Distributed Route Aggregation on the Global Network

www.route-aggregation.net

João Luís Sobrinho
Instituto de Telecomunicações
Instituto Superior Técnico
Universidade de Lisboa
joao.sobrinho@lx.it.pt

Laurent Vanbever
Princeton University
vanbever@cs.princeton.edu

Franck Le
IBM T. J. Watson Research
fle@us.ibm.com

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

ABSTRACT

The Internet routing system faces serious scalability challenges, due to the growing number of IP prefixes it needs to propagate throughout the network. For example, the Internet suffered significant outages in August 2014 when the number of globally routable prefixes went past 512K, the default size of the forwarding tables in many older routers. Although IP prefixes are assigned hierarchically, and roughly align with geographic regions, today's Border Gateway Protocol (BGP) and operational practices do not exploit opportunities to aggregate routes. We present a distributed route-aggregation technique (called DRAGON) where nodes analyze BGP routes across different prefixes to determine which of them can be filtered while respecting the routing policies for forwarding data-packets. DRAGON works with BGP, can be deployed incrementally, and offers incentives for ASs to upgrade their router software. We present a theoretical model of route-aggregation, and the design and analysis of DRAGON. Our experiments with realistic assignments of IP prefixes, network topologies, and routing policies show that DRAGON reduces the number of prefixes in each AS by about 80% and significantly curtails the number of routes exchanged during transient periods of convergence.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing Protocols; C.2.3 [**Network Operations**]: Network management; C.2.6 [**Internetworking**]: Routers

General Terms

Algorithms; Management; Performance; Theory

Keywords

Inter-domain Routing; Route Aggregation; Scalability; BGP

CoNEXT'14, December 2–5, 2014, Sydney, Australia.

http://dx.doi.org/10.1145/2674005.2674999.

1. INTRODUCTION

Ideally, a global routing system should scale by having each node maintain detailed information about nearby destinations and only coarse-grained information about far-away destinations [18, 34, 19]. Such a scalable system would be possible under careful, centralized planning of the network topology, address assignment, and route computation. However, the Internet is anything but centralized, and its growth has been anything but orderly. Each Autonomous System (AS) makes its own decisions about where to connect, how to acquire address space, and what routes to select and make available to others. Increasingly, even ASs at the perimeter of the Internet are multi-homed, making it necessary to propagate their routing information beyond their providers. As a result, more than half a million IP prefixes are distributed by the global routing system [1].

The growing number of globally routable IP prefixes has serious consequences for the Internet. The number of prefixes determines the size of *forwarding-tables* (or, Forwarding Information Base, or FIB) stored in expensive high-speed memory on the routers, as well as that of *routing-tables* (or, Routing Information Base, or RIB). Many older routers devote a default size of 512K entries to the IPv4 forwarding table, which lead to the Internet outages of August 12, 2014, when the number of prefixes crossed this threshold [10, 22]. The number of prefixes also affects the message overhead and convergence time of the Border Gateway Protocol (BGP), and the time required to bring up a single BGP session [15, 37]. BGP's scalability challenges also hinder the deployment of security enhancements like S-BGP, since the substantial computational overhead for signing and verifying BGP routes grows with the number of prefixes.

Fortunately, the underlying structure of the global routing system makes better route aggregation possible. The assignment of IP prefixes is mostly aligned with the AS-level topology and business relationships, since blocks of IP addresses are allocated hierarchically by Regional Internet Registries (RIRs) to Internet Service Providers (ISPs) who, in turn, assign sub-blocks to their customers. Some ASs acquire Provider-Independent (PI) prefixes directly from the RIRs. Nonetheless, these prefixes still align roughly with geographic regions. Likewise, despite the prevalence of multi-homing, most ASs connect to multiple providers in the same geographic area, leaving potential for far-away ASs to route based on coarser-grained routing information.

Existing routing protocol implementations and operational practices do *not* exploit these opportunities for route aggregation. According to best current practices, an ISP configures its routers to filter BGP routes from single-homed customers with IP addresses allocated out of the ISP's address space, but not from customers who are multi-homed or have PI addresses. The reason is simple: network operators cannot reason about how more aggressive filters would affect how other parts of the Internet reach their customers. In the face of uncertainty, ISPs are understandably conservative in applying route filters. Worse yet, some ISPs do not filter at all, out of ignorance, sloppy operational practices, or legitimate concerns that a previously single-homed customer might later become multi-homed.

In this paper, we introduce DRAGON (Distributed Route Aggregation on the Global Network), a route-aggregation solution for inter-domain routing. DRAGON operates with today's BGP protocol and any routing policies that ensure correct operation of BGP. We show that by comparing routes for different prefixes, an AS can determine which prefixes can be filtered locally without worsening the type of route used to forward data-packets. DRAGON can be deployed incrementally and has built-in incentives for ASs to participate. Our experiments with realistic AS-level topologies, IP prefix assignments, and routing policies show that DRAGON dispenses with about 80% of the IP prefixes in each AS.

Inter-domain routing policies are neither arbitrary nor random. They reflect business goals and are constrained by BGP's configuration mechanisms. Seen in this light, it is not surprising that many routing policies end-up satisfying properties which leave a distinctive mark on the global routing system. One such property is isotonicity [8, 32], enjoyed by the Gao-Rexford (GR) routing policies [13] among others [31, 23]. In loose terms, isotonicity means that if an AS prefers one route over another, a neighbor AS does not have the opposite preference after its local processing of the two routes. We show that, in the face of isotone routing policies, DRAGON attains an optimal aggregated state that preserves the global routes traversed by data-packets on their way to the destinations.

The fundamentals of DRAGON are valid for any prefix-based routing system substantiated on a routing vector-protocol. We honor that generality in the way we present DRAGON even if our examples and experiments pertain to inter-domain routing. The paper has three main parts: the design of DRAGON, illustrated with examples, Section 3; the theoretical justification for DRAGON, Section 4; and experiments quantifying the scalability benefits of DRAGON, Section 5. The next section establishes the routing model. There is also a section on related work, Section 6, and a concluding section, Section 7.

2. ROUTING AND FORWARDING MODEL

A network is composed of nodes joined by links. Addresses are strings of bits of fixed length. A prefix is a string of bits of length shorter than that of the addresses, representing all the addresses whose first bits coincide with those of the prefix. Prefixes are assigned to nodes and made known to all other nodes in the network through a routing vector-protocol, in accordance with the routing policies configured at the various nodes. In inter-domain routing, nodes are ASs, prefixes are IP prefixes, the routing vector-protocol is BGP, and there is a two-way link between two ASs if at least two border routers, one on each AS, established a BGP-session between them.

A route is an association between a prefix and an attribute.[1] Attributes are totally ordered by preference. A route pertaining to prefix p is called a p-route. A standard vector-protocol instantiates a distinct computation process for every prefix. The node to which p has been assigned is the origin of p. This node attaches an attribute to p thus forming a p-route that it announces to its neighbors. Each node stores in its routing-table, for each one of its neighbors, a candidate p-route that extends the last of the p-routes announced by the neighbor that was received by the node. The node elects the candidate p-route with the most preferred attribute and, in turn, announces the elected p-route to its neighbors. Every time a node elects a p-route, it makes an entry in its forwarding-table associating p to the forwarding neighbors for p, those being the neighbor nodes for which the candidate p-route coincides with the elected p-route. Allowing for multi-path routing, a prefix may be associated with more than one forwarding neighbor. Routing policies specify the relative preference among attributes and how the attribute of an elected route at one node is extended to the attribute of a candidate route at a neighbor node.

The prototypical inter-domain routing policies are the Gao-Rexford (GR) routing policies [13], which postulate that neighbor nodes establish either a customer-provider or a peer-peer relationship. The policies are supported on just the three attributes "learned from a customer," "learned from a peer," and "learned from a provider." Following standard terminology, we use the term "customer route" as shorthand for "route with attribute 'learned from customer'," and similarly for the terms "peer route" and "provider route," and talk about the preference among routes signifying the preference among their attributes. A customer route is preferred to a peer route which is preferred to a provider route.[2] Customer routes are exported to all neighbors, all routes are exported to customers, and these are the only exportations allowed. A route originated by a node can be assumed to have attribute "learned from a customer," since it is subjected to the same treatment as if it were learned from a customer, namely, the route is exported to all of the node's neighbors.

A vector-protocol is correct in a network if it terminates in a stable state that guides data-packets to their destinations. Reference [32] gives a condition on the cycles of a network that guarantees correctness. In the case of the GR routing policies, that condition stipulates the absence of cycles where each node is a customer of the next around the cycle.

Figure 1 shows a network operating according to the GR routing policies. Solid lines join a provider and a customer, with the provider drawn higher than the customer, and a dashed line joins two peers. For instance, u_2 is a provider of both u_3 and u_4, and a peer of u_1. Node u_6 is multi-homed to two providers, u_3 and u_4. Prefix p was assigned to node u_4 which originates a p-route that it exports to all its neighbors. Once the vector-protocol terminates, u_2 elects a customer p-route, learned from u_4, which becomes u_2's forwarding neighbor for p; u_1 elects a peer p-route, learned from u_2; and u_5 elects a provider p-route, learned both from u_1 and u_3.

