Understanding Route Aggregation

Franck Le[†] **Geoffrey G. Xie**[*] **Hui Zhang**[‡]

March 9, 2010
CMU-CS-10-106

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[†]Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA.
[*]Computer Science, Naval Postgraduate School, Monterey, CA, USA.
[‡]Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

This research was sponsored by the NSF under the 100x100 Clean Slate Project [1] (NSF-0331653), the 4D Project [2] (NSF-0520187), a Graduate Research Fellowship, and grants CNS-0520210, CNS-0721574. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF, or the U.S. government.

Keywords: routing anomalies, route aggregation, forwarding loops

Abstract

Route aggregation, the method to supersede a set of routes by a single, more general route, is a universal mechanism that is either explicitly included in a routing protocol specification or added by router vendors as a configuration option. Widely deployed for both intra-domain and inter-domain routing purposes, route aggregation (RA) can be vulnerable to routing anomalies, and is fingered to be the cause of many reported loops and blackholes. In this paper, we posit that the problem arises from a lack of fundamental understanding of the RA mechanism. Moreover, we present the first rigorous and comprehensive analysis of route aggregation based on an abstract model. We show that the range of potential anomalies from RA configurations is much wider than previously documented. We demonstrate that existing RA configuration guidelines are inadequate. We further prove that determining whether the collection of RA configurations in a network can result in a persistent forwarding loop is NP-hard. Given this complexity, we identify a sufficient condition for ensuring convergence and loop-free forwarding paths. Finally, from the condition, we derive a migration strategy to harden existing and future network designs against RA-induced route oscillations and forwarding loops.

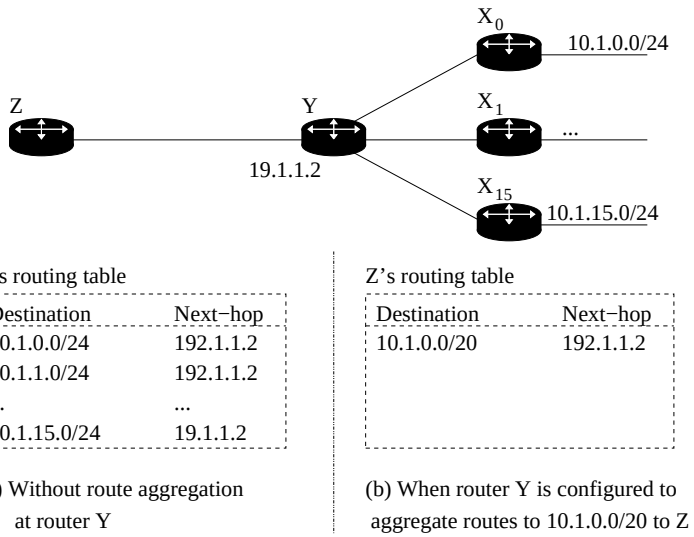

Figure 1: Route aggregation allows router Y to combine multiple routes (10.1.0.0/24, 10.1.1.0/24, ..., 10.1.15.0/24) into a single one (10.1.0.0/20).

1 Introduction

Route aggregation (also commonly called *route summarization* or *supernetting*) designates the method to supersede a set of routes with a single more general route. To illustrate this feature, consider the network depicted in Figure 1. Suppose all routers run a common routing protocol (e.g., RIP or EIGRP). Every router X_i ($0 \leq i \leq 15$) is directly connected to an interface with IP prefix $10.1.i.0/24$. Consequently, router Y's routing table contains at least 16 entries corresponding to the network addresses 10.1.0.0/24, 10.1.1.0/24, ..., 10.1.15.0/24. Rather than advertising these 16 prefixes to router Z, route aggregation allows router Y to combine all of them into a single destination prefix 10.1.0.0/20, and announce only one route to Z.

Once an aggregate route (e.g., 10.1.0.0/16) is configured at a router, all the more specific routes (e.g., 10.1.0.0/24, 10.1.1.0/24, etc.) become *child* or *contributing* routes at that router. The router will start announcing the aggregate route only after it is activated, by the presence of at least one of its child routes. The announcement will stop if no child route is present.

Route aggregation can be triggered by static routes or dynamic features built into routing protocols [7]. All routing protocols (e.g., BGP, OSPF, RIP, EIGRP, IS-IS) of commercial routers support automated ways to aggregate routes and some implementations even enable it by default.

1.1 A fundamental & prevalent mechanism

Route aggregation is an essential design primitive for network operations. It is used to meet important design objectives [18, 16], some of which are explained below.

• *Reduction of routing table size:* The success of the Internet led to an exponential growth of the routing table size in the early 1990s. The growth rate raised concerns regarding a core router's ability to support the exploding number of routing entries. In response, the networking community

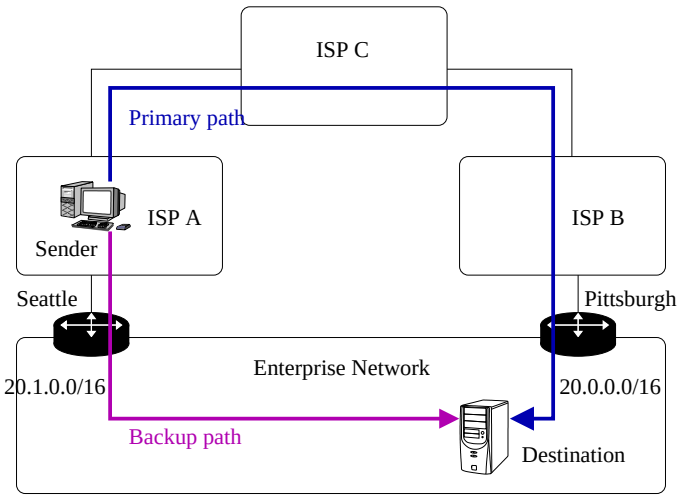

Figure 2: Multi-homed networks rely on route aggregation to implement domain backup. Border routers advertise both specific and aggregate routes.

introduced the concepts of Classless Inter-Domain Routing (CIDR) and route aggregation [7]. As illustrated in Figure 1, this feature allows routers to reduce the number of route entries and has been crucial in curbing the routing table size growth [10].

• *Reduction of route flaps:* Route aggregation increases the Internet routing stability. By restricting the scope of route advertisements, route flaps at the network edge are contained. For example, in Figure 1, let us assume that the interface connected to router X_0 and corresponding to 10.1.0.0/24 flaps. A hardware failure may cause the router to continuously announce that interface alternately as "up" and "down". With route aggregation deployed at router Y, those routing instabilities are hidden from router Z.

• *Network wide design goals:* Operators rely on route aggregation to implement a variety of essential design goals. One example is domain backup for multihomed networks [15]. Let us consider the scenario depicted in Figure 2. The enterprise network has two service providers, ISP A on the West coast, and ISP B on the East coast. The East coast branch is allocated IP range 20.0.0.0/16 and the West coast branch has the subnet 20.1.0.0/16. Incoming traffic from the Internet destined for the East coast should enter through the "Pittsburgh" router, while incoming traffic for the West coast should enter through the "Seattle" router. Additionally, incoming traffic from the Internet destined for the East (respectively, West) coast should be able to enter through the "Seattle" (respectively, "Pittsburgh") router as a backup path. To satisfy these requirements, operators rely on route aggregation. They configure "Pittsburgh" to simultaneously announce the specific 20.0.0.0/16, and more general 20.0.0.0/15 prefixes. Similarly, "Seattle" is configured to advertise both 20.1.0.0/16 and 20.0.0.0/15.

The above list of objectives is not exhaustive but to illustrate some of the important reasons why operators deploy route aggregation. In fact, when networks are connected to their provider(s) through multiple border routers, operators also frequently rely on concurrent advertisements of general and more specific routes at their border routers to load balance incoming traffic. Furthermore, our own discussions with operators revealed that route aggregation is also used to hide

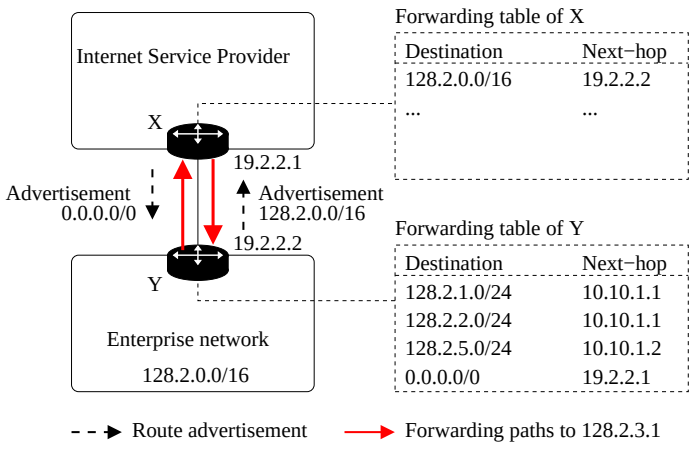

Figure 3: A persistent forwarding loop because of default route. Packets sent to the unused IP prefix 128.2.3.0/24 keep bouncing between X and Y.

network topology.

The importance of route aggregation has been substantiated by two recent empirical studies [19, 20]. They confirm that route aggregation is widely deployed in operational networks. In our own research, we analyzed the router configuration files of Abilene [3] and a large-scale campus network [17]. We found that those networks depend on route aggregation in both intra and inter-domain settings, involving IS-IS, EIGRP, and BGP protocols.

1.2 An extremely vulnerable mechanism

A recent study [19] discovered a surprisingly large number of persistent forwarding loops in the Internet and concluded route aggregation to be the root cause behind 50% of them. Route aggregation can also result in blackholes [18], which are surprisingly prevalent in the Internet [11]. We illustrate these known anomalies with two simple examples[1] below.

