Understanding Route Redistribution

Franck Le
Carnegie Mellon University
franckle@cmu.edu

Geoffrey G. Xie
Naval Postgraduate School
xie@nps.edu

Hui Zhang
Carnegie Mellon University
hzhang@cs.cmu.edu

***Abstract*—Route redistribution (RR) has become an integral part of IP network design as the result of a growing need for disseminating certain routes across routing protocol boundaries. While RR is widely used and resembles BGP in several nontrivial aspects, surprisingly, the safety of RR has not been systematically studied by the networking community. This paper presents the first analytical model for understanding the effect of RR on network wide routing dynamics and evaluating the safety of a specific RR configuration. We first illustrate how easily inaccurate configurations of RR may cause severe routing instabilities, including route oscillations and persistent routing loops. At the same time, general observations regarding the root causes of these instabilities are provided. We then introduce a formal model based on the general observations to represent and study the safety of route redistribution. Using the model, we prove that given a RR configuration, determining whether the redistributions result in a cycle is NP-hard. Given this complexity, we present a sufficient condition, which can be checked in polynomial time with the proposed analytical model, for ensuring the safety of a RR configuration. Finally, the paper proposes potential changes to the current RR protocol to guarantee safety.**

I. Introduction

Recent studies show that some enterprise networks rival carrier networks in terms of scale and complexity of routing design [1]. One may even argue that because of a more dynamic business environment fueled by acquisitions and mergers, large enterprise networks may be more difficult to control and manage than carrier networks. One source of this difficulty stems from the fact that the routing structure of a large enterprise network typically consists of multiple domains or routing instances [1]. Routing instances form for many reasons. Company acquisitions, departments administered by different teams, and multi-vendor equipments may lead to such situations [2]. Alternatively, network administrators may intentionally create separate routing instances to filter routes, limit reachability and enforce policies [3].

Routers within one routing instance typically run the same routing protocol to fully share reachability information and they by default do not exchange routing information with routers in other routing instances. Consider the network depicted in Figure 1. It consists of two routing instances. Routers in the RIP instance do not have visibility of the addresses and subnet prefixes in the OSPF instance and vice versa. To allow the exchange of routing information between different routing instances, router vendors have introduced a feature called *route redistribution*. Route redistribution (RR) is a configuration option local to a router. It designates the dissemination of routing information from one protocol process to another within the same router. For routers in the RIP instance to learn the prefixes in the OSPF instance, a router (e.g., D or E) needs to run *both* a process of RIP and a process of OSPF and inject the OSPF routes into the RIP instance.

As such, router vendors introduced RR to address a need from network operations. We recently looked at the configurations of some large university campus networks and found that RR is indeed widely used. However, contrary to traditional routing protocols, there is no standard or RFC formally defining the functionality of RR. Significant efforts are usually associated with the design and analysis of a routing protocol to ensure its correctness and stability but the specification of RR did not receive as much attention. Consequently, RR is often misconfigured leading to sub-optimal routing and even severe instabilities such as route oscillations and persistent routing loops.

The risks of route redistribution have been acknowledged by router vendors and network operators, but there is currently no general guideline to configure RR correctly. Solutions are developed on an ad-hoc basis for specific situations [3] but most existing solutions do not satisfy network design goals. RR has two main objectives. The first one is to propagate routing information between different routing instances for connectivity purposes as described earlier. The second objective is route back up: in the event of a network failure (e.g., link B-C of Figure 1 being down), routing instances should provide alternate forwarding paths to each other (e.g., router C should still be able to reach router A through the C-E-F-D-A path.) However, to avoid route oscillations and routing loops, according to current vendor recommendation [2], a route received from a routing instance must not be re-injected back into that same instance. Such a strong restriction prevents domain back-up. In addition, most of existing solutions apply to scenarios with only two routing instances, but large operational networks often include more than two routing instances [1].

In short, RR has become an integral part of IP network design but its stability seems sensitive to network failures and configuration errors. To the best of the authors' knowledge this paper is the first to analyze route redistribution and attempt to identify the origins of the observed instabilities. Our work is based on two key insights. First, we observe that the way RR effects the flow of routes is very similar to that of a distance vector routing protocol, albeit with a larger scope

[1]Minor revisions, February 9, 2008.
In Proceedings of ICNP'07, Beijing, China.

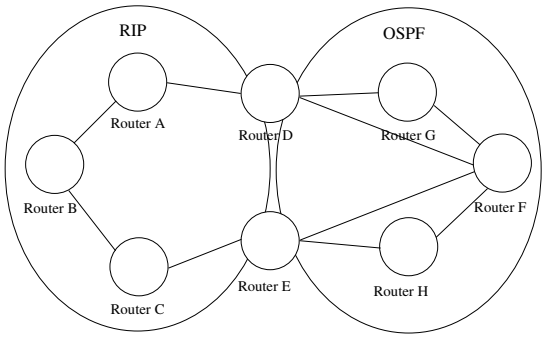

Fig. 1. An example enterprise network that consists of two routing instances. Without route redistribution, the routers in the RIP instance do not have knowledge of the destinations in the OSPF instance, and vice-versa.

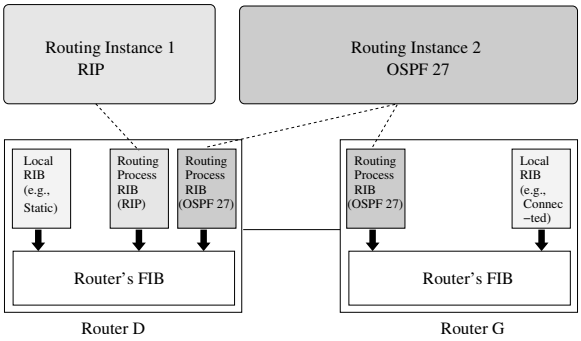

Fig. 2. Router's route selection procedure. Router D may need to perform route selection among three routing processes while G may need to select between two. D and G belong to the same OSPF routing instance.

since the routes are passed between routing instances. Second, the policy-based functionality of RR makes it resemble BGP in several nontrivial ways.

The rest of the paper is organized as follows. Section II provides a tutorial of the concepts related to the RR procedure. Section III illustrates some of the instabilities that route redistribution can cause. Section IV introduces a model to abstract the dynamics of the RR procedure. The model helps to identify the formation of route oscillations and persistent routing loops as will be demonstrated in Section V. In Section VI, we show that determining whether a configuration of route redistribution can converge to a state containing a cycle is NP-hard. This complexity has motivated us to develop a sufficient condition for RR safety. The result is presented in Section VI. Finally in Section VII, we propose a set of changes to the RR procedure itself aimed at creating a protocol that is *inherently safe*, i.e., free of instabilities even if misconfigured.

II. Background

Although a router may run multiple routing protocols at the same time, it forwards packets according to routes stored in a protocol-agnostic table called Forwarding Information Base (FIB). Routing instabilities such as oscillations and loops are always the consequence of some routers installing some wrong routes in their FIBs. As background information for the rest of the paper, this section briefly explains how a router populates its FIB and describes the role of route redistribution in that procedure. Without loss of generality, all discussions are with respect to a single destination prefix, denoted by P, unless noted otherwise.

A. Route Selection

A router that runs multiple routing protocols actually instantiates a separate *routing process* for each protocol. Each instantiated routing process has its own Routing Information Base (RIB) to store protocol-specific routing information. These processes may offer routes to P at the same time and the router then must select one of the routes to include in the FIB. To add flexibility to the selection procedure, router vendors have introduced a configurable integer parameter per-routing process, called *administrative distance* [4], to facilitate a logic for ranking a set of routing processes that supply routes to a same destination prefix. Upon creation, each routing process is assigned a protocol-specific default administrative distance. For example, the default administrative distance (AD) is 110 for OSPF and 120 for RIP. The administrator may override the default AD of a protocol, according to local policy and on a per-router per-prefix basis, as part of the protocol configuration. The administrative distance values are local to a router and are not propagated in any signaling message.