A prefix q is *more specific* than a prefix p if it is longer than p and its first bits coincide with those of p. Delegation of prefixes from providers to customers combined with operational practices, such as those related to multi-homing and traffic engineering, cause prefixes at different levels of specificity to be propagated throughout the network and maintained in the routing- and forwarding-tables of nodes. The longest prefix match rule [12] prescribes that data-packets are forwarded at a node according to the elected route of the most specific of the prefixes that contains the destination address of the data-packet. In Figure 1, prefix q, assigned to u_6, is more specific than prefix p. Node u_3 elects a customer q-route, learned from u_6, and a provider p-route, learned from u_2. Data-packets arriving

[1] Our use of the term "attribute" is generic and not meant to single out the parameters of BGP, such as LOCAL-PREF and AS-PATH.

[2] Peer routes do not have to be preferred to provider routes [13]. We make this extra assumption because it seems to be valid in practice and it simplifies the exposition.

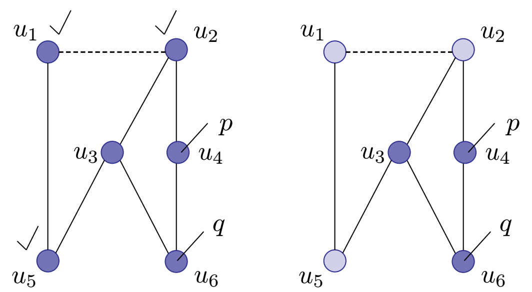

Figure 1: Providers are drawn above customers and joined with solid lines. Peers are joined with dashed lines. Node u_6 is multi-homed to u_3 and u_4. Node u_4 is assigned p and delegates q to its customer u_6, with q more specific than p. Left. Standard stable state. Checks mark nodes that satisfy the condition for filtering q. Right. Stable state after all nodes execute code $\mathbf{CR}$ in whatever order, with light-shaded nodes forgoing q.

at u_3 with destination in q are forwarded to u_6, whereas those arriving with destination in p but *not* in q are forwarded to u_2.

3. MECHANISMS OF DRAGON

DRAGON relies on standard routing messages of a vector-protocol and augments local routing decisions with filtering of prefixes and generation of aggregation prefixes. Section 3.1 presents basic filtering code for DRAGON and Section 3.2 presents a rule for originating routes that ensures that the filtering code does not cause black holes. Section 3.3 introduces a property of routing policies known as isotonicity and shows that, in their presence, DRAGON attains optimally filtered routing states. Section 3.4 deals with partial deployment. Section 3.5 discusses alternative filtering codes. Section 3.6 concerns multiple levels of prefixes and Section 3.7 presents aggregation prefixes. Section 3.8 discusses the reaction of DRAGON to network events, such as link failures. Last, Section 3.9 shows how DRAGON accommodates prefix de-aggregation for the purposes of traffic engineering.

3.1 Filtering code

The goal of DRAGON is for many nodes to dispense with routes pertaining to more specific prefixes with little or no change in the properties of paths traversed by data-packets. Towards this goal, some nodes filter some prefixes. Filtering of a prefix means that no entry for the prefix is installed in the forwarding-table of the node and the prefix is not announced to neighbor nodes. Routes pertaining to the prefix are still kept in the routing-table of the node for a prompt reaction to network events (Section 3.8).

Let q be more specific than p. We investigate the following code to filter q, to be executed autonomously at every node.

Code $\mathbf{CR}$: If the node is not the origin of p and the attribute of the elected q-route equals or is less preferred than the attribute of the elected p-route, then filter q. Otherwise, do not filter q.

This code is intuitively reasonable as it maintains or improves the attribute of the route according to which data-packets are forwarded at a node. Certainly, the origin of p should not filter q-routes. Otherwise, data-packets arriving there with destination in q would have nowhere to go and would have to be dropped. For a node other than the origin of p, if the attribute of the elected q-route equals that of the elected p-route, then the node filters q. On filtering, the node saves on forwarding state while it still forwards data-packets with destination in q according to an elected route—that for p—whose attribute is the same as that of the elected q-route without filtering. Last, if the attribute of the elected q-route is less preferred than the attribute of the elected p-route, then all the more reason for the node to filter q. On filtering, the node saves on forwarding state and improves the attribute of the route according to which it forwards data-packets with destination in q.

Throughout the paper, we will study the global effect of local code $\mathbf{CR}$. For now, we exemplify that effect with Figure 1 assuming the GR routing policies. Node u_6 acquired its address space from its provider u_4. The acquired address space is represented by prefix q which is more specific than prefix p. Despite this acquisition, u_6 wants to send and receive data-packets to and from both providers u_4 and u_3. Thus, both p and q are propagated by the vector-protocol throughout the network. The stable state is depicted on the left-hand side of the figure and described next alongside the possibility of filtering q upon execution of code $\mathbf{CR}$.

- Node u_4 is the origin of p. Thus, u_4 cannot filter q.
- Node u_6, which is the origin of q, elects a customer q-route (originated by itself) and a provider p-route. Thus, u_6 cannot filter q-routes.
- Node u_3 elects a customer q-route, learned from u_6, and a provider p-route, learned from u_2. Thus, it cannot filter q-routes.
- Node u_2 elects both a customer q-route, learned from u_3 and u_4, and a customer p-route, learned from u_4. Thus, it can filter q-routes.
- Node u_1 elects both a peer q-route and a peer p-route, both routes learned from u_2. Thus, it can filter q-routes.
- Node u_5 elects both a provider q-route and provider p-route, both routes learned from u_1 and u_3. Thus, it can filter q-routes.

Suppose that u_2 executes $\mathbf{CR}$, thereby filtering q. Despite the absence of a q-route, u_2 still forwards data-packets with destination in q according to a customer route, that elected for p. Because u_2 filters q, u_1 no longer receives a q-route from u_2 and, hence, does not elect any q-route. It forwards data-packets with destination in q according to the elected p-route which was also learned from u_2. Since u_1 does not elect a q-route, it exports none to its customer u_5. Node u_5 still elects a provider q-route learned from u_3. In this state, suppose that u_5 executes $\mathbf{CR}$. It, too, filters q and starts forwarding data-packets with destination in q according to the elected provider p-route, learned from u_3 and u_1. In summary, if u_2 then u_5 execute $\mathbf{CR}$, then we arrive at the routing state depicted in the right-hand side of Figure 1 and commented upon next.

- Nodes u_2 and u_5 filter q while u_1 is oblivious of q. We say that a node *forgoes* q if either it filters q or is oblivious of q. In real topologies, most nodes will forgo q. Of these, a few will filter q, while the majority will be oblivious of q. Routing state pertaining to q only needs to be kept in some small vicinity of the node originating q.
- Data-packets are delivered to the destinations, there being no route oscillations, forwarding loops, or black holes. When this happens, we say that DRAGON is *correct* (Section 3.2).
- Data-packets are forwarded at each node according to an elected route whose attribute equals that of the elected route used to forward them when there was no filtering. Such a

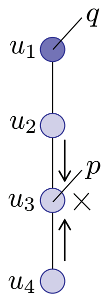

Figure 2: Prefix q is more specific than p. Node u_1 originates q, and u_3 originates p by sending a p-route to all its neighbors. Node u_2 executes $\mathbf{CR}$, in the process creating a black hole at u_3. Arrows indicate the expedition of data-packets with destination in q.

desirable global state is called *route consistent*. A route-consistent state is *optimal* if the set of nodes forgoing q is maximal (Section 3.3). The routing state depicted in the right-hand side of Figure 1 is optimal and can be arrived at by executing $\mathbf{CR}$ once at each node in whatever order.

3.2 Announcement rule and correctness

Through code $\mathbf{CR}$, DRAGON subordinates the computation of q-routes to the computation of p-routes. Therefore, even if the vector-protocol is correct for p and q, taken individually as two unrelated prefixes, it is legitimate to ask whether DRAGON is correct, always delivering data-packets to their destinations. The main concern is that filtering of q by some nodes may create black holes for data-packets with destination in q. In Section 4.2, we prove that the following rule for originating prefixes guarantees correctness of DRAGON.

Rule $\mathbf{RA}$: The origin of p announces p with a route whose attribute is equal or less preferred than the attribute of the elected q-route.

The necessity of rule $\mathbf{RA}$ can be appreciated with the example of Figure 2. Node u_1 is the origin of q and u_3—which a customer of a customer of u_1—is the origin of p. Node u_3 elects a provider q-route. Suppose that it originates p with a customer route, thus violating rule $\mathbf{RA}$: the attribute of the p-route with which u_3 originates p ("learned from a customer") is preferred to the attribute of the elected q-route ("learned from a provider"). Node u_2 elects a provider q-route and a customer p-route. On executing $\mathbf{CR}$, u_2 filters q. As a consequence, no q-route arrives at u_3 and u_4. Data-packets arriving at u_2 and u_4 with destination in q are forwarded to u_3 by the elected p-route to be dropped there. Node u_3 becomes a black hole for q. In order to satisfy rule $\mathbf{RA}$, u_3 can originate p only with a provider route, meaning that it can export p-routes only to its customers; in this case, to node u_4. If u_3 does export a p-route to u_4, then u_4 elects both a provider q-route and a provider p-route. Node u_4 may filter q-routes that data-packets with destination in q will be delivered to u_1.