Persistent forwarding loops

Consider the scenario depicted in Figure 3. The owner of the enterprise network is allocated a full class B address range with the prefix 128.2.0.0/16. Suppose within the enterprise network, subnets with prefixes of 128.2.1.0/24, 128.2.2.0/24, and 128.2.5.0/24 are deployed, while many other sub-prefixes including 128.2.3.0/24 are *unused*. We show how packets sent to one of the unused addresses may be trapped in persistent forwarding loops.

Suppose router Y is configured with an aggregate route to 128.2.0.0/16 and it has routes to 128.2.1.0/24, 128.2.2.0/24, and 128.2.5.0/24 in its forwarding table. Because of the presence of the three child routes, Y advertises the aggregate route to router X of the ISP network. Suppose at the same time, X advertises the default route (i.e., 0.0.0.0/0) to Y. Now consider a packet arriving at X and with a destination address that is part of the unused child prefix 128.2.3.0/24. X

[1] We have validated both examples in a lab environment.

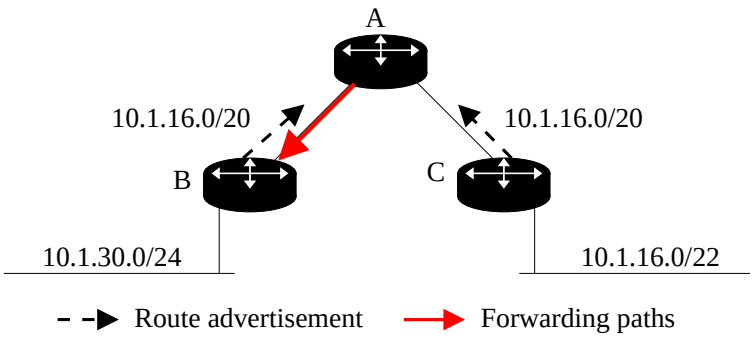

Figure 4: Illustration of a blackhole.

forwards the packet to Y because of the aggregate route advertised by Y, but Y returns the packet to X because Y has no specific route to the destination and must use the default route. In other words, the packet will be stuck in a persistent forwarding loop until its TTL is down to 0. Although the anomaly does not affect traffic of valid users directly, researchers have warned about potential security exploits of the anomaly to cause network congestions and losses of valid packets.

To prevent this anomaly, operators commonly install a *sink route* in the router. It is a route corresponding to the aggregate address and pointing to the Null interface. Its goal is to drop all packets that match the aggregate address but do not match any more specific route. For example, installing a static route for 128.2.0.0/16 pointing to the Null interface at router Y will prevent the above forwarding loop. Packets destined to 128.2.3.0/24, upon reaching Y, will be discarded at Y. Previous studies [19] have speculated the lack of sink routes (e.g., because of accidental omissions) to be the root cause of many observed loops in the Internet.

Blackholes

We consider the scenario depicted in Figure 4. Router B is the gateway of a network with prefix 10.1.1.30/24 and router C is the gateway of another network with prefix of 10.1.16.0/22. Suppose B and C are (mis)configured to advertise to router A the same aggregate route with destination prefix 10.1.16.0/20. As a result, router A receives two routes to 10.1.16.0/20 with identical metrics. A may select B as its next-hop for packets destined to 10.1.16.0/22. However, upon arriving at B, the packets are dropped because B does not have a route for them. In other words, these packets are blackholed despite the existence of a valid path to their destination (i.e., A-C).

1.3 Contributions of paper

We conducted a series of controlled experiments to understand route aggregation. The results were intriguing. Not only did the behaviors vary with router vendors, which is understandable given a lack of clear standards on the mechanism, a greater range of potential anomalies was also discovered. The observations have motivated us to develop a comprehensive and rigorous analysis of route aggregation in order to uncover the root cause of the problems and find solutions. In this paper, we present our findings, which can be summarized as follows:

1. We report experimental results indicating that the range of routing anomalies that can result from route aggregation is much larger than previously reported. We disclose four new types of anomalies, and explain why the existing guidelines are not sufficient.

2. We introduce a canonical router model that precisely defines the per-router behavior of route aggregation with two primitives. The model is able to predict the impact of a router's local configuration of route aggregation on its FIB content. We show that the network-wide effect of route aggregation can be inferred by combining the new per-router model with the concept of activation sequence for route propagation [8].

3. With aid of the new analytical model, we have discovered two important but previously unknown properties about route aggregation. The first property can be thought of as a correctness criterion for the route aggregation primitives. The second property is an unexpected behavior: the configuration of route aggregation on one router interface can impact how routes are advertised on other interfaces of the same router. Route aggregation can impact a network's reachability in surprising ways because of this property.

4. We show that the problem of determining whether the collection of route aggregation configurations in a network can result in persistent forwarding loops is NP-hard. In other words, it may not always be possible to statically check the correctness of route aggregation configurations.

5. We identify a sufficient condition for route aggregation parameters to guarantee convergence and loop-free forwarding paths. We further derive a configuration guideline as well as a migration strategy to harden current and future route aggregation designs against route oscillations and forwarding loops.

2 Variety of behaviors

The existing literature on route aggregation is limited and spotty, primarily consisting of documents from router vendors about their own proprietary mechanisms. As such, we conducted a number of experiments to determine the consistency of route aggregation behaviors across router vendors and routing protocols. The testbed consisted of Cisco 2600 (running IOS version 12.2 or 12.3) and Juniper 4300 routers (running JUNOS version 8.2R1.7).

2.1 Cisco IOS

The experiments reveal that Cisco routers handle route aggregation differently for each routing protocol. We describe the main properties of route aggregation in RIP, EIGRP, BGP, and OSPF.

2.1.1 RIPv2

Cisco's RIPv2 supports two types of route aggregation: (1) interface route summarization, and (2) auto summarization. The first type enables routers to advertise an explicitly configured aggregate route, instead of the child routes, out of a given interface. It is configured through the

```
Codes:  C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        ...
D        10.2.0.0/16 is a summary, 00:00:11, Null0
D        10.2.1.0/24 [90/307200] via 192.168.1.1, 00:00:46, FastEthernet0/0
C        192.168.1.0/24 is directly connected, FastEthernet0/0
```

Figure 5: Illustration of an EIGRP sink route (shaded) in a FIB.

ip summary-address rip command on a per interface basis. In contrast, auto-summarization automatically summarizes subprefixes (e.g., 10.1.0.0/16, 10.2.0.0/16, etc.) to a classful network prefix (e.g., 10.0.0.0/8). The metric of an aggregate route is set to the lowest metric of all child routes that are present.

2.1.2 EIGRP

Like RIP, EIGRP supports both interface route summarization and auto summarization, and it determines the metric of an aggregate route in the same way as RIP. A notable difference is that EIGRP creates a sink route as soon as it advertises an aggregate route. The sink route is presented to the route selection procedure [5], which determines the route to be installed in the forwarding table. By default, an EIGRP sink route is assigned a relatively small administrative distance (AD) of 5. The administrative distance is used to rank routes from different routing processes (EIGRP 17, OSPF 10, RIP, static, etc.) announcing the same destination prefix. The preference is given to the route with the smallest administrative distance [4]. As described in Section 1.2, a sink route can prevent forwarding loops by dropping all packets that match the aggregate route but do not match any of the more specific routes in the forwarding table. Figure 5 illustrates a sink route (which is the shaded entry) installed in a router's forwarding table.

2.1.3 BGP

BGP supports route aggregation. In addition, BGP can append an AS-SET (a union of AS numbers) to the AS-PATH attribute[2] to annotate which ASes the child routes are from.

To illustrate, we consider the network in Figure 6, which consists of four BGP Autonomous Systems (AS 10, AS 20, AS 30, AS 40). Suppose router D receives routes to prefixes 10.1.1.0/24 and 10.1.2.0/24 from ASes 10 and 20, respectively. D will aggregate the two routes into a single 10.1.0.0/16 route and advertises it to AS 40.

2.1.4 OSPF

Contrary to the previous routing protocols, OSPF is a link-state protocol, with link state information (not routes) flooded to all participating routers. However, OSPF offers the notion of areas

[2]We note that without the keyword *as-set* at the end of the *aggregate-address* command, the AS-SETs are not appended to the AS-PATH. Our own discussions with operators revealed that this keyword is frequently overlooked, effectively making the corresponding BGP aggregate routes undetectable from BGP table dumps.

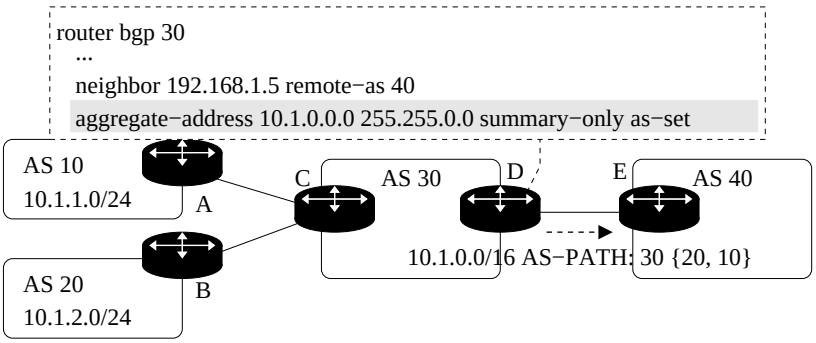

Figure 6: Route summarization in BGP.

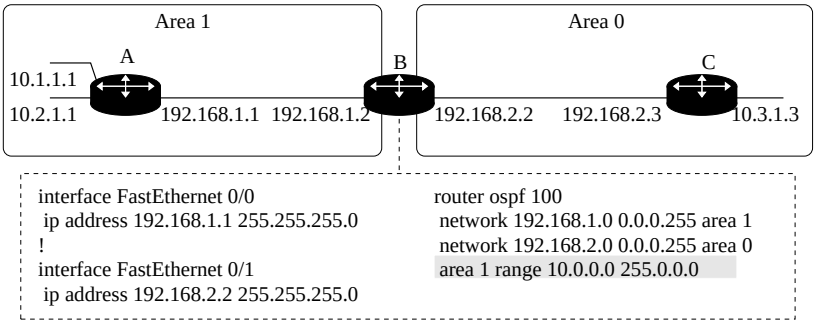

Figure 7: Route aggregation in OSPF.

to allow hierarchical network design. All non-zero areas must be directly connected to the backbone area (area 0), and routers are required to share identical information only within their own area. Between these OSPF areas, routes are exchanged in a vectoring manner, and hence can be aggregated.