Among the routing processes announcing a route to P, the one with the lowest administrative distance will be selected. We will refer to it as the *selected routing process* for P and the route subsequently installed in the FIB the *active* route for P. If multiple processes present the same lowest administrative distance, the router picks one of them using a nondeterministic and vendor-specific algorithm. Fig. 2 illustrates the administrative distance based route selection procedure on two routers (D and G in the enterprise network shown in Fig. 1). Router D has three concurrent routing processes: OSPF, RIP, and the built-in process for handling local static routes or direct connected subnets. When the OSPF and RIP processes present independent routes to P at the same time, the router will select the route from the OSPF processs because of its lower AD value (110 vs. 120).

B. Route Redistribution

A router running multiple routing processes does not by default redistribute routes among these processes. Route redistribution (RR) must be explicitly configured. The RR configuration and operation can be complex. Contrary to most routing protocols which optimize a metric, RR is driven by policies, making it very similar to BGP. As in BGP, access control lists can be applied and tags can be assigned to the different prefixes. In Cisco, *route-map* allows network administrators to filter the routes, prioritize the received announcements (by assigning different AD values) and modify the attributes of the redistributed routes. Figure 3(a) provides an example of a Cisco configuration redistributing routes from a RIP process into an OSPF one. The *route-map* statements filter the route and modify the attributes of the redistributed routes. In Juniper routers, RR is specified through routing policies (i.e., import and export statements). Figure 3(b), extracted from [5], provides an example of a JUNOS configuration redistributing

```
router ospf 27
 redistribute rip metric 200 subnets
    route−map rip−2−ospf
 distance ospf external 200
!
route−map rip−2−ospf permit 10
 match ip address 100
 set metric−type type−1
 set tag 22
```

(a)

```
policy−statement rip−2−ospf {
      from protocol rip;
      then {
          metric 5;
          accept;
      }
   }
}

ospf {
      export rip−2−ospf;
      }
```

(b)

Fig. 3. (a) Excerpt of a Cisco RR configuration for router D of the example enterprise network. The "redistribute" command enables RR from the RIP process into the OSPF process. A route-map is applied to enforce policies. (b) Excerpt of a Juniper RR configuration for router E of the same network.

the routes from a RIP process into an OSPF one.

When configuring a route redistribution, a number of per-route parameters are involved. A metric value is assigned to each new route in the target routing process (i.e., the process that receives the route). The metric can be manually configured. If not specified, a default value is assigned to the redistributed route. The value of the metric is important since the target routing process may disseminate a redistributed route to other routers within the same routing instance and when that happens the value of the metric may influence the router selection outcomes at these routers. Additional route attributes can be specified depending on the target routing process. As an example, in OSPF, a route can be injected either as a Type 1 or Type 2 route. Each route type presents its own properties [6]. All these attributes affect the route selection (within a routing instance) and when misconfigured can cause instabilities as illustrated in the subsequent sections.

While RR is a complex, policy-driven vector protocol like BGP, there is no RFC or other form of standards about it. Vendor Web sites or manuals usually only provide guidelines and sample configuration files. It took us many hours of reading and experimentation to identify the following two very basic operational characteristics of RR. First, *a route is redistributed only if it is active*, i.e., present in the router's FIB. Consequently, RR is not a transitive operation. For example, let's consider a router running three routing processes X, Y and Z. Suppose that redistributions from X to Y and from Y to Z are configured. If initially no route to P exists in the FIB and the router learns a route to P through X, the route is redistributed into Y. However, the route will not be further redistributed into Z from Y because Y is not the selected process for P. Second, a router seems able to differentiate locally redistributed routes from routes that routing processes learn from their protocol operations. Routes redistributed from the local FIB apparently will not be considered by the route selection procedure. Continuing from the previous example, suppose Y has a lower AD than X. The router will not switch to designate Y as the selected process for P after the route redistribution from X to Y. In other words, *RR does not impact local route selection*. However, since it is possible (and quite often) that a routing process disseminates redistributed routes to other routers in the same routing instance, RR does have a profound impact on how the network as a whole populates the FIBs.

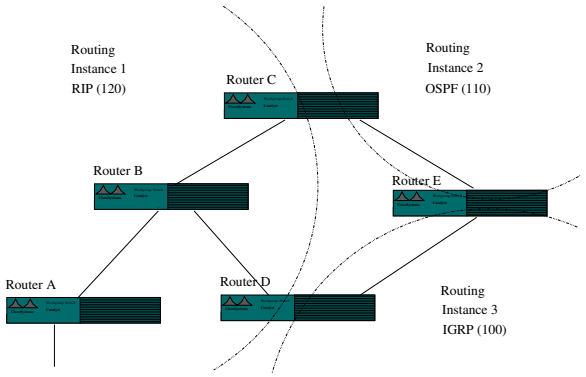

Fig. 4. Scenario 1. The network consists of three routing instances and routers C, E and D are performing mutual route redistribution. The values in brackets represent the AD of the routing instances.

III. Illustration of Routing Instabilities

Inaccurate RR configurations can cause sub-optimal routing, route oscillations and routing loops. This section illustrates the two latter issues. For the other instabilities, please see [2] and [3]. We have validated all the described scenarios (including the permanent oscillations in section V) – with the exception of scenario 2 (section III-B). It was difficult to create the required race conditions. The validation environment consisted of 5 routers (Cisco 2600, IOS Version 12.2).

A. Scenario 1: Persistent Routing Loop

The first scenario is a network composed of three routing processes (see Figure 4). The experiment revealed a persistent routing loop because of route redistribution.

The example network can be that of a company with three office branches, each running their own routing instance. A separate group may administer each site. RR is enabled between every two routing instances to allow connectivity between the sites, and every redistributing router performs mutual RR so that the routing instances can backup each other in the event of network failure.

We consider a prefix P connected to router A and redistributed into the RIP instance. Alternatively, P can be originated from another RIP router or injected into RIP through other means including BGP. The following describes the sequence of observed events that caused a persistent routing loop for packets sent to destinations in P.

We use the following notation throughout the paper: routing instances are numbered (1, 2, ...), routers are labeled (A, B, ...), and routing processes are denoted by <router>.<routing-instance>, e.g., $D.1$ designates the RIP process in router D.

- t_1 Initially, only A's FIB has an entry to P. After starting the RIP process on A, the connected route is redistributed into RIP (i.e., $A.1$).
- t_2 D's FIB includes an entry to P. The route is learnt from RIP and points to B.
- t_3 E's FIB contains a route to P with D as the next-hop: This is because D redistributes the route from

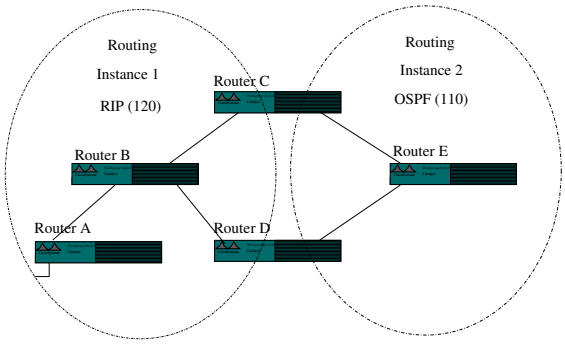

Fig. 5. Routers C and D are performing mutual route redistribution. The redistribution can result in persistent routing loops (C-B-D-E-C) for prefixes originated from the RIP domain or injected into the RIP domain (e.g., through router A)

RIP into IGRP (i.e., $D.1$ into $D.3$).

t_4 C's FIB presents a route to P. The installed route is from OSPF (i.e., $C.2$) and E is the next-hop: C receives two routes from RIP (from B) and OSPF (from E). Since OSPF presents a lower AD (110 vs. 120), the latter route is selected.

t_5 As B receives the route from A and the redistributed one from C, we observe 2 cases.

- If the route from C presents a lower metric, C becomes the next-hop, causing a routing loop (B-C-E-D-B).
- If the route from A presents a lower metric, A remains the next-hop resulting in a sub-optimal path (D, E, C, B, A) from D to A.

