Scaling the Internet Routing System Through Distributed Route Aggregation

João Luís Sobrinho, *Senior Member, IEEE, Member, ACM*, Laurent Vanbever, *Member, IEEE, ACM*, Franck Le, André Sousa, and Jennifer Rexford, *Senior Member, IEEE, Fellow, ACM*

***Abstract*— The Internet routing system faces serious scalability challenges due to the growing number of IP prefixes that needs to be propagated throughout the network. 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 routing information. We present DRAGON, a distributed route-aggregation technique whereby 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 Autonomous Systems (ASs) to upgrade their router software. We illustrate the design of DRAGON through a number of examples, prove its properties while developing a theoretical model of route aggregation, and evaluate its performance. Our experiments with realistic AS-level topologies, assignments of IP prefixes, and routing policies show that DRAGON reduces the number of prefixes in each AS by at least 70% with minimal stretch in the lengths of AS-paths traversed by data packets.**

***Index Terms*— Inter-domain routing, routing scalability, BGP, routing algebra.**

I. Introduction

A ROUTING system scales if the amount of state information needed to support data-packet forwarding grows sub-linearly with the number of destinations. There is a long history of work on scalable routing systems that assumes full control over the format of data-packets, network topology, and address assignment, along with centralized routing decisions [1]–[3]. However, none of these premises apply to the Internet routing system. The headers of data-packets carry IP addresses identifying the final end-points of communication. Each component network, or Autonomous System (AS), of the Internet makes its own decisions about where to connect, how to acquire IP address space, and which routing policies to use, with the Border Gateway Protocol (BGP) maintaining the routing system in the presence of frequent changes. Against this scenario, the vast majority of the IP prefixes assigned to the various ASs of the Internet are routed globally. The Internet routing system does not scale.

At the same time, the number of globally routed IP prefixes continues to grow at a fast pace [4], with serious consequences for the performance and cost of the Internet routing system. This number determines the size of *forwarding-tables*, or Forwarding Information Bases (FIBs), stored in expensive high-speed memory on the routers, as well as the time it takes to look-up an address in those tables [5]. Recently, on August 12, 2014, the Internet suffered outages because the forwarding-tables of some older routers could not support the more than 512K IPv4 prefixes that they were receiving [6], [7]. The number of globally routed IP prefixes also determines the size of *routing-tables*, or Routing Information Bases (RIBs), the time it takes to bring up a single BGP session, the churn, and the convergence time of BGP [5], [8], [9]. The evolution towards IPv6 may exacerbate the scalability problems of the Internet routing system. IPv6 enlarges the size of IP addresses from 32 bits to 128 bits [10]. Without an allocation and assignment plan for IPv6 that takes routing scalability into account, many more IPv6 prefixes than IPv4 prefixes are potentially injected into the Internet routing system.

While the current Internet routing system does not scale, there is an underlying structure to the system which ought to allow for better aggregation of routing information. There are two sides to this structure. First, the AS-hierarchy, as determined by the provider-customer relationships, aligns with the prefix-hierarchy, in the sense that customer ASs acquiring address space from provider ASs originate prefixes that are more specific than those originated by the providers. Second, ASs that originate aggregatable address space are roughly clustered together, since address space is allocated by Regional Internet Registries (RIRs) to ASs in a given geographic region and ASs tend to connect to providers in the same geographic region.

Existing implementations and configurations of BGP do not exploit these opportunities for route aggregation. According to best current practices, an Internet Service Provider (ISP) configures its routers to filter prefixes from single-homed

Manuscript received May 24, 2015; revised November 10, 2015; accepted January 21, 2016; approved by IEEE/ACM Transactions on Networking Editor N. A. L. Reddy. This work was supported in part by the Fundação para a Ciência e Tecnologia within the Portuguese Ministry of Science and Higher Education under Grant UID/EEA/50008/2013, in part by the National Science Foundation under Grant CNS-1409056, and in part by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Grant W911NF-06-3-0001.

J. L. Sobrinho is with the Instituto de Telecomunicações and the Instituto Superior Técnico, Lisbon 1049-001, Portugal (e-mail: joao.sobrinho@lx.it.pt).

L. Vanbever is with ETH Zürich, Zürich 8092, Switzerland (e-mail: lvanbever@ethz.ch).

F. Le is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: fle@us.ibm.com).

A. Sousa is with Prodrive Technologies B. V., Eindhoven 5501, The Netherlands (e-mail: andre.magalhaes.sousa@gmail.com).

J. Rexford is with Princeton University, Princeton, NJ 08540-5233 USA (e-mail: jrex@cs.princeton.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2016.2527842

customers with address space assigned out of the ISP's own address space, but not from customers that are multi-homed or have been allocated prefixes directly from a RIR. The reason is simple: network operators cannot reason about the way 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 present a route aggregation solution to scale the Internet routing system called DRAGON—Distributed Route Aggregation on the GlObal Network. At heart, DRAGON is a distributed algorithm that exploits the inherent structure of the Internet routing system to significantly reduce the amount of routing and forwarding state that needs to be stored and exchanged. Thus, it contrasts with forwarding-table aggregation techniques [11]–[13], which are based on sequential algorithms executed independently at each AS affecting only the forwarding state. DRAGON augments the usual BGP routing decisions with a filtering strategy and an aggregation strategy, while relying on standard BGP routing messages to exchange routing information. The filtering strategy provides criteria for ASs to discard prefixes that are covered by less specific prefixes, effectively confining their propagation to a vicinity of the ASs that originate them. The aggregation strategy provides criteria for the introduction of a few aggregation prefixes, each covering a number of existing prefixes, thereby allowing the latter to be subject to the filtering strategy.

DRAGON exactly respects network-wide routing policies whenever these policies satisfy the isotonicity property [14], [15]. In loose terms, isotonicity means that if an AS prefers one route over another, then a neighbor AS does not have the opposite preference after its local processing of the two routes. Isotonicity is a common property of routing policies even if network operators are not explicitly aware of this fact. For instance, the Gao-Rexford (GR) routing policies [16], which provide a baseline of understanding for how network operators set their policies, are isotone. Our experiments with realistic AS-level topologies, IP prefix assignments, and the GR routing policies show, for example, that DRAGON dispenses with at least 70% of the IP prefixes in the forwarding-table of each AS.

In a previous conference paper [17], we presented an overview of the filtering and aggregation strategies of DRAGON, and we discussed some preliminary performance results. Here, we expound the theoretical underpinnings of DRAGON building up from the framework provided by the algebraic theory of routing [15]. In addition, we present a comprehensive set of stable-state results, including an assessment of the savings in the sizes of forwarding-tables and routing-tables, and an assessment of the (small) stretch induced in the lengths of AS-paths followed by data-packets. Due to space limitations, a comprehensive evaluation of the dynamics of DRAGON is left for a future publication.

The fundamentals of DRAGON are valid beyond inter-AS routing. We honor the generality of DRAGON by using generic terms such as network, prefix, node, routing vector protocol, instead of Internet, IP prefix, AS or router, BGP, even if our examples and experiments pertain to inter-AS routing. The remainder of the paper is organized as follows. Section II presents the routing and forwarding model. Section III introduces DRAGON's filtering strategy and aggregation strategy through examples. Section IV rigorously proves the properties of DRAGON. Section V provides an assessment of the stable-state performance of DRAGON. Section VI discusses related work and Section VII draws final conclusions.

II. Routing and Forwarding

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.

A route is an association between a prefix and an attribute, with the set of all possible attributes totally ordered by preference.[1] A route pertaining to prefix p is called a p-route. The node to which p has been assigned is the origin of p. A standard routing vector protocol instantiates a different computation process for each prefix p, which starts when the origin of p forms a p-route that it sends to all its neighbors. Whenever a node receives a p-route from a neighbor it extends the attribute of that route into the attribute of a candidate route to reach p via that neighbor. Among the candidate p-routes learned from its neighbors, the node elects the one with the most preferred attribute and sends it 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. 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. A rigorous, algebraic formulation of a routing vector protocol for arbitrary routing policies is given in [15] and reviewed in Section IV.

The prototypical inter-AS routing policies are the Gao-Rexford (GR) routing policies [16], which postulate that neighbor nodes establish either a provider-customer or a peer-peer relationship. The policies are supported on just three attributes, which we will call the GR 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 a customer'," and similarly for the terms "peer route" and "provider route," and we 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

[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 [16]. We make this extra assumption because it seems to be valid in practice and it simplifies the exposition.

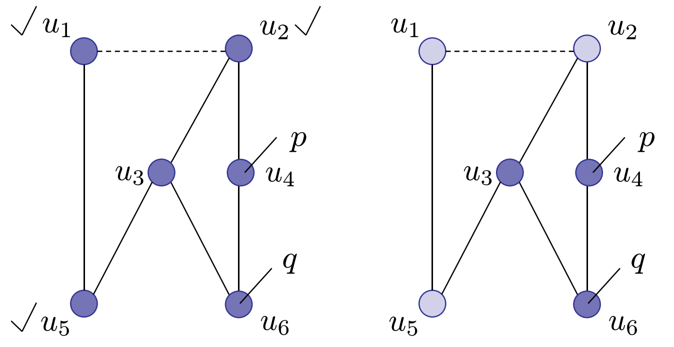

Fig. 1. Providers are drawn above customers and joined to them with solid lines. Peers are joined with dashed lines. Node u_6 is multi-homed to u_3 and u_4. Node u_4 originates p and its customer u_6 originates q. Prefix q is more specific than p. Left. Standard stable state. Checks mark nodes that satisfy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

are the only exportations allowed. The origin of a prefix can be assumed to form a route with attribute "learned from a customer," since such a route is subjected to the same treatment as if it were learned from a customer, namely, the route is exported to all of the origin's neighbors.