It must be noted that the assignment of prefixes in Figure 2 is unlikely to be found in the Internet where blocks of IP addresses are delegated from providers to customers rather than the other way round.

3.3 Isotonicity and optimal route-consistency

Routing policies are isotone [8, 32] whenever the relative preference among attributes of elected routes is respected among the attributes of the candidate routes derived from them. Let u and v be two neighbor nodes and α and β be any two attributes such that α is preferred to β. Suppose that elected routes with attributes α and β at v are extended to candidate routes with attributes α' and β' at u, respectively. The combined routing policies of u and v are isotone if α' equals β' or α' is preferred to β'.

The GR routing policies are isotone. For instance, suppose that u is a customer of v. All of a customer route, a peer route, and a provider route at v are exported by v to u, and all become provider routes at u. Thus, isotonicity holds. Suppose, instead, that u is a provider of v. A customer route is preferred to both a peer route and a provider route at v. The customer route is exported by v to u where it becomes a customer route too, whereas the peer route and the provider route are not exported by v to u. Clearly, the customer route at u is preferred to no route. Isotonicity holds as well. A similar argument can be made if u is a peer of v. Many other practical routing policies are isotone. For example, the next-hop routing policies proposed in [31] for inter-domain routing generalize the GR routing policies and are isotone as well. So are the routing policies that incorporate siblings in the landscape of Internet business relationships [23].

A global routing state attained by DRAGON is *route consistent* if it is stable and always forwards data-packets according to an elected route whose attribute is the same as that of the elected route used to forward them without DRAGON. A route-consistent routing state is *optimal* if the set of nodes that forgoes q is maximal. Ideally, DRAGON would lead to optimal route-consistent states and this is exactly what happens if routing policies are isotone and all nodes execute code $\mathbf{CR}$ once in whatever order. A proof of this result is given in Section 4.3. The intuition is the following. With isotonicity, if the attribute of the elected q-route is the same or less preferred than the attribute of the elected p-route at a node v—as in the premise of code $\mathbf{CR}$—then, at a neighbor u of v, the attribute of the q-route learned from v is also the same or less preferred than the attribute of the p-route learned from v. Therefore, the filtering of q at v is consistent with the filtering decision that would be made at u based on routes learned from v.

If routing policies are not isotone, then code $\mathbf{CR}$ may not conduce to route-consistent states. Consider the network of Figure 3 where the GR routing policies are used with two important exceptions.

- Node u_5 prefers routes learned from provider u_3 to routes learned from provider u_1.
- Node u_3 exports provider routes to its customer u_5, but it does not export customer routes to u_5. Thus, the combined routing policies of u_3 and u_5 are *not* isotone. A customer route is preferred to a provider route at u_3, but the former is not exported to u_5 whereas the latter is.

Node u_3 elects a customer q-route and a provider p-route. It exports a p-route to u_5, but it does not export a q-route to u_5. Thus, u_5 elects a p-route learned from its *most preferred* provider, u_3, but a q-route learned from its *least preferred* provider, u_1. The longest prefix match rule directs data-packets arriving at u_5 with destination in q exclusively to u_5's least preferred provider, u_1. Suppose that u_1 executes $\mathbf{CR}$. In this case, u_5 is oblivious of q. It starts forwarding data-packets with destination in q according to the elected p-route which is the one learned from its most preferred provider, u_3. DRAGON changed the attribute of the route according to which data-packets with destination in q are forwarded at u_5, from "learned from least preferred provider" to "learned from most preferred provider." It can be argued that the preference of the route according to which u_5 forwards data-packets with destination

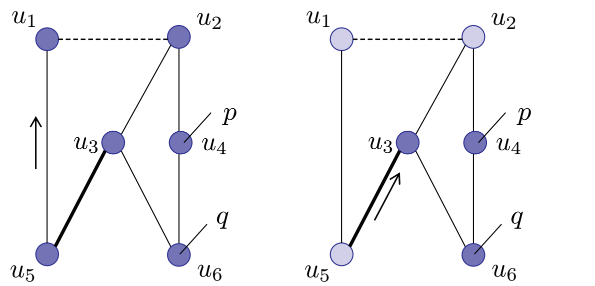

Figure 3: The thicker line means that u_5 prefers provider u_3 to provider u_1. Node u_3 exports only provider routes to u_5. Left. Standard stable state. Right. State after all nodes execute $\mathbf{CR}$. Arrows indicate the expedition of data-packets at u_5 with destination in q. Route consistency is not satisfied.

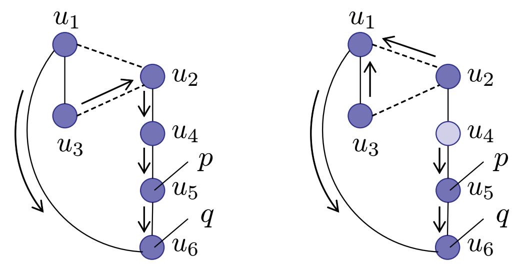

Figure 4: Arrows indicate the expedition of data-packets with destination in q. Left. Initial, standard stable state. Deployment of DRAGON first by u_3, then by u_2, and last by u_4 guarantees route-consistent states at all stages. Right. Filtering of q by u_4 alone decreases the preference of the elected q-routes at u_2 and u_3. This decrease in preference reinforces u_2's and u_3's readiness to filter q.

in q has improved. That is one more counter-intuitive behavior of vector-protocols when routing policies are not isotone [36]. However, the point we are making here is that DRAGON did not lead to a route-consistent state (maybe u_3 agreed with u_5 to forward u_5's data-packets further *except* for those with destination in q).

The following comments summarize the significance of isotonicity for DRAGON.

- The incentive for an individual node to deploy DRAGON is embodied in code $\mathbf{CR}$ and does *not* depend on isotonicity.
- The correctness of DRAGON derives from the correctness of a standard vector-protocol and rule $\mathbf{RA}$, and does *not* depend on isotonicity.
- Isotonicity guarantees an optimal route-consistent state if all nodes adopt DRAGON, by executing code $\mathbf{CR}$, but it does say if that state is *efficient*, in the sense of having many nodes forgoing q. For example, the routing policies that substantiate shortest-path routing are isotone, while it is well-known that we cannot compact routing and forwarding state without stretching distances, in general [19]. In inter-domain routing, the efficiency of DRAGON stems from the hierarchy established by the provider-customer relationships and the alignment, even if imperfect, between this hierarchy and prefix assignment (Section 5).
- Isotonicity guarantees that there is an order for adoption of DRAGON among all nodes that is route-consistent at all stages. This is shown in the next section.

3.4 Partial deployment

DRAGON can be deployed progressively, one node at a time. With isotone routing policies, it is always possible to sequence the adoption of DRAGON so that route-consistency is ensured at all stages of deployment. In the particular case of the GR routing policies, all sequences of adoptions obeying the following general condition ensure route-consistency.

Condition $\mathbf{PD}$: First, execute $\mathbf{CR}$ at nodes that elect either a peer or a provider q-route, *in whatever order*. Next, execute $\mathbf{CR}$ at nodes that elect a customer q-route top-down in the provider-customer hierarchy, that is, only execute $\mathbf{CR}$ at a node that elects a customer q-route after the code has been executed at its providers.

If condition $\mathbf{PD}$ is violated, then there may exist stages of deployment that are not route-consistent. However, these stages entail incentives for nodes to adopt DRAGON. Pursuing these incentives, nodes settle quickly in route-consistent states.

We illustrate with Figure 4. Node u_5 is the origin of p and u_6 is the origin of q. Node u_1 is a provider of both u_3 and u_6. Node u_2 is a peer of both u_1 and u_3. The left-hand side of the figure shows the initial stage, where DRAGON is not deployed at all. At this stage, only u_2, u_3, and u_4 will be able to filter q on executing $\mathbf{CR}$. Node u_3 elects a peer p-route and a peer q-route, whereas u_2 and u_4 both elect a customer p-route and a customer q-route. In order to satisfy condition $\mathbf{PD}$, u_3 is first in adopting DRAGON, executing $\mathbf{CR}$ and filtering q. On doing so, it continues to forward data-packets with destination in q to its peer u_2. Next, u_2 must execute $\mathbf{CR}$ before u_4 because it is a provider of u_4. On executing $\mathbf{CR}$, u_2 filters q while still forwarding data-packets with destination in q to its customer u_4. Last, u_4 executes $\mathbf{CR}$, filters q, and forwards data-packets with destination in q to u_5, the same as without DRAGON. All intermediate stages of deployment are route-consistent.