When repeating the experiment, the loop sometimes formed in the opposite direction (B-D-E-C-B) because of a different order of routing message arrivals.

This configuration consisted of three routing instances. However, networks with only two routing instances can be vulnerable as well. The configuration in Figure 5 shows an example. The network may be migrating from RIP to OSPF. Mutual redistribution is performed at two redistributing routers to exchange routing information and to allow domain back up. When C first redistributed the route from $C.1$ into $C.2$, D received two choices and preferred $D.2$ because of the lower AD. Then, D redistributed $D.2$ into $D.1$ and B switched to use D as its next-hop when the redistributed route had a lower metric value, resulting in a routing loop C-B-D-E-C.

B. Scenario 2: Route Oscillation

The configuration in Figure 5 can also experience severe route oscillations. [3] describes a similar scenario and explains that the following hypothetical sequence of events can cause route flapping. If C and D simultaneously redistribute P into routing process 2 and then update their FIB at the "same" time, the route flaps between the 2 routing instances:

t_1 C (respectively, D) learns a route from $C.1$ (respectively, $D.1$). C (respectively, D) redistributes P from $C.1$ into $C.2$ (respectively, from $D.1$ into $D.2$.)

t_2 D (respectively, C) receives two routes from 1 and 2. As the AD of 2 is lower, D (respectively, C) updates its FIB, preferring the route from 2. Consequently, D (respectively, C) stops announcing the route into 2 but redistributes the route from 2 to 1.

t_3 Because both routers cease to advertise the route into 2, this routing instance no longer has a route to the destination.

t_4 C and D only have one way to reach the destination (i.e., through $C.1$ and $D.1$). Therefore, they each update their FIB and redistribute the route from 1 into 2.

Noting that the states at (t_1) and (t_4) are identical, the route oscillates between the two routing instances.

One may argue that such oscillations, even though possible, are not likely to occur in practice. For these oscillations to happen, the signaling messages from the redistributing routers must be processed in a specific sequence order. Routers' load, link delay and other factors will most probably disrupt this synchronization putting an end to the oscillation. We later show that RR can actually cause permanent route oscillations, independent of any race conditions.

IV. A Model for Route Redistribution

To analyze RR, we introduce a model consisting of three distinct components – 1) Per-Router Route Selection and RR Logics, 2) Route Propagation Graph, and 3) Network-wide RR Logic. The first component abstracts the router behavior. We introduce two logics corresponding to the local route selection and RR procedures. These logics enable the derivation of the actual active routes and of the routes that are redistributed at each redistributing router. While these logics provide a local view of the operations at each router, we also need a global view of the interactions between the redistributing routers in order to determine the paths taken by the data traffic from each routing instance. To analyze the exchange of routing information between the redistributing routers, we propose a second component of the model – the route propagation graph. This graph provides a global view of the routing instances present in a network and their RR configuration. Finally, an integration of the two previous components produces the third component – a network-wide RR logic. This last component models the dynamic exchange of routing information between the routers and consequently allow us to predict the paths taken by the data packets. Understanding these data flows can reveal the formation of oscillations, routing loops and other instabilities.

The model applies to existing routing protocols with the exception of BGP because both the route selection and route redistribution logics of this protocol differ from those of other routing protocols. In terms of route selection logic, BGP processes first look at the BGP *AS_PATH* parameter to decide whether to consider a route whereas other routing protocols consider all incoming routes. BGP's route redistribution logic also differs from that of other routing protocols: each vendor has defined its own set of proprietary options for redistributing BGP from and into existing IGPs [7], [8], [9].

The proposed model could be extended to include BGP, but

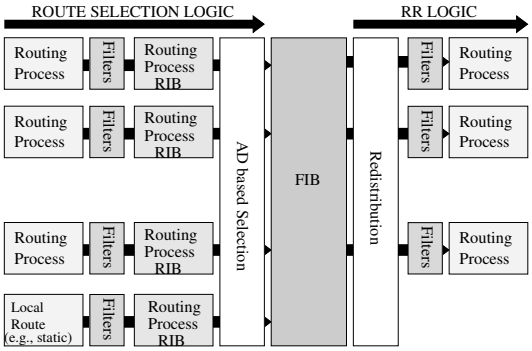

Fig. 6. Per-router route selection and route redistribution logics. A router first selects the preferred routing process based on the AD values. Then, it installs the best route in the FIB, and redistributes the active route according to the configured policies.

we decide to set aside this protocol to keep the model simple.

A. The Per-Router Route Selection and RR Logics

A router's operations related to RR can be decomposed into two main steps (see Figure 6). First, the router selects the best route according to the route selection logic[1], and installs it in the FIB. A routing process possesses a route to P in its RIB either because it has originated the prefix or because it has received a route advertisement from a neighbor router and the route has passed the import filter(s) configured for this routing process. Among the routing processes offering a route to P, the router chooses the one with the lowest AD. And, within this selected routing process, the router picks the *best* route. The installed route is also called the *active* route. Then, the router redistributes the active route according to the RR logic.

The following route selection logic is triggered either when a new route to P is installed in the RIB of one of the routing processes or when the previously active route in the FIB is no longer available (e.g., the router announcing the initial route may have withdrawn it after a network failure).

Considering a router r, let RP be the set of routing processes running on r. Each routing process $x \in RP$ has a RIB, $x.RIB$, and an AD $x.ad$. For a destination P, let *selected-process(P)* be the selected routing process for P and *active-route(P)* the installed route used for forwarding purposes at r. Initially, we set *selected-process(P) = NULL* and *active-route(P) = NULL*.

Procedure *Route selection at router* r

1: **for all** routing process $x \in RP$ that receives a route to P **do**
2: Apply local filters to the route
3: **end for**
4: **for all** routing process $x \in RP$ such that $P \in x.RIB$ **do**
5: **if** (*selected-process(P) = NULL*) OR ($x.ad <$ *selected-process(P).ad*) OR ($x.ad ==$ *selected-process(P).ad* AND rand(0, 1) == 1) **then**
6: *selected-process(P)* $\leftarrow x$
7: *selected-process(P).ad* $\leftarrow x.ad$
8: Select best route from *selected-process(P)* (according to metric) and install it in FIB
9: *active-route(P)* $\leftarrow$ selected best route
10: **end if**
11: **end for**

Then, the router redistributes the active route according to the RR logic.

Procedure *RR at router* r

1: **for all** routing process $y \in RP$ such that redistribution of P from *selected-process(P)* to y is enabled **do**
2: **if** permitted by RR filter between the two processes **then**
3: redistribute $active(P)$ into y, and
4: set route attributes according to the RR filter
5: **end if**
6: **end for**

It is important to note that a redistributed route is marked with the local FIB as the source of the route or stored outside the target process's regular RIB, and as such it will not be offered back to the same FIB. For the same reason, a redistributed route will be removed if the original active route is no longer present in the FIB.

B. The Route Propagation Graph

To analyze RR, we extend the routing instance graph, introduced in [1] to represent the network-wide static RR configuration among routing instances.

We define a weighted directed graph $G = <V, R, O, E, D>$ called the route propagation graph to model the complete route redistribution configuration with respect to a prefix P. Each routing instance is represented by a vertex $u \in V$, and associated with its default AD (see Figure 7). Static routes and directly connected subnets are often redistributed into routing protocols. We use two special vertices to represent the origins of these two types of routes. The set $O \subseteq V$ is the set of routing instances that originate the prefix P. Each vertex $o \in O$ is represented as a striped vertex. A destination may be originated by several routing instances. The set R designates the routers that perform route redistribution among routing instances of V. For each route redistribution from routing instance u to routing instance y performed by $r \in R$, we represent such redistribution by a dashed edge from u to v. The edge is marked with r to indicate the router that is redistributing the route between the two routing instances. We note such edge $<u, v, r>$. As described in section II, the default AD of a routing process can be overridden by a *customized* value at each router. In such case, we add the customized AD of the routing process(es) at router r to the edge. A redistribution from u to v by router r, with customized AD values for both processes $r.u$ (d_1) and $r.v$ (d_2), is represented by an edge from u to v labeled "d_1, r, d_2" (see Figure 9). For RR using the default AD values, the d values are omitted from the edge.