A routing vector protocol is correct in a network if it terminates in a stable state that guides data-packets to their destinations. In general, the correctness of a routing vector protocol depends upon the configuration of routing policies around the cycles of a network. In the particular case of the GR routing policies, a routing vector protocol is correct if there is no cycle in the network where each node is a customer of the next around the cycle. Not all paths in a network can transport data-packets. Those that can are called *usable*. In the case of the GR routing policies, the usable paths are of the form PRC, where P is either trivial or a path where each link joins a customer to a provider, R is either trivial or a link joining one peer to another, and C is either trivial or a path where each link joins a provider to a customer. A network is *policy-connected* if there is a usable path from every node to every other.

In the network of Figure 1, nodes operate a routing vector protocol with 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. The address space assigned to u_4 is represented by prefix p. Node u_4 is the origin of p. Once the routing 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. Routes for prefixes at different levels of specificity are propagated throughout the network. The *longest prefix match rule* [18] 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, u_6 acquired address space from its provider u_4, represented by prefix q which is more specific than p. Node u_6 is the origin of q. Node u_3 elects a customer q-route, learned from u_6, and a provider p-route, learned from u_2. Data-packets arriving 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.

III. MECHANISMS

DRAGON relies on standard routes to convey routing information between neighbor nodes and augments local routing decisions with a filtering strategy and an aggregation strategy. Section III-A presents basic filtering code for DRAGON and Section III-B presents a rule for originating prefixes that ensures that the filtering code does not create black holes. Section III-C introduces a property of routing policies known as isotonicity and illustrates its implications on the optimality and the partial deployment of DRAGON. Section III-D discusses the filtering strategy in the context of inter-AS routing policies that take path-lengths into account. Section III-E concerns multiple levels of prefixes. Section III-F presents the aggregation strategy. Last, Section III-G shows the reaction of DRAGON to network events such as link failures.

A. Filtering Code

The goal of DRAGON is for many nodes to dispense with routes pertaining to the more specific prefixes with little or no change in the properties of paths traversed by data-packets. Towards this goal, 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. Candidate routes for the prefix are still kept in the routing-table of the node and we still say that the node elects the candidate route with the most preferred attribute.

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

Code CR:

> In the presence of p, filter q if and only 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.

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. If it did, then 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. 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. Last, if the attribute of the elected q-route is preferred to that of the elected p-route, then filtering would worsen the attribute of the route used to forward data-packets with destination in q. Thus, in this case, the node does not filter q.

Throughout the paper, we will study the global, network-wide effects of local code $\mathbf{CR}$. For now, we exemplify that effect with Figure 1. Recall that nodes follow the GR routing policies. Both p and q are propagated by the routing 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 (formed by itself) and a provider p-route. Thus, u_6 cannot filter q.
- 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.
- 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.
- 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.
- 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.

Suppose that u_2 executes $\mathbf{CR}$, thereby filtering q. Despite the absence of an entry for q in its forwarding-table, 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 routing 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_1 and u_3. In summary, if u_2 then u_5 execute $\mathbf{CR}$, then we arrive at the routing state depicted on 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 realistic Internet topologies, most nodes forgo q. Of these, a few will filter q, while the majority will be oblivious of q. In other words, routing state pertaining to q only needs to be disseminated in some small vicinity of the origin of q.
- Data-packets are delivered to their destinations, there being no route oscillations, forwarding loops, or black holes. When this happens, we say that DRAGON is *correct*.
- 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 desirable global routing state is called *route-consistent*. A route-consistent state is *optimal* if the set of nodes forgoing q is maximal. The routing state depicted on the right-hand side of Figure 1 is optimal route-consistent.
- Node u_2 lost u_3 as a neighbor to which it could 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 refine the filtering code with the aim of preserving not only the attributes of the routes used to forward data-packets, but also the forwarding neighbors [19]. With the refined filtering code, u_2 would not be allowed to filter q.

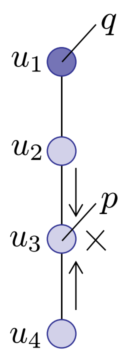

Fig. 2. Prefix q is more specific than prefix p. Node u_3 is the origin of p, exporting p to all its neighbors. Node u_1 is the origin of q exporting q to all its neighbors. When u_2 executes code $\mathbf{CR}$, it creates a black hole at u_3, indicated by the cross. Arrows indicate the expedition of data-packets with destination in q.

B. Origination Rule

Through code $\mathbf{CR}$, DRAGON subordinates the propagation of q-routes to the elected p-routes. Therefore, even if the routing 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 a black hole for data-packets with destination in q. In Section IV, we prove that the following origination rule guarantees correctness of DRAGON.

Rule RO:

In the presence of q, the origin of p announces p in a p-route whose attribute is equal to or less preferred than the attribute of the elected q-route.

The necessity of rule $\mathbf{RO}$ 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 announces p with a customer route, thus violating rule $\mathbf{RO}$: 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 $\mathbf{RO}$, u_3 can originate p only with a provider route,

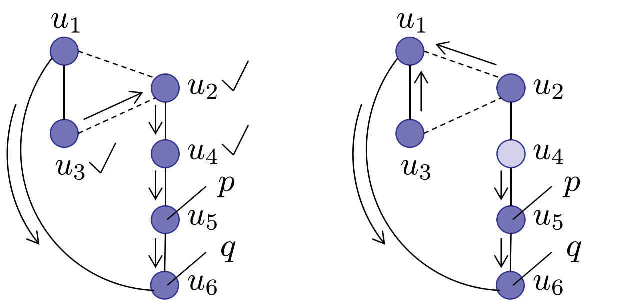

Fig. 3. Prefix q is more specific than prefix p. Node u_5 is the origin of p and u_6 is the origin of q. Left. Standard stable state. Checks mark nodes that satisfy the premise for filtering q. Right. Node u_4 executes code $\mathbf{CR}$, worsening the elected q-routes at u_2 and u_3. This worsening reinforces u_2's and u_3's incentive to execute code $\mathbf{CR}$. Arrows indicate the expedition of data-packets with destination in q.

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 through u_3 to u_1.

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

C. Isotonicity, Optimality, and Partial Deployment

The routing policies of link uv are isotone [14], [15] whenever the relative preference among attributes of elected routes at v is respected among attributes of candidate routes derived from them at u (see Section IV-C).

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 routing policies used in practice or proposed are isotone [20], [21].

We illustrate the implications of isotonicity on DRAGON with the network of Figure 3. Node u_6 is the origin of q and u_5 is the origin of p. Node u_1 is a provider of both u_3 and u_6, and a peer of u_2. Node u_2 is a provider of u_4 and a peer of both u_1 and u_3. The left-hand side of the figure shows the initial stable state, before DRAGON is deployed. Nodes u_2, u_3, and u_4 will filter q if they execute code $\mathbf{CR}$: u_2 elects a customer q-route and a customer p-route, both routes learned from u_4; u_3 elects a peer q-route and a peer p-route, both routes learned from u_2; and u_4 elects a customer q-route and a customer p-route, both routes learned from u_5.

Suppose that u_4 is the first node to execute code $\mathbf{CR}$, thereby filtering q. As a consequence, the elected q-route at u_2 worsens from customer to peer, the latter learned from u_1, and the elected q-route at u_3 worsens from peer to provider, as shown on the right-hand side of the figure. The resulting routing state is not route-consistent. Node u_2, say, will now forward data-packets with destination in q to its peer u_1 rather than to its customer u_4. However, u_2 and u_3 gained an extra incentive to execute code $\mathbf{CR}$. If, say, u_2 executes code $\mathbf{CR}$, then it saves on forwarding state and it reverts to forwarding data-packets with destination in q to its customer u_4 on account of the elected p-route. Once u_2 and u_3 execute code $\mathbf{CR}$, thereby filtering q, DRAGON reaches a final state that is optimal route-consistent.

With isotonicity, it is even possible to sequence the adoption of DRAGON among the nodes such that all intermediate routing states are route-consistent. In the particular case of the GR routing policies, all sequences of adoption obeying the following condition ensure continual route-consistency.

Route-consistent partial deployment

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.

In Figure 3, agreement with the previous condition would have u_3 be the first node to execute code $\mathbf{CR}$. The consequent filtering of q would still allow u_3 to forward data-packets with destination in q to a peer, u_2, guided by the elected p-route, and it would not affect the elected q-routes at other nodes. After u_3, node u_2 would execute the filtering code and, last, u_4 would do so.

The role played by isotonicity on DRAGON can be summarized as follows.

- DRAGON attains an optimal route-consistent state after all nodes execute code $\mathbf{CR}$ in whatever order.
- The incentive to filter is embodied in code $\mathbf{CR}$. However, that incentive is exacerbated when some nodes execute code $\mathbf{CR}$, because filtering can only worsen the attribute of elected routes at other nodes. Nodes at which the attribute of elected routes worsen can recover the attribute of the route used to forward data-packets by filtering as well.
- There is a sequence for adoption of DRAGON that is route-consistent throughout all intermediate stages of deployment.
- The correctness of DRAGON does *not* depend on isotonicity. It follows from rule $\mathbf{RO}$ and from the same property of routing policies around the cycles of a network that guarantees correctness of a routing vector protocol (see Section IV).
- Despite ensuring an optimal route-consistent state, isotonicity by itself does *not* say if that state is efficient, in the sense of having many nodes forgoing q.