Back to the initial stage, suppose that u_4, rather than u_3, is first in adopting DRAGON. The resulting state is depicted in the right-hand side of Figure 4. Node u_4 forwards data-packets with destination in q to u_5, as before, but it stops exporting a q-route to u_2. As a consequence, u_2 elects a peer q-route, learned from u_1. It no longer exports a q-route to u_3 whose reaction is to elect a provider q-route, learned as well from u_1. The routing state after u_4 alone adopts DRAGON is not route-consistent. On the other hand, both u_2 and u_3 now have strong incentives to deploy DRAGON because, by doing so, they improve the attribute of the route used to forward data-packets. On executing $\mathbf{CR}$, u_2 forwards data-packets with destination in q to its customer u_4 rather than to its peer u_1; on executing $\mathbf{CR}$, u_3 forwards data-packets with destination in q to its peer u_2 rather than to its provider u_1. In addition, on executing $\mathbf{CR}$, both u_2 and u_3 save on forwarding state.

3.5 Alternative filtering codes

Preserving forwarding neighbors. Code $\mathbf{CR}$ may reduce the multi-path potential of a vector-protocol. For example, in the right-hand side of Figure 1, after all nodes execute $\mathbf{CR}$, u_2 loses u_3 as a neighbor to which it can forward data-packets with destination in q, since u_3 is a forwarding neighbor for q that is not a forwarding neighbor for p. It is possible to tighten code $\mathbf{CR}$ so that it preserves or improves, not only route attributes, but also the sets of forwarding neighbors. Reference [33] outlines such a code for the especial case of the GR routing policies.

Notwithstanding, application of $\mathbf{CR}$ at the router-level, with router-level attributes, provides a multi-path effect at the AS-level which may be sufficient in practice. For example, suppose that nodes in Figure 1 represent ASs containing several routers. On executing $\mathbf{CR}$ at the routers, it likely happens that u_2's border routers connected to u_3 do not filter q whereas those connected to u_4 do filter it. At the AS-level, it is as if data-packets arriving at u_2 with destination in q were forwarded to both u_3 and u_4.

Relaxing AS-paths. Attributes of BGP routes (our use of the term "attributes") can be seen as composed of two component-attributes: those implemented with BGP's parameter LOCAL-PREF, which we will call L-attributes in the remainder of this section; and AS-paths, which exactly correspond to BGP's parameter AS-PATH. L-attributes typically reflect the business relationships between neighbor ASs and take precedence in route election, with the lengths of AS-paths serving as tie-breakers among routes having the same L-attributes.

For filtering, code $\mathbf{CR}$ requires the whole attribute of the elected q-route to equal or be less preferred than the whole attribute of the elected p-route. However, attempting to preserve or improve the lengths of AS-paths does not lead to significant savings in routing state, in general. On the other hand, since AS-paths only play a secondary role to L-attributes in route election and, additionally, they are not even good indicators of network performance [31], we may specialize code $\mathbf{CR}$ to allow for some slack in the lengths of AS-paths. A node other than the origin of p will filter q if and only if: the L-attribute of the elected q-route is less preferred than that of the elected p-route; or the L-attribute of the elected q-route equals that of the elected p-route and the AS-path of the elected q-route is not shorter than that of the elected p-route by more than X links, where $X \geq 0$ becomes a parameter of the filtering strategy. The limiting case of $X = +\infty$ amounts to code $\mathbf{CR}$ applied exclusively to L-attributes.

3.6 Multiple levels of prefixes

A very large number of prefixes at different levels of specificity is announced in the network. We define the *parent* of a prefix q in a set of prefixes as the most specific of the prefixes that are less specific than q in the set. DRAGON operates by having every node contrast each prefix q against its parent prefix in the set of prefixes learned from the vector-protocol, in the same way that q is contrasted against p in code $\mathbf{CR}$.

We can impose that every node executes code $\mathbf{CR}$ on the list of prefixes it learns from the vector-protocol, from the least to the most specific ones. However, the executions of code $\mathbf{CR}$ at different nodes are not correlated in time and may depend on route dynamics outside the control of network operators. Thus, the parent of a prefix may vary from node to node at any given time, and throughout time. Despite the asynchrony, DRAGON remains correct for the same routing policies that make the vector-protocol correct. In addition, if routing policies are isotone, then DRAGON leads to an optimal route-consistent state.

3.7 Aggregation prefixes

Provider-independent (PI) prefixes are those acquired by nodes directly from registrars. These prefixes do not have a parent in the routing system and, as it stands, cannot be filtered. DRAGON promotes the generation of a few aggregation prefixes to allow filtering of many of the PI prefixes. Each node determines autonomously which aggregation prefixes it originates, if any. The specification for an aggregation prefix requires it to be as short as possible without introducing new address space and to be announced with an attribute that respects rule $\mathbf{RA}$. This specification can be realized with a quick algorithm that traverses twice the binary tree of prefixes that has the empty prefix for root and the prefixes without parent for leaves, as they are known locally at a node.

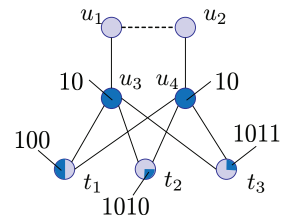

Figure 5: Both u_3 and u_4 originate prefix 10. Both u_1 and u_2 filter prefixes 100, 1010, and 1011.

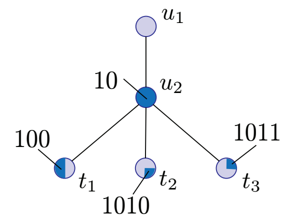

Figure 6: The announcement of 10 by u_2 subsumes that by u_1, and allows u_1 to filter prefixes 100, 1010, and 1011.

DRAGON self-organizes when more than one node originates the same aggregation prefix. We illustrate two distinct aspects of this self-organization. In Figure 5, nodes t_1, t_2, and t_3 were assigned PI prefixes 100, 1010, and 1011, respectively. Both u_3 and u_4 elect customer routes for each of these prefixes. Independently of each other and possibly at different moments in time, both u_3 and u_4 originate aggregation prefix 10. Then, the vector-protocol anycasts to 10 with both u_3 and u_4 knowing how to further data-packets with destination in 10. There is an advantage in having both u_3 and u_4 originate 10. This way, both u_1 and u_2 can filter the PI prefixes. If, say, u_3 alone originated 10, then u_2 would elect a peer 10-route, but customer routes for 100, 1010, and 1011. Node u_2 would not be able to filter any of the PI prefixes.

In Figure 6, nodes t_1, t_2, and t_3 were again allocated PI prefixes 100, 1010, and 1011, respectively. Suppose that u_1 originates aggregation prefix 10 while u_2 does not. Therefore, u_2 elects a provider 10-route, learned from u_1, but customer routes for all the PI prefixes. Node u_2 cannot filter any of the latter. However, it, too, can originate 10. Suppose it does originate 10. As a consequence, u_1 learns a customer 10-route from u_2, elects that route, and stops originating 10. Because u_1 also elects a customer route for each of the PI prefixes, it may filter them. Any data-packet arriving at u_1 with destination in 10 is forwarded to u_2 which delivers it to the appropriate customer.

3.8 Network dynamics

DRAGON reacts automatically to network events, such as link failures or additions. We illustrate that reaction calling again upon the network of Figure 1 and starting from the route-consistent state depicted on its right-hand side. The failure of a link that does not affect the election of a customer q-route at the origin of p, u_4, is handled solely by code $\mathbf{CR}$. For instance, suppose that two-way link $\{u_3, u_6\}$ fails. Consequently, the elected q-route at u_3 changes from customer to provider, the latter route learned from u_2. Since u_3 also elects a provider p-route, it now filters q on executing $\mathbf{CR}$.

Suppose, instead, that two-way link $\{u_4, u_6\}$ fails. Node u_4 no longer elects a customer q-route. Hence, it cannot announce p

with a customer route as such an action would violate rule **RA**. Rather, u_4 de-aggregates p into longer prefixes that can be announced with customer routes. For instance, suppose that $p = 10$ and $q = 10000$. Node u_4 withdraws p and announces the three prefixes 10001, 1001, and 101, which together with the missing prefix $q = 10000$ partition the address space of p. Node u_2 elects customer routes for 10001, 1001, and 101, learned from u_4. Through u_3, it also elects a customer 10000-route. Therefore, u_2 no longer filters q, but rather pieces together the prefixes 10001, 1001, 101, and $q = 10000$ to originate aggregation prefix $p = 10$. In global terms, the failure of $\{u_4, u_6\}$ moved the origin of p upwards in the provider-customer hierarchy, from u_4 to its provider u_2. Node u_1 elects a peer p-route and a peer q-route, filtering q on executing **CR**. Similarly, u_5 elects a provider p-route and a provider q-route, filtering q on executing **CR**.

In practice, if u_4 assigned q to u_6 and two-way link $\{u_4, u_6\}$ fails, then u_4 would likely wait some time for two-way link $\{u_4, u_6\}$ to be repaired before de-aggregating p.

3.9 Traffic engineering

Customers connected to multiple providers sometimes perform in-bound traffic engineering by de-aggregating their assigned prefixes and announcing different sub-prefixes to different providers [9]. DRAGON allows for route consistent filtering of the sub-prefixes.