In OSPF, route summarization can be performed in two cases: (1) at the area boundaries by the Area Border Routers (ABR) (e.g., router B in Figure 7) and, (2) when external routes are redistributed into OSPF. In both cases, the cost of a summary route is set to the maximum cost of all child routes. Finally, we observe that OSPF also creates a sink route when advertising an aggregate route.

2.2 JUNOS

In Juniper routers, route aggregation is configured in a consistent way across all routing protocols. The configuration consists of three steps (Figure 8): The first step (lines 1 to 5), begins with a *routing-options* statement, and creates an aggregate route. Characteristics of the aggregate route can also be specified in this statement (e.g., community, metric, tags, etc.) Then, a *policy-options* statement (lines 6 to 13) defines the export policies for the aggregate route. Finally, the export rules are applied to the protocols (lines 15 to 18) where the aggregate routes are to be advertised.

The design of JUNOS differs from that of Cisco IOS in two aspects. First, JUNOS requires a child route to be in the forwarding table for an aggregate route to be considered by the route

```
routing-options {
   aggregate {
      route 10.1.0.0/24;
   }
}
policy-options {
   policy-statement aggregate-into-rip {
      term first-term {
         from protocol aggregate;
         then accept;
      }
   }
}
protocols {
   rip {
      export aggregate-into-rip;
   }
}
```

Figure 8: Configuration of supernetting in Juniper routers.

selection procedure. In contrast, Cisco IOS requires the presence of a child route in a routing process specific routing information base (RIB).

Second, Juniper distinguishes two types of aggregate routes: aggregate and generated. These two types of routes differ in how the next-hop is generated. The next-hop of an aggregate route is set to *discard*: A packet that matches this aggregate route but no other more specific routes is then dropped. This functionality is very similar to that of sink routes. In contrast, the next-hop of a generated route is set to the next-hop of the first child route.

Finally, JUNOS by default assigns aggregate routes a distinct default administrative distance value of 130.

3 Disclosure of new anomalies

From the observed behaviors, we infer four new types of routing anomalies that can happen with route aggregation. We validated all the presented anomalies in our tested, with the exception of the route oscillations presented in Section 3.1 as this anomaly requires specific race conditions that were difficult to produce in our environment. We describe the anomalies in the order of their degree of severity.

3.1 Route oscillations

We observe that route aggregation can result in route oscillations. We assume the network depicted in Figure 9. Routers X and Y are Juniper routers. They are configured to advertise only the aggregate route 10.1.0.0/16 into the OSPF routing instance. The following cycle illustrates the occurrence of a route oscillation:

t_1 Because of a directly connected child route (10.1.1.0/24) in their FIBs, X and Y each creates an aggregate route and passes it to the route selection procedure. Being the only available route

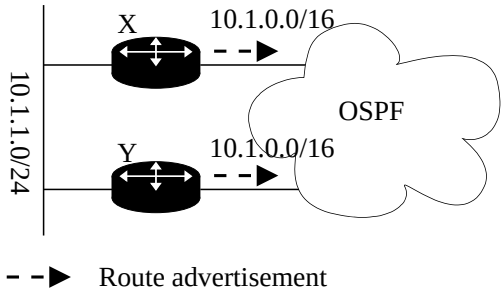

Figure 9: Route oscillations.

to 10.1.0.0/16, the aggregate route is installed in the FIB, and then advertised into the OSPF instance according to the export policies.

t_2 Router X receives the aggregate route advertised by router Y. As a result, X has two routes to 10.1.0.0/16: (1) the local aggregate route, and (2) the OSPF route received from Y. As explained previously, its route selection procedure ranks these two routes based on their administrative distances. Suppose OSPF processes at X and Y are configured with a lower administrative distance than that of aggregate routes (e.g., 100 vs. 130). As such, X prefers the OSPF route and installs it in the FIB. Consequently, the local aggregate route is no longer the active route and is no longer advertised. Router Y may concurrently perform the same actions, preferring the OSPF route to 10.1.0.0/16 from X, and consequently, stop advertising the aggregate route to 10.1.0.0/16 in OSPF.

t_3 Because X has stopped advertising the aggregate route into OSPF, Y no longer receives any route to 10.1.0.0/16 from X. Therefore, Y reverts back to the local aggregate route as the active route installed in the FIB. Similarly, X reverts back to its local aggregate route. The resulting state is identical to that of step t_1, i.e., we obtain a route oscillation. The cycle continues as long as routers X and Y are synchronized and process the received messages at about the same time.

3.2 Forwarding loop due to one aggregate route

Section 1.2 described the potential formation of a persistent forwarding loop. The anomaly results from the interplay between two aggregate routes (0.0.0.0/0, and the more specific 128.2.0.0/16) announced by two distinct routers. Because neither of the two advertising routers has a route to some IP prefix, traffic sent to those destinations keeps bouncing between the two routers.

Our experiments reveal that a single aggregate route can also result in persistent forwarding loops. In other words, even in the absence of a default route (or a more general aggregate route), a single aggregate route can create a loop. As depicted in Figure 10, our experiment consisted of Cisco routers running RIP. Router B is configured to advertise an aggregate route to 10.2.0.0/16 out of its interface to C. The following sequence of actions resulted in a persistent forwarding loop:

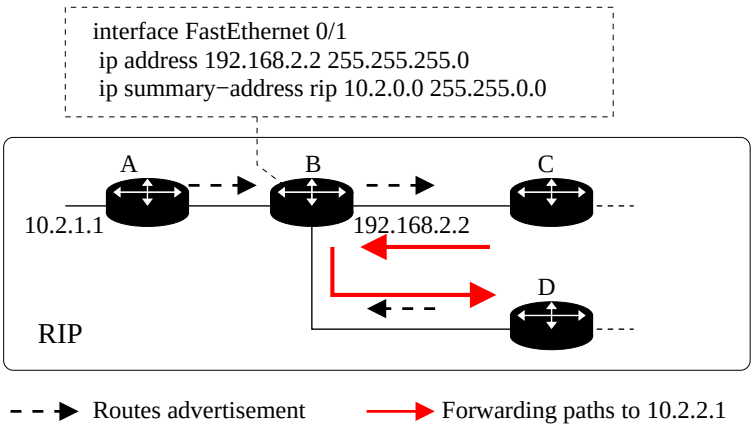

Figure 10: Persistent forwarding loops due to one aggregate route.

t_1 Because router A has a directly connected interface to 10.2.1.0/24, it learns a route to this prefix and then advertises it to B .

t_2 Router B receives the announcement from A and installs a route to 10.2.1.0/24 in its FIB. Then, as this route is a child route of 10.2.0.0/16, B sends an aggregate route to 10.2.0.0/16 to router C with a hop-count of 2. We note that although B advertises a route to 10.2.0.0/16 to C, B does not have any entry for this prefix in its FIB.

t_3 The route gets propagated and comes back to router B, from router D. Because B has no route to 10.2.0.0/16, it accepts the route from D and installs it in the FIB. Since this route from D has a higher hop-count than the initial aggregate route advertised by B, B will not propagate it further. As such, the network converges with a persistent forwarding loop for traffic destined to an unused IP address within 10.2.0.0/16.

To illustrate it, let us focus on an unused sub-prefix of 10.2.0.0/16, e.g., 10.2.2.0/24. We sent packets to a destination within that unused prefix (e.g., 10.2.2.1) from C. C forwards the packets to B because of the aggregate route advertised by B. Then, B forwards them to D because of the 10.2.0.0/16 route advertised by D. Finally, the packets come back to C. We have a persistent forwarding loop.

3.3 Forwarding loop for allocated IPs

The previous cases illustrated the formation of persistent forwarding loops for unused IP prefixes. Our experiments reveal that a single aggregate route can actually also cause loops that directly impact traffic sent to IP addresses assigned to users.

We implemented the network depicted in Figure 11 consisting of Cisco routers running RIP. Router A is the gateway of a group of networks (10.1.1.0/24, 10.1.2.0/24, etc.), and it is configured to send an aggregate route with prefix 10.0.0.0/8 to router B. Router E is the gateway of another group of networks (10.20.1.0/24, 10.30.1.0/24, etc.), and is configured to send the same aggregate route 10.0.0.0/8 to router F. We have seen similar type of configurations in real networks [17].

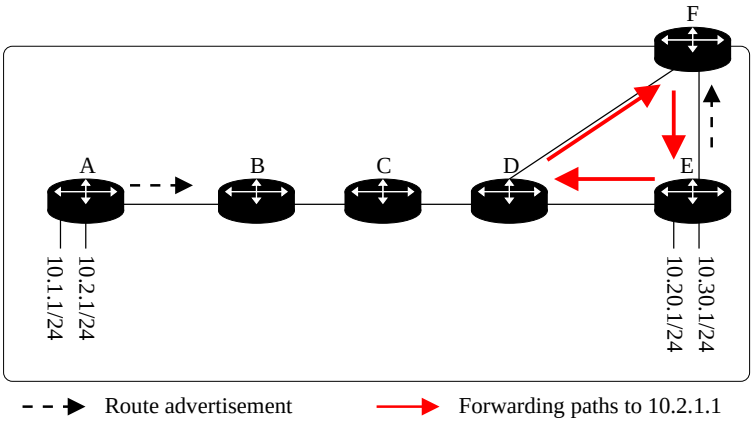

Figure 11: Persistent forwarding loops affecting directly connected IP prefixes.

