



Persistent route oscillations in inter-domain routing [☆]

Kannan Varadhan ^{a,*}, Ramesh Govindan ^b, Deborah Estrin ^b

^a Lucent Technologies, Room MH 2B-230, 600 Mountain Avenue, Murray Hill, NJ 07974, USA

^b USC/Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90292, USA

Abstract

Hop-by-hop inter-domain routing protocols, such as border gateway protocol (*BGP*) and inter-domain routing protocol (*IDRP*), use *independent route selection* to realize domains' local policies. A domain chooses its routes based on *path attributes* present in a route. It is widely believed that these inter-domain routing protocols always converge. We show that there exist domain policies that cause *BGP/IDRP* to exhibit persistent oscillations. In these oscillations, each domain repeatedly chooses a sequence of routes to a destination. Complex oscillation patterns can occur even in very simple topologies. We analyze the conditions for persistent route oscillations in a simple class of inter-domain topologies and policies. Using this analysis, we evaluate ways to prevent or avoid persistent oscillations in general topologies. We conclude that if a hop-by-hop inter-domain routing protocol allows unconstrained route selection at a domain, the protocol may be susceptible to route oscillations. Constraining route selection to a provably "safe" procedure (such as *shortest path*) can reduce the number of realizable policies. Alternatively, a routing policy registry can help detect unsafe policies. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Routing; Policy; Inter-domain; BGP; IDRP; Non-convergence

1. Introduction

Internet resources, such as hosts, routers and transmission facilities, are partitioned into different administrative *domains*. In general, domains

fall into two categories: subscribers and providers. A university campus network or a corporate internal network is an example of a subscriber domain. Provider domains facilitate data exchange between subscriber domains. For economic reasons, a provider domain may wish to allow only certain classes of transit traffic to traverse its facilities. Similarly, a subscriber domain may prefer to route its traffic through a designated provider (e.g., a national backbone).

In a hop-by-hop routing infrastructure, such policies can be realized by selective dissemination of routing information. Both the border gateway protocol version 4 (*BGP* [26]), the widely deployed Internet standard for inter-domain routing) and the inter-domain routing protocol (*IDRP* [15]) provide this functionality. *BGP* and *IDRP* are sometimes called *path-vector* protocols,

[☆] This work was supported by the National Science Foundation under Cooperative Agreement NCR-9321043. The work of K. Varadhan and D. Estrin was supported by the National Science Foundation under contract number NCR-92-06418. Systems research at USC is supported through NSF infrastructure grant, award number CDA-9216321. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

* Corresponding author.

E-mail addresses: kannanv@research.bell-labs.com (K. Varadhan), govindan@isi.edu (R. Govindan), estrin@isi.edu (D. Estrin).

after the routing loop suppression mechanism they use.

BGP and IDRP also share another characteristic – they both use a similar distributed routing algorithm for hop-by-hop routing. We have coined the phrase *path-attribute-based, independent, route selection (PAIRS)*¹ to describe this type of distributed route computation. A simplified description of PAIRS follows. Each domain receives one or more *routes* from each of its neighbors. A route indicates its sender’s reachability to an *address prefix* (a network-layer address aggregate). Each route also contains one or more *path attributes*. For each received loop-free route to a given address prefix, the domain first computes an integer *preference*, then selects the route with the highest preference. Route preference assignment reflects domains’ policies. The preference function takes as input a route’s path attributes. However, *domains can independently choose their preference functions*.

It is widely believed that this distributed route computation converges, regardless of the preference functions at participating domains [23]. In this paper, we demonstrate the contrary. Specifically, we show that there exist domain preference functions for which PAIRS exhibits *persistent route oscillations*, even in the absence of topology changes. In these oscillations, each domain in a cycle of domains repeatedly selects the same sequence of routes, never converging on a single route. We construct a formalism that helps us evaluate the different solutions to the route oscillations problem. This problem is also discussed in [30,31].

The rest of the paper is organized as follows. In Section 2, we describe inter-domain topologies and

preference functions for which PAIRS exhibits persistent route oscillations. We show that these oscillations may be attributed to route “feedback”, caused by inter-dependent domain preference functions. By appropriately configuring a public-domain BGP implementation with such preference functions, we have re-created these oscillations in our laboratory testbed. However, despite the widespread deployment of BGP in the Internet, there is no anecdotal evidence of observed route oscillations of the form discussed in this paper. Existing provider policies are safe probably because the commercial Internet infrastructure is still in its infancy – therefore, the range of policies currently expressed is still limited. We think conditions for route oscillations are more likely to occur as the commercial Internet matures, and as the Internet transitions to the more expressive IDRP. It is important to understand the pathological situations in any protocol, however rare they may be, so as to be able to avoid these situations, and otherwise recognize and recover from them even if they should occur [22].