We illustrate with Figure 7. Node u_7 is a customer of both u_4 and u_5. It was assigned prefix p. In order to balance its incoming traffic between its two providers, u_7 de-aggregates prefix p into sub-prefixes $p0$ and $p1$. It announces p and $p0$ to u_4, while it announces p and $p1$ to u_5. All data-packets with destination in $p0$ arrive at u_7 via its provider u_4, and all data-packets with destination in $p1$ arrive at u_7 via its provider u_5. The left-hand side of the figure shows the flow of data-packets with destination in $p0$. Note that u_5 forwards data-packets with destination in $p0$ to its own providers u_1 and u_2, although the block of addresses represented by $p0$ belongs to its customer u_7 (!). Node u_5 could just thwart the intention of its customer u_7 by filtering $p0$-routes, so that all data-packets arriving at u_5 with destination in $p0$ would be forwarded directly to u_7.

Thus, we assume that the traffic engineering intention of u_7 is respected by its providers. Specifically:

- If a provider of u_7 elects both a $p0$-route and a $p1$-route, then it announces p according to rule **RA**.
- If a provider of u_7 learns a p-route from a neighbor other than u_7, then it refrains from electing the customer p-route learned from u_7.

Node u_4 elects a customer $p0$-route and a provider $p1$-route. Hence, in order to obey rule **RA**, u_4 announces p with a provider route, exporting a p-route only to its customers; in this case, to u_6. Node u_6 elects both a provider $p0$-route and a provider p-route. Executing **CR** on $p0$, u_6 filters $p0$. The situation at u_5 is analogous. It originates a provider p-route that it exports to its customer u_8. Node u_8 elects both a provider $p0$-route and a provider p-route. Executing **CR** on $p0$, u_8 filters $p0$.

If nothing else were done, only u_6 and u_8 would be able to filter $p0$. However, note that u_1 elects both a customer $p0$-route and a customer $p1$-route. Therefore, u_1 can originate aggregation prefix p with a customer p-route, announcing p to all its neighbors. Suppose it does so. Node u_3 now elects both a provider $p0$-route and a provider p-route. It filters $p0$ after executing **CR** on $p0$. Node u_2 elects a peer $p0$-route and a peer p-route. It, too, filters $p0$ after executing **CR** on $p0$. Node u_5 learns a p-route from u_1 and

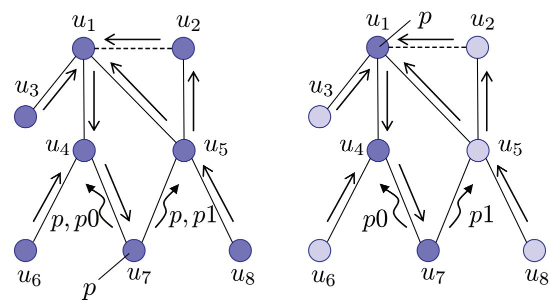

Figure 7: Node u_7 is a multi-homed to u_4 and u_5. It is assigned prefix p which it de-aggregates into prefixes $p0$ and $p1$. Node u_7 announces p and $p0$ to u_4, and p and $p1$ to u_5. Wiggled arrows indicate prefix announcements and straight arrows indicate the expedition of data-packets with destination in $p0$. Left. Standard stable state for $p0$. Right. Route consistent state for $p0$ with u_1 originating p and light-shaded nodes forgoing $p0$.

u_2, thereby electing a provider p-route. If u_5 now executes **CR** on $p0$, it filters $p0$, because it elects both a provider $p0$-route and a provider p-route. The resulting routing state for prefix $p0$ is shown at the right-hand side of Figure 7. The routes used to forward data-packets with destination in $p0$ are the same as without DRAGON. Route-consistency is satisfied.

If two-way link $\{u_4, u_7\}$ fails, then $p0$ is eliminated from the network, u_1 no longer originates p, and the only p-route that u_5 learns is the one from u_7. Thus, u_5 elects a customer p-route, learned from u_7, announcing p to all its neighbors. All data-packets with destination in $p0$ are guided by elected p-routes, ending up in u_7 via u_5.

4. THEORY OF DRAGON

In Section 4.1, we briefly review the algebraic framework of [32] which allows us to reason with generality about vector-protocols and about DRAGON. Based on this framework, we sketch the proof of correctness of DRAGON in Section 4.2, and the proof of optimality of DRAGON under isotonicity in Section 4.3.

4.1 Correctness of vector-protocols

The set of attributes is denoted by Σ and their order by $\preceq$. For $\alpha, \beta \in \Sigma$, the inequality $\alpha \prec \beta$—equivalently, $\beta \succ \alpha$—means that α is preferred to β. The especial attribute $\bullet$ denotes unreachability, being the least preferred of all attributes. For simplicity, we assumed that routing policies do not depend on prefixes. The transformations of attributes as routes propagate in a network are modeled by maps on Σ. A map L on Σ such that $L(\bullet) = \bullet$ is called a *label*. Links point in the direction of traffic flow. Each link uv is associated with a label $L[uv]$ telling how the attribute α of a route elected at v extends into the attribute $L[uv](\alpha)$ of a candidate route at u. The label of a walk $P = u_n u_{n-1} \cdots u_1 u_0$, denoted by $L[P]$, is obtained through composition of the labels of its constituent links: $L[P] = L[u_n u_{n-1}] \cdots L[u_1 u_0]$.

Cycle $C = u_0 u_{n-1} \cdots u_0$ is *strictly absorbent* if

$$\forall_{\alpha_0 \prec \bullet, \ldots, \alpha_{n-1} \prec \bullet} \ \exists_{0 \leq i < n} \ \alpha_{i+1} \prec L[u_{i+1} u_i](\alpha_i), \qquad (1)$$

with subscripts interpreted modulus n. Intuitively, C is strictly absorbent if whatever the attributes of routes learned by each node *externally* to C at least one of the nodes prefers the attribute of that route to the attribute of the route it learns from its neighbor around C. This suggests that the propagation of routes around C eventually subsides and that once a stable state is reached C does not contain a forwarding loop. Indeed, the following theorem is proven in [32].

Theorem 1 *If all cycles in a network are strictly absorbent, then a vector-protocol is correct in that network.*

We assume that all cycles in every network are strictly absorbent. The origin of p is denoted by t^p. It announces a p-route with an attribute denoted by $R^*[t^p;p]$. Let $R[u;p]$ be the attribute of the p-route elected at u in the stable state ($R[u;p] = \bullet$ if u does not elect a p-route). Node t^p elects the p-route it announced, $R[t^p;p] = R^*[t^p;p]$, and every node u other than t^p elects the candidate p-route with the most preferred attribute. If v is a forwarding neighbor of u for p, then $R[u;p] = L[uv](R[v;p])$; otherwise, if v is a neighbor of u, but not a forwarding neighbor of u for p, then $R[u;p] \prec L[uv](R[v;p])$. A forwarding path for p is a path ending at t^p that joins every node, other than t^p, to one of the node's forwarding neighbors for p.

4.2 Correctness of DRAGON

Given cycle C and two distinct nodes of the cycle, u and v, uCv denotes the unique path in C from u to v. The next lemma states an easy, but useful, implication of strict-cycle-absorbency. The proof is omitted.

Lemma 1 *Suppose cycle C is strictly absorbent. Let u and v be two distinct nodes of C and $\alpha_u \prec \bullet$ and $\alpha_v \prec \bullet$ be two attributes. Then, either $\alpha_u \prec L[uCv](\alpha_v)$ or $\alpha_v \prec L[vCu](\alpha_u)$ (or both).*

We consider two prefixes p and q such that q is less specific than p.

Theorem 2 *If all cycles in a network are strictly absorbent, then DRAGON is correct in that network whatever the set of nodes executing* $\mathbf{CR}$.

PROOF. We divide the proof into three claims.

Claim 1*: DRAGON terminates*. The vector-protocol terminates for prefix p, Theorem 1. Some nodes execute $\mathbf{CR}$ which may lead them to filter q. Filtering does not compromise termination for prefix q.

Claim 2*: DRAGON does not yield forwarding loops.* The stable states for p and q do not contain forwarding loops, Theorem 1. A data-packet with destination in q may be forwarded from a node that *does not* elect a q-route to one that does, but never back. Once a data-packet reaches a node that elects a q-route, it is guided all the way along a forwarding path for q.

Claim 3*: DRAGON does not yield black holes.* A black hole could only exist at t^p, and only if t^p did not elect a q-route. We will show that every node on a forwarding path for q starting at t^p, other than t^p itself, elects a q-route whose attribute is *preferred* to that of the elected p-route. Hence, even if the node executes $\mathbf{CR}$, it does not filter q. Rather, it keeps announcing q-routes according to the rules of the vector-protocol, so that t^p always elects a q-route.