[1]The route selection procedure may be vendor specific. Some implementations may maintain the existing selected routing process when the AD of the routing process advertising the new route is not strictly smaller than the currently selected one.

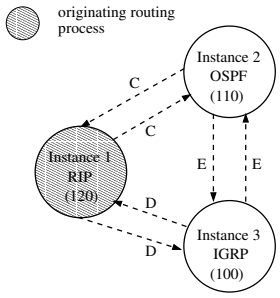

Fig. 7. Illustration of the route propagation graph of the network in scenario 1. The values in brackets indicate the default AD of the routing instances.

In addition, for every router $r \in R$, we need to represent all the routing processes running on r, including the ones that are not part of the RR configuration. The reason is that these routing processes affect the route selection and RR logics. For example, consider a router r running three routing processes x, y, and z. Even though the redistribution from x into y is enabled, x may not redistribute any route into y when z is the selected routing process at r. A routing process of r which participates in routing instance w but is not part of the RR configuration is represented by an edge from w to w marked by r (Figure 10).

It is important to note that each vertex is a routing instance, but a router deals with routing processes, each of which is a member of a routing instance. When a route is redistributed into a routing instance, every routing process of that instance should eventually receive an advertisement of that route.

The RR graph of the network in Figure 4 is illustrated in Figure 7. The dashed edge from RIP to OSPF, labeled C, indicates that router C is configured to redistribute P from the RIP process into the OSPF process.

C. The Network-wide RR Logic

The two previous components of the model are not sufficient to derive the data paths. The first component, per-router route selection and RR logics, focuses only on a single router. However, the received routing messages at each router depend on that router's interaction with other routers. Looking at a single router at a time does not allow us to derive the actual inputs and outputs of the router. The second component, the route propagation graph, only represents the topology and the configuration of the network. In both components, the network-wide dynamics are missing. To model this crucial information, we introduce a third component, the Network-wide RR Logic.

To analyze the exchange of routing information among the routing instances and derive the active routes, we adopt the activation sequence concept suggested in [10]. At each time step, a set of redistributing routers in R are activated. When activated, a node performs the route selection and RR procedures. Because the set of routers that are activated at each time step is arbitrary, we may obtain a different outcome for each run.

The following describes a logic to derive the propagation of signaling messages and identify the actual redistributions in the RR graph for a specific activation sequence. The network-wide RR logic uses a variable called candidate-list, CL, for storing the list of all routers from R to be activated, and another variable S for tracking the subset of routers activated at each time step. We represent actual route redistribution events by solid edges. A redistribution edge changes from dashed to solid, i.e., *is activated*, if (1) the route to be redistributed is active and (2) the policy configured for that edge permits this route to be redistributed. Also, we represent the data paths by thick solid arrows. Finally, a routing instance colored in white indicates that it does not have a route to P, whereas a routing instance colored in dark means that the routing instance does.

Procedure *Network-wide RR*

Initialization

1: $t = 0$
2: Insert the redistributing routers that have an edge either from or to an originating vertex into CL.

Main loop

1: **while** $CL \neq$ *EMPTY SET* **do**
2: *t++*
3: Remove a subset S of routers from CL
4: **for all** router $r \in S$ **concurrently do**
5: Execute *route-selection logic* at r
6: **if** r's selected routing process has changed **then**
7: //let $r.u$ denotes the new selected routing process
8: De-activate all edges labeled r
9: **if** there is no configured redistribution from nor to $r.u$ **then**
10: Activate edge <u,u,r> //to mark the selected routing process at r
11: **else**
12: Execute *route-redistribution logic* at r
13: **for all** routing processes $r.v$ running on r **do**
14: **if** r is redistributing a route to P from $r.u$ into $r.v$ **then**
15: Activate edge <u,v,r>
16: **end if**
17: **end for**
18: **for all** vertex v to which <u,v,r> is activated **do**
19: insert routers that have an edge from or to v into CL (if not already present)
20: **end for**
21: **end if**
22: **end if**
23: **end for**
24: **for all** vertice $u \in V \setminus O$ **do**
25: **if** there is no active edge from a routing instance other than u pointing into u **then**
26: Color u in white
27: Insert routers that have an active edge from u into CL (if not already present)
28: **else if** there is an active edge from a routing instance

other than u pointing into u **then**
29: Color u in dark
30: **end if**
31: **end for**
32: **end while**

We refer to the sequence of RR graphs in an activation sequence as *routing states*. We say that a routing state is stable when all the redistributing routers have selected their best option among the available choices and there is no signaling message in transit in the network (i.e., CL and S are empty).

V. Disclosure of Instabilities Using RR Model

This section and the next one (section VI) present two applications of the RR model. This section studies the network wide routing dynamics: section V-A explains how the developed RR model allows us to analyze the propagation of routing information and to detect potential instabilities in a network. Based on the derived insight, section V-B reveals new instabilities that RR can cause.

A. Detection of Instabilities

The routing states can expose instabilities that can result from RR. Considering a routing state, we define a cycle as a set of k distinct routers $r_1, r_2, ..., r_k \in R$ such that $<u_1, u_2, r_1>$, $<u_2, u_3, r_2>$, ..., $<u_k, u_{k+1}, r_k> \in E$, $<u_1, u_2, r_1>$, $<u_2, u_3, r_2>$, ..., $<u_k, u_{k+1}, r_k>$ are active at time *t*, and $u_1 = u_{k+1}$. A cycle is *permanent* if there exists a time t_c, such that for all $t > t_c$, the cycle is present in G.

A permanent cycle in a RR graph does not imply the existence of a persistent routing loop. As described in section III, depending on the internal metrics within the routing instances, either a persistent routing loop may form, or a sub-optimal routing path may arise. Both consequences are unwanted and consequently, permanent cycles are undesirable.

The RR model allows us to analyze the propagation of routing information for different activation sequences and consequently, detect the potential formation of cycle. Figure 8 illustrates an activation sequence which converges to a permanent cycle in the network of Figure 4.

t=0 A route P is originated by the RIP routing instance (1). Routers C and D are connected to 1. Consequently, $CL(t=0) = \{C, D\}$.

t=1 We assume $S(t=1) = \{C, D\} \subseteq CL(t=0)$. We run the route selection and RR logics at routers C and D. C learns a route to P through $C.1$ and redistributes it from $C.1$ into $C.2$. The corresponding edge (from $C.1$ to $C.2$) becomes solid to indicate that the redistribution is active. Router E which is connected to 2 is inserted into CL. Similarly, D learns a route to P and redistributes it from $D.1$ to $D.3$. The edge from $D.1$ to $D.3$ also becomes solid. We have $CL(t=1) = \{E\}$.

t=2 We assume $S(t=2) = \{E\}$. We run the route selection and RR logics at router E. Router E is receiving a route to P through $E.2$ and $E.3$. Because $E.3$ presents a lower AD (100), E selects $E.3$ as its selected routing process and redistributes the route from $E.3$ into $E.2$. The edge from $E.3$ to $E.2$ becomes solid to indicate that the corresponding redistribution is active. Since C is connected to 2, C is added to $CL(t=2)$. $CL(t=2) = \{C\}$.

t=3 We assume $S(t=3) = \{C\}$. Router C has two choices to reach P (through $C.1$ and $C.2$). Because $C.2$'s AD is lower than the one from $C.1$, $C.2$ becomes the selected routing process at C and C redistributes from $C.2$ into $C.1$. The edge from $C.1$ to $C.2$ is de-activated and instead the one from $C.2$ to $C.1$ is active. We have $CL(t=3) = \{D\}$.

t=4 We assume $S(t=4) = \{D\}$. D has only one choice to the destination (i.e., through $D.1$). No other router than D announces a route to P into 3. Therefore, D maintains its selected routing process ($D.1$), and $CL(t=4) = EMPTY\ SET$.