In Section 3, we study these oscillations in a restricted class of inter-domain topologies. For these topologies, we describe a representation of domain preference functions that we call return graphs. Using this representation, we derive necessary and sufficient conditions for the existence of route oscillations in these topologies. Our derivation shows that these oscillations can happen in relatively complex ways even in simple topologies.

The existence of route oscillations in inter-domain routing points to a routing protocol design failure. In Section 4, we show that constraining PAIRS to consider only a small “safe” subset of path attributes can significantly reduce the number of policies realizable in those protocols. Not surprisingly, perhaps, realizing richer policies through independent route selection can adversely affect route convergence in PAIRS.

However, in the existing commercial Internet infrastructure, mechanisms to realize policy through independent route selection are already widely deployed. In this situation, a combination of the following two approaches can be adopted (Section 5). The first approach analyzes domain policies a priori to detect the likelihood of route

¹ Even though BGP and IDRP are path-vector protocols, the routing behavior described in this paper is not caused by the loop suppression mechanism they use. If a routing protocol were to use a different loop suppression mechanism [8], but allow independent route selection, it would also be susceptible to the routing behavior we describe here. Moreover, this routing behaviour is independent of the nature of the specific attributes that could be carried in a route, and can occur even if arbitrary information is included as a path attribute. To focus on the independent route selection aspect of these protocols, we classify them as PAIRS protocols.

oscillations; one or more domain policies can then be modified to avoid oscillations. A routing policy registry (such as the Internet Routing Registry [3,4,1]) is useful for this. The second approach introduces additional protocol mechanisms that detect the existence of an oscillation, and modify one or more domains' policies to suppress the oscillation. Unsafe policies can then be realized using explicit routes [9,11].

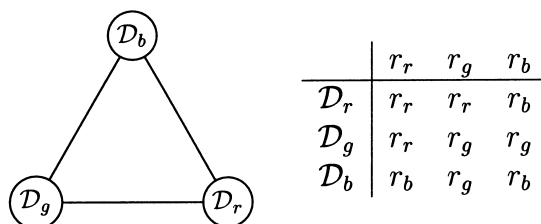
2. Examples and motivation

In this section, we show how persistent route oscillations can result from PAIRS route computation. We first introduce a simple model of PAIRS route computation at each participating domain \mathcal{D} . In this paper, we assume that all references to routes pertain to address prefix x , unless otherwise stated. Domain \mathcal{D} maintains the last route advertisement to x heard from each of its neighbors. \mathcal{D} also maintains the last route r to x

that it advertised. Suppose \mathcal{D} hears a new route advertisement for x from its neighbor. It assigns a preference to this route and recomputes the most preferred route to x . If this route is different from r , \mathcal{D} propagates this route. PAIRS route computation is said to *converge* if at some future point in the computation, no further route advertisements occur.

Even in a relatively small inter-domain topology, PAIRS can exhibit persistent route oscillations (i.e., non-convergence) for routes to x . Consider three domains \mathcal{D}_r , \mathcal{D}_g , and \mathcal{D}_b connected together as shown in Fig. 1. Suppose that domain \mathcal{D}_r (respectively \mathcal{D}_g and \mathcal{D}_b) has a “direct” route r_r colored “red” (respectively r_g colored “green”, and r_b colored “blue”) to the destination. With the preference functions shown in Fig. 1, the PAIRS algorithm exhibits persistent route oscillations (Fig. 1). Intuitively, this is because the policies of the three domains are not simultaneously satisfiable.

The preference functions of Fig. 1 are not simply based on the identity of the domain that



<i>Domain Policies</i>		
\mathcal{D}_r	prefers “blue” over “red” over “green” routes	
\mathcal{D}_g	prefers “red” over “green” over “blue” routes	
\mathcal{D}_b	prefers “green” over “blue” over “red” routes	

Fig. 1. Example of a cyclic domain policy that leads to non-convergence in PAIRS. In the first part of the figure (topology) each domain is represented by a circle. The policies of each domain are shown in the third part of the figure. The second part of the figure is a compact representation of the preference functions at all of the domains (\mathcal{D}_r , \mathcal{D}_g , and \mathcal{D}_b), and for all possible routes that can occur in this topology. The entries in each column are the routes that a domain will select when it receives a route corresponding to that column. Notice that the preference functions are inter-dependent: \mathcal{D}_r 's most preferred route is r_b , \mathcal{D}_g 's most preferred route is r_r , and \mathcal{D}_b 's most preferred route is r_g . Also, \mathcal{D}_r will never select r_g , \mathcal{D}_g will never select r_b , and \mathcal{D}_b will never select r_r .