In order to arrive at a contradiction, assume that there is a node in a forwarding path P for q starting at t^p such that the attribute of the elected q-route equals or is less preferred than the attribute of the elected p-route. Let u be the first such node along P. Hence, by hypothesis, we have

$$R[u;q] \succeq R[u;p]. \tag{2}$$

Let Q be a forwarding path for p starting at u and let v be the first node along Q that meets P. We distinguish two cases. If $v = t^p$, then, from rule $\mathbf{RA}$, we obtain $R^*[v;p] = R[v;p] \succeq R[v;q]$. Otherwise, if $v \neq t^p$, then since u is the first node along P such that $R[u;q] \succeq R[u;p]$, we must have $R[v;q] \prec R[v;p]$. In both cases,

$$R[v;p] \succeq R[v;q]. \tag{3}$$

Path vPu is the sub-path of P running from v to u and uQv is the sub-path of Q running from u to v. The union of vPu and uQv is a cycle, denoted by $uQvPu$. Because P is a forwarding path for q, we write $R[v;q] = L[vPu](R[u;q])$; because Q is a forwarding path for p, we write $R[u;p] = L[uQv](R[v;p])$. With $\alpha_u = R[u;q]$ and $\alpha_v = R[v;p]$, Inequalities (2) and (3) become $\alpha_u \succeq L[uQv](\alpha_v)$ and $\alpha_v \succeq L[vPu](\alpha_u)$, respectively. From Lemma 1, we conclude that $uQvPu$ is not strictly absorbent, contradicting the premise of the theorem. Thus, the attribute of the elected q-route is preferred to the attribute of the elected p-route at any node along a forwarding path for q starting at t^p. $\square$

4.3 Optimal route-consistency

A label L is isotone if $\alpha \preceq \beta$ implies $L(\alpha) \preceq L(\beta)$ for all attributes α and β. A link, or a walk, is isotone if its label is isotone. A walk all links of which are isotone is itself isotone. Isotonicity confers optimality to the attributes of routes elected at the various nodes. The next theorem is proven in [32].

Theorem 3 *If all links in a network are isotone, then the attribute of an elected route at a node equals or is preferred to the attribute of any route propagated to the node from its origin along a walk in that network.*

In symbols, the previous theorem states that, for all nodes u, all prefixes p, and all walks P from u to t^p, we have

$$R[u;p] \preceq L[P](R^*[t^p;p]). \tag{4}$$

Theorem 4 *Suppose that all links in a network are isotone. Then, DRAGON attains the optimal route-consistency state once all nodes execute* $\mathbf{CR}$ *in whatever order.*

PROOF. We divide the proof into four claims, the first two of which assert properties of the stable state previous to the execution of DRAGON.

Claim 1: *At any node u, the attribute of the elected q-route is the same or preferred to the attribute of the elected p-route.*

In symbols, we need to show that $R[u;q] \preceq R[u;p]$, for all nodes u and prefixes p. Let P be a forwarding path for q starting at u; Q be a forwarding path for p starting at u; and T be a forwarding path for q starting at t^p. Denote by Qt^pT the walk composed of path Q followed by path T. From rule $\mathbf{RA}$, we have

$$L[T](R^*[t^q;q]) = R[t^p;q] \preceq R^*[t^p;p].$$

Applying label $L[Q]$ to the previous inequality yields, due to isotonicity,

$$L[Qt^pT](R^*[t^q;q]) \preceq L[Q](R^*[t^p;p]) = R[u;p]. \tag{5}$$

Hence,

$$\begin{aligned} R[u;q] &\preceq L[Qt^pT](R^*[t^q;q]) && \text{(from Theorem 3)} \\ &= R[u;p]. && \text{(from (5))} \end{aligned}$$

Claim 2: *Let E be the set of nodes, not containing t^p, for which the attribute of the elected q-route equals that of the elected p-route. If u does not belong to E neither does any of its forwarding neighbors for q.*

We prove the contrapositive statement. Let v be a forwarding neighbor of u for q, $R[u;q] = L[uv](R[v;q])$, and assume that v belongs to E, $R[v;q] = R[v;p]$. Thus,

$$R[u;p] \preceq L[uv](R[v;p]) = L[uv](R[v;q]) = R[u;q].$$

From Claim 1, we know that $R[u;q] \preceq R[u;p]$. Thus, $R[u;q] = R[u;p]$, meaning that u belongs to E as well.

Claim 3: *Set E is the optimal set of nodes that can forgo q while maintaining route-consistency.*

Nodes in E, and only those, can forgo q while preserving the attribute of the route according to which they forward data-packets with destination in q. Moreover, from Claim 2, filtering of q by nodes in any subset of E does not affect the elected q-routes of nodes *not* in E.

Claim 4: *Suppose that an arbitrary subset of nodes of E filter q and let S be the subset of nodes of E that do not forgo q. Any node u in S will filter q on executing* **CR***.*

A node that filters q is a node that can be removed from the network as far as the computation of elected q-routes for all other nodes is concerned. Removal of nodes reduces the set of paths in the network. Invoking Theorem 3, we conclude that the attribute of the elected q-route at any node either remains the same or becomes less preferred. Hence, the premise of **CR** remains valid at any node in S. ☐

5. EVALUATION

We evaluate the scalability benefits brought to inter-domain routing when all ASs deploy DRAGON. Section 5.1 describes the network topologies, prefix assignments, and routing policies we used. The savings in routing and forwarding state achieved by DRAGON are presented in Section 5.2, and the impact on convergence is discussed in Section 5.3.

5.1 Methodology and datasets

We ran DRAGON on inferred Internet topologies provided by UCLA [2] and lists of IP prefixes originated by each AS collected by CAIDA [3]. In the inferred topologies, each link is labeled as provider-to-customer, customer-to-provider, or peer-to-peer. Accordingly, we assumed the GR routing policies. We used data from November 2012 and September 2013, but only consider the 2013 results in the following as the findings are almost identical in the two cases.

Fixing inaccuracies in the datasets. We first fixed inaccuracies usually found in the data [30]. Regarding the Internet topology, we broke every customer-provider cycle and ensured that the topology is policy-connected [33] —meaning that there is a valid path from every AS to every other—by removing ASs that prevented this from happening. From 46,455 ASs and 184,024 links, we ended up with 39,193 ASs (keeping 84% of them) and 165,235 links (keeping 90% of them). Most of these ASs (84% of them) are at the perimeter of the provider-customer hierarchy having no customers. These ASs are called stubs. Regarding the prefixes, we removed any prefixes originated by multiple ASs and those whose parent prefix is not originated by a direct or indirect provider. From 491,936 prefixes, we ended up with 433,244 prefixes (keeping 88% of them). The median number of prefixes originated by an AS is 2, with a 95-th percentile of 33 prefixes and a 99-th percentile of 159 prefixes.

Accounting for missing peering links. Inferred Internet topologies typically underestimate the number of peering links [5]. In order to compensate this distortion, we experimented introducing peering links between ASs connected at common Internet Exchange

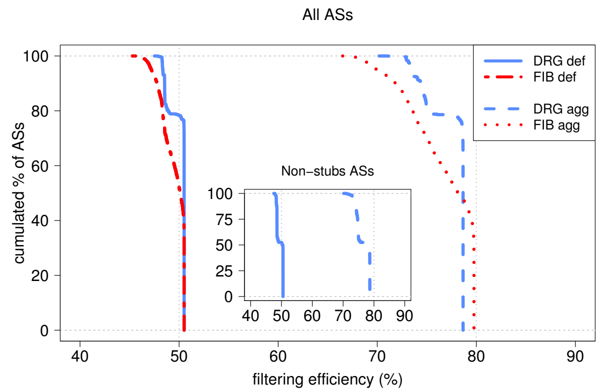

Figure 8: CCDF of filtering efficiency. Around 80% of the ASs reach the maximum filtering efficiencies of 50% (without aggregation prefixes, `DRG def`) and 79% (with aggregation prefixes, `DRG agg`). Inset. CCDF of filtering efficiency of non-stubs only, exhibiting a similar behavior.

Points, following the approach used in [24]. We found that the performance of DRAGON was only marginally affected with median values within less than 1% of the original ones. In hindsight, this is reasonable. The performance of DRAGON mostly results from the alignment between the provider-customer hierarchy and the assignment of prefixes, and, therefore, it is not significantly impacted by peering links.

Introducing aggregation prefixes. Half of the prefixes in our dataset are PI prefixes that do not have a parent prefix in the routing system. In order to subject these prefixes to filtering, we introduced around 45,000 aggregation prefixes subject to the conditions specified in Section 3.7, increasing the total number of prefixes by ~11%. Among all ASs, 8% originate at least one aggregation prefix, with a median value of 3 prefixes, a 95th (resp. 99th) percentile of 66 (resp. 306) prefixes.

5.2 Filtering efficiency

We define *filtering efficiency* as the normalized difference between the number of entries in the forwarding-table of an AS before and after DRAGON is deployed on all ASs.[3] The results for routing-tables mimic those for forwarding-tables and are not referred to explicitly. Without aggregation prefixes, the maximum possible filtering efficiency is 50% as half of the prefixes do not have a parent. With aggregation prefixes, the maximum filtering efficiency rises to 79%.

Figure 8 plots the filtering efficiency according to whether or not aggregation prefixes are introduced (`DRG` curves). Results are presented as Complementary Cumulative Distribution Functions (CCDFs). A point (x, y) of a curve means that y% of the ASs have a filtering efficiency of more than x%. The main plot depicts the filtering efficiency for *all* ASs, whereas the inset focuses on *non-stubs*.

DRAGON enables near-maximum filtering results for each AS. We first consider the case without aggregation prefixes. Every AS forgoes more than 47.5% of the prefixes. This is partially justified by the fact that for a large number of prefixes having a parent (83% of them), the prefix and its parent have the same origin AS [9, 4].