D. Path-Lengths and Relaxation of the Filtering Code

BGP registers the AS-path traversed by routes as they propagate away from their origins, with inter-AS routing policies

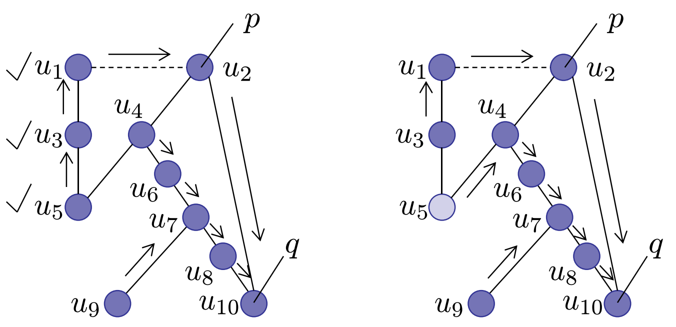

Fig. 4. GR-with-path-lengths routing policies. Prefix q is more specific than prefix p. Node u_{10} originates a q-route with 0 hops, while u_2 originates a p-route with 1 hop. Left. Standard stable state. Checks mark nodes that satisfy the premise for filtering q. Right. Stable state after u_5 filters q. Arrows indicate the expedition of data-packets with destination in q.

using AS-path-length to break ties among routes with the same GR attribute. The GR-with-path-length routing policies have attributes composed of GR attributes and hop-counts. A GR-with-path-length attribute is preferred to another if the GR attribute of the former is preferred to that of the latter, or the two GR attributes are the same, but the hop-count of the former is smaller than that of the latter. A GR-with-path-length attribute extends to another by extending the GR attribute of the former and incrementing its hop-count. The GR-with-path-length routing policies are not isotone and, as a consequence, do not yield route-consistent global states after filtering, in general. In addition, use of code **CR** with those routing policies does not lead to big savings in routing state, in general. We will discuss a relaxed use of code **CR** that leads to very efficient routing states while accepting a small stretch in the lengths of paths traversed by data-packets.

In the network of Figure 4, nodes operate a routing vector protocol with GR-with-path-lengths routing policies. We first observe that the routing policies of link u_5u_4 are not isotone. At u_4, a customer route with 4 hops is preferred to a provider route with 2 hops. Both routes are exported by u_4 to u_5. At u_5, the customer route with 4 hops becomes a provider route with 5 hops whereas the provider route with 2 hops becomes a provider route with 3 hops. Since a provider route with 5 hops is *less* preferred than a provider route with 3 hops, isotonicity is violated.

In the figure, node u_{10} is the origin of q, announcing q in a q-route with 0 hops. Hence, u_2 elects a customer q-route with 1 hop. In order to satisfy rule **RO**, node u_2, which is the origin of p, announces p in a p-route with 1 hop (in BGP practice, this would be accomplished with AS pre-pending).

The initial stable state is shown on the left-hand side of the figure. Node u_5 elects the provider q-route with 4 hops learned from u_3, to the detriment of the provider q-route with 5 hops learned from u_4. Data-packets with destination in q arriving at u_5 traverse path $u_5u_3u_1u_2u_{10}$ to reach q. At the same time, u_5 elects the provider p-route with 3 hops learned from u_4, to the detriment of the provider p-route with 4 hops learned from u_3. Suppose that u_5 executes code **CR** leading to the stable state depicted on the right-hand side of the figure. Since the elected q-route is less preferred than the elected p-route at u_5, this node filters q. From then on, it forwards data-packets with destination in q according to the elected p-route, that is, it forwards them to u_4 which deflects them on path $u_4u_6u_7u_8u_{10}$. In short, before filtering, data-packets with destination in q arriving at u_5 traverse path $u_5u_3u_1u_2u_{10}$ of length 4 hops; after filtering, they traverse path $u_5u_4u_6u_7u_8u_{10}$ of length 5 hops. Nodes u_1 and u_3 also filter q upon execution of code **CR**. After all nodes execute code **CR**, the resulting routing state is still route-consistent with respect to the GR attributes alone, but node u_5 experiences a stretch in the length of the path traversed by data-packets with destination in q due to the lack of isotonicity.

More importantly than the absence of isotonicity, applying DRAGON to the GR-with-path-lengths routing policies may not lead to efficient routing states. In Figure 4, u_9 elects a provider q-route with 3 hops learned from u_7 and a provider p-route with 5 hops learned as well from u_7. The elected q-route is preferred to the elected p-route. Thus, u_9 does not filter q upon execution of code **CR**. However, u_9 would not even witness any distortion in the path traversed by data-packets if it filtered q.

As a matter of fact, we know that a compact routing state is not possible, in general, with shortest paths without some stretch [3], [22]. Therefore, we relax code **CR** by applying it only to the GR attributes, disregarding hop-counts. In Figure 4, since u_9 elects both a provider q-route and a provider p-route, it filters q upon execution of the relaxed code **CR**. As before, the resulting routing state obtained after all nodes execute the relaxed code **CR** is route-consistent in terms of the GR attributes, but more efficient than if nodes executed code **CR** on the full attributes of the GR-with-path-lengths routing policies. Rule **RO** can also be relaxed to apply only to the GR attributes. In Figure 4, u_2 could form and send to all its neighbors a p-route with a hop-count of 0 instead of 1.

We can even tradeoff efficiency with stretch by allowing a node with equal GR attributes for the q-route and the p-route to filter q only if the hop-count of the elected q-route is not shorter than that of the elected p-route by more than some pre-specified number of hops. However, simply neglecting hop counts in the filtering code already yields very small stretch (see Section V).

E. Multiple Levels of Prefixes

A very large number of prefixes at different levels of specificity is globally routed 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. Prefix q is a *child* of its parent prefix. DRAGON operates by having every node contrast each prefix q against its parent prefix in the set of prefixes learned from the routing vector protocol, in the same way that q is contrasted against p in code **CR**.

We can impose that every node executes code **CR** on the list of prefixes it learns from the routing vector protocol from the least specific to the most specific one. However, the executions of code **CR** at different nodes are uncorrelated 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 condition on the routing policies around the cycles of a network that guarantee correctness of a routing vector protocol. In addition, if routing policies are isotone, then DRAGON leads to an optimal route-consistent state after network-wide deployment.

F. Aggregation Strategy

Provider-Independent (PI) prefixes, which are those acquired directly from a RIR, do not have a parent prefix in the routing system which would allow them to be filtered at ASs far-away from their origin ASs. The aggregation strategy of DRAGON authorizes nodes to originate so called aggregation prefixes thereby potentiating the filtering of PI prefixes. An aggregation prefix must satisfy the following two conditions:

- it does not create new routable address space;
- it becomes the parent of as many PI prefixes as possible and at least of two of them.

As with any other prefix, an aggregation prefix is subjected to rule $\mathbf{RO}$. In a previous conference paper [17], we describe examples to illustrate how the routing system self-organizes if more than one node originates the same aggregation prefix.

G. Reaction to Network Events

DRAGON reacts automatically to network events, such as link failures and additions. We illustrate that reaction calling again on the network of Figure 1 and starting from the state depicted on its right-hand side. Most link failures do not trigger either code $\mathbf{CR}$ or rule $\mathbf{RO}$. For example, if any of the links u_1u_2, u_2u_1, u_1u_5, u_5u_1, u_2u_3, or u_3u_2 fails, then there is no change whatsoever in the routing state pertaining to prefix q. If u_5u_3 fails, then u_5 becomes oblivious to q and if u_3u_6 fails, then u_3 becomes oblivious to q.

A few link failures trigger code $\mathbf{CR}$ alone. For instance, if link u_2u_4 fails, then node u_2 no longer elects a p-route. Faithful to code $\mathbf{CR}$, u_2 re-installs the elected customer q-route learned from u_3, which it exports to all its neighbors. Eventually, nodes u_1, u_3, and u_5 stop electing a p-route and re-elect a q-route.

The rarest, but more challenging failure is the one of the link joining the origin of p to a unique forwarding neighbor for q. Such a failure summons both code $\mathbf{CR}$ and rule $\mathbf{RO}$. Suppose that link u_4u_6 fails. Node u_4 no longer elects a q-route. It cannot announce p as such an action would violate rule $\mathbf{RO}$. Hence, u_4 withdraws p. As a consequence, u_2 stops electing a p-route and re-installs the elected customer q-route learned from u_3, which it exports to all its neighbors. Eventually, every node stops electing a p-route and re-elects a q-route.

While u_4 withdraws p upon the failure of u_4u_6, it re-aggregates the address space of p to the exclusion of that represented by q into sub-prefixes that can be announced in customer routes. For instance, suppose that $p = 10$ and $q = 10000$. Node u_4 announces the three prefixes 10001, 1001, and 101 to all its neighbors, 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, all learned from u_4. It also elects a customer 10000-route, learned from u_3. If the aggregation strategy is active at u_2, then this node pieces together prefixes 10001, 1001, 101, and $q = 10000$ to originate aggregation prefix $p = 10$, allowing a subsequent filtering of q at u_1 and u_5.

The failure of a link that joins the origin of p to a unique forwarding neighbor for q may temporarily create a black hole for q at the origin of p, and it generates more routes than any other type of link failure. Such links ought to be built with added redundancy or protected with a timer.

IV. Theory

The design of DRAGON is grounded on the solid foundations provided by the algebraic theory of routing [15]. Section IV-A presents the basic elements of this theory. Sections IV-B and IV-C briefly review correctness and optimality of routing vector protocols, respectively. Sections IV-D and IV-E prove correctness and optimality of DRAGON, respectively, for isotone routing policies. Section IV-F discusses the correctness of DRAGON in the absence of isotonicity. The appendix to the paper contains notation concerning paths and cycles.