The depicted topology is free of anomalies. In particular, any host can reach any assigned IP address. However, let us now assume that a link is added between routers D and F. (The link may be added to alleviate the load of router E.) In this new setting, all traffic arriving at F and destined to a subnet within 10.0.0.0/8 and directly connected to A will be caught in a persistent forwarding loop F-E-D-F. We observed the following sequence of events:

t_1 Routers A and E advertise an aggregate route with prefix 10.0.0.0/8 to B and F, respectively.

t_2 F learns a route to 10.0.0.0/8 from E and advertises it to D. In parallel, the aggregate route from A gets propagated to B, C and D.

t_3 D receives two routes to 10.0.0.0/8. Since the route from F has a lower hop-count value, D selects the route from F pointing to F as its next-hop. Then, D further advertises the route to C and E.

t_4 E initially has no route to 10.0.0.0/8 in the FIB. Upon receiving the route from D, E accepts it and installs it in the FIB, pointing to D as its next-hop. Since this route from D has a higher hop-count than the initial aggregate route advertised by E, E will not propagate it further. As such, the network converges with a persistent forwarding loop for traffic destined to a subnet within 10.0.0.0/8 and directly connected to A. For example, when sending packets to 10.1.1.1 from F, the packets are forwarded to E, which sends them to D; but D returns them back to F.

3.4 Perpetual Count-to-Infinity

We re-implemented the scenario of Figure 10. All routers were configured to run RIPv2 and router B is configured to advertise an aggregate route with prefix 10.2.0.0/16 to C. As the only change in setup, we substituted router B with a Juniper router. We observed a perpetual count-to-infinity problem:

t_1 Router B, receiving a child route (10.2.1.0/24) from A, creates an aggregate route in its FIB for 10.2.0.0/16 and advertises it to router C.

t_2 Router C accepts the received route, installs a route to 10.2.0.0/16 in its FIB, increments the hop-count by one and re-advertises the route to its neighbor(s).

t_3 The route propagates in the RIP instance, and router D receives the route initially advertised by C, installs a route to 10.2.0.0/16 in its FIB, increments the hop-count and sends it to B.

t_4 Router B receives the RIP route to 10.2.0.0/16 from D. B has two routes to 10.2.0.0/16: the local aggregate route and the RIP route received from D. By default, JUNOS assigns aggregate routes an administrative distance of 130, and RIP routes a smaller administrative distance of 100. As a result, the RIP route is preferred and installed in the FIB instead of the local aggregate route. B increments the hop-count of the RIP route, and sends the route to C.

t_5 The hop count keeps incrementing as illustrated in the previous steps until it reaches the maximum authorized value (i.e., 16). The route is then withdrawn, and the cycle repeats from Step 1.

This anomaly is particularly undesirable because the network is permanently unstable. In addition to the continuous RIP messages, routers are continually busy processing the control messages. Also, traffic sent to unused sub-prefixes of 10.2.0.0/16 results in a forwarding loop B-D-...-C-B[3].

3.5 Inadequacy of current guideline

Current specifications [7] have identified the following rule for routers performing route aggregation:

A routing domain which performs summarization of multiple routes must discard packets which match the summarization but do not match any of the explicit routes which makes up the summarization. This is necessary to prevent routing loops in the presence of less-specific information (such as a default route). Implementation note - one simple way to implement this rule would be for the border router to maintain a "sink" route for each of its aggregations. By the rule of longest match, this would cause all traffic destined to components of the aggregation which are not explicitly known to be discarded.

We observe that in both the JUNOS and EIGRP experiments presented in Section 3.4, the operation of router B complies to the above rule. Whenever it advertises the aggregate route (10.2.0.0/16), it creates a local aggregate route pointing to *discard* as its next-hop. However, as illustrated by the experiments, the network remains vulnerable to routing anomalies.

[3]We repeated the same experiment with all Cisco routers running EIGRP. When the EIGRP sink route at router B is assigned a higher administrative distance than that of regular EIGRP routes, we did not observe a count-to-infinity problem. However, we observed that the aggregate route was sometimes present and sometimes absent. In other words, we still obtained a route oscillation.

4 A model for route aggregation

The previous section revealed that the range of routing anomalies that can occur with route aggregation is in fact much larger than previously reported. In addition, existing guidelines are not sufficient to address the problem. These results motivated us to develop an analytical model to reason about route aggregation. In particular, a key question is: *Given the route aggregation configuration of a network, can we determine whether it is free of routing anomalies?*

To answer this question, we first introduce a canonical router model that precisely defines the route aggregation logic performed at a router. Our key insight is that the route aggregation logic is composed of two simple primitives, which may be invoked differently with different vendor implementations.

The model allows us to predict the FIB content and the route advertisements at each router. The model actually also reveals two important properties of route aggregation. First, inaccurate implementations could result in local instabilities. Second, unexpectedly, configuration of aggregate routes on a router interface can impact route advertisements on other interfaces of the same router.

Finally, we explain how one can build upon the per-router model to derive a network-wide view of the interactions between aggregating routers, and analyze the paths adopted by each router. As such, the model provides the ability to predict potential routing anomalies.

4.1 Terminology

We formally define a prefix, a route and the relation *more specific than*.

- A destination prefix represents a range of contiguous IP addresses. For example, the destination prefix 10.1.1.0/24 represents the range of IP addresses $\{10.1.1.0, 10.1.1.1, 10.1.1.2, \ldots, 10.1.1.255\}$.
- A destination prefix P_1 is said to be strictly more specific than a destination prefix P_2 when $P_1 \subset P_2$.
- A route r consists of three components: (1) a destination prefix P, (2) a set of metrics m (to rank routes), (3) a next-hop h, where traffic destined to P should be forwarded to. We assume that router software implementations include means to differentiate aggregate routes from other routes (e.g., routes received from peers, static routes, direcly connected routes).
- Considering two routes r_1 and r_2, r_1 is said to be strictly more specific than r_2 when $r_1.P \subset r_2.P$. For brevity, we also note it $r_1 \subset r_2$.

4.2 Route aggregation functions

The essence of route aggregation lies in two primitives which we term *add-sink()* and *adv-aggr()*.

Primitive 1 *add-sink(E, A)*

Input: A set of routes E present at the router, and a set of configured aggregate routes A

$S = \{\}$
We remove the aggregate routes from E
for all $a \in A$ **do**
  if $\exists\ e \in E$: $e \subset a$ **then**
    Add a to S
    break
  end if
end for
Present S to the route selection procedure

Primitive 2 *adv-aggr(E, A)*

Input: A set of routes E present at the router, a set of aggregate routes A configured on a given interface

We remove the aggregate routes from E
for all $a \in A$ **do**
  if there exists a child route of a in E **then**
    $a.m$ = $metric(a, E)$
    Remove all child routes of a in E
    Add a to E
    break
  end if
end for
Advertise E on the interface

4.2.1 *add-sink()*

As described in Section 2, vendor implementations attempt to create a sink route in the FIB upon receiving a child route of a configured aggregate route.

The *add-sink()* primitive actually presents two characteristics. First, the sink route is not always installed in the FIB. Instead, in the presence of a child route, a sink route is created with its own administrative distance value. For example, sink routes created from an EIGRP routing process have a default AD value of 5. Sink routes derived from Cisco OSPF processes are assigned a value of 110, and those from Cisco BGP processes have an AD of 200. In Juniper routers, aggregate routes have an AD of 130. After creation, the sink routes are passed to the route selection procedure [5] which determines a best route to be installed in the FIB, from all routes offered by the different routing processes (e.g., RIP, OSPF, static), for each given destination prefix. As illustrated in the experiment of Section 3.4, the best route is the one with the lowest AD value.

Second, the location where the primitive examines for the presence of child routes differs depending on the implementation. For example, as long as a child route is present in the FIB,

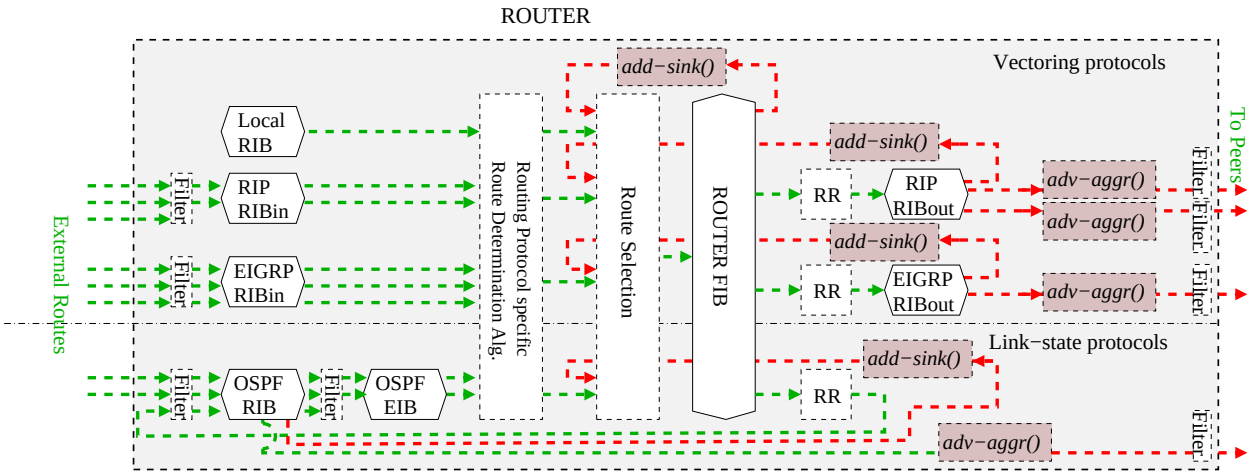

Figure 12: Per-router model of RA. The essence of RA lies in two main primitives *add-sink()* and *adv-aggr()*. Different implementations apply them at different locations of the router.