[3]Equivalently, it is the difference between the number of prefixes forgone and the number of aggregation prefixes introduced divided by the number prefixes in the original routing system.

Thus, these prefixes are immediately filtered by the neighbors of those common ASs and become oblivious to the rest of the Internet. Note, however, that DRAGON finds filtering opportunities for all prefixes whatever their origin.

Moreover, DRAGON enables close to 80% of the ASs to realize the maximum possible filtering efficiency of 50%. This is not surprising since the topology is policy-connected and the majority of ASs are stubs. When the topology is policy-connected, stubs *without peers* retain only the parent prefixes. Interestingly, the subplot in Figure 8 asserts the good performance of DRAGON *even* for non-stubs, where around 50% of them still attain the maximum possible filtering efficiency.

As expected, aggregation prefixes improve performance. With aggregation prefixes, every AS has a filtering efficiency of more than 70% with close to 80% of the ASs attaining the maximum filtering efficiency of 79%. It is rarely the case that the origin AS of an aggregation prefix coincides with the origin ASs of the prefixes it covers, that is, of the prefixes that have the aggregation prefix for parent. On the other hand, the algorithm to generate aggregation prefixes ensures that their origin ASs are as close as possible, in terms of the provider-customer hierarchy, to the origin ASs of the covered prefixes. Consequently, although the covered prefixes are not necessarily filtered by the neighbors of their origin ASs, they are filtered along a small vicinity of those ASs, justifying the good performance of DRAGON.

DRAGON performs better than traditional FIB compression techniques for the majority of the ASs. FIB compression refers to algorithms run locally at each AS with the purpose of reducing the size of forwarding-tables without change in the forwarding of data-packets [38, 35, 29]. FIB compression does not affect routing-tables or the dynamics of BGP. Next to the curves for DRAGON, Figure 8 plots the filtering efficiency obtained when running a typical FIB compression algorithm [38] on our dataset (`FIB` curves), using the code provided by the authors.

Like DRAGON, FIB compression allows aggregation prefixes to be introduced. Without aggregation prefixes, DRAGON *performs better* than FIB compression on the majority of the ASs, and at least as good on all of them. This is because DRAGON is more relaxed on its filtering assumptions. DRAGON keeps only the attributes of routes used to forward data-packets whereas FIB compression preserves the exact forwarding neighbors, ties broken by the length of AS-PATH.

With aggregation prefixes, FIB compression can perform slightly better than DRAGON (~1% better). This is because the introduction of aggregation prefixes is not optimized in DRAGON. An AS originates an aggregation prefix only if it elects customer routes for all covered prefixes. There are PI prefixes whose address space could be aggregated except that no AS elects a customer route for all of them. Hence, no AS originates an aggregation prefix for those PI prefixes, preventing them from being subject to filtering. In contrast, FIB compression does not care about elected routes when it aggregates prefixes. It is possible to optimize the introduction of aggregation prefixes in DRAGON in order to match the filtering efficiency of FIB compression. An AS could originate an aggregation prefix covering prefixes for which it elects a peer or a provider route as long as the aggregation prefix is announced with a provider route, meaning that it is announced only to the customers of the AS, so as to satisfy rule $\mathbf{RA}$.

5.3 Convergence upon link failures

We implemented a DRAGON simulator on top of SimBGP [28], an event-driven simulator for BGP. Using our simulator, we compare the transient behavior of DRAGON against that of standard BGP upon link failures. We focus on link failures as they are more demanding on the routing system than link additions [20]. For simplicity, we do not consider the case where new aggregation prefixes are introduced.

We run our simulations on *prefix-trees*. A prefix-tree is a subset of the prefixes of the routing system composed of a prefix without a parent and all less specific prefixes. The dynamics of DRAGON is independent from one prefix-tree to another. The majority of the prefix-trees are trivial, containing a single prefix. For them, DRAGON and BGP share the same convergence behavior as filtering is impossible. Since we are interested in understanding the difference between DRAGON and BGP, we only consider non-trivial prefix-trees. There are 25,266 of them having a median of 5 prefixes. We present results for 250 randomly selected non-trivial prefix-trees.

For each non-trivial prefix-tree, we distinguish between link failures that trigger de-aggregation of the root of the prefix-tree and those that do not (see Section 3.8). The probability of a link failure triggering de-aggregation is small. In the worst case, only 0.03% of all link failures can cause the root of the prefix-tree to be de-aggregated. For each prefix-tree, we *exhaustively* fail all links that can cause de-aggregation. For the rest of the links, we run 4,000 independent trials, each trial corresponding to a link failure selected randomly. In both cases, we measure the total number of routes (advertisements and withdrawals) exchanged network-wide after the failure until a new stable state is reached. We used the default Minimum Route Advertisement Interval (MRAI) value of 30 seconds.

Figure 9 plots the results. On the left, we show the CCDF of the number of routes exchanged network-wide for link failures that do not cause de-aggregation. On the right, we show the CCDF for those link failures that cause de-aggregation.

DRAGON exchanges fewer routes than BGP upon link failures. For non-trivial prefix-trees, DRAGON exchanges less routes than BGP in 95% of the cases and less than half those exchanged with BGP in more than 50% of the cases.

When de-aggregation is not required (99.97% of the failures), a link failure in DRAGON produces, at worst, a network-wide effect for the prefix at the root of the prefix-tree and only a local effect for each of the other prefixes. In contrast, in BGP, all prefixes in the prefix-tree are prone to network-wide effects. The left plot in Figure 9 corroborates this. DRAGON generates more than 100 routes for only 5% of the cases whereas BGP generates more than 100 routes for more than 15% of the cases. Furthermore, DRAGON does not generate *any* route for 40% of the link failures, while BGP generates routes for more than 98% of the link failures.

When a link failure causes de-aggregation (0.03% of the failures), DRAGON can generate more routes than BGP. The right plot in Figure 9 shows that DRAGON announces more routes in 60% of the cases. The number of de-aggregated prefixes originated by an AS for a given prefix-tree is bounded by the difference in length between the most and the least specific prefixes in that prefix-tree. Consistent with this observation, the plot shows that the number of routes exchanged by DRAGON never exceeds that exchanged by BGP by more than one order of magnitude. While the results are already good, we believe that they can be improved further by ensuring that the combination of de-aggregates into an aggregation prefix at a different AS (see Section 3.8) occurs before the de-aggregates are propagated network-wide.

6. RELATED WORK

Scalability limits of routing. The study of the scalability limits of routing has a long and rich history [18, 34, 19]. However,

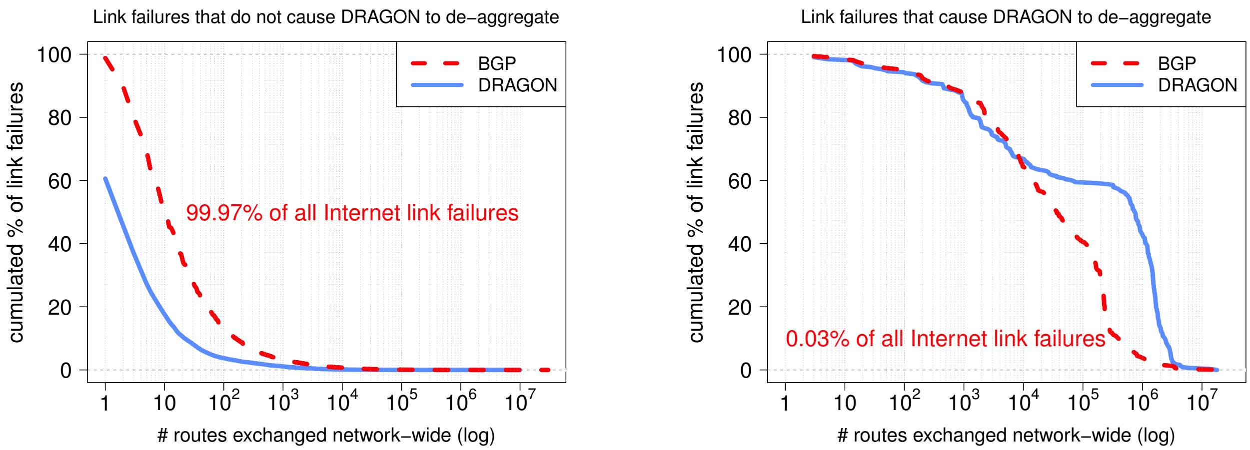

Figure 9: In 95% of the cases, BGP exchanges more routes than DRAGON for non-trivial prefix-trees. When a failure does not cause de-aggregation (99.97% of the cases), DRAGON (resp., BGP) sends more than 100 routes for 5% (resp. 15%) of the cases (left plot). When a failure causes de-aggregation (0.03% of the cases), DRAGON can generate more routes than BGP, but never more than one order of magnitude (right plot).

almost all of the scalability work is premised on shortest-path routing, showing a fundamental trade-off between the sizes of forwarding tables and the stretch in the lengths of paths followed by data-packets in relation to distances. A recent paper embarks on the scalability limits of policy-based routing [14], suggesting that the export rules of the GR routing policies lead to efficient routing, a result that is consistent with our findings. In contrast to the routing strategies of [14], DRAGON is a distributed algorithm proposed for the Internet's existing IP addressing scheme and has been framed for arbitrary routing policies.