A. Attributes and Labels

Route attributes form a finite set Σ equipped with a linear order $\preceq$. If $\alpha \prec \beta$ (resp. $\alpha \succ \beta$), then we say that α is preferred to β (resp. α is less preferred than β). From the linear order, we define an election operation $\sqcap$ which yields the most preferred of two attributes: $\alpha \sqcap \beta = \alpha$, if $\alpha \preceq \beta$; and $\alpha \sqcap \beta = \beta$, otherwise. Binary operation $\sqcap$ is selective, associative, and commutative. Thus, there is always a most preferred attribute from among a set T of attributes, which is denoted by $\sqcap T$. There is a special attribute $\bullet$ to indicate that a prefix is not reachable. Attribute $\bullet$ is the least preferred of all attributes.

A link uv represents the possibility of forwarding data-packets from node u to node v. Routes travel in the opposite direction. The routing policies of link uv tell how the attribute of a route elected at v is extended into the attribute of a candidate route stored at u to reach the prefix announced in the route via v. They subsume the export policy of v with regard to u and the import policy of u with regard to v. We assume that routing policies do not depend on the prefix. Therefore, the routing policies of link uv are modeled by a map on the set of attributes which we denote by $L[uv]$ and call the label of uv. Label $L[uv]$ extends attribute α into attribute $L[uv](\alpha)$. If v cannot reach a prefix, then u cannot reach that same prefix via v. Therefore, $L[uv](\bullet) = \bullet$.

Given walk $u_0u_1 \cdots u_n$, we say that a route propagates from u_n to u_0 along the walk if u_i elects the route learned from u_{i+1} and sends it to u_{i-1}, for every $0 < i < n$. The label of walk $u_0u_1 \cdots u_n$, denoted by $L[u_0u_1 \cdots u_n]$, is the composition of the labels of its links, $L[u_0u_1 \cdots u_n] = L[u_0u_1]L[u_1u_2] \cdots L[u_{n-1}u_n]$, and describes how attributes of routes at u_n propagated from u_n to u_0 along the walk transform into attributes of routes at u_0. The label of a trivial walk composed of a single node is, by definition, the identity map.

B. Correctness of Routing Vector Protocols

A routing vector protocol is correct in a given network if it terminates in a stable state devoid of forwarding loops, whatever the initial global routing state and whatever the delays in the delivery of routes between neighbor nodes. Correctness of a routing vector protocol depends on how routing policies are configured around the cycles of the network. A cycle $C = u_0u_1 \cdots u_{n-1}u_0$ is *strictly absorbent* [15] if

$$\forall_{\alpha_0 \prec \bullet, \alpha_1 \prec \bullet, \ldots, \alpha_{n-1} \prec \bullet} \exists_{0 \le i < n} \, \alpha_i \prec L[u_iu_{i+1}](\alpha_{i+1}), \quad (1)$$

where indexes are modulus n. In words, a cycle is strictly absorbent if, for every combination of routes learned by its nodes externally to the cycle and sent to their neighbors around the cycle, at least at one of the nodes the attribute of the route learned externally to the cycle is preferred to the attribute of the route learned from the neighbor around the cycle. The following theorem is proven in [15].

Theorem 1: If all cycles in a network are strictly absorbent, then the routing vector protocol terminates in a stable state that is free of forwarding loops.

Under the hypothesis that all cycles in the network are strictly absorbent, we provide a characterization of the stable states. Let t^p be the origin of p and $R^*[t^p;p]$ be the attribute of the p-route formed by t^p; $R^*[u;p] = \bullet$ if $u \neq t^p$. Let $R[u;p]$ be the attribute of the elected p-route at u in the stable state. Elected p-routes satisfy the following system of fixed-point equations:

$$R[u;p] = \sqcap\{L[uv](R[v;p]) \mid v \text{ neigh. of } u\} \sqcap R^*[u;p], \quad (2)$$

with $R[t^p;p] = R^*[t^p;p]$, meaning that t^p elects the p-route it formed. Directly from the fixed-point equations, we deduce, for every link uv, that

$$R[u;p] \preceq L[uv](R[v;p]). \quad (3)$$

If u elects a p-route, $R[u;p] \prec \bullet$, then v is *forwarding neighbor* of u for p if

$$R[u;p] = L[uv](R[v;p]). \quad (4)$$

Path $P = u_0u_1 \cdots u_n$ with $u_n = t^p$ is a *forwarding path for* p if u_{i+1} is a forwarding neighbor of u_i for p, for $0 \le i < n$. An elementary proof by induction shows that $R[u_i;p] = L[u_iP](R^*[t^p;p])$, for $0 \le i \le n$.[3] Generally, we say that P is a *walk for* p if it terminates at t^p. The attribute of walk P is $L[P](R^*[t^p;p])$.

C. Optimality of Routing Vector Protocols Under Isotonicity

A label L of a link or of a walk is isotone if it is an increasing map on the ordered set of attributes:

$$\forall_{\alpha,\beta} \, \alpha \preceq \beta \Rightarrow L(\alpha) \preceq L(\beta). \quad (5)$$

[3]See the appendix for the notation concerning paths and cycles.

A link, and a walk, is isotone if its label is isotone. The composition of isotone labels is itself isotone. Thus, a walk all links of which are isotone is isotone. We first state the implication of isotonicity on strictly absorbent cycles [15].

Theorem 2: Let C be a cycle all links of which are isotone, and let u be an arbitrary node of the cycle. Then, C is strictly absorbent if and only if every route propagated by u all the way around C arrives back at u with an attribute that is less preferred than the one it started out with:

$$\forall_{\alpha \prec \bullet} \, \alpha \prec L[uCu](\alpha).$$

Isotonicity is associated with optimality of the routing vector protocol computation in the sense made precise by following theorem [15].

Theorem 3: Suppose that all links in the network are isotone and all cycles are strictly absorbent. Then, the attribute of the elected p-route at an arbitrary node u equals or is preferred to the attribute of any walk P for p starting at u:

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

D. Correctness of DRAGON Under Isotonicity

In this section, we prove correctness of DRAGON under the assumption that all cycles in the network are strictly absorbent and all links are isotone. In particular, correctness stays proven for the GR routing policies if there is no cycle in the network where each node is a customer of the next around the cycle. It also stays proven for the GR-with-path-lengths routing policies (see Section III-D) under the same condition on the cycles, since path-lengths are only used as tie-breaks for any given GR attribute and, thus, do not influence correctness.

We consider the following scenario. There is a prefix p and a prefix q that is more specific than p. Node t^p is the origin of p, forming a p-route with attribute $R^*[t^p;p]$; t^q is the origin of q, forming a q-route with attribute $R^*[t^q;q]$. In the *initial (stable) state*, rule **RO** is satisfied and none of the nodes has yet executed code **CR**. Starting from the initial state, nodes execute code **CR** one at a time until all of them have done so. A sequence of nodes executing code **CR** is called a *filtering sequence*. Filtering sequences correspond to permutations of the nodes of the network.

A node that executes code **CR** but does not filter q leaves the routing state unchanged. On the other hand, a node that executes code **CR** and filters q leads the routing system to a new stable state. As nodes of a sequence execute code **CR**, the routing system progresses through a sequence of stable states. Every stable state is fully characterized by the set of nodes that filter q. We denote by $R_S[u;q]$ the attribute of the elected q-route at node u when a set S of nodes filters q. The stable state obtained at the end of a filtering sequence is called a *final state*. Stable states, other than the initial and final ones, are called *intermediate states*. They correspond to stages in the deployment of DRAGON.

We start with three lemmas that will be used in this section and the next. The first lemma allows us to conclude that if the premise for filtering q embodied in code **CR** is *not* satisfied at a given node u, then it is also *not* satisfied at any node along a forwarding path for q starting at u.

Lemma 4: Suppose that a set of nodes filters q. If the attribute of the elected q-route is preferred to the attribute of the elected p-route at node u, then, as well, the attribute of the elected q-route is preferred to the attribute of the elected p-route at every node along a forwarding path for q starting at u.

Proof: Let S be the set of nodes that filters q. Let P be a forwarding path for q under S starting at u. In order to obtain a contradiction, assume that there is a node w along P such that the attribute of its elected p-route equals or is preferred to the attribute of its elected q-route:

$$R_S[w;p] \preceq R_S[w;q]. \tag{6}$$

Further assume that w is the first node along P satisfying the previous condition. Clearly, $w \neq u$, since, by hypothesis, $R_S[u;p] \succ R_S[u;q]$. Let v be the predecessor of w along P, $R_S[v;q] = L[vw](R_S[w;q])$. We write (iso. stands for isotonicity)

$$\begin{aligned} R_S[v;p] &\preceq L[vw](R_S[w;p]) \quad \text{(from (3))},\\ &\preceq L[vw](R_S[w;q]) \quad \text{(from (6) and iso. of } vw),\\ &= R_S[v;q]. \end{aligned}$$

The previous inequality contradicts the choice of w as the first node along P for which the attribute of the elected p-route equals or is preferred to the attribute of its elected q-route. ■

Filtering q at a node is equivalent to removing the node from the network insofar as the election of q-routes at all other nodes is concerned. The next two lemmas scrutinize the effect of filtering q at a node on the attributes of elected q-routes of all nodes.

Lemma 5: Suppose that a set of nodes filters q. If a new node z filters q, then the attribute of the elected q-route at an arbitrary node u remains the same or becomes less preferred.

Proof: Let S be the set of nodes that filters q before z does. Let P be a forwarding path for q under S starting at u, $R_S[u;q] = L[P](R^*[t^q;q])$, and Q be a forwarding path for q under $S+z$ starting, as well, at u, $R_{S+z}[u;q] = L[Q](R^*[t^q;q])$. Path Q is also a path for q under S.[4] We write

$$\begin{aligned} R_S[u;q] &= L[P](R^*[t^q;q])\\ &\preceq L[Q](R^*[t^q;q]) \quad \text{(Theorem 3)},\\ &= R_{S+z}[u;q]. \end{aligned}$$

■

Lemma 6: Suppose that a set of nodes filters q. If a new node z filters q, then the attribute of the elected q-route remains the same at every node along a forwarding path for q that does not contain z.