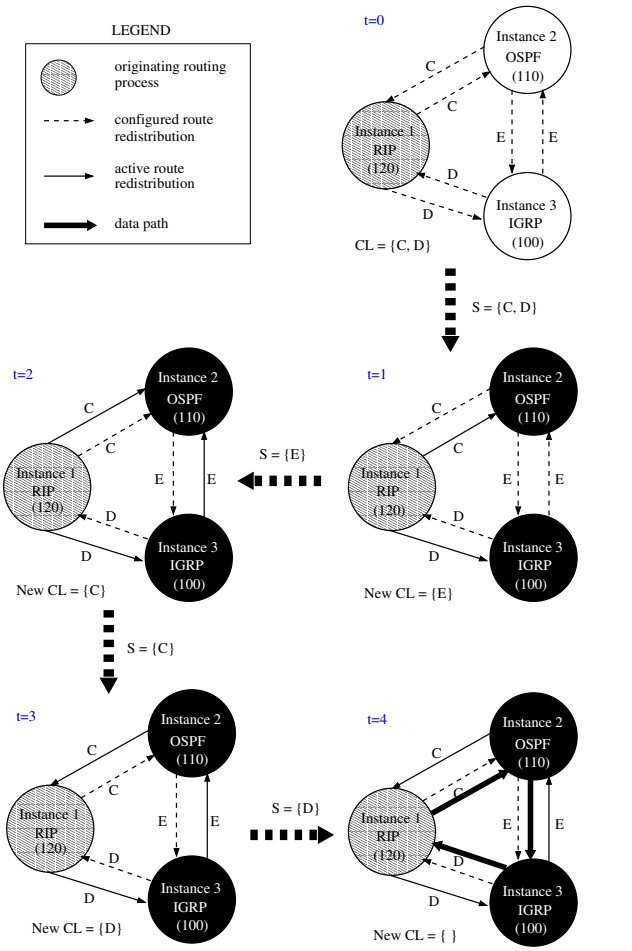

Fig. 8. Illustration of an activation sequence and the corresponding routing states for the network in scenario 1.

The paths (thick arrows) taken by the data packets to reach P are represented in the final graph. They form a cycle that reveals the potential formation of a routing loop.

Some configurations of RR always converge to a state containing a cycle (independently of the activation sequence) (e.g., Figure 4). Other configurations always converge to a

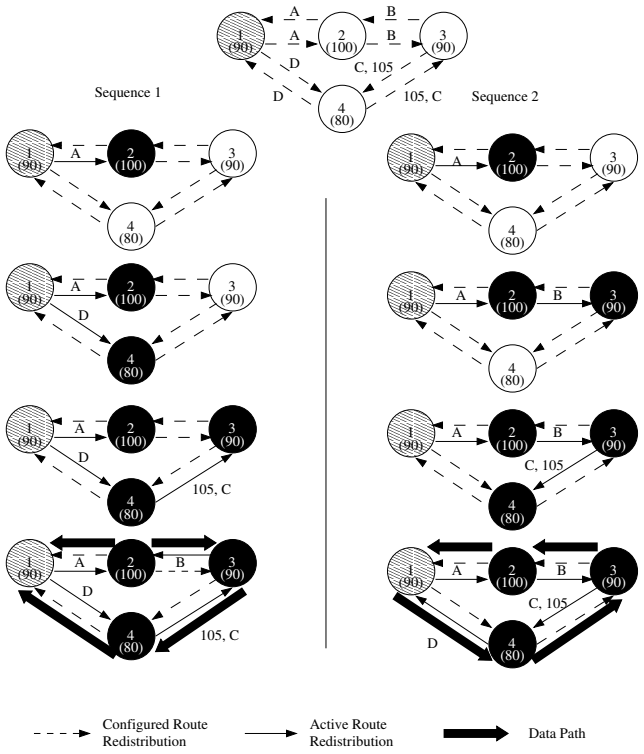

Fig. 9. Scenario 3. Detection of persistent routing loops. Sequence 1 does not result in any routing loop but sequence 2 discloses a potential routing loop. The values on the edges represent the customized AD: router C has a customized AD (105) value for routing process 4, overriding the default value (80).

cycle-free state. Some configurations can converge to a state which includes a cycle depending on the activation sequence (for example, scenario 3 of Figure 9). Finally, some configurations may converge after some arbitrary time (section III-B, Figure 5) and others may always diverge (section V-B).

We say that a RR configuration is *safe* if for all activation sequences, the execution of the network-wide RR logic converges to a cycle free state.

B. Disclosure of New Instabilities

Vendor documentation [2], [3] mentioned the potential formation of routing loop, sub-optimal routing and route flapping in certain race conditions as illustrated in section III. Our model shows new instabilities that can result from RR. Permanent route oscillations, independent from any race conditions, can indeed appear due to improper RR configuration.

To demonstrate the formation of permanent route oscillations, we consider the topology in scenario 4 (Figure 10). The corresponding RR graph is depicted at the top of the figure (t=0). A prefix is originated by routing instance 1.

- t=1 C learns a route to the prefix through $C.1$ and redistributes the route from $C.1$ into $C.2$.
- t=2 A learns a route through $A.2$ and redistributes the route from $A.2$ into $A.3$.
- t=3 B receives the route through $B.3$, installs it in its FIB and redistributes the route from $B.3$ into $B.4$.
- t=4 A receives two routes to the destination: from $A.2$ and from $A.4$. Because $A.4$ has a lower AD, $A.4$ becomes the selected routing process. Consequently, A stops redistributing from $A.2$ into $A.3$.
- t=5 Because A stopped redistributing from $A.2$ into $A.3$, B no longer receives any announcement. B removes the route and stops announcing it into $B.4$.
- t=6 Consequently, $A.4$ no longer receives a route to the destination either. Instead, A learns the route from $A.2$ and redistributes it into $A.3$. We note that this state is identical to the one at t=2. A permanent route oscillation has formed. This instability has been observed in the conducted experiments.

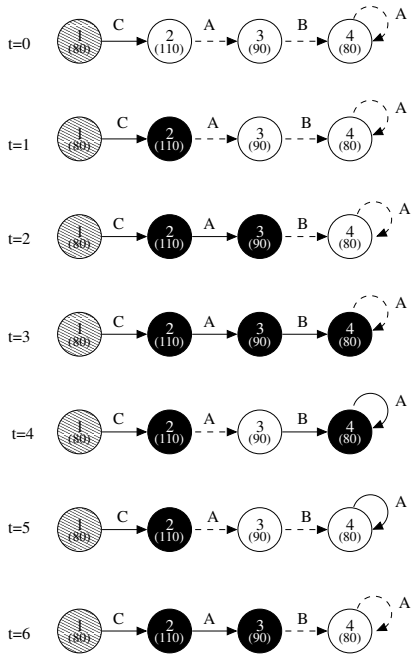

Fig. 10. Scenario 4. Illustration of permanent route flap. Router A runs 3 routing processes but no redistribution is enabled neither from nor to $A.4$. The routing states highlight the formation of permanent oscillations since the states at t=2 and t=6 are identical.

These permanent oscillations come from the fact that the redistribution of the routes into the routing processes depends on the routes present in the FIB, which in turn depend on the routes present in the routing processes. Section II explained that routes redistributed from a router does not directly impact the local route selection. However, since the redistributed routes may get further redistributed, those routes can come back and ultimately affect the route selection outcomes.

VI. Towards Instability-Free RR Configuration

In this section, we investigate how to determine if a given RR configuration can result in instabilities. First, we demonstrate that determining whether a configuration can converge to a state containing a cycle is a NP-hard problem. The complexity of the task motivates the search for sufficient conditions. In section VI-B, we introduce a sufficient condition guaranteeing that a RR configuration does not result in any persistent routing loop or oscillations.