Intuitively, we can see that there is no unique route assignment, such that each node is assigned a route that satisfies its local policies. Therefore, if each of the domains has a route to destination x , then the selection and advertisement of its route to x by any domain will lead to a conflict in another domain's route; that other domain will then change its route, and advertise a new route. Hence this set of nodes and routes will never converge on their routes to a destination x .

Later in this section, we will see that this topology can oscillate in a number of complex patterns.

advertises a route; for instance, even though \mathcal{D}_r hears green and blue routes from \mathcal{D}_b , it selects one and not the other. \mathcal{D}_r 's preference functions express its policies regarding the routes it hears from \mathcal{D}_g and \mathcal{D}_b . It is not unusual for a provider to specify such a policy in the existing Internet. We think that inter-dependent policies similar to that in Fig. 1 are not unlikely in the future Internet.

In Fig. 1, each domain alternates between two routes – its own “direct” route and that of its anti-clockwise neighbor. At some instant t , a domain can have selected exactly one of these routes. The selected route defines that domain's *route state* (or *r-state*) at t . If a domain is in an *r-state* r at t , it must have last advertised r . When a domain receives a route advertisement, its *r-state* may change. At different times, a domain can be in different *r-states*. The *r-states* at a domain are determined by its neighbors' *r-states*, and its own preference functions. For example, \mathcal{D}_r has exactly two states: r_r and r_b . \mathcal{D}_r never selects r_g , since the direct route r_r is always available.

From Fig. 1, we can make the following observations:

1. These persistent route oscillations occur in the absence of topology changes.
2. This topology oscillates regardless of route processing times and route propagation delays at the three domains.
3. The lack of a global metric in PAIRS causes each domain to oscillate between loop-free paths.
4. If packet forwarding is synchronized with route exchange, packets could loop indefinitely.
5. Independent of the initial *r-states* of the domains, PAIRS always exhibits persistent oscillations in this topology.
6. The original hop-by-hop distance vector algorithms of [17,27,29] are provably loop-free; local policies that are configured at each domain introduces the oscillations.

In the rest of the paper, we use a more extensible notation to consider this problem in other more general topologies. We rewrite the domains, \mathcal{D}_r , \mathcal{D}_g , and \mathcal{D}_b , as \mathcal{D}_0 , \mathcal{D}_1 , \mathcal{D}_2 , respectively; their corresponding direct routes are then r_0 , r_1 , and r_2 . The topology and the corresponding preference functions are then shown in Fig. 2.

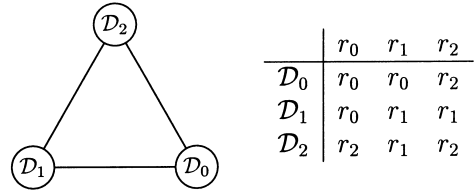


Fig. 2. Generalized description of Fig. 1.

There exist preference functions that cause PAIRS to behave differently for different initial *r-states*. Fig. 3 shows preference functions for a cycle of four domains. PAIRS converges in this topology for a particular assignment of initial states; following convergence, each domain has a route that is stable and non-oscillatory. With other initial *r-states*, the topology exhibits two different kinds of oscillations. In one of them, \mathcal{D}_0 repeatedly selects r_0 and r_2 . In the other, \mathcal{D}_0 repeatedly oscillates among r_0 , r_2 , and r_3 . In this example, PAIRS can exhibit a persistent route oscillation *despite the existence of a stable route assignment*.

Notice that, in the above results, we make no assertion about how the cycle of four domains arrives at any particular initial *r-state*. We consider protocol operation from an initial configuration in which each domain in the cycle has selected a direct route, either its own, or that of another in the cycle that it acceptable to it given its policy configuration. We then assert that, if the cycle is in that particular initial configuration, then oscillation, if it occurs, occurs independent of message propagation delays and route computation speeds. However, it may be the case that, for a cycle of domains to reach that initial configuration may depend on initial route selections at each domain, as well as route computation and propagation delays. How a cycle of domains could get to a particular initial *r-state* is beyond the focus of our present work.

What causes the oscillations described in Figs. 2 and 3? In Fig. 2, observe that a domain's *r-state* can “feedback” into another, possibly different, *r-state*. Informally, when \mathcal{D}_0 advertises r_0 , \mathcal{D}_1 transitions to *r-state* r_0 , as its preference function dictates. Then, \mathcal{D}_1 's advertisement of r_0 causes \mathcal{D}_2 to select and advertise r_2 . This advertisement causes \mathcal{D}_0 to select r_2 . We say that *r-state* r_0 returns to r_2 at \mathcal{D}_0 . Intuitively, route oscillations happen in