Proof: Let S be the set of nodes that filters q before z does. Let P be a forwarding path for q under S not containing node z. In order to obtain a contradiction, assume that there is a node u along P that does not preserve the attribute of its elected q-route after filtering by z:

$$R_{S+z}[u;q] \neq R_S[u;q]. \tag{7}$$

Further assume that u is the last node along P satisfying the previous condition. Clearly, $u \neq t^q$, since $R_{S+z}[t^q;q] = R^*[t^q;q] = R_S[t^q;q]$. Let v be the successor of u along P, $R_S[u;q] = L[uv](R_S[v;q])$. Because u is the last node along P satisfying Inequality (7), we have

$$R_{S+z}[v;q] = R_S[v;q]. \tag{8}$$

We write

$$\begin{aligned} R_{S+z}[u;q] &\preceq L[uv](R_{S+z}[v;q]) \quad \text{(from (3))},\\ &= L[uv](R_S[v;q]) \quad \text{(from (8))},\\ &= R_S[u;q]. \end{aligned}$$

On the other hand, from Lemma 5, we know that $R_{S+z}[u;q] \succeq R_S[u;q]$. Hence, $R_{S+z}[u;q] = R_S[u;q]$, contradicting Inequality (7). ■

The following theorem proves that any forwarding path for q starting at t^p is an invariant, the theorem being the cornerstone of the subsequent proof of correctness of DRAGON.

Theorem 7: A forwarding path for q starting at t^p remains a forwarding path for q after a new node executes code **CR**.

Proof: Consider an arbitrary filtering subsequence. We will show that if P is a forwarding path for q starting at t^p at the end of the subsequence, then P remains a forwarding path for q after the next node in the sequence executes code **CR**.

Of the nodes of the subsequence, a set S filters q. If a new node *not* on path P executes code **CR** and filters q, then Lemma 6 asserts that P remains a forwarding path for q. We now show that any node on path P does *not* satisfy the premise for filtering q embodied in code **CR**, leaving the routing state intact after execution of the code. As a matter of fact, it suffices to show that the attribute of the elected q-route is preferred to the attribute of the elected p-route at the first node after t^p along P. Then, from Lemma 4, we conclude that the attribute of the elected q-route is preferred to the attribute of the elected p-route at all nodes of P.

Let u be the first node after t^p along P and let Q be a forwarding path *for p* starting at u, $R[u;p] = L[Q](R^*[t^p;p])$. We assume that the premise for filtering q is satisfied,

$$R_S[u;q] \succeq R[u;p], \tag{9}$$

to arrive at the contradiction that cycle $t^p uQt^p$ is not strictly absorbent. We write (iso. stands for isotonicity)

$$\begin{aligned} R^*[t^p;p] &\succeq R_S[t^p;q] \quad \text{(rule } \mathbf{RO}),\\ &= L[t^p u](R_S[u;q])\\ &\succeq L[t^p u](R[u;p]) \quad \text{(from (9); iso. of } t^p u),\\ &= L[t^p u](L[Q](R^*[t^p;p]))\\ &= L[t^p uQt^p](R^*[t^p;p]). \end{aligned}$$

Therefore, there is an attribute α, $\alpha = R^*[t^p;p] \prec \bullet$, such that $\alpha \succeq L[t^p uQt^p](\alpha)$. From Theorem 2, we deduce that $t^p uQt^p$ is not strictly absorbent. ■

[4]The inverse need not be true. If z is a node of P other than u, then P is not a forwarding path for q under $S+z$.

Theorem 8: DRAGON is correct, always delivering data-packets to their intended destinations.

Proof: Both the stable states for p and q are devoid of forwarding loops. A data-packet with destination in q arriving at a node that does not elect a q-route is forwarded according to the elected p-route. The data-packet is guided along elected p-routes until it arrives at a node that elects a q-route. From then on, the data-packet is forwarded along elected q-routes to the origin of q. Therefore, DRAGON never introduces forwarding loops. Whenever t^p satisfies rule $\mathbf{RO}$, it is not a black hole for q, because if it elects a p-route it also elects a q-route. By hypothesis, rule $\mathbf{RO}$ is satisfied in the initial state. Theorem 7 implies that rule $\mathbf{RO}$ remains valid through all intermediate states and final state of every filtering sequence. ∎

E. Optimality of DRAGON Under Isotonicity

Again, we assume that all cycles in the network are strictly absorbent and all links are isotone. A state is route-consistent if the attribute of the route used to forward data-packets is the same as that in the initial stable state, without any filtering. A route-consistent state is optimal if any additional filtering would break route-consistency. We shall prove two results. First, that the final state of every filtering sequence is optimal route-consistent, meaning that the attribute of the route used to forward data-packets is the same as that in the initial state. Second, that there is a filtering sequence all intermediate states of which are route-consistent. We start with a lemma that allows us to classify nodes into those that elect a q-route with an attribute that equals that of the elected p-route in the initial state, represented by set F, and those that elect a q-route with an attribute that is preferred to that of the elected p-route in the initial state, represented by set $\overline{F}$.

Lemma 9: In the initial state, the attribute of the elected q-route equals or is preferred to the attribute of the elected p-route at every node.

Proof: We want to show that $R[u;q] \preceq R[u;p]$ for every node u. Let P be a forwarding path for q starting at u, $R[u;q] = L[P](R^*[t^q;q])$; Q be a forwarding path for p starting at u, $R[u;p] = L[Q](R^*[t^p;p])$; and T be a forwarding path for q starting at t^p, $R[t^p;q] = L[T](R^*[t^q;q])$. Walk Qt^pT is a walk from u to t^q. We write (iso. stands for isotonicity)

$$\begin{aligned}
R[u;q] &= L[P](R^*[t^q;q]) \\
&\preceq L[Qt^pT](R^*[t^q;q]) \quad \text{(Theorem 3)}, \\
&= L[Q](L[T](R^*[t^q;q])) \\
&= L[Q](R[t^p;q]) \\
&\preceq L[Q](R^*[t^p;p]) \qquad \text{(rule } \mathbf{RO}\text{; iso. of } Q\text{)}, \\
&= R[u;p].
\end{aligned}$$

∎

The following theorem proves that the attribute of elected q-routes of nodes in $\overline{F}$ is an invariant, and so is the worsening of the attribute of elected q-routes of nodes in F. The theorem is the basis for the proof of optimal route-consistency of DRAGON.

Theorem 10: A node at which the attribute of the elected q-route is preferred to the attribute of the elected p-route sees the attribute of the elected q-route remain the same when a new node executes code $\mathbf{CR}$. A node at which the attribute of the elected q-route equals or is less preferred than the attribute of the elected p-route sees the attribute of the elected q-route either remain the same *or* worsen when a new node executes code $\mathbf{CR}$.

Proof: Consider an arbitrary filtering subsequence and let z be the next node to execute code $\mathbf{CR}$. Let u be any node such that the attribute of the elected q-route is preferred to the attribute of the elected p-route before z executes code $\mathbf{CR}$. From Lemma 4, we learn that any node along a forwarding path P for q starting at u also has an attribute of the elected q-route that is preferred to the attribute of the elected p-route. Therefore, if z is a node along P, then it does not satisfy the premise for filtering q, it does not filter q, and the routing state remains the same. On the other hand, if z is not a node along P, and z ends up filtering q, then, from Lemma 6, we conclude that the attribute of the elected q-route at any node of P, in particular that of u, is unchanged by the filtering.

Now, let u be any node such that the attribute of the elected q-route equals or is less preferred than the attribute of the elected p-route. Directly from Lemma 5, we obtain that the attribute of the elected q-route either remains the same of becomes less preferred. ∎

Theorem 11: DRAGON is optimal route-consistent.

Proof: Set F is the set of nodes such that the attribute of the elected q-route equals the attribute of the elected p-route in the initial state, and $\overline{F}$ is the set of nodes such that the attribute of the elected q-route is preferred to the attribute of the elected p-route in the initial state. From Lemma 9, we know that every node u belongs either to F or to $\overline{F}$.

Consider any filtering sequence. Suppose, first, that u belongs to $\overline{F}$. From Theorem 10, the attribute of the elected q-route remains preferred to the attribute of the elected p-route throughout all intermediate states and final state of the filtering sequence. Thus, u does not forgo q in the final state.

Now, suppose that u belongs to F. If u becomes oblivious of q at some intermediate state, then, from Theorem 10, we conclude that it remains oblivious of q throughout all succeeding intermediate states and the final state. If u has not become oblivious of q when its turn arrives of executing code $\mathbf{CR}$, then, again from Theorem 10, we deduce that the attribute of its elected q-route equals or is less preferred than the attribute of its elected p-route. Therefore, upon execution of code $\mathbf{CR}$, u filters q.

In summary, in the final state, nodes in F forgo q while nodes in $\overline{F}$ do not forgo q. A node u that forgoes q forwards data-packets according to the elected p-route, which, from the definition of set F, has an attribute equal to that of the elected q-route in the initial state: the final state is route-consistent. In addition, no node in $\overline{F}$ could dispense with q-routes without breaking route consistency: the final state is optimal route-consistent. ∎

The proof of the following, and last, theorem exhibits a filtering sequence for which all intermediate states are route-consistent. In practice, any filtering sequence where each node u executes code **CR** after all nodes that have u as forwarding neighbor for q have done so yields route-consistent intermediate states.

Theorem 12: There is a filtering sequence for which all intermediate states are route-consistent.

Proof: In the initial state, the set of nodes that elect a q-route and their forwarding neighbors for q form an acyclic digraph. We construct a filtering sequence to validate the following invariant: the state is route-consistent, and the set of nodes that do *not* forgo q and the set of links joining those nodes to their forwarding neighbors for q form an acyclic digraph.