Juniper routers create the aggregate route (Section 2.2). In contrast, the presence of child routes in the FIB is not a sufficient condition for Cisco routers. For example, we configured a Cisco router to run EIGRP and to advertise an aggregate route to 10.2.0.0/16 on one of its interfaces. We further configured the router with a static route to 10.2.1.0/24. Although a child route of 10.2.0.0/16 is present in the FIB, we observed that the aggregate route is not advertised. The aggregate route is advertised only when a child route is one of the routes to be announced to the EIGRP peers, as is the case when the static route is redistributed into EIGRP.

The *add-sink()* primitive takes two input arguments and outputs a set of sink routes to the route selection procedure. The first argument E is a set of routes present at the router. However, depending on the implementations, this set may correspond to the routes in either the FIB (for JUNOS), or a protocol specific routing information base (RIB) (for Cisco). The second argument A is the set of all aggregate routes that are configured on the router.

4.2.2 *adv-aggr()*

The second primitive, *adv-aggr()* handles the creation and advertisement of aggregate routes to the router's peers. JUNOS implementations rely on *export* policies to announce the aggregate routes from the FIB into the routing processes (Section 2.2). As such, JUNOS routers rely on route redistribution [13] to advertise the aggregate routes. In contrast, Cisco implementations depend on a separate primitive that we call *adv-aggr()*.

As highlighted in Section 2, the configuration of aggregate routes can be performed per interface. Operators may customize different aggregate routes to be announced on each interface. Consequently, *adv-aggr()* is performed for each interface and per routing process. The primitive takes two input arguments: E, the set of routes present in a RIBout – the part of a RIB for storing routes to be advertised out, and A, the set of aggregate routes configured on a given interface. It determines and advertises a set of aggregate routes on each interface.

To determine the set of routes to advertise, *adv-aggr()* first removes the aggregate routes from E. Then, it substitutes all the child routes present in E for the aggregate routes configured on the interface. The metric of the aggregate routes is set by the *metric()* function which is routing process specific. For example, *metric(a, E)* returns the maximum of all the child routes of a in E for OSPF, but returns the minimum of all the child routes of a in E for RIP. Certain implementations offer operators the option to simultaneously advertise child routes and aggregate routes. However, given a routing process (e.g., RIP, EIGRP), existing implementations do not allow operators to configure two overlapping aggregate routes (i.e., a_1, a_2 with $a_1.P \cap a_2.P \neq \{\ \}$) on a same interface.

4.3 Per-router Model

A router may run multiple routing protocols (e.g., BGP, RIP, OSPF, IS-IS). In fact, a router can also run multiple instances of a same routing protocol (e.g., OSPF 1, OSPF 2, etc.) For each instance, the router creates a separate routing process.

Prior studies [13, 14] have proposed a router-level model to study route selection and route redistribution, i.e., how routers rank routes received from different routing processes and exchange routing information between them. We extend this previously proposed model to include route aggregation. We illustrate where the route aggregation primitives are applied and how they impact the content of the different data structures.

As depicted in Figure 12, the model differs slightly for vectoring and link-state routing processes. The differences come from the fact that link-state processes forward all the received information, whereas vectoring processes advertise only the best routes.

4.3.1 Vectoring processes

Each vectoring routing process (e.g., RIP or EIGRP) is assigned a *RIBin* for incoming route announcements and a *RIBout* for outgoing advertisements. A set of filters first discard invalid advertisements (e.g., RIP updates whose hop count exceeds 16) and routes that violate preconfigured routing policies. The remaining valid routes are added to the *RIBin*. Then, among all the entries in the *RIBin*, a protocol specific route determination algorithm selects the best route for each destination prefix. For example, RIP prefers routes with the lowest hop-count. The best routes are forwarded to the route selection procedure which selects the most preferred route across the processes (e.g., RIP, EIGRP) based on the administrative distances. The results from route selection are installed in the FIB.

Depending on the vendor implementation, the first primitive *add-sink()* may then be applied to the FIB entries. In particular, Juniper routers execute the *add-sink()* primitive, taking routes from the FIB, and the set of configured aggregate routes as input. The function returns a set of new sink routes to the route selection procedure.

Routes from the FIB are then processed by route redistribution (RR). RR allows the exchange of routes across routing processes. RR installs a route automatically to the RIBout of the route's own routing process. For example, RR always install a RIP route into the RIBout of the RIP process. In contrast, redistribution between different processes (e.g., redistributing an OSPF or EIGRP route into a RIP process) must be explicitly enabled. The outcomes of RR are stored in the

*RIBout*s. For further details of the RR procedure, we refer the reader to two recent papers by Le *et al.* [13, 14].

Again, depending on the implementation, functions of route aggregation may then be applied to the *RIBout*s. For example, Cisco routers perform both the *add-sink()*, and *adv-aggr()* primitives, taking the routes in the *RIBout*s as the first argument, and the set of aggregate routes configured on a given interface as the second argument. *add-sink()* is executed per routing process, and creates new routes to be presented to the route selection procedure. In contrast, *adv-aggr()* is executed per interface and per routing process. This primitive determines the routes to be advertised on a given interface by a given routing process.

4.3.2 Link-state processes

We model link-state routing processes with a different set of data structures. Rather than two types of RIBs, each link-state process is associated with a RIB and an Eligible Information Base (EIB). The RIB stores the information received from the peers, as well as the locally originated and locally redistributed routes. All members of a link-state routing instance should have identical information in their RIBs. EIB is a separate database and stores a subset of routes from the RIB: those eligible for installation in the FIB. The filter between RIB and EIB prevents locally redistributed routes from entering the EIB. The protocol specific route determination algorithm processes the entries from the EIB to compute the best routes. These best routes are then passed to the router wide route selection procedure.

Although the data structures for link-state routing processes differ than those used for vectoring routing processes, the functionality of the route selection, route redistribution and route aggregation procedures remain the same independently of whether the routes are learned from a vectoring or link-state routing process. In particular, from a route aggregation perspective, the only difference lies in the fact that, instead of taking routes from a RIBout, *add-sink()* and *adv-aggr()* take routes from a RIB for link-state processes.

4.4 Route aggregation properties

The per-router model of route aggregation reveals two important properties. The first one is a necessary guideline for correctness. The second property is unexpected, and may be considered as a pathology due to the current design.

Property 1: *A child route must be strictly more specific than the aggregate route.*

Implementations of route aggregation that violate Property 1 may result in persistent local instabilities. This is because the outcome of *add-sink()* impacts the input to the route selection procedure, which in turn affects the input to *add-sink()* (Figure 12). This cycle may create local instabilities. To illustrate it, we assume an implementation of route aggregation that is not compliant with Property 1. We assume a router configured to advertise an aggregate route with prefix 10.2.0.0/16. Suppose another 10.2.0.0/16 route is received from a neighbor of the router. The noncompliance to Property 1 implies that the route from the neighbor is considered a child route

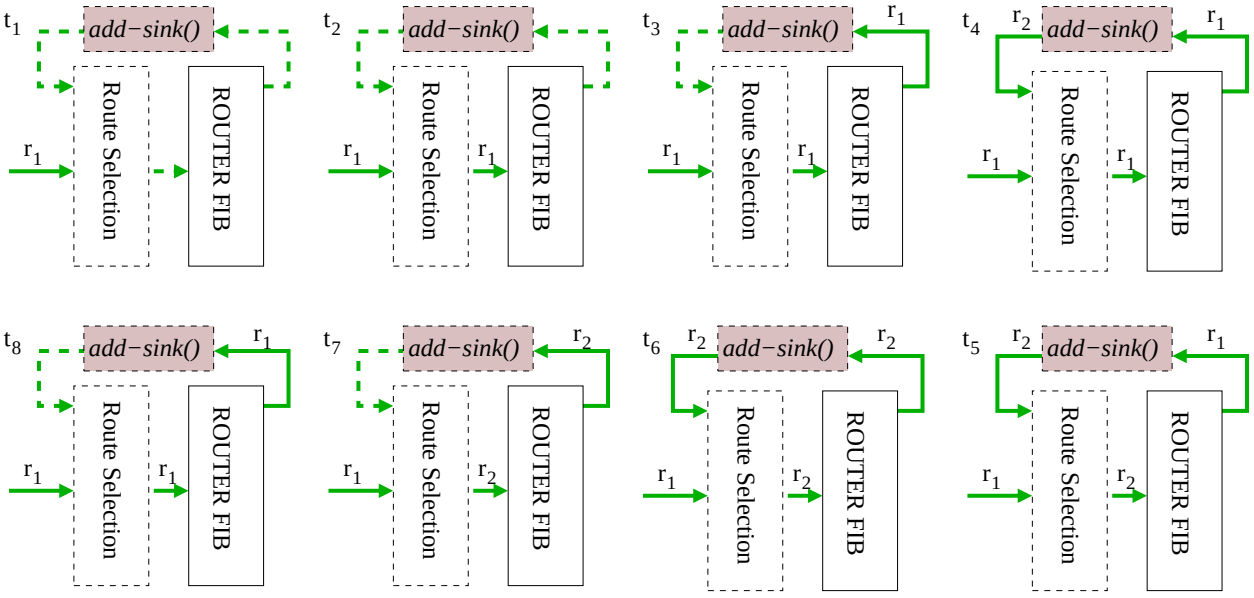

Figure 13: Local instabilities that can derive from violation of Property 1.

and thus, activates the aggregate route. A permanent route oscillation may occur as follows, which is also demonstrated in Figure 13.

t_1 The route received from the neighbor, denoted by r_1 is presented to the route selection procedure.

t_2 Because it is the only available route, r_1 is installed in the FIB.

t_3 *add-sink()* is executed, taking r_1 as the first input, and the aggregate route, denoted by r_2, as the second argument. Because r_1 is a child route to r_2, *add-sink()* returns r_2.

t_4 r_2 is then presented to the route selection procedure.

t_5 Assuming that r_2 has a lower AD value than r_1, r_2 is preferred and installed in the FIB instead of r_1.

t_6 The function *add-sink()* is now executed with r_2 as the first argument.

t_7 *add-sink()* presents the empty set to the route selection procedure. As such, only r_1 is presented to the route selection procedure.

t_8 r_1 is installed in the FIB and *add-sink()* is executed with r_1 as the first argument. We notice that this state is identical to the one at t_3. We obtain a permanent route oscillation.