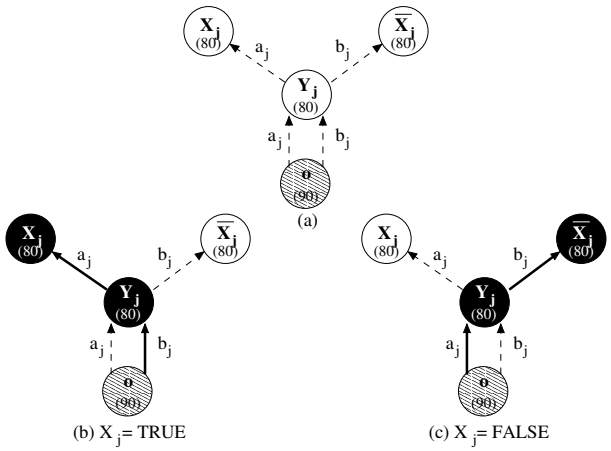

Fig. 11. Conversion of variable into subgraph. Each variable X_j is represented by the topology in figure (a). Depending on the activation sequence, we can obtain two different stable routing states depicted by figures (b) and (c). We associate the redistribution outcome of figure (b) with TRUE assigned to X_j and that of figure (c) with FALSE assigned to X_j.

A. Complexity of Route Redistribution Analysis

We show that given a RR configuration, determining whether the redistributions converge to a state containing a cycle is NP-hard.

Problem: considering a route propagation graph <*V, R, O, E, D*>, we ask whether there exists an activation sequence such that the redistributions converge to a state which includes a cycle of active redistributions. We will call this problem, the Route Redistribution Configuration - Cycle Detection (RRC-CD) problem.

Theorem 6.1: the RRC-CD problem is NP-hard.

Proof: The proof is inspired from [11].

To show that RRC-CD is NP-hard, we rely on the fact that the 3-CNF SAT problem is a NP-complete problem [12]. We more specifically prove that RRC-CD $\geq_p$ 3-CNF SAT. We consider an instance of 3-CNF SAT, i.e., a set of *m* clauses of length at most 3 over *n* Boolean variables: $B = C_1 \wedge C_2 \wedge ... \wedge C_m$ with each clause $C_k, (1 \leq k \leq m)$, composed of at most three distinct literals: $C_k = l_1^k \vee l_2^k \vee l_3^k$. Each $l_i^k (1 \leq i \leq 3)$ is of simple form of a single variable: X_j or $\overline{X_j}$ $(1 \leq j \leq n)$. We construct a route propagation graph G such that B is satisfiable if and only if there exists an activation sequence such that G converges to a state including a cycle of active redistributions. We construct G as follows:

1) O contains an originating vertex o.
2) For each clause C_k, insert new node labeled C_k into V
   For each literal l_i^k $(1 \leq i \leq 3)$ of the form X_j or $\overline{X_j}$ $(1 \leq j \leq n)$.
   If $X_j \notin V$, add the subgraph Figure 11a into G,
   If $l_i^k == X_j$, insert edge from X_j to C_k labeled g_j
   If $l_i^k == \overline{X_j}$, insert edge from $\overline{X_j}$ to C_k labelled $\overline{g_j}$
3) For each $C_k (1 \leq k \leq m)$, add the nodes U_k, V_k, Z_k, W_k and the associated edges as illustrated in Figure 12.
4) For each $C_k (1 \leq k \leq m)$, add edge from U_k to U_{k+1} (U_{m+1}=U_1) with label "d_k, e_k" where the customized AD of d_k(= 90) forces e_k to prefer Z_k over U_k over U_{k+1}.

Such graph can be computed from B in polynomial times. Figure 13 provides an illustration of the graph for $B = (X_1 \vee X_2 \vee X_3) \wedge (X_1 \vee \overline{X_2} \vee \overline{X_3}) \wedge (\overline{X_1} \vee X_2 \vee \overline{X_3})$. We now show that the transformation of B into G is a reduction

$\Rightarrow$ We show that if B has a satisfying assignment, there is an activation sequence such that G converges and the final state includes a cycle of active redistributions $<U_1, U_2, e_1>$, $<U_2, U_3, e_2>$, ..., $<U_m, U_1, e_m>$. B having a satisfying assignment implies that each clause C_k contains at least one literal l_i^k $(1 \leq i \leq 3)$ with a TRUE value. By definition of the variable assignment (Figures 11b, 11c) and by construction of G, the router g_j $(1 \leq j \leq n)$ or $\overline{g_j}$ redistributes a route from X_j or $\overline{X_j}$ into the routing instance C_k. As such, each routing instance C_k has a route to P. Then, $\forall k \in [1, m]$, the router f_k redistributes the route from C_k into U_k.

Every router e_k is running four processes (U_k, U_{k+1}, W_k and Z_k). We show that U_k is the selected routing process at e_k: Only processes U_k and U_{k+1} have a route. Z_k does not have a route since router d_k picks C_k as its selected routing process. Consequently, no router announces a route to Z_k, and neither Z_k nor W_k have a route to P. Between U_k and U_{k+1}, U_k presents a lower administrative distance at e_k and is therefore preferred. As such $\forall k \in [1, m]$, the router e_k learns a route to P from U_k and redistributes it into U_{k+1}.

The redistributions converge, and $<U_1, U_2, e_1>$, $<U_2, U_3, e_2>$, ..., $<U_m, U_1, e_m>$ form a cycle (Figure 13).

$\Leftarrow$ For the other direction, we want to show that if G has converged to a stable state containing a permanent cycle of active redistributions, then there exists an assignment such that the Boolean expression B is true.

Suppose that G has converged and contains a cycle. The only possible cycle is composed of the edges $<U_1, U_2, e_1>$, $<U_2, U_3, e_2>$, ..., $<U_m, U_1, e_m>$.

We show that routing instances $C_1, C_2, ..., C_m$ all have a stable route to P. We prove it by contradiction: we assume that there exists a routing instance $C_k (1 \leq i \leq m)$, with no stable route to P. This imples that none of the g_j nor $\overline{g_j}$ $(1 \leq j \leq n)$ redistributes a stable route into C_k.

Considering router d_k, it runs 4 routing processes (C_k, o, V_k and Z_k). Because only o has a stable route to P, d_k redistributes from o into Z_k. (C_k does not have a stable route to P by assumption. Consequently, because of the topology of G, V_k does not either). Then, router e_k learns a route to P from Z_k and redistributes it into W_k. Independent of whether U_k has a route to P or not (e.g., from U_{k-1}), e_k picks Z_k as its selected routing process since Z_k is the routing process with the lowest AD at e_k. The edge $<U_k, U_{k+1}, e_k>$ is not active.

This contradicts with the initial assumption that $<U_1, U_2, e_1>$, $<U_2, U_3, e_2>$, ..., $<U_m, U_1, e_m>$ form a cycle of active redistributions.