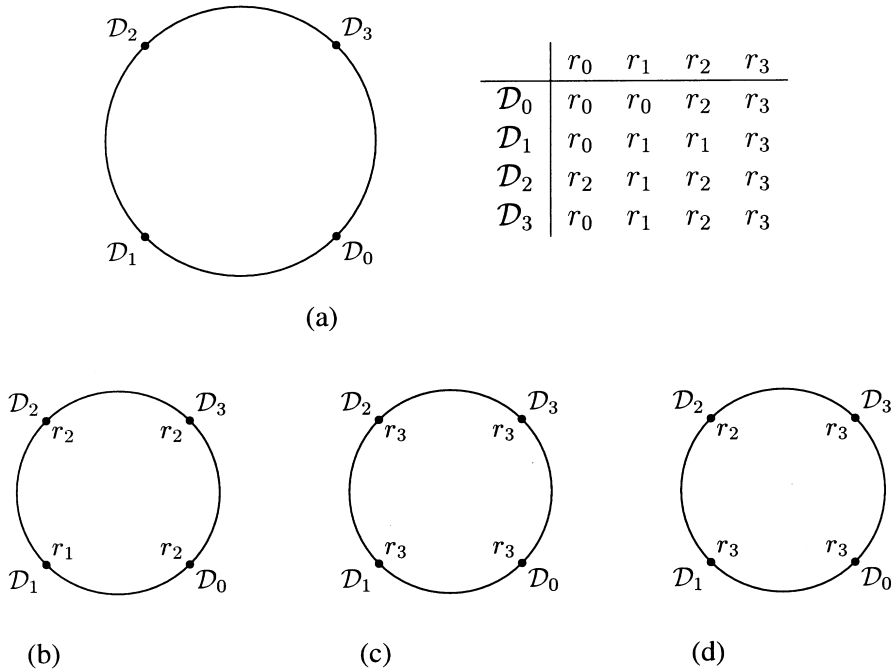


Fig. 3. **PAIRS behavior with four nodes.** (a) Preference functions at four domains: Each domain has a direct route to x . (b) Initial r -states for \mathcal{D}_0 to oscillate as $\langle r_0, r_2, r_0 \rangle$: these initial r -states can be realized, for example, if \mathcal{D}_2 's advertisement reaches \mathcal{D}_3 , \mathcal{D}_0 , and \mathcal{D}_1 before those domains have processed their direct route. (c) A possible stable assignment: if \mathcal{D}_3 's advertisement for r_3 reaches all other domains first, these r -states result. (d) Initial r -states for \mathcal{D}_0 to oscillate as $\langle r_0, r_3, r_2, r_3, r_0 \rangle$: if \mathcal{D}_2 and \mathcal{D}_3 select their direct route, and the advertisement for r_3 reaches \mathcal{D}_0 and \mathcal{D}_1 , these initial r -states are achieved.

Fig. 2 because there exists a cycle of returns at \mathcal{D}_1 : r_1 returns to r_2 , and r_2 returns to r_1 . In Fig. 3, \mathcal{D}_0 has *two* such cycles, in one of which r_3 returns to itself.

3. Characterizing route oscillations in simple topologies

In this section, we attempt to analyze persistent route oscillations in a particular class of inter-domain topologies. In these topologies, domain preference functions can be represented as return graphs, based on the notion of return states. We derive necessary and sufficient conditions on return graphs for the existence of route oscillations in these topologies. We will use these conditions on return graphs to evaluate different mechanisms as solutions to this problem in the following section (Section 4).

Table 1 summarizes the various terms introduced in this paper.

3.1. Assumptions and problem statement

Suppose that the three domains of Fig. 2 were part of a larger inter-domain topology. Depending on the policies of adjacent domains, the oscillations at these three domains could affect a number of other domains, perhaps triggering other “sympathetic” oscillations. Visualizing, and reasoning about, these complex oscillation patterns in general topologies is difficult.

For this reason, we consider a more restricted class of topologies which exhibit route oscillations. Informally, we believe that the kind of route feedback described in the previous section cannot happen in acyclic topologies.² So, we consider a class of simple cyclic topologies, which we call D.

² The one exception, that we are aware of, is the acyclic topology of two nodes connected directly to each other. We can model this as the two-node topology in D.