Consider a filtering subsequence, letting S be the set of nodes that filters q and $D(S)$ be the acyclic digraph formed by nodes that do *not* forgo q and by links joining those nodes to their forwarding neighbors for q. We choose the next node to execute code **CR** to be a node belonging to F without links pointing to it in $D(S)$. If there is a such node u, then, upon execution of code **CR**, it filters q. Since all nodes that have u for forwarding neighbor for q already forgo q, filtering of q by u does not affect the elected q-routes of any other node, while $D(S+u)$ remains an acyclic digraph composed of nodes that do not forgo q and links joining them to their forwarding neighbors. In addition, since u belongs to F, filtering of q preserves route-consistency. If, on the other hand, all nodes without links pointing to them in $D(S)$ belong to $\overline{F}$, then, from Lemma 4, we conclude that none of the nodes of $D(S)$ satisfy the premise embodied in code **CR**. Hence, the optimal route-consistent state has been reached: nodes of $D(S)$ can complement the filtering sequence in any order. ■

F. Correctness of DRAGON Without Isotonicity

Without isotonicity, route-consistency in the final state is lost, in general. However, correctness is valid, depending exclusively on all cycles of the network being strictly absorbent. The proof of correctness given in Section IV-D was supported on the invariance of forwarding paths for q starting at t^p. That invariance also depends exclusively on strict-cycle-absorbency. In turn, the invariance was supported on the fact that a node that filters q does not affect a forwarding path for q that does not contain that node (Lemma 6). The latter statement no longer holds without isotonicity, in general.

We omit the general proof of correctness of DRAGON without isotonicity because of space limitations. Instead, we carry out the proof for the specific example of Figure 5, highlighting the proof strategy that can be used in the general case. In the figure, the origins of p and q are $u_1 = t^p$ and $u_4 = t^q$, respectively. The attributes of the p-route and the q-route formed by u_1 and u_4 are denoted by $\beta^p = R^*[t^p; p]$ and $\beta^q = R^*[t^q; q]$, respectively. Path $u_2u_0u_1u_3u_4$ is a forwarding path for q and $u_2u_0u_1$ is a forwarding path for p. If the routing policies of link u_3u_2 are not isotone, then the following sequence of events can be conceived. Node u_0 executes code **CR** and filters q. As a consequence, u_2 no longer learns a q-route from u_0. It elects the q-route learned directly from u_4. That q-route is sent to u_3, with u_3 electing the q-route learned from u_2 to the detriment of the q-route learned directly from u_4 along the forwarding path for q before filtering by u_0. Hence, filtering of q by u_0 ends up affecting forwarding path $u_1u_3u_4$. The q-route now elected at u_3 could possibly not be exported to u_1 creating a black hole there.

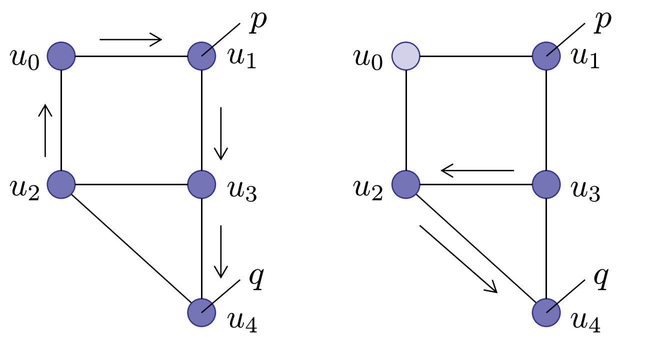

Fig. 5. Arrows point to forwarding neighbors for q. In addition, u_1 is a forwarding neighbor of u_0 for p and u_0 is a forwarding neighbor of u_2 for p. For an execution of code **CR** at u_0 to destroy forwarding path $u_1u_3u_4$, cycle $u_3u_2u_0u_1u_3$ would have to be non-strictly-absorbent.

However, we show that the sequence of events described above implies that cycle $u_3u_2u_0u_1u_3$ is not strictly absorbent. After u_0 filters q and u_2 elects the q-route learned from u_4, u_3u_4 stops being a forwarding path for q and $u_3u_2u_4$ becomes one. Thus,

$$L[u_3u_4](\beta^q) \succ L[u_3u_2u_4](\beta^q). \tag{10}$$

Because path $u_3u_2u_4$ has a different attribute from that of u_3u_4, path $u_2u_0u_1u_3u_4$ must have a different attribute from that of u_2u_4. Path $u_2u_0u_1u_3u_4$ is a forwarding path for q before u_0 filters q. Hence,

$$L[u_2u_4](\beta^q) \succ L[u_2u_0u_1u_3u_4](\beta^q). \tag{11}$$

Since u_0 filtered q, it satisfied the premise embodied in code **CR**. Therefore,

$$L[u_0u_1u_3u_4](\beta^q) \succeq L[u_0u_1](\beta^p). \tag{12}$$

Rule **RO** is satisfied at u_1. So,

$$\beta^p \succeq L[u_1u_3u_4](\beta^q). \tag{13}$$

Let

$$\begin{aligned} \alpha_3 &= L[u_3u_4](\beta^q); \\ \alpha_2 &= L[u_2u_4](\beta^q); \\ \alpha_0 &= L[u_0u_1u_3u_4](\beta^q); \\ \alpha_1 &= \beta^p. \end{aligned}$$

Then, Inequalities (10), (11), (12), and (13) become, respectively,

$$\begin{aligned} \alpha_3 &\succ L[u_3u_2](\alpha_2); \\ \alpha_2 &\succ L[u_2u_0](\alpha_0); \\ \alpha_0 &\succeq L[u_0u_1](\alpha_1); \\ \alpha_1 &\succeq L[u_1u_3](\alpha_3). \end{aligned}$$

The previous set of inequalities reveal that cycle $u_3u_2u_0u_1u_3$ is not strictly absorbent.

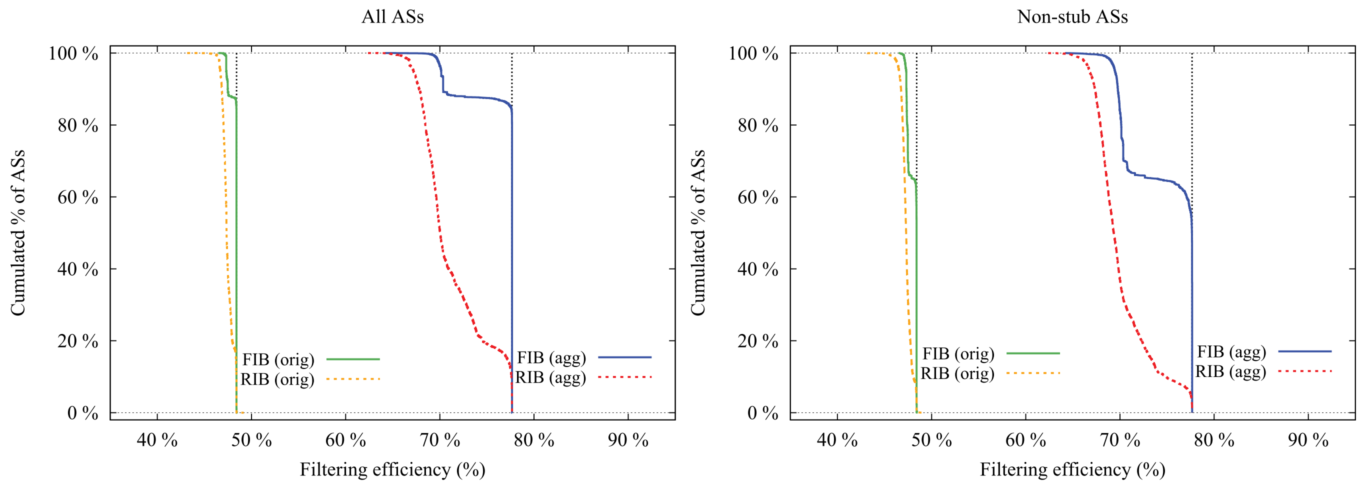

Fig. 6. CCDF of FIB and RIB filtering efficiencies. Without and with aggregation prefixes, the maximum filtering efficiencies are 49% and 78%, respectively, marked by vertical dashed lines. Left. CCDF for all ASs. Around 87% of them reach the maximum FIB filtering efficiency. Right. CCDF for non-stub ASs. Around 63% of them reach the maximum FIB filtering efficiency.

The general proof of correctness follows the strategy above. We start with a node z that satisfies the premise embodied in code **CR** and presume that a forwarding path for q starting at the origin of p is affected. Then, we will be able to find an alternate sequence of forwarding paths for q, before and after z filters q, which, together with a forwarding path for p starting at z, are entangled in such a way that has to presuppose the presence in the network of a cycle that is not strictly absorbent.

V. Evaluation

We evaluate the stable-state performance of DRAGON when deployed on all ASs. Section V-A describes the network topologies, prefix assignments, and routing policies that were used. The savings in the sizes of forwarding-tabes and routing-tables are presented in Section V-B. The stretch in AS-path-lengths is discussed in Section V-C.

A. Data-Sets and Methodology

We ran DRAGON on data-sets of inferred Internet topologies and data-sets of IPv4-prefixes-to-origin-AS mappings made available by CAIDA [23], [24]. We experimented with several topologies and several mappings, corresponding to different dates, and concluded that the results are consistent across them. We present results for the inferred topology and IPv4-prefixes-to-origin-AS mappings of February 2015.