To summarize, we have proven that $\forall k \in [1, m]$, C_k has a stable route to P. In other words, $\forall k \in [1, m]$, one of the g_j (or $\overline{g_j}$) announces a stable from X_j (or $\overline{X_j}$) into C_k. By assigning the corresponding X_j (or $\overline{X_j}$) the value true, we obtain an assignment that satisfies B since every clause includes a literal

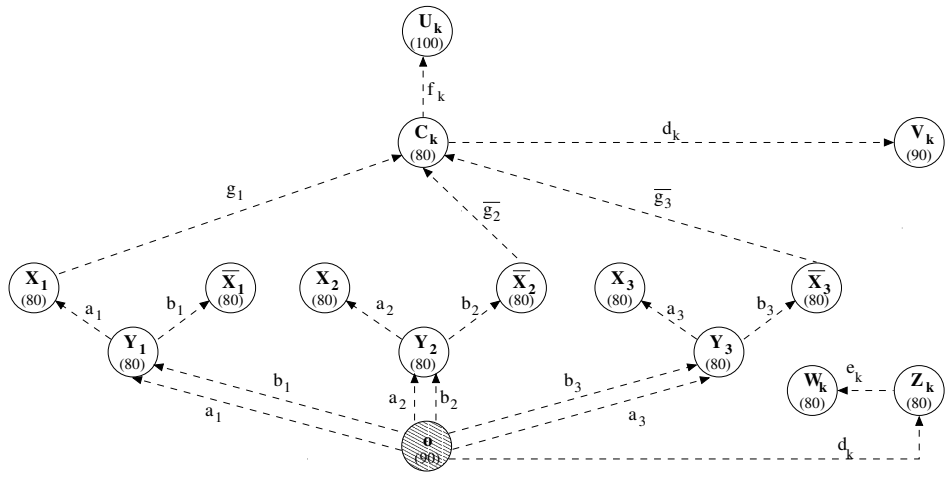

Fig. 12. Construction of the sub-graph for clause $C_k = X_1 \vee \overline{X_2} \vee \overline{X_3}$

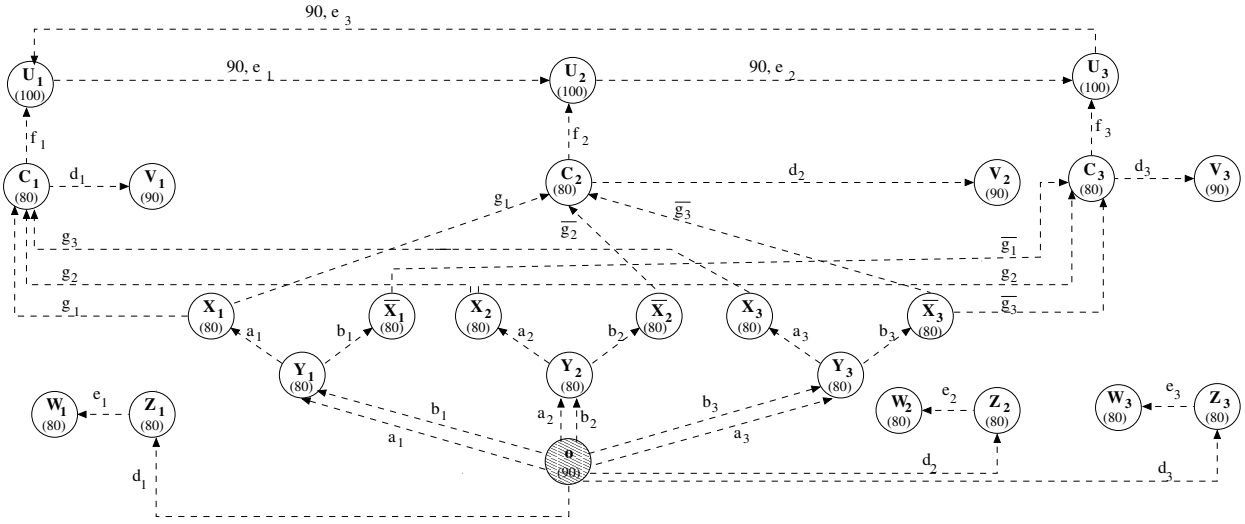

Fig. 13. Graph for $B = (X_1 \vee X_2 \vee X_3) \wedge (X_1 \vee \overline{X_2} \vee \overline{X_3}) \wedge (\overline{X_1} \vee X_2 \vee \overline{X_3})$.

that has a true value. ■

B. A Sufficient Condition for Safety

Given the complexity of the RRC-CD problem, we present a sufficient condition, which can be checked in polynomial time, to guarantee that a RR configuration will result in neither persistent routing loops nor oscillations.

For this purpose, we introduce the concept of a *primary route propagation graph <V, R, O, E', D>*. The graph represents the preferred redistributions between the different routing instances. It is extracted from the route propagation graph *<V, R, O, E, D>* as follows:

- We initialize $E' = NULL$.
- For each redistributing router, we only conserve the redistribution(s) from the routing process(es) presenting the lowest AD. We insert these edges into E' and discard all the other ones.

Theorem 6.2: A RR configuration whose primary route propagation graph satisfies the three following conditions is safe from permanent routing loops and oscillations:

1) For all vertices $u \in V$, there is a *redistribution path* from an originating vertex $o \in O$ to u. We say that there is a redistribution path from o to u if there exists u_1, u_2, ..., $u_k \in V$, and r_0, r_1, ..., $r_k \in R$, such that $<o, u_1, r_0>$, $<u_1, u_2, r_1>$, ..., $<u_k, u, r_k> \in E'$.
2) The graph is acyclic
3) Every redistributing router is redistributing from a single routing instance.

Proof: We consider a network whose primary route propagation graph satisfies the three identified conditions. For each vertex u in the route propagation graph, following condition (1), there exists a redistribution path from an originating vertex $o \in O$ to u. Such path is valid since each edge on the path is active. This derives from the fact that each redistribution on the path is a preferred one (by construction of the primary route propagation graph) and from condition (3) that states that each redistributing router redistributes from only a single routing instance. A network configuration whose

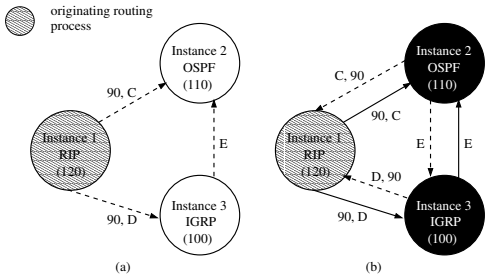

Fig. 14. The AD for $C.1$ and $D.1$ are customized to the value 90. The graph in (a) represents the corresponding primary route propagation graph. The graph in (b) depicts the state whereto RR converges.

primary route propagation graph satisfies conditions (1) and (3) converges to a stable state: each redistributing router receives a route from its preferred routing process and will select it. Finally, condition (2) ensures that no routing loop can exist. The construction of the graph and the verification of the 3 conditions can clearly be performed in polynomial times. ■

To illustrate the utility of the sufficient condition, again consider the network configuration used for scenario 1 of section III. By assigning the value 90 to $C.1$'s and to $D.1$'s AD, the configuration complies with the sufficient condition and is guaranteed to converge to an acyclic routing state. The corresponding primary route propagation graph is depicted in Figure 14a. In contrast to Figure 8, for all activation sequences the route redistributions converge to the same safe state (Figure 14b) where the edges from the primary route propagation graph are active. Figure 15 illustrates the transient and final states for one of the activation sequences.

The router redistribution model and the sufficient condition allow one to perform "what if" analysis and detect the configuration errors before deployment. [11] introduces the concept of a *robust* configuration which remains safe after node failures. A similar analysis can be carried out for route redistribution but the details are outside the scope of this paper.

A new area of research consists in identifying operational guidelines (i.e., constraints on actual configuration parameters) for achieving properties such as safety and robustness. [13] constitutes a first step in this line of research.

VII. Towards a Safe-by-Design RR Protocol

The previous sections address problems associated with the current RR procedure defined by vendors. They focused on methods that can be implemented using existing router features. In this section, we investigate longer-term solutions. We consider possible extensions and modifications to the RR procedure so that desired properties can be safely supported. The aim is to derive a RR protocol that is safe by design, i.e., it will always converge to an acyclic routing state, even in the presence of configuration errors.

