASes are deployed at Tier 1 ASes. The application collects traffic statistics of the stub AS' inbound links and from FIRC ASes, uses such information to compute the load balancing decisions and assign available routes to different flows, and sends the flexible routing actions to corresponding FIRC ASes. In each experiment run, we randomly select a multi-homed stub AS as the one hosting the application, and let it specify flexible routing policies for a destination IP prefix to 1~4 FIRC ASes while following the stability guidelines proposed in Sections 2 and 3. We run experiments for 1k multi-homed stub ASes. For each (stub AS, number of FIRC ASes) combination, we run experiments 20 times, each of which randomly selects FIRC ASes to specify flexible routing policies.

Results. Figure 5 presents the CDF of the percentage of ASes with a change of selected BGP route. The key observation is: by following the proposed guidelines when specifying flexible interdomian route policies, the application can not only realize its inbound traffic load balancing goal, but also make the global Internet converge efficiently, causing a small number of AS rerouting. Specifically, if only requesting fine-grained routing control on a single FIRC AS, for 90% of policies to a destination, less than 10% of ASes are affected. Even when an application specifies flexible routing policies at 4 FIRC ASes, only less than 15% of ASes are affected.

5 DISCUSSION

Use policy self-conflicting graph in practice. This paper designs the policy self-conflicting graph to eliminate policy self-induced instability. It is a powerful theoretical tool, but it requires collecting the selection and export policies of ASes to a centralized server to construct this graph and check for loop on it. This is problematic because selection and export policies are considered private. One future work we plan to investigate is to develop a privacy-preserving loop detection algorithm using secure multi-party computation [7]. In this way, ASes can collaboratively check whether the policy self-conflicting graph has a loop without exposing their selection or export policies.

Impact of interdomain NAI on interdomain routing protocol. Previous sections show that the co-existence of flexible interdomain NAI and the de facto interdomain routing protocol (*i.e.*, BGP) may lead to severe stability issues. Other than restricting the flexibility of interdomain NAI (*i.e.*, use policy self-conflicting graph and full RIB exposure with self-stabilizing filter), another direction to cope with such stability issues is to design a new interdomain routing protocol. Our previous work [21] SFP is an example of such interdomain routing protocol. In a nutshell, SFP extends BGP to allow neighboring ASes to exchange fine-grained interdomain routing information (*e.g.*, routes for different flows). By maintaining the consistency between fine-grained route announcement and fine-grained flexible interodomain routing, SFP can also eliminate the policy self-induced instability. However, it does not eliminate the policy overwriting instability. We leave this as another problem to investigate in the future.

6 RELATED WORK

Several systems have been proposed to provide such flexible interdomain routing services (*e.g.*, [1, 2, 5, 10, 11, 13–16, 21–24]). They can be categorized into two classes. The basic idea of this category is to make participating ASes express their fine-grained routing policies to the infrastructure of a trusted third-party that composes and enforces their fine-grained interdomain policies [2, 6, 10, 11, 13, 14]. An important representative system of this category is SDX [11] and its variants [2, 6, 10]. The second category is tunnel-based overlay [15, 20, 24, 25]. MIRO [24], ARROW [15] and RCS [20] are the most recent systems in this category. The basic idea is to let a stub AS negotiate with a remote AS to select routes different from the BGP route, and then build a tunnel between stub and remote ASes to utilize the negotiated routes. In addition, Google and Facebook [17, 26] develop flexible peering systems to take route selection back to edge by overriding the BGP. But they do not provide services for clients. Despite the benefits of these systems, however, none of them investigated the potential stability issues when integrating such flexible interdomain route control with BGP, the de facto interdomain routing protocol. To the best of our knowledge, this paper is the first to fill this gap by identifying two new stability issues and proposing corresponding stability mechanisms.

7 CONCLUSION

We identify two stability issues in an interdomain network running FIRC and BGP simultaneously, analyze their root causes, propose a series of stability mechanisms, and validate the mechanisms using trace-driven experiments. These results shed lights toward stable interdomain NAI. As future work, we plan to investigate (1) how applications can enforce the proposed mechanisms when specifying FIRC policies while allowing each AS to keep its routing policies private, and (2) how to design fine-grained interdomain routing protocols that avoids stability issues..

ACKNOWLEDGMENT

The authors thank the NAI reviewers and our shepherd Chunshan Xiong for their extensive and valuable feedback. The authors thank Haitao Yu, Danny Lachos, Daniel Ernst and Yan Zhu for their help during the preparation of this paper. Qiao Xiang is the corresponding author. This work is funded in part by the Facebook Research Award, the Google Research Award and the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001.

Qiao Xiang dedicates this paper to Xi Chen, a close friend from high school who recently passed away.