An implementation compliant with Property 1 ensures that the newly created aggregate route (i.e., r_2 in Figure 13) is strictly more general than any of its child routes, i.e., $r_1.P \subset r_2.P$. This then guarantees that at time t_5, the child route is not withdrawn.

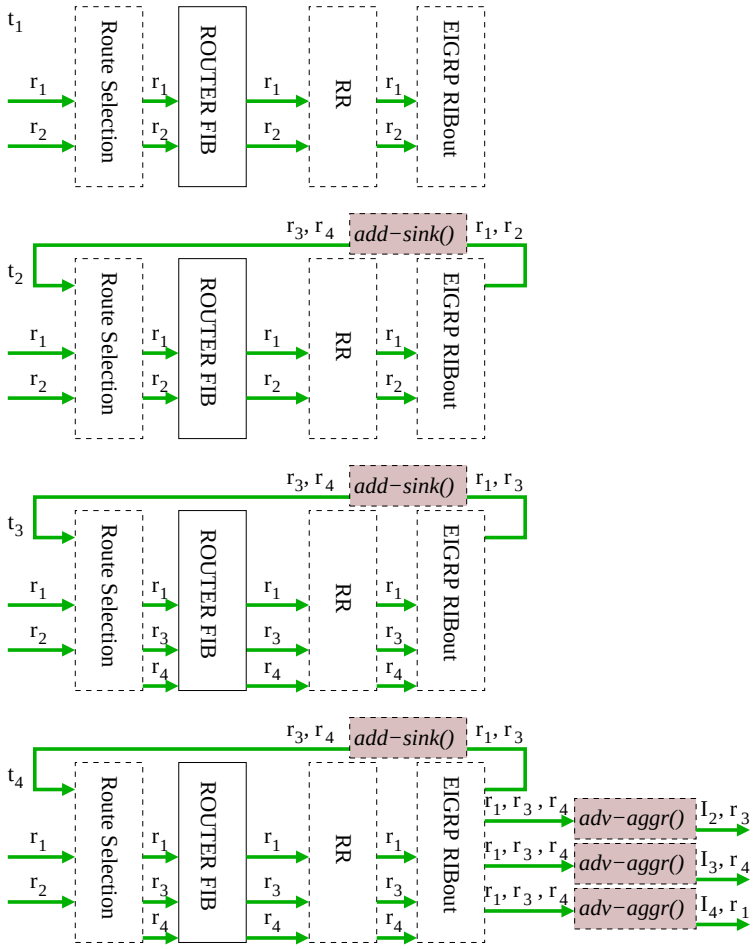

Figure 14: Configuration of aggregate routes on interfaces I_2 and I_3 impacts the announcements on interface I_4. Although the router receives two routes r_1, r_2 to different prefixes, only one of the routes (r_1) is advertised on I_4.

Property 2: *Configuration of route aggregation on an interface can prevent the advertisements of routes on other interfaces.*

The per-router model allows us to infer the FIB content and the advertisements at each router. In particular, from the model, we discover a surprising and counterintuitive result. Configuring an aggregate route (e.g., with prefix 10.2.0.0/16) on an interface (e.g., Ethernet 0/0) may prevent a route received from a neighbor from being announced on another interface (e.g., Ethernet 0/1).

To illustrate it, we assume a Cisco router running EIGRP on four interfaces I_1, I_2, I_3 and I_4. We assume that the router receives only two routes: r_1 and r_2 to destination prefixes 192.168.1.0/24 and 192.168.0.0/16, respectively on interface I_1 (Figure 14). We further assume that the router is configured to advertise an aggregate route r_3 to 192.168.0.0/16 out of I_2, and aggregate route r_4 to 192.0.0.0/8 out of I_3.

t_1 First, the two received routes, r_1 and r_2, are installed in the FIB, and the EIGRP RIBout.

t_2 Then, after executing *add-sink(*$\{r_1, r_2\}, \{r_3, r_4\}$*)*, the output $\{r_3, r_4\}$ is presented to the route selection procedure.

t_3 Because r_3 has an AD value of 5 whereas r_2 has an AD value of 90, r_3 is preferred and installed in the FIB, serving as a sink route. As for r_4, it is the only route to 192.0.0.0/8 and is therefore also installed in the FIB.

The EIGRP RIBout is updated. r_2 is removed, while r_3 and r_4 are added. *add-sink(*$\{r_1, r_3, r_4\}, \{r_3, r_4\}$*)* is executed returning $\{r_3, r_4\}$.

t_4 *adv-aggr()* is performed for each interface.
For I_2, *adv-aggr(*$\{r_1, r_3, r_4\}, \{r_3\}$*)* returns r_3. As such, r_3 is advertised out on I_2.
For I_3, *adv-aggr(*$\{r_1, r_3, r_4\}, \{r_4\}$*)* returns r_4. Finally, on I_4, *adv-aggr(*$\{r_1, r_3, r_4\}, \{\}$*)* returns r_1.

Although the advertisements on interfaces I_2 and I_3 may be as expected, the announcements on I_4 is surprising. No route aggregation is configured on I_4. When the router receives two routes r_1 and r_2 to different prefixes, one may expect both routes to be advertised out of I_4. However, it turns out that only r_1 is announced. r_2 has been "filtered out". We implemented the above scenario and validated the results using Cisco routers.

4.5 Network wide predictions

The per-router model gives us the ability to predict the FIB and the announcements at each router. Adopting the previously proposed concept of activation sequence [8], we can then model route propagation between routers, and analyze the network-wide impact of route aggregation.

5 A NP-hard problem

Theorem 1: *The problem of determining whether the collection of route aggregation configurations in a network is vulnerable to persistent forwarding loops is NP-hard.*

Proof: The proof is inspired by previous works [9, 13]. We call LOOP-RA the problem of determining whether a network configuration of route aggregation is vulnerable to persistent forwarding loops. We prove that LOOP-RA $\geq_P$ 3-CNF SAT. Because the 3-CNF SAT problem is a NP-complete problem, this implies that the LOOP-RA problem is NP-hard.

To show that 3-CNF SAT is polynomial time reducible to LOOP-RA, we consider an instance of 3-CNF SAT. Let $C = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ be a set of k clauses of length at most 3 over n Boolean variables $\{X_1, X_2, \ldots, X_n\}$. Every clause C_i $(1 \leq i \leq k)$ is of the form $l_1^i \vee l_2^i \vee l_3^i$ with each l_j^i $(1 \leq j \leq 3)$ being of the form of X_m or $\overline{X_m}$ $(1 \leq m \leq n)$.

We construct a graph $G = (V, E)$ (where V represents the set of routers, and E the set of physical links between the routers) such that C is satisfiable if and only if there exists an activation sequence [8, 13] such that G converges to a state which includes a persistent forwarding loop. We construct $G = (V, E)$ as follows.

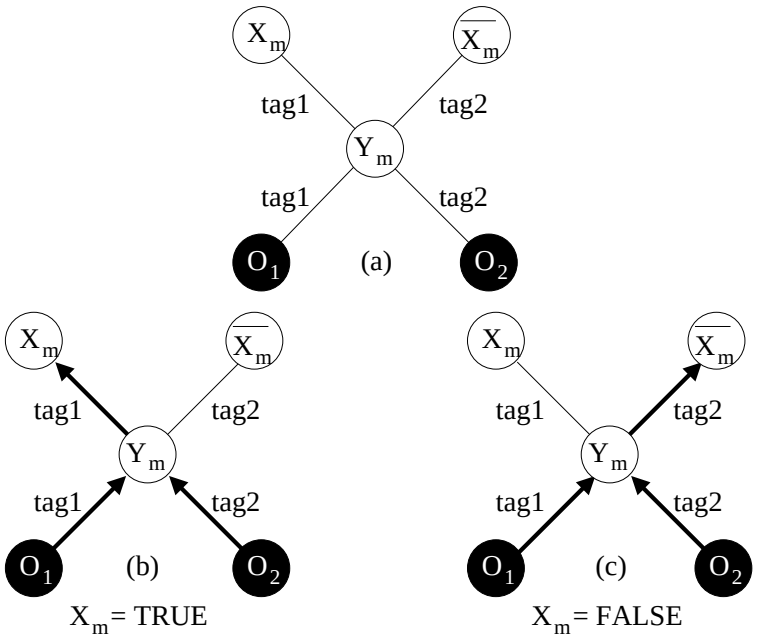

Figure 15: Conversion of a variable into a subgraph. Each variable X_m is represented by the topology depicted in (a). All routers run RIPv2. Routes advertised from O_1 are tagged with *tag1* whereas those advertised by O_2 are labeled with *tag2*. Y_m re-advertises the active routes tagged with *tag1* to X_m and those tagged with *tag2* to $\overline{X_m}$. Because Y_m receives two equal routes to d, it randomly selects one of them to become the active route. Depending on the selection outcome, we can obtain two possible states illustrated in (b) and (c). We associate the outcome of (b) (respectively, (c)) with the value TRUE (respectively, FALSE) assigned to X_m. In the case of (b) (respectively, (c)), X_m (respectively, $\overline{X_m}$) learns a route to P.

1. We focus on a destination prefix P (e.g., 192.168.1.0/24) and add two nodes O_1 and O_2 directly connected to P to V.

2. For each clause C_i ($1 \leq i \leq k$), insert a new node labeled C_i into V.

   For each literal l^i_j ($1 \leq j \leq 3$) of the form X_m or $\overline{X_m}$ ($1 \leq m \leq n$),

   - If $X_m \notin V$, add the subgraph depicted in Figure 15(a) into G.
   - If $l^i_j == X_m$, insert edge $<X_m, C_i>$ into E.
   - If $l^i_j == \overline{X_m}$, insert edge $<\overline{X_m}, C_i>$ into E.