Table 1
Glossary of terminology and notation

D	A cyclic inter-domain topology in which no domain selects routes from its clockwise neighbor
\mathcal{D}_i	A domain in D. Domains in D are numbered with integer subscripts. $\mathcal{D}_{i\oplus 1}$ is \mathcal{D}_i 's clockwise neighbor
r_i	\mathcal{D}_i 's direct route. We assume that this route is always present as a fall-back route
$pref_i$	A function that takes two routes, and returns the route that has a higher preference at \mathcal{D}_i
r -state	At any instant t , the route selected by a domain. At different instants, a domain can select different routes
\mathcal{R}_i	The collection of possible r -states of \mathcal{D}_i
return relation	We say r_a returns to r_b at \mathcal{D}_i , if when \mathcal{D}_i advertises r_a , route feedback causes \mathcal{D}_i to select r_b
return graph	For each \mathcal{D}_i , the directed graph whose nodes are the r -states in \mathcal{R}_i and whose arcs express the return relationships between those states
G_0	The return graph at \mathcal{D}_0 . G_i is the return graph for any \mathcal{D}_i in D
return cycle	A cycle in a component of the return graph. Every component in D has exactly one cycle
\mathcal{C}	A single cycle in G_0 . \mathcal{C}_i denotes the return cycle isomorphic to \mathcal{C} in G_i

D contains n domains $\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_{n-1}$. Each domain \mathcal{D}_i peers with $\mathcal{D}_{(i-1)\bmod n}$ and $\mathcal{D}_{(i+1)\bmod n}$ (respectively notated $\mathcal{D}_{i\oplus 1}$ and $\mathcal{D}_{i\oplus 1}$). Without loss of generality, assume $\mathcal{D}_{i\oplus 1}$ is \mathcal{D}_i 's clockwise neighbor. Assume further that, each domain \mathcal{D}_i has a direct route r_i that is always available as a fall-back. Fig. 4 describes preference functions for a persistent route oscillation in D. In this oscillation, each domain repeatedly selects $n - 1$ r -states.

In this paper, we study those route oscillations in D that occur in the absence of topology changes and are independent of route computation times and route processing speeds. We say a \mathcal{D}_i oscillates if it repeatedly selects a sequence of r -states r_a, r_b, \dots, r_x . One of these r -states can be r_i . The other r -states must correspond to routes heard from $\mathcal{D}_{i\oplus 1}, \mathcal{D}_{i\oplus 1}$, or both. Here, we consider those oscillations in which \mathcal{D}_i 's r -states correspond either to r_i or to routes heard from $\mathcal{D}_{i\oplus 1}$. That is, we

restrict the class of preference functions in D to those in which a \mathcal{D}_i never selects a route from $\mathcal{D}_{i\oplus 1}$. Our analysis also applies to oscillations in which \mathcal{D}_i selects either r_i or routes from $\mathcal{D}_{i\oplus 1}$. Section 3.5 discusses the likelihood of oscillations in which \mathcal{D}_i 's r -states include routes from both $\mathcal{D}_{i\oplus 1}$ and $\mathcal{D}_{i\oplus 1}$.

With these assumptions, we attempt to answer the following two questions:

- Among the class of preference functions we consider, which ones can cause route oscillations in D?
- For given preference functions in D, what are the different ways (if any) in which D can oscillate?

One possible answer to these questions is suggested by the following approach. If we represent the current state of D by a vector of r -states, we can represent the next state of D as a product of a

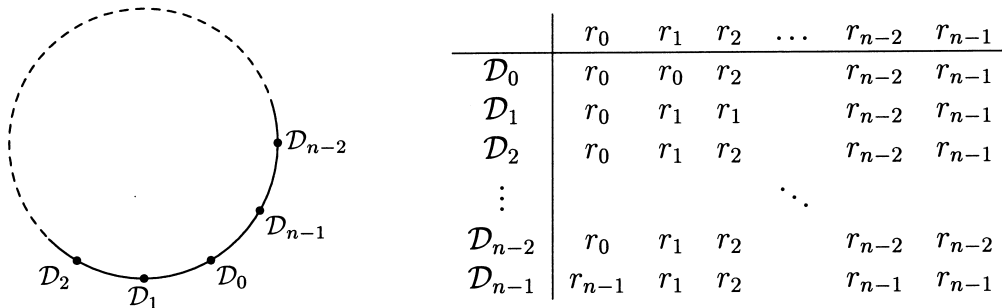


Fig. 4. A persistent route oscillation among n domains: each domain, with a direct route and the table of domain preference functions are shown. Each domain prefers its anti-clockwise neighbor's direct route more than its own. \mathcal{D}_1 oscillates as $\langle r_0, r_{n-1}, r_{n-2}, \dots, r_2, r_0 \rangle$, if some domain initially advertises its direct route.

state transformation matrix and the current state of D. This transformation matrix is determined by the given preference functions. Conditions on the eigenvalues of this matrix determine whether D can oscillate or not [21]. In this paper, we describe an alternative representation of D’s preference functions, which we call a *return graph*. The choice of this term was motivated by the roughly analogous control-theoretic notion of a *first return map* (sometimes also called a Poincaré map) [12].