A. RR and Strict Monotonicity

The analysis of RR operations reveals that RR behaves like a distance vector protocol. When a routing process x redistributes a route to a prefix P to a routing process y,

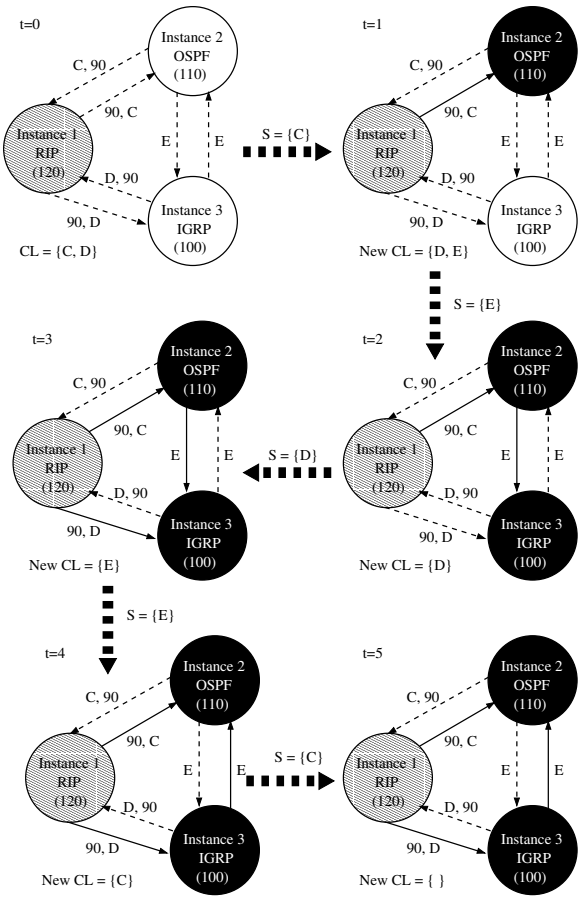

Fig. 15. Illustration of an activation sequence when the AD for $C.1$ and $D.1$ are set to 90. The redistributions converge to a cycle-free state.

x mainly announces to y that it has a route to P. y does not have a global view of the topology but only know that x is the next-hop for P. [14], [15] studied properties for vectoring algorithms and showed that strict monotonicity (SM) was an important one. Monotonicity means that the weight of a redistributed route should be larger than the one of the initially received route. However, current RR does not satisfy monotonicity. The AD value of a routing process can be set to any value and consequently, at a same router, a redistributed route may present a higher preference than the initially received route.

B. An Approach Based on Dynamic Route Tag

Modern routing protocols (e.g., RIPv2, OSPFv2, EIGRP), which support the CIDR format, allow an integer tag to be associated with an external route. This tag was designed to provide a label of the external route being redistributed. The usages of this field were left out of the routing protocol specifications and were to be specified in a separate document.

We propose to utilize this unused external route tag to ensure that RR is strictly monotonous. The route tag should carry a counter that is incremented by at least one when re-injected from a routing instance to another one. The route selection procedure needs to be upgraded to take the route tag into consideration. When receiving multiple routes to a

same destination, a router should first consider the route tag. Routes with lower route tag value should be preferred. Finally, to address the count-to-infinity problem, existing solutions can be adopted: a maximum counter value can be defined (RIP defines the maximum count as 16 [16]) or hold down timers [17] can be implemented.

Such modifications make RR SM [15]. Applying the framework from [15], RR is well captured by the *scope product* operator ([15], section 3.2): RR is used between the routing instances, and within each routing instance, the routing instance's routing protocol is applied. As long as this latter algebra – i.e., the one used in each routing instance – is SM, the resulting routing remains SM [15].

An important question meriting further investigation is how to design new modifications to the RR procedure that not only guarantee the correctness of the procedure but also preserve most of the benefits offered by route redistribution. In order to achieve this goal, we need to fully understand the operational needs of the RR procedure first. To this end, empirical studies of route redistribution usages in operational networks are required.

VIII. Related Work

Few documents address route redistribution. [18] and [19] have short sections on routing protocol interactions. They briefly describe the challenges and risks of route redistribution. [20] is an IETF standard specifying the interaction between OSPF and BGP/IDRP. However, the document is specific to these 3 protocols and does not deal with other routing protocols. [2] and [3] are vendor reports presenting possible consequences of redistribution (such as sub-optimal routing, routing loops or delayed convergence.) To prevent those undesired effects, [2] recommends preventing information received from a routing process u from being re-advertised back into u. However, such an approach violates one of the goals of route redistribution i.e., the ability for routing instances to back-up each other in case of failures. [3] focuses on redistribution between multiple instances of OSPF. The report mentions that OSPF routes are susceptible to instabilities. A number of solutions are proposed. Some of the approaches allow partial backup but do not satisfy all objectives. When network partitions happen within a routing instance, internal routers still loose connectivity. Finally, [7], [8] and [9] are proprietary extensions addressing the redistribution from BGP into specific protocols (e.g., OSPF, EIGRP). However, these extensions can still result in instabilities and do not address the redistributions between IGP processes.

IX. Conclusion

Route redistribution continues to be a popular choice for disseminating routes between routing protocols because it is relatively easy to configure and it has the flexibility to support a wide range of policy-based scenarios. However, RR misconfigurations are also common and the consequences include permanent routing loops and persistent route flaps.

This paper takes the position that most of the misconfigurations are due to a lack of understanding of the network wide RR logic and provides an analytical model to address the problem. The model precisely defines how a RR configuration influences the route selection outcomes at different routers and as such provides the first formal specification of the RR procedure. Detailed applications of the model are also presented. Not only can the model be used to explain why loops may form and routes may oscillate for certain RR configurations, it can also be used to derive sufficient conditions for an instability-free RR configuration. The analytical results give clues indicating that the current RR procedure may have fundamental weaknesses and raise the question if it requires non-trivial changes to the RR procedure to address these weaknesses. The paper exposes this issue and proposes a set of functional modifications as the first step towards deriving a safe RR protocol.

X. Acknowledgments

The authors thank John Gibson, Orathai Sukwong and Marcin Pohl for their help in setting up the presented experiments, and Sihyung Lee as well as Steve McManus for their comments. This research was sponsored by the NSF under both NeTS Grant CNS-0520210 and a Graduate Research Fellowship. 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 the U.S. government.

References

[1] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg, "Routing Design in Operational Networks: A Look from the Inside," in *Proc. ACM SIGCOMM*, 2004.
[2] "Redistributing Routing Protocols," Cisco, September 2006.
[3] "OSPF Redistribution Among Different OSPF Processes," Cisco, January 2006.
[4] "What is Administrative Distance?" Cisco, March 2006.
[5] Dwyer, Chowbay, Pavlichek, Downing, Sonderegger, Thomas, and Pavlichek, *Juniper Networks Reference Guide: JUNOS Routing, Configuration, and Architecture*. Addison-Wesley, 2002.
[6] "OSPF Design Guide," Cisco, April 2006.
[7] "Using OSPF in an MPLS VPN Environment," Cisco, 2002.
[8] "EIGRP MPLS VPN PE-CE Site of Origin (SoO)," Cisco, 2003.
[9] "ERX Routing Protocols Configuration Guide, Vol. 2," Juniper.
[10] L. Gao and J. Rexford, "Stable Internet Routing Without Global Coordination," in *Proc. ACM SIGMETRICS*, 2000.
[11] T. Griffin, F. B. Shepherd, and G. T. Wilfong, "The stable paths problem and interdomain routing." *IEEE/ACM Trans. Netw.*, 2002.
[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.
[13] F. Le and G. Xie, "On Guidelines for Safe Route Redistributions," in *Proc. ACM SIGCOMM Internet Network Management Workshop*, 2007.
[14] J. L. Sobrinho, "Network routing with path vector protocols: Theory and applications," in *Proc. ACM SIGCOMM*, 2003.
[15] T. G. Griffin and J. L. Sobrinho, "Metarouting," in *Proc. ACM SIGCOMM*, 2005.
[16] C. Hedrick, *Routing Information Protocol*, Request for Comments 1058, 1998.
[17] J. Doyle, *OSPF and IS-IS: Choosing an IGP for Large-Scale Networks*. Addison-Wesley, 2005.
[18] F. B. (Editor), *Requirements for IP Version 4 Routers*, Request for Comments 1812, 1995.
[19] J. T. Moy, *OSPF Anatomy of An Internet Routing Protocol*. Addison-Wesley, 1998.
[20] K. Varadhan, S. Hares, and Y. Rekhter, *BGP4/IDRP for IP—OSPF Interaction*, Request for Comments 1745, 1994.