3. For each node C_i, add edge $< C_i, C_{i+1} >$ ($C_{k+1} = C_1$) into E. We further configure node C_i to run RIPv2 and to advertise an aggregate route to C_{i+1}. The aggregate route is for a destination prefix P_a that is a strict supernet of P, $P \subset P_a$ (e.g., $P_a = 192.168.0.0/16$). We select the supernet such that there exists at least one unused IP subprefix P_u in P_a (e.g., $P_u = 192.168.2.0/24$). In addition, we assume that C_i has an offset-list of 10 for routes advertised to C_{i+1}, i.e., the offset value of 10 is added to the initial metric of routes advertised to C_{i+1}.

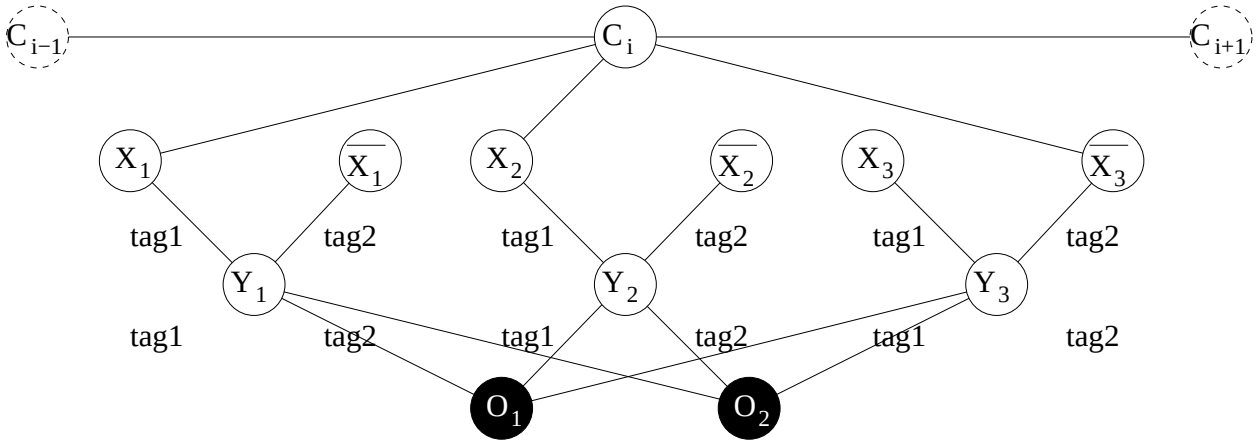

Figure 16: Construction of the sub-graph for clause $C_i = X_1 \vee X_2 \vee \overline{X_3}$.

Figure 16 represents the subgraph for the clause $C_i = X_1 \vee X_2 \vee \overline{X_3}$. The graph G can be computed from C in polynomial times. We now show that the transformation of C into G is a reduction, i.e., C is satisfiable if and only if G can result in a persistent forwarding loop.

$\Rightarrow$ We assume that C has a satisfying assignment. We show that G contains a persistent forwarding loop. Since C has a satisfying assignment, each clause C_i contains at least one literal l_j^i ($1 \leq j \leq 3$) with a TRUE assignment. By definition of the variable assignment and by construction of G, the node corresponding to that literal advertises a route to destination P to the node C_i. As such, C_i receives a contributing route to P_a and subsequently advertises the aggregate route to P_a to its neighbor C_{i+1}. Every node C_i advertises an aggregate route to P_a to C_{i+1} with a hop-count of 15. So, every node C_{i+1} learns a route to P_a pointing to C_i as its next-hop. The network converges and we obtain a persistent forwarding loop $C_k - C_{k-1} - \ldots - C_1 - C_k$. Any packet sent to P_u and that reaches one of the nodes C_i ($1 \leq i \leq k$) results in the persistent forwarding loop.

$\Leftarrow$ We assume that G contains a persistent forwarding loop. We want to demonstrate that C has a satisfiable assignment.

We show that every node C_i ($1 \leq i \leq k$) receives a route to P from at least l_1^i, l_2^i or l_3^i. We prove it by contradiction. We assume that there exists a node C_j ($j \in [1, k]$) such that C_j receives no route to P from l_1^j, l_2^j nor l_3^j. Consequently, C_j does not advertise any route to P_a to C_{j+1}. We note that C_j may receive a route to P_a from C_{j-1}. However, because the route to P_a received from C_{j-1} has a hop-count of 15, that route is not further propagated to C_{j+1}. As such, C_{j+1} has no route to P_a. We can then verify that packets sent to any destination prefix (e.g., P, P_a, or P_u) either reach the destination or are dropped by one of the routers because of the lack of route. We do not obtain any persistent forwarding loop, which contradicts the initial assumption.

To summarize, every node C_i ($1 \leq i \leq k$) receives a route to P from at least l_1^i, l_2^i or l_3^i. For each $i \in [1, k], j \in [1, 3]$, for the node l_j^i that advertises a route to P to C_i, we assign the TRUE value to the corresponding literal. We obtain an assignment that satisfies C since every clause C_i ($1 \leq i \leq k$) contains at least one literal with the TRUE value. ■

6 Mitigation of anomalies

Given the complexity of the LOOP-RA problem, we identify a sufficient condition to ensure convergence and loop-free forwarding paths. The key insight is that route aggregation can only be performed in a vectoring manner. We take advantage of this observation, and leverage the results of the previous Metarouting work [9] to ensure correct routing. When routers exchange routing information in a vectoring method, the Metarouting theory [9] establishes that *strict monotonicity* (SM) is a sufficient condition for convergence to loop-free paths. The SM property stipulates that when a router propagates a route, its preference should strictly decrease. Although only a sufficient condition – and not a necessary one – we first show that all described anomalies actually come from a violation of this property (Section 6.1). Then, from this condition, we derive a configuration guideline (Section 6.2) and a migration path to harden existing and future network designs against route oscillations and forwarding loops (Section 6.3).

6.1 Analysis of Root Causes

The presented anomalies can be attributed to two main causes. First, without a sink route, an aggregating router effectively advertises a route to some destinations that it does not have route to. For example, considering a router configured with an aggregate route (e.g., to 192.168.0.0/16), and having only one child route (e.g., to 192.168.1.0/24). The router has no route to many destinations in the aggregate address space (e.g., 192.168.2.1) yet still announces a route for them to its neighbor(s). The SM property is clearly violated for these destinations. This violation is at the origins of the blackholes and persistent forwarding loops presented in Section 1.2, Section 3.2 and Section 3.3. Current best practice suggests to create a sink route to address this problem. However, the use of sink route is not sufficient. As demonstrated in Sections 3.1 and 3.4, route aggregation can still result in routing anomalies even in the presence of sink routes. This leads to the second cause behind the observed problems.

A router may receive multiple routes to the most specific prefix (e.g., 192.168.10.0/16) from different routing processes (e.g., OSPF, RIP, EIGRP, static, sink). In such a case, the router relies on the administrative distance (AD) to determine the best route. Routes with lower AD values are more preferred. Again, we observe that with regards to the AD-based ordering, route aggregation violates SM. In particular, local aggregate routes may be assigned a larger AD value than that of the advertised routes. For example, by default, JUNOS assigns aggregate routes a default AD value of 130. The aggregate route may be advertised in RIP, which has a default AD value of 100 in JUNOS. Without customizing the AD values, the aggregate route is actually more preferred after it is advertised, clearly a violation of the SM property. This violation is responsible for the anomalies described in Section 3.1 and Section 3.4.

6.2 Configuration guidelines

With the insights above, it may seem straightforward to create a guideline to eliminate the problems. For example, one may propose the following strawman guideline.

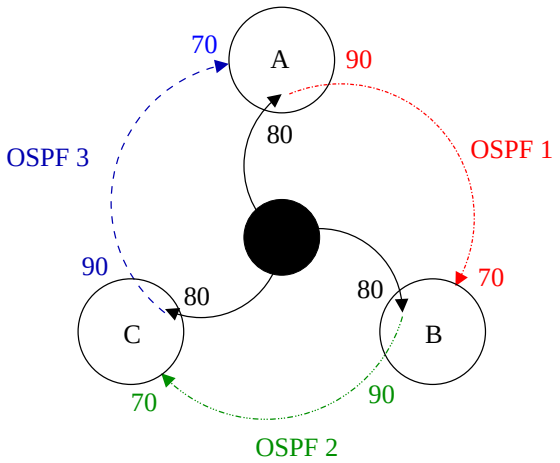

Figure 17: The strawman guideline is still vulnerable to anomalies (e.g., oscillations, loops.) The numbers indicate the AD values assigned to the different routing processes.

Strawman configuration guideline: *For each configured aggregate route, create a sink route, and ensure that the AD of the sink route is strictly lower than that of any aggregate route originated at the same router.*

To illustrate how the configuration guideline works, consider a router running two routing processes (OSPF and RIP) with AD values of 90 and 100, respectively. Suppose both routing processes are configured to announce an aggregate route with the same prefix. According the guideline, a sink route must be created and configured with an AD value strictly smaller than 90.

It turns out that the problems are more profound and the above strawman guideline is not sufficient. Configurations compliant with this guideline can still result in anomalies. To illustrate this point, let us consider the configuration of Figure 17. It consists of three routers A, B and C, each running different routing processes of OSPF (OSPF 1, OSPF 2, and OSPF 3). All three routers are configured to advertise a same aggregate route (e.g., 192.168.0.0/16), and are directly connected to a child route. The arrows represent routes. At router A, the sink route (solid black arrow) is configured with an AD value of 80, and the aggregate route is advertised into OSPF 1 (AD = 90). Router A is also running a routing process of OSPF 3 (AD = 70). We assume routers B abd C to have similar configurations. We note that configuration is compliant with the above guideline. However, when implementing the described scenario, we observed a persistent forwarding loop A-C-B-A for unused sub-prefixes in the aggregate address range. The anomaly is due to how the AD values are configured. As explained previously, when receiving multiple routes to the same destination prefix, routers select the route with the lowest AD value. However, this parameter is not propagated in route advertisements (e.g., RIP Reply, OSPF LSA) but is configured locally at each router. In the depicted scenario, the aggregate route has an initial AD value of 90 when originated from A, but is given an AD value of 70 when received at B. The preference of the advertised route has increased, violating the SM property.