The topology data-set contains a list of pairs of ASs classified into "customer-provider" or "peer-peer." Accordingly, we used the GR routing policies. We fixed some inaccuracies in the data-set. Cycles where each AS is a customer of the next around the cycle were broken. ASs that prevented the topology from being policy-connected were removed. From 49,755 ASs and 379,674 links, we ended up with 48,999 ASs (keeping 98% of them) and 364,916 links (keeping 96% of them). Of the 48,999 ASs, 41,664 are stubs (85% of them). Of the 364,916 links, 93,695 join a customer to a provider (26% of them), 93,695 join a provider to a customer (26% of them), and 177,526 join a peer to another peer (48% of them).

The IPv4-prefixes-to-origin-AS mappings data-set mapped a few prefixes to ASs that were not present in the topology. These prefixes were removed. The data-set also contained a few prefixes with more than one origin AS. Without loss of generality, we chose just one origin AS per prefix. Last, the data-set also contains a few pairs of child-prefix-parent-prefix where the origin of the child prefix is higher up in the AS-hierarchy. In this case, either the child prefix or the parent prefix were removed. From 562,467 IPv4 prefixes, we ended up with 530,444 IPv4 prefixes (keeping 94% of them). Out of these prefixes, 51% have no parent prefix in the routing system and 40% have the same origin AS as their parent prefix. In order to subject the prefixes without a parent to the filtering strategy, we included 58,843 prefixes according to the aggregation strategy discussed in Section III-F, increasing the total number of prefixes by 11%.

B. Filtering Efficiency

The *filtering efficiency* of a FIB (of a RIB) is the normalized difference between the number of entries in the FIB (in the RIB) before and after DRAGON is deployed on all ASs. The size of a FIB is reduced by one for every prefix forgone at the AS. The size of a RIB is reduced by one for every prefix forgone at a neighbor AS that announced the prefix when DRAGON was not deployed. The *maximum filtering efficiency* of a FIB (of a RIB) is obtained when only the prefixes without a parent prefix in the routing system are stored in the FIB (in the RIB). The maximum filtering efficiency is the same for FIBs and RIBs. Its value is 49% without aggregation prefixes and 78% with aggregation prefixes.

Figure 6 plots filtering efficiencies of FIBs and RIBs without and with aggregation prefixes, orig curves and agg curves, respectively. Results are presented as Complementary Cumulative Distribution Functions (CCDFs). A point (x, y) of a curve means that $y\%$ of ASs have a filtering efficiency of more than $x\%$. The left-hand side plot shows filtering efficiencies across *all* ASs, whereas the right-hand side plot focuses on non-stub ASs.

DRAGON enables near-maximum FIB filtering efficiency at *each* AS. Without aggregation prefixes, each AS has a FIB filtering efficiency of at least 47%, differing from the maximum possible by 2%. This is partially explained by the predominance of child prefixes that have the same origin AS as their parent prefix (84% of all child prefixes; 40% of all prefixes). Such child prefixes are filtered, rather trivially, by the neighbors of the ASs that originated them. However, with the introduction of aggregation prefixes, child prefixes and parent prefixes are invariably originated at different ASs, while DRAGON still attains near-maximum FIB filtering efficiencies. With aggregation prefixes, each AS has a FIB filtering efficiency of at least 69%, differing from the maximum possible by 9%.

Without and with aggregation prefixes, DRAGON enables approximately 87% of the ASs to attain the maximum FIB filtering efficiency. This is partially explained by the prevalence of stub ASs without peers (75% of all ASs). Because the topology is policy-connected, these stub ASs elect provider routes for all prefixes. Hence, they forgo all prefixes which have a parent prefix in the routing system, thereby reaching the maximum filtering efficiency. However, interestingly, many non-stub ASs also attain the maximum FIB filtering efficiency. The plot on the right-hand side of Figure 6 shows that to be the case for 63% of all non-stub ASs. The FIB efficiency results fare well with those obtained through FIB aggregation techniques [17], while DRAGON scales all the routing state and not just the FIBs.

The RIB filtering efficiency is slightly worse than the FIB filtering efficiency, offset, in the worst case, by 2% without aggregation prefixes and by 10% with aggregation prefixes. This is expected since when an AS filters a prefix it is already saving on its FIB size but not on its RIB size. It will only save on the RIB size of its neighbor ASs (see Section III-A).

C. AS-Path-Length Stretch

We applied the filtering code to the GR-attributes alone, ensuring route-consistency with respect to these attributes, but accepting a distortion in the AS-paths traversed by data-packets (see Section III-D). AS-path distortion can only affect data-packets whose destination address match to a child prefix, since only these prefixes are subject to filtering. Thus, in order to better appreciate the small stretch introduced by DRAGON, we focus exclusively on child prefixes. We consider pairs composed of an AS and a child prefix, and determine the number of hops in the AS-path followed by data-packets that start at the AS with destination address matching the child prefix, when DRAGON is deployed and when it is not. The *stretch* of an AS-child-prefix pair is the ratio between the two numbers defined above. When DRAGON is not deployed, data-packets are guided by elected routes pertaining to the child prefix throughout their journeys. The number of hops associated with an AS-child-prefix pair is read off directly from the route that the AS elects for the child prefix.

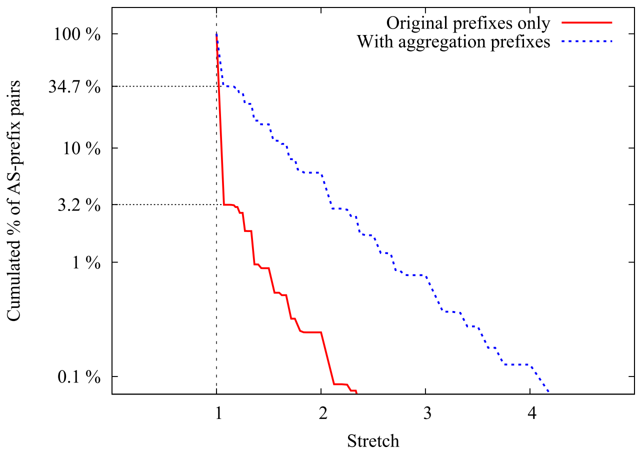

Fig. 7. CCDF of stretch, with a logscale y-axis. Without and with aggregation prefixes, 97% and 65% of all AS-child-prefix pairs, respectively, have no stretch.

However, when DRAGON is deployed, data-packets may be guided by elected routes pertaining to ever more specific prefixes throughout their journey ending at the AS that originated the child prefix. We computed the number of hops in such AS-paths by emulating the forwarding decisions made at each AS. When there was more than one forwarding neighbor to choose from, we used a uniform distribution to randomly select just one. Although this procedure adds a random factor to the results, the emulation was run multiple times to guarantee robustness of the results, and it showed little variation across different executions.

The median AS-path-length of an AS-child-prefix pair when DRAGON is not deployed is 4. When DRAGON is fully deployed it remains at 4 without aggregation prefixes and increases to 5 with aggregation prefixes. Figure 7 plots the stretch over all AS-child-prefix pairs without and with aggregation prefixes, orig curves and agg curves, respectively. Results are presented as CCDFs, where each point (x, y) means that for $y\%$ of the AS-child-prefix pairs, the stretch is greater than x. Without aggregation prefixes, we see that 97% of the AS-child-prefix pairs bear no stretch at all. Again, this is partially explained by the predominance of child prefixes that are originated by the same AS as their parent prefixes. With aggregation prefixes, 65% of the AS-child-prefix pairs bear no stretch, whereas 94% of the AS-child-prefix pairs have a stretch not greater than 2.

VI. Related Work

A. Scalability Limits of Routing

Existing work on the scalability limits of routing [1]–[3] presuppose full control over the parameters of routing. Moreover, almost all of that work is premised on shortest-path routing, exploring the fundamental trade-off between the size of forwarding-tables and the stretch in the lengths of paths traversed by data-packets. A recent paper embarks on the scalability limits of policy-based routing [25], 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 previous work, DRAGON is a distributed algorithm proposed for the Internet's existing IP addressing scheme and has been framed for arbitrary routing policies.

B. Characterizing Growth of the Internet Routing System

Measurement studies track the growth in the number of IP prefixes and BGP update messages over time [4], [26], and identify the *causes* of growth such as multi-homing, traffic engineering, and address allocation policies [27]–[31]. DRAGON *reduces* the number of globally routed IP prefixes and BGP update messages.

C. Reducing Forwarding-Table Size

As the number of globally routed IP prefixes grew, researchers explored ways to reduce the size of *forwarding-tables* by aggregating related entries [11]–[13], keeping entries only for recently used prefixes [32], compressing information [33], or directing some traffic to routers with room for larger tables [34]. These optimization techniques do not reduce the size of *routing-tables* or the number of BGP update messages, and require re-optimizations when routing decisions change.

D. 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 originated several hops away, and can sometimes lead to black holes and loops [35]. Other techniques for reducing the size of routing-tables work only within a single AS, missing opportunities for global scalability gains [9].

VII. Conclusions

DRAGON is a distributed route-aggregation algorithm that operates with standard routes of a routing vector protocol. It comprises a filtering strategy and an aggregation strategy. The filtering strategy is composed of filtering code and an origination rule that together allow nodes to dispense with many prefixes without creating black holes in a stable state. DRAGON works with 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 can be made route-consistent as well. DRAGON's aggregation strategy boosts the filtering possibilities within the network.

Applied to the Internet, DRAGON harnesses whatever hierarchical structure there is in inter-AS routing to create an efficient routing system. Evaluation of DRAGON on inferred topologies of the Internet with realistic prefix assignments and routing policies show reductions in the amount of routing and forwarding state close to 70% in each ASs, reaching up to 80% in some ASs, with minimal stretch in the lengths of AS-paths traversed by data-packets. We leave for future work an evaluation of DRAGON under partial deployment and an evaluation of its dynamics upon link failures and additions. Toward this end, we developed a simulator and are currently implementing DRAGON in the open source router software BIRD.