REFERENCES

[1] Gilad Asharov, Daniel Demmler, Michael Schapira, Thomas Schneider, Gil Segev, Scott Shenker, and Michael Zohner. 2017. Privacy-preserving interdomain routing at Internet scale. *Proceedings on Privacy Enhancing Technologies* 2017, 3 (2017), 147–167.

[2] Rüdiger Birkner, Arpit Gupta, Nick Feamster, and Laurent Vanbever. 2017. SDX-based flexibility or Internet correctness? Pick two!. In *Proceedings of the Symposium on SDN Research*. ACM, 1–7.

[3] CAIDA. 2016. AS relationships dataset. http://www.caida.org/data/as-relationships/

[4] CAIDA. 2016. The CAIDA anonymized Internet traces 2016 dataset. http://www.caida.org/data/passive/passive_2016_dataset.xml

[5] Yichao Cheng, Ning Luo, Jingxuan Zhang, Timos Antonopoulos, Ruzica Piscac, and Qiao Xiang. 2021. Looking for the maximum independent set: a new perspective on the stable path problem. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE.

[6] Arnaud Dethise, Marco Chiesa, and Marco Canini. 2018. Prelude: ensuring interdomain loop-freedom in SDN-enabled networks. *arXiv preprint arXiv:1806.09566* (2018).

[7] David Evans, Vladimir Kolesnikov, and Mike Rosulek. 2017. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security* 2, 2-3 (2017).

[8] Open Networking Foundation. 2013. OpenFlow Switch Specification 1.4.0. Open Networking Foundation (on-line). https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

[9] Timothy G Griffin, F Bruce Shepherd, and Gordon Wilfong. 2002. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking (ToN)* 10, 2 (2002), 232–243.

[10] Arpit Gupta, Robert MacDavid, Rüdiger Birkner, Marco Canini, Nick Feamster, Jennifer Rexford, and Laurent Vanbever. 2016. An industrial-scale software defined Internet exchange point. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, Vol. 16. 1–14.

[11] Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P. Donovanand Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. 2014. SDX: a software defined Internet exchange. In *Proceedings of SIGCOMM 2014*. 233–239.

[12] Internet2. 2019. Internet2. https://www.internet2.edu.

[13] Vasileios Kotronis, Xenofontas Dimitropoulos, and Bernhard Ager. 2012. Outsourcing the routing control logic: better internet routing based on SDN principles. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 55–60.

[14] Karthik Lakshminarayanan, Ion Stoica, Scott Shenker, and Jennifer Rexford. 2004. *Routing as a Service*. Computer Science Division, University of California Berkeley.

[15] Simon Peter, Umar Javed, Qiao Zhang, Doug Woos, Thomas Anderson, and Arvind Krishnamurthy. 2014. One tunnel is (often) enough. In *ACM SIGCOMM 2014*, Vol. 44. ACM, 99–110.

[16] Shahrooz Pouryousef, Lixin Gao, and Arun Venkataramani. 2020. Towards logically centralized interdomain routing. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 739–757.

[17] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. 2017. Engineering egress with edge fabric: steering oceans of content to the world. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 418–431.

[18] Joao Luis Sobrinho. 2003. Network routing with path vector protocols: Theory and applications. In *Proceedings of ACM SIGCOMM 2003*.

[19] João Luís Sobrinho, David Fialho, and Paulo Mateus. 2017. Stabilizing BGP through distributed elimination of recurrent routing loops. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, 1–10.

[20] Yangyang Wang, Jun Bi, and Keyao Zhang. 2017. A SDN-based framework for fine-grained inter-domain routing diversity. *Mobile Networks and Applications* 22, 5, 906–917.

[21] Qiao Xiang, Chin Guok, Franck Le, John MacAuley, Harvey Newman, and Y. Richard Yang. 2018. SFP: toward interdomain routing for SDN networks. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*.

87–89.

[22] Qiao Xiang, Jingxuan Zhang, Kai Gao, Yeon-sup Lim, Franck Le, Geng Li, and Y Richard Yang. 2020. Toward optimal software-defined interdomain routing. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1529–1538.

[23] Qiao Xiang, Jensen Zhang, Franck Le, and Y Richard Yang. 2020. Toward programmable interdomain routing. In *Proceedings of the Applied Networking Research Workshop*. 22–24.

[24] Wen Xu and Jennifer Rexford. 2006. MIRO: Multi-path interdomain routing. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. 171–182.

[25] Xiaowei Yang, David Clark, and Arthur W Berger. 2007. NIRA: a new interdomain routing architecture. *IEEE/ACM Transactions on Networking (ToN)* 15, 4 (2007), 775–788.

[26] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. 2017. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 432–445.