3.2. Return graphs

In Section 2, we informally introduced the notion of a domain’s *r-state*. At any instant t , the *r-state* of a domain is the route it has selected. In D, the *r-states* of \mathcal{D}_i can include r_i and some other domains’ direct routes heard from $\mathcal{D}_{i\oplus 1}$. A direct route r_j is an *r-state* of \mathcal{D}_i if and only if, when \mathcal{D}_j advertises r_j , all domains between \mathcal{D}_j and \mathcal{D}_i (going clockwise in D, \mathcal{D}_i inclusive), select that route. Denote the preference function at \mathcal{D}_i by $pref_i$; $pref_i(r_a, r_b)$ is the more preferred of r_a and r_b at \mathcal{D}_i . Formally, r_j is an *r-state* of \mathcal{D}_i if and only if $pref_k(r_j, r_k)$ is r_j for all k in $j \oplus 1, \dots, i \ominus 1, i$. Thus, the set of possible *r-states* of \mathcal{D}_i (denoted by \mathcal{R}_i) can be determined entirely from domain preference functions.

In Section 2, we also introduced the returns relation between two states. We said that, at \mathcal{D}_i , r_a

returns to r_b if, when \mathcal{D}_i advertises r_a , route feedback causes \mathcal{D}_i to transition to r_b . Equivalently, the returns relation is can be defined in terms of domain preference functions in D. Suppose that r_a and r_b are two *r-states* at \mathcal{D}_i . Then, r_a returns to r_b at \mathcal{D}_i if and only if

$$r_b = pref_i(pref_{i\oplus 1}(pref_{i\oplus 2}(\dots(pref_{i\oplus 1}(r_a, r_{i\oplus 1}), \dots), r_{i\oplus 2}), r_{i\oplus 1}), r_i). \quad (1)$$

That is, when \mathcal{D}_i selects r_a , $\mathcal{D}_{i\oplus 1}$ selects $pref_{i\oplus 1}(r_a, r_{i\oplus 1})$, $\mathcal{D}_{i\oplus 2}$ selects $pref_{i\oplus 2}(pref_{i\oplus 1}(r_a, r_{i\oplus 1}), r_{i\oplus 2})$, and so on, exactly once around D.

Given the preference functions in D, we can define at \mathcal{D}_i a directed return graph G_i , whose nodes are the *r-states* in \mathcal{R}_i . G_i has a directed arc from r_a to r_b if and only if r_a returns to r_b . Fig. 5 shows the return graph for the example in Fig. 3. This collection of return graphs is an alternative representation of the preference functions shown in Fig. 3.

3.3. Properties of return graphs

We can make several general observations about return graphs. Eq. (1) implies that each node in a return graph has exactly one outgoing arc. Such a directed graph has a well-defined structure; it may be disconnected, and each connected component generally contains one or more chains

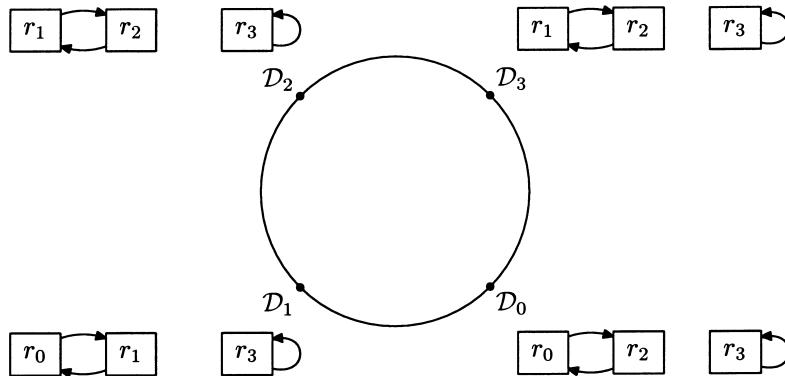


Fig. 5. Return graphs for the topology of Fig. 3(a). In this topology, the return graph corresponding to each domain is shown adjacent to that domain. Each return graph has two components. One component is a cycle consisting of two nodes. The second component is a cycle consisting of one node.

“leading into” *exactly one* cycle. (For, if a connected component contained two cycles, some node in that component must have two outgoing arcs.) A cycle in a return graph may have one or more nodes (Fig. 5). A one-node cycle corresponds to an r -state that returns to itself.