Appendix
Notation

A *path* P is a network with node-set $\{u_0, u_1, \ldots, u_n\}$ and link-set $\{u_0u_1, u_1u_2, \ldots, u_{n-1}u_n\}$. We refer to path P by its linear sequence of nodes, $P = u_0u_1 \cdots u_n$. A *walk* in a network is a linear sequence of nodes $P = u_0u_1 \cdots u_n$ such that u_iu_{i+1} is a link in that network, for $0 \le i < n$. If the nodes of a walk are all distinct, then the walk defines a path. Given walk $P = u_0u_1 \cdots u_n$, we write, for $0 \le i \le j \le n$:

$$\begin{aligned} u_iPu_j &= u_iu_{i+1} \cdots u_j; \\ u_iP &= u_iu_{i+1} \cdots u_n; \\ Pu_j &= u_0u_1 \cdots u_j. \end{aligned}$$

A *cycle* C is a network obtained by linking the last node of a path to its first. Its node-set and link-set are of the form $\{u_0, u_1, \ldots, u_n\}$ and $\{u_0u_1, u_1u_2, \ldots, u_{n-1}u_0\}$, respectively. We refer to cycle C by its circular sequence of nodes, $C = u_0u_1 \cdots u_{n-1}u_0$. The walk around cycle C starting and ending at node u, crossing each link exactly once, is denoted by uCu.

Acknowledgment

The authors thank the anonymous reviewers for comments that led to a better presentation of their work. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

References

[1] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks performance evaluation and optimization," *Comput. Netw.*, vol. 1, no. 3, pp. 158–174, Jan. 1977.

[2] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," in *Proc. ACM SIGCOMM*, Aug. 1988, pp. 35–42.

[3] D. Krioukov, K. C. Claffy, K. Fall, and A. Brady, "On compact routing for the Internet," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 41–52, Jul. 2007.

[4] *BGP Routing Table Analysis Reports*. [Online]. Available: http://bgp.potaroo.net

[5] G. Huston, "What's so special about 512?" *Internet Protocol J.*, vol. 17, no. 2, pp. 2–18, Dec. 2014.

[6] C. Edwards. (Sep. 2014). *Internet Routing Failures Bring Architecture Changes Back to the Table*, ACM News. [Online]. Available: http://cacm.acm.org/news/178293-internet-routing-failures-bring-architecture-changes-back-to-the-table/fulltext

[7] R. Lemos. (Aug. 2014). *Internet Routers Hitting 512K Limit, Some Become Unreliable, ArsTechnica*. [Online]. Available: http://ars.to/1r9AbxJ

[8] Z. B. Houidi, M. Meulle, and R. Teixeira, "Understanding slow BGP routing table transfers," in *Proc. Internet Meas. Conf.*, Nov. 2009, pp. 350–355.

[9] E. Karpilovsky, M. Caesar, J. Rexford, A. Shaikh, and J. van der Merwe, "Practical network-wide compression of IP routing tables," *IEEE Trans. Netw. Service Manage.*, vol. 9, no. 4, pp. 446–458, Dec. 2012.

[10] R. Hinden and S. Deering, *IP Version 6 Addressing Architecture*, document RFC 4291, Feb. 2006.

[11] R. P. Draves, C. King, S. Venkatachary, and B. D. Zill, "Constructing optimal IP routing tables," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 88–97.

[12] X. Zhao, Y. Liu, L. Wang, and B. Zhang, "On the aggregatability of router forwarding tables," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[13] Z. A. Uzmi *et al.*, "SMALTA: Practical and near-optimal FIB aggregation," in *Proc. ACM CoNEXT*, 2011, pp. 29:1–29:12.

[14] B. Carré, *Graphs and Networks*. Oxford, U.K.: Clarendon, 1979.

[15] J. L. Sobrinho, "An algebraic theory of dynamic network routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, pp. 1160–1173, Oct. 2005.

[16] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 681–692, Dec. 2001.

[17] J. L. Sobrinho, L. Vanbever, F. Le, and J. Rexford, "Distributed route aggregation on the global network," in *Proc. ACM CoNEXT*, Dec. 2014, pp. 161–172.

[18] V. Fuller and T. Li, *Classless Inter-Domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*, document RFC 4632, Aug. 2006.

[19] J. L. Sobrinho and F. Le, "A fresh look at inter-domain route aggregation," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2556–2560.

[20] M. Schapira, Y. Zhu, and J. Rexford, "Putting BGP on the right path: A case for next-hop routing," in *Proc. ACM SIGCOMM Workshop Hot Topics Netw.*, Oct. 2010, pp. 3:1–3:6.

[21] Y. Liao, L. Gao, R. Guérin, and Z.-L. Zhang, "Safe interdomain routing under diverse commercial agreements," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, pp. 1829–1840, Dec. 2010.

[22] M. Thorup and U. Zwick, "Compact routing schemes," in *Proc. ACM Symp. Parallel Algorithms Archit. (SPAA)*, 2001, pp. 1–10.

[23] *The CAIDA AS Relationships Dataset*. [Online]. Available: http://www.caida.org/data/active/as-relationships/, accessed Feb. 2015.

[24] *The CAIDA Routeviews Prefix to as Mappings Dataset for IPv4 and IPv6*. [Online]. Available: http://www.caida.org/data/routing/routeviews-prefix2as.xml, accessed Feb. 2015.

[25] A. Gulyas, G. Retvari, Z. Heszberger, and R. Agarwal, "On the scalability of routing with policies," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1610–1618, Oct. 2014.

[26] G. Huston, "Analyzing the Internet's BGP routing table," *Internet Protocol J.*, vol. 4, no. 1, pp. 2–15, Mar. 2001.

[27] H. Narayan, R. Govindan, and G. Varghese, "The impact of address allocation and routing on the structure and implementation of routing tables," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 125–136.

[28] T. Bu, L. Gao, and D. Towsley, "On characterizing BGP routing table growth," *Comput. Netw.*, vol. 45, no. 1, pp. 45–54, May 2004.

[29] X. Meng *et al.*, "IPv4 address allocation and the BGP routing table evolution," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 1, pp. 71–80, Jan. 2005.

[30] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the scalability of BGP: The role of topology growth," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 8, pp. 1250–1261, Oct. 2010.

[31] L. Cittadini *et al.*, "Evolution of Internet address space deaggregation: Myths and reality," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 8, pp. 1238–1249, Oct. 2010.

[32] Y. Liu, S. O. Amin, and L. Wang, "Efficient FIB caching using minimal non-overlapping prefixes," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 1, pp. 14–21, Jan. 2013.

[33] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger, "Compressing IP forwarding tables: Towards entropy bounds and beyond," in *Proc. ACM SIGCOMM*, Aug. 2013, pp. 111–122.

[34] H. Ballani, P. Francis, T. Cao, and J. Wang, "Making routers last longer with ViAggre," in *Proc. USENIX NSDI*, 2009, pp. 453–466.

[35] F. Le, G. G. Xie, and H. Zhang, "On route aggregation," in *Proc. ACM CoNEXT*, Dec. 2011, Art. ID 6.

João Luís Sobrinho received the Licenciatura and Ph.D. degrees in electrical and computer engineering from the Instituto Superior Técnico, Universidade de Lisboa, Portugal, in 1990 and 1995, respectively. Before joining academia in 1997, he was with Bell Labs, Lucent Technologies, for two years. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Instituto Superior Técnico, and a Senior Researcher with the Instituto de Telecomunicações.

He won an Internet Society Applied Networking Research Prize 2015, the IEEE Communications Society William R. Bennett Prize 2006, and an IEEE PIMRC 1994 Best Ph.D. Student Paper Award.

Laurent Vanbever received the Ph.D. degree in computer science from the University of Louvain, Belgium, in 2012. He was a Post-Doctoral Research Associate with Princeton University, where he collaborated with Prof. J. Rexford. He is currently an Assistant Professor with ETH Zürich, where he leads the Networked Systems Group since 2015.

He won several awards for his research, including the ACM SIGCOMM 2015 Best Paper Award, the ACM SIGCOMM Doctoral Dissertation Award (runner-up), the University of Louvain Best Thesis Award, the ICNP 2013 Best Paper Award, and three Internet Society Applied Networking Research Prizes for his work on inter-domain routing and software-defined networking.

Franck Le received the Diplome d'Ingénieur degree from the École Nationale Supérieure des Télécommunications de Bretagne, France, in 2000, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, in 2010. He is currently a Research Scientist with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

He won the ICNP 2007 Best Paper Award and a NSF Graduate Research Fellowship.

André Sousa received the B.S. and M.S. degrees in electrical and computer engineering from the Instituto Superior Técnico, Universidade de Lisboa, Portugal, in 2012 and 2014, respectively. He was an M.S. Research Associate with the Instituto de Telecomunicações. He is currently a Software Engineer with Prodrive Technologies B.V., Son, The Netherlands.

Jennifer Rexford received the B.S.E. degree in electrical engineering from Princeton University, in 1991, and the Ph.D. degree in electrical engineering and computer science from the University of Michigan, in 1996. She was with AT&T Labs-Research for eight years. She joined Princeton University in 2005, where she is currently the Gordon Y. S. Wu Professor of Engineering and the Chair of Computer Science.

She has co-authored the book entitled *Web Protocols and Practice* (Addison-Wesley, 2001). She was an ACM Fellow (2008), and a member of the American Academy of Arts and Sciences (2013) and the National Academy of Engineering (2014). She was the 2004 winner of ACM's Grace Murray Hopper Award for outstanding young computer professional. She served as the Chair of ACM SIGCOMM from 2003 to 2007.