To derive a true configuration guideline that ensures convergence to loop-free forwarding paths,

JUNOS

```
routing-options {
  aggregate {
    route 10.2.0.0/24 {
      preference 80;
    }
  }
}
```

Cisco OSPF

```
router ospf 100
   discard-route external 80 internal 80
   area 1 range 10.2.0.0 255.255.0.0
   summary-address 10.3.0.0 255.255.0.0
   redistribute static subnets
```

Cisco EIGRP

```
interface Ethernet 0/0
   ip address 192.168.1.1 255.255.255.0
   ip summary-address eigrp 1 10.2.0.0 255.255.0.0 80
```

Cisco BGP

```
router bgp 65000
   aggregate-address 10.2.0.0 255.255.0.0 summary-
   only as-set
   distance 90 100 80
```

Figure 18: Commands to harden existing configurations of route aggregation against oscillations and loops.

we borrow the results from previous works by Le *et al.* [13, 14]. First, we observe that sink routes constitute their own route type. Sink routes are compared with other routes (e.g., OSPF, RIP, static, etc.) based on the AD values. For network with multiple routing instances (e.g., sink, static, RIP, OSPF), Le *et al.* proposed the following guideline [14]:

Guideline 1: For a destination prefix P, all processes of a routing instance shall share the same AD value and every routing instance shall be assigned a globally unique AD value.

A routing instance is defined to be a collection of routing processes running a same routing protocol (e.g., OSPF), and exchanging routing information [14]. More precisely, two routing processes are said to belong to the same routing instance when they are hosted at different routers and establish adjacencies.

Furthermore, in the presence of sink routes, aggregate routes can actually be conceptually represented as the redistribution of the sink routes into another routing process. Le *et al.* identified the following guideline [12] to guarantee safe route redistribution:

Guideline 2: Only redistribute a route to P to a routing instance where the prefix is configured with a higher AD.

6.3 Migration Path

The above configuration guidelines together provide us with a migration path to protect existing and future network designs from permanent route oscillations and persistent forwarding loops that may result from route aggregation.

In particular, for every configured aggregate route, one needs to ensure that a sink route is also configured and assigned a strictly lower AD value than any aggregate route originated at the same router. In addition, all processes of each routing instance shall share one globally unique AD value for the destination prefix of this aggregate route.

JUNOS and Cisco IOS offer the ability for operators to customize the AD of the sink routes. The only exception is the Cisco RIP implementation. For this special case, we propose to migrate

from RIP to EIGRP. EIGRP supports all the features in RIP (e.g., off-set lists, distribute-lists, etc.) For the other routing protocols (e.g., OSPF, BGP, EIGRP), we describe the commands to add to the route aggregation configurations. For illustration purposes, we assume the configuration of a single aggregate route to 10.2.0.0/16. Figure 18 describes and shades the relevant commands to set the AD of the sink route to the value of 80[4].

6.4 Limitations

The identified configuration guidelines introduce new restrictions on the assignments of the AD values. The restrictions have little effect on most of the objectives described in Section 1 (e.g., reduction of routing table size and traffic engineering). In future work, we will attempt to further understand the implications of the restrictions on operational requirements, and possibly relax the guidelines.

In addition, we note that configurations compliant with the guidelines can still result in blackholes. We focused on oscillations and forwarding loops, as these anomalies are usually considered to be among the most severe ones: divergence leaves a network in a permanent state of instability while loops can cause congestions and can be exploited by malicious parties [19]. We leave configuration guidelines for preventing blackholes to future work.

7 Related Work

Although an essential primitive of routing designs, there are few studies on route aggregation. RFC 1339 [7] discusses aggregation as a means to curb the routing table size growth. It introduces the notion of, and emphasizes the need for, sink routes to prevent routing loops. However, we show in this paper that sink routes are not sufficient. In the presence of sink routes, route aggregation can still cause routing anomalies.

Few studies have looked at the impact of route aggregation on BGP: Geoff Huston [10] analyzed the growth of the BGP table size, and described several design objectives (e.g., load balancing, mutual backup for multihomed networks) why operators may simultaneously advertise aggregate and more specific routes. A more recent analysis [20] studied the impact of general and more specific BGP routes on IP reachability. The authors found some surprising and counterintuitive results. For example, a BGP withdrawal does not imply that a destination is no longer reachable. A more general route may still provide a path to the destination. It has also been recently documented [6] that the advertisements of general and more specific routes can cause violations of the traditional BGP policies (e.g., peer, customer, provider relationships). While these studies have revealed provoking results and all suggest that route aggregation deserves further attention from the networking community, they concentrate on BGP. Instead, route aggregation is also deployed in intra-domain settings and can cause instabilities to the global Internet routing system. Our analysis

[4]For Cisco OSPF, as explained in Section 2.1.4, this implementation supports two types of route aggregation: (1) intra-area, and (2) external routes. To illustrate and differentiate them, we assume that the router is configured to advertise an aggregate route with prefix 10.2.0.0/16 between areas, and another aggregate route with prefix 10.3.0.0/16 from external child routes.

considers the behaviour of route aggregation not only in BGP, but also in other routing protocols (e.g., OSPF, EIGRP, RIP).

In terms of routing anomalies, it has been reported that route aggregation can cause loops and blackholes [7, 18]. This feature is believed to be at the root causes of recently observed persistent forwarding loops [19]. Supernetting could also be responsible for reported Internet blackholes [11]. We show that the range of routing anomalies that can derive from supernetting is actually much larger.

8 Conclusion

We make several contributions in this paper. First, we present experimental results that show that the range of routing anomalies resulting from route aggregation is much larger than previously reported. We then present a canonical router model that is first to capture the behaviors of router aggregation and interactions between router aggregation and other functional blocks in a router. The model enables us to perform a number of analysis for a single router and a network of routers. We show that existing RA configuration guidelines are inadequate. In addition, we find that route aggregation is tightly intertwined with other essential functional components of a router (e.g., route selection, route redistribution), and these complex interactions can lead to unexpected outcomes. We further prove that determining whether the collection of RA configurations in a network can result in a persistent forwarding loop is NP-hard. Given this complexity, we identify a sufficient condition, for ensuring convergence and loop-free forwarding paths. Finally, from the condition, we derive a migration strategy to harden existing and future network designs against RA-induced route oscillations and forwarding loops.

Several questions remain open. In this paper, we have focused on oscillations and loops. Future study is needed to study how to prevent black holes. Also, while the proposed guidelines ensure safety, do they also introduce restrictions that may limit important network design objectives? If so, can we relax the guidelines? Finally, if we were to define route aggregation from a clean-slate, how should we design it?

9 Acknowledgement

The authors thank Robert Beverly for helpful discussions and suggestions.

References

[1] 100x100 Clean Slate Project. www.100x100network.org.

[2] 4D Project. www.cs.cmu.edu/∼4D.

[3] Internet 2 – Router Configurations. http://vn.grnoc.iu.edu/Internet2/configs/configs.html.

[4] Cisco. What is Administrative Distance?, March 2006.

[5] Cisco. Route Selection in Cisco Routers, 2008.

[6] Pierre Francois and Bruno Quoitin. Threat to BGP Policies : limited-scope more specific prefix injection, October 2009. Internet-Draft.

[7] V. Fuller, T. Li, J. Yu, and K. Varadhan. Supernetting: an Address Assignment and Aggregation Strategy, 1992. Request for Comments 1338.

[8] L. Gao and J. Rexford. Stable Internet Routing Without Global Coordination. In *Proc. ACM SIGMETRICS*, 2000.

[9] Timothy G. Griffin and Joao Luis Sobrinho. Metarouting. In *Proc. ACM SIGCOMM*, 2005.

[10] Geoff Huston. Analyzing the Internet's BGP Routing table, January 2001.

[11] Ethan Katz-Bassett, Harsha V. Madhyastha, John P. John, Arvind Krishnamurthy, David Wetherall, and Thomas Anderson. Studying Black Holes in the Internet with Hubble. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[12] Franck Le and Geoffrey Xie. On Guidelines for Safe Route Redistributions. In *Proc. ACM INM Workshop*, 2007.

[13] Franck Le, Geoffrey Xie, and Hui Zhang. Understanding Route Redistribution. In *Proc. IEEE ICNP*, 2007.

[14] Franck Le, Geoffrey Xie, and Hui Zhang. Instability Free Routing: Beyond One Protocol Instance. In *Proc. ACM CoNEXT*, 2008.

[15] Priscilla Oppenheimer. *Top-Down Network Design (2nd Edition)*. Cisco Press, 2004.

[16] Peter Rybaczyk. *Cisco Network Design Solutions for Small-Medium Businesses*. Cisco Press, 2004.

[17] Yu-Wei Eric Sung, Sanjay Rao, Geoffrey Xie, and David Maltz. Towards Systematic Design of Enterprise Networks. In *Proc. ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2008.

[18] Russ White, Don Slice, and Alvaro Retana. *Optimal Routing Design*. Cisco Press, 2005.

[19] Jianhong Xia, Lixin Gao, and Teng Fei. A measurement study of persistent forwarding loops on the Internet. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 2007.

[20] Yaping Zhu, Jennifer Rexford, Shubho Sen, and Aman Shaikh. Impact of prefix-match changes on IP reachability. In *Proc. Internet Measurement Conference (IMC)*, 2009.