Cycles in a return graph G_i have several interesting properties:

1. Since every node in a return graph has exactly one outgoing arc, a node in G_i can be in at most one cycle. Moreover, the directed path leading out from any node in a return graph eventually leads into a cycle.
2. A one-node cycle corresponds to a stable route assignment. That is, if r_a returns to r_a at \mathcal{D}_i , then the following is a stable route assignment in D : r_a at \mathcal{D}_i , $\text{pref}_{i\oplus 1}(r, r_{i\oplus 1})$ at $\mathcal{D}_{i\oplus 1}$, and so on.
3. From the previous property, it is trivially true that for a one-node cycle in G_i , there exists a “corresponding” one-node cycle in $G_{i\oplus 1}$. If there exists a two-node cycle in G_i , there exists a “corresponding” two-node cycle in $G_{i\oplus 1}$. To see this, suppose that r_a and r_b constitute the two-node cycle in G_i . Clearly the states $\text{pref}_{i\oplus 1}(r_a, r_{i\oplus 1})$ and $\text{pref}_{i\oplus 1}(r_b, r_{i\oplus 1})$ (say r_c and r_d , respectively) must be in $\mathcal{D}_{i\oplus 1}$. If r_a returns to r_b in G_i , then r_c returns to r_d in $G_{i\oplus 1}$. Conversely, if r_b returns to r_a in G_i , then r_d returns to r_c in $G_{i\oplus 1}$. Finally, r_c and r_d cannot be identical; if they were, r_a and r_b must also be identical. Extending this argument, if there exists a k -node cycle in G_i , there exists a k -node cycle in every other domain’s return graph. Since a node in G_i can be in at most one cycle, the cycles in other domains’ return graphs are *isomorphic* to the cycles in G_i .

From Property 3, cycles in G_0 are representative of cycles in all G_i .

In D , one of the r -states of every domain \mathcal{D}_i is its direct route r_i . This r -state must lead into some cycle in G_i (from Property 1). That cycle corresponds to a cycle (call it \mathcal{C}) in G_0 . We say that r_i can *activate* \mathcal{C} . Thus, in Fig. 3(c), r_3 can *activate* the one-node cycle. Intuitively, if only \mathcal{D}_i were to advertise r_i initially, cycle \mathcal{C} would eventually be realized. More than one direct route can *activate* the same cycle. In Fig. 3(b), any one of r_0 , r_1 , or r_2 can *activate* the two-node cycle. The collection of

initially *activated* cycles defines the initial r -states of domains in D .

3.4. Persistent route oscillations in D

In this section, we describe necessary and sufficient conditions on cycles in G_0 for the existence of persistent route oscillations in D . Earlier, we said \mathcal{D}_i oscillates if it repeatedly visits the same sequence of k r -states, for some k . We say an oscillation exists in D if at least one domain in D oscillates. In D , if \mathcal{D}_i oscillates among k r -states, then $\mathcal{D}_{i\oplus 1}$ must oscillate among at least k r -states (any state transition in \mathcal{D}_i can only be triggered by a state transition in $\mathcal{D}_{i\oplus 1}$, since r_i is always available and, by our assumption, \mathcal{D}_i never selects a route advertised by $\mathcal{D}_{i\oplus 1}$). Actually, $\mathcal{D}_{i\oplus 1}$ must oscillate among exactly k routes. Otherwise, $\mathcal{D}_{i\oplus 2}$ and all other domains in D , including $\mathcal{D}_{i\oplus 1}$ must oscillate among more than k routes. This is a contradiction. We call the smallest repeated sequence of r -states at \mathcal{D}_i its *period*.

Intuitively, if G_0 has a multi-node cycle \mathcal{C} , D can exhibit persistent route oscillations. This happens if r_i can *activate* \mathcal{C} , and \mathcal{D}_i initially advertises r_i . Thus, there exists an oscillation in the topology of Fig. 2 if \mathcal{D}_0 initially advertises r_0 .

Less obviously, two (or more) one-node cycles can also cause oscillations in D . For example, Fig. 6 shows a topology in D . In this topology, G_0 has three one-node cycles. There exists an oscillation in this topology if \mathcal{D}_0 and \mathcal{D}_1 initially advertise r_0 and r_1 , respectively.

The following theorem (Theorem 1) formalizes these two observations.

Theorem 1. D can exhibit persistent route oscillations if and only if either G_0 has at least one k -node cycle ($k > 2$), or G_0 has more than one one-node cycle.

Proof. We now sketch a proof for Theorem 1. The proof sketch has two parts. In the first part, we show that if G_0 has exactly one one-node cycle, D cannot oscillate. In the second part, we show ini-