Characterizing growth of the routing system. Measurement studies track the growth in the number of IP prefixes and BGP update messages over time [1, 16], and identify the *causes* of growth such as multi-homing, traffic engineering, and address-allocation policies [26, 7, 25, 11]. DRAGON *reduces* the number of prefixes and update messages.

Reducing forwarding-table size. As the number of IP prefixes grew, researchers explored ways to compress the *forwarding-tables* by aggregating related entries [38, 35, 29] or directing some traffic to routers with room for larger tables [6]. These optimization techniques do not reduce the size of *routing-tables* or the number of BGP update messages, and require re-optimization when forwarding decisions change.

Reducing routing-table size. Best current practices for reducing the size of routing-tables rely on the diligence of network operators to apply static filters to BGP routes learned from each neighbor. However, these techniques cannot aggregate routes originating several hops away, and can sometimes lead to black holes and loops [21]. Other techniques for reducing the size of routing-tables [17] work only within a single AS, missing opportunities for global scalability gains.

Inter-domain route aggregation. Our recent short paper [33] presented filtering strategies to scale the Internet routing system, assuming the GR routing policies and only one parent prefix. One of these strategies applies exclusively to nodes that elect provider routes whereas the other requires changes to both the control- and data-planes. In the present paper, the filtering strategy and accompanying announcement rule are valid for general routing policies and arbitrary levels of prefixes, apply to all nodes, and operate exclusively at the control-plane. In addition, in the present paper, we build the filtering strategy into the design of DRAGON, addressing route-consistency, aggregation prefixes, partial deployment, traffic engineering, and network dynamics.

Clean-slate architectures. Many "clean-slate" routing architectures have been proposed (see [27] for a survey) that route on AS numbers or loose source routes. However, these architectures require a major overhaul of Internet routing, whereas DRAGON works with today's BGP.

7. CONCLUSIONS

DRAGON is a distributed route-aggregation algorithm that operates with standard routes of a vector-protocol. It comprises a filtering code and an announcement rule that together allow nodes to forgo prefixes while avoiding black holes. DRAGON applies to any routing policies, but if these policies are isotone, then DRAGON leads to an optimal route-consistent state, reachable through intermediate stages of deployment all of which are route-consistent.

In inter-domain routing, DRAGON harnesses the existing alignment between prefix allocation and the provider-customer Internet hierarchy to produce a very efficient routing system. Evaluation of DRAGON on inferred topologies of the Internet show reductions in the amount of forwarding and routing state on the order of 80% and a parallel decrease in the route activity needed to deal with network events.

8. ACKNOWLEDGMENTS

We would like to thank Sharon Goldberg, Michael Schapira, Stefano Vissicchio, Lucca Cittadini, and José Brázio for their feedback on this work. João's research was supported in part by Fundação para a Ciência e a Tecnologia (FCT), Portugal, under project PEst-OE/EEI/LA0008/2013. Franck's research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Laurent's and Jennifer's research was supported in part by the National Science Foundation under grant CNS-1409056.

9. REFERENCES

[1] BGP routing table analysis reports. bgp.potaroo.net.

[2] Internet AS-level Topology Archive. irl.cs.ucla.edu/topology.

[3] Routeviews Prefix to AS mappings Dataset. www.caida.org/data/routing/routeviews-prefix2as.xml.

[4] CIDR report, 2014. www.cidr-report.org/as2.0.

[5] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a large European IXP. In *ACM SIGCOMM*, pages 163–174, August 2012.

[6] H. Ballani, P. Francis, T. Cao, and J. Wang. Making routers last longer with ViAggre. In *USENIX NSDI*, pages 453–466, April 2009.

[7] T. Bu, L. Gao, and D. Towsley. On characterizing BGP routing table growth. *Computer Networks*, 45(1):45–54, May 2004.

[8] B. Carré. *Graphs and Networks*. Clarendon Press, Oxford, UK, 1979.

[9] L. Cittadini, W. Muhlbauer, S. Uhlig, R. Bush, P. Francois, and O. Maennel. Evolution of Internet address space deaggregation: Myths and reality. *IEEE Journal on Selected Areas in Communications*, 28(8):1238–1249, October 2010.

[10] C. Edwards. Internet routing failures bring architecture change back to the table. *ACM News*, September 2014.

[11] A. Elmokashfi, A. Kvalbein, and C. Dovrolis. On the scalability of BGP: The role of topology growth. *IEEE Journal on Selected Areas in Communications*, 28(8):1250–1261, October 2010.

[12] V. Fuller and T. Li. Classless inter-domain routing (CIDR): The Internet address assignment and aggregation plan. RFC 4632, August 2006.

[13] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692, December 2001.

[14] A. Gulyas, G. Retvari, Z. Heszberger, and R. Agarwal. On the scalability of routing with policies. *IEEE/ACM Transactions on Networking*, 2014. Early access through IEEE Xplore.

[15] Z. B. Houidi, M. Meulle, and R. Teixeira. Understanding slow BGP routing table transfers. In *Internet Measurement Conference*, pages 350–355, November 2009.

[16] G. Huston. Analyzing the Internet's BGP routing table. *The Internet Protocol Journal*, 4(1), March 2001.

[17] E. Karpilovsky, M. Caesar, J. Rexford, A. Shaikh, and J. van der Merwe. Practical network-wide compression of IP routing tables. *IEEE Transactions on Network and Service Management*, 9(4):446–458, December 2012.

[18] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks: Performance evaluation and optimization. *Computer Networks*, 1:155–170, January 1977.

[19] D. Krioukov, kc claffy, K. Fall, and A. Brady. On compact routing for the Internet. *ACM SIGCOMM Computer Communications Review*, 37(3):41–52, July 2007.

[20] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *ACM SIGCOMM*, pages 175–187, August 2000.

[21] F. Le, G. G. Xie, and H. Zhang. On route aggregation. In *ACM CoNEXT*, December 2011.

[22] R. Lemos. Internet routers hitting 512K limit, some become unreliable, August 2014. ArsTechnica, ars.to/1r9AbxJ.

[23] Y. Liao, L. Gao, R. Guérin, and Z.-L. Zhang. Safe interdomain routing under diverse commercial agreements. *IEEE/ACM Transactions on Networking*, 18(6):1829–1840, December 2010.

[24] R. Lychev, S. Goldberg, and M. Schapira. BGP security in partial deployment: Is the juice worth the squeeze? In *ACM SIGCOMM*, pages 171–182, August 2013.

[25] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 address allocation and the BGP routing table evolution. *ACM SIGCOMM Computer Communications Review*, 35(1):71–80, January 2005.

[26] H. Narayan, R. Govindan, and G. Varghese. The impact of address allocation and routing on the structure and implementation of routing tables. In *ACM SIGCOMM*, pages 125–136, August 2003.

[27] J. Pan, S. Paul, and R. Jain. A survey of the research on future Internet architectures. *IEEE Communications Magazine*, 49(7):26–36, July 2011.

[28] J. Qiu. SimBGP: Python Event-driven BGP simulator. www.bgpvista.com/simbgp.php.

[29] G. Retvari, J. Tapolcai, A. Korosi, A. Majdan, and Z. Heszberger. Compressing IP forwarding tables: Towards entropy bounds and beyond. In *ACM SIGCOMM*, pages 111–122, August 2013.

[30] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 lessons from 10 years of measuring and modeling the Internet's autonomous systems. *IEEE Journal on Selected Areas in Communications*, 29(9):1810–1821, October 2011.

[31] M. Schapira, Y. Zhu, and J. Rexford. Putting BGP on the right path: A case for next-hop routing. In *ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 3:1–3:6, October 2010.

[32] J. L. Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Transactions on Networking*, 13(5):1160–1173, October 2005.

[33] J. L. Sobrinho and F. Le. A fresh look at inter-domain route aggregation. In *IEEE INFOCOM Mini-Conference*, pages 2556–2560, April 2012.

[34] P. F. Tsuchiya. The Landmark hierarchy: A new hierarchy for routing in very large networks. In *ACM SIGCOMM*, pages 35–42, August 1988.

[35] Z. Uzmi, M. Nebel, A. Tariq, S. Jawad, R. Chen, A. Shaikh, J. Wang, and P. Francis. SMALTA: Practical and near-optimal FIB aggregation. In *ACM CoNEXT*, 2011.

[36] S. Vissicchio, L. Cittadini, L. Vanbever, and O. Bonaventure. iBGP deceptions: More sessions, fewer routes. In *IEEE INFOCOM*, pages 2122–2130, April 2012.

[37] L. Wang, M. Saranu, J. Gottlieb, and D. Pei. Understanding BGP session failures in a large ISP. In *IEEE INFOCOM*, pages 350–355, May 2007.

[38] X. Zhao, Y. Liu, L. Wang, and B. Zhang. On the aggregatability of router forwarding tables. In *IEEE INFOCOM*, pages 848–856, March 2010.