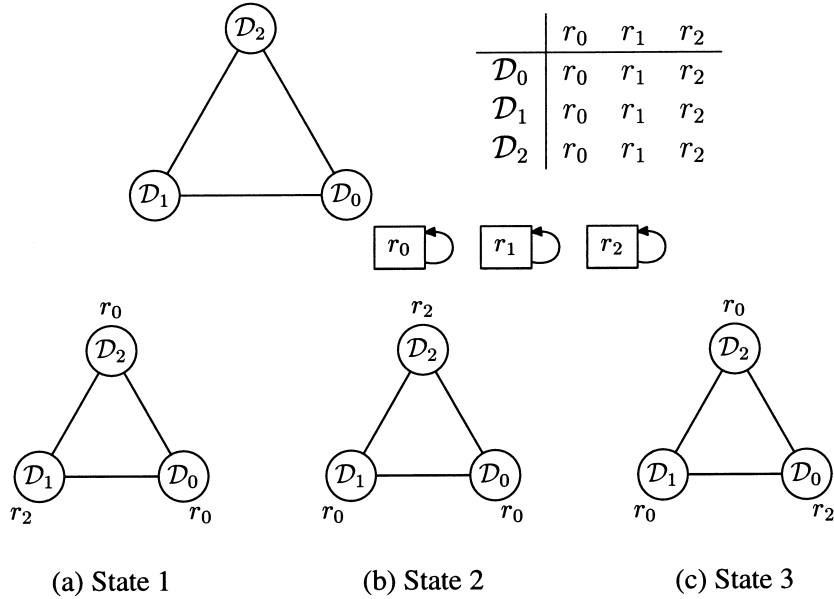


Fig. 6. **Multiple one-node cycles can cause an oscillation:** The first part shows a three-domain topology in \mathcal{D} with the preference functions at \mathcal{D}_0 . It also shows G_0 , with three one-node cycles. There exists an oscillation in this topology if at least two of these three cycles are initially activated. Further, this figure shows the three states of the oscillation, when r_0 and r_2 are initially activated.

tial conditions for an oscillation in \mathcal{D} if G_0 has one k -node cycle ($k > 2$), or G_0 has more than one one-node cycle.

1. Suppose G_0 has exactly one one-node cycle. Then, since the cycles in G_0 are representative of cycles in other domains' return graphs, all other G_i must have exactly one one-node cycle. Assume to the contrary that \mathcal{D} exhibits a persistent route oscillation. Suppose that \mathcal{D}_i 's period has two routes r_a and r_b (the proof for the case when \mathcal{D}_i 's period has k routes is similar). Now, in G_i , the directed path out of r_a must lead into a cycle (from Property 1). The same is true for r_b . Suppose r_c constitutes the one-node cycle in G_i .

If r_c is different from both r_a and r_b , then when \mathcal{D}_i advertises either r_a or r_b , it will eventually attain r -state r_c . But that is a contradiction, since we assume that \mathcal{D}_i repeatedly selects only r_a and r_b .

If r_c is r_b , then advertisement of both r_a and r_b by \mathcal{D}_i results in an eventual transition into r_b (i.e., r_a cannot recur in the sequence), a contradiction.

A similar contradiction occurs if r_c is r_a .

2. Suppose G_0 has one k -node cycle. Without loss of generality, assume that r_j can activate this cycle. Then, a start state in which only r_j is initially advertised will result in persistent route oscillations. A period of the oscillation at each \mathcal{D}_i contains the r -states of the k -node cycle. This follows from the definition of the returns relation. Suppose G_0 has two one-node cycles. Without loss of generality, assume that r_i and r_j can activate these two cycles. An initial state in which only r_i and r_j 's advertisements initially traverse \mathcal{D} will result in an oscillation whose period contains the r -states in their two one-node cycles. \square

3.5. Discussion

In this section, we discuss the implications of the conditions for the existence of an oscillation in \mathcal{D} . We also consider the effect of relaxing some of our assumptions about the topology and the preference functions.

Given a set of preference functions in \mathcal{D} , we can use Theorem 1 to determine the different ways in

which domains in D can oscillate. The theorem describes two ways: when either a single multi-node cycle, or two one-node cycles are initially *activated*. Oscillations with more complex periods are possible. Suppose r_a and r_b activate two different cycles \mathcal{C}_a and \mathcal{C}_b . The period of the resulting oscillation contains the r -states of \mathcal{C}_a and \mathcal{C}_b . The oscillation of Fig. 3(d) is an example of this. However, if r_a and r_b activate \mathcal{C} , it is possible for the period of the oscillation to contain two instances of each r -state in \mathcal{C} .

When two or more cycles are initially *activated*, the order of the r -states in a period of the oscillation depends on the routes used to *activate* the cycles. Fig. 7 demonstrates this.

If G_0 has one multi-node cycle, only one cycle need be *activated* to cause route oscillations. If G_0 has only multi-node cycles, it follows from Property 1 above that *any* initial state leads to route oscillations. This is the case with Fig. 2; there exists no stable route assignment in D . We call such return graphs *unsatisfiable*. PAIRS admits preference functions which can result in unsatisfiable return graphs.

We considered a particular kind of oscillation, one in which route advertisements “flow” clockwise around D . In D , can \mathcal{D}_i 's r -states include routes advertised both by $\mathcal{D}_{i\oplus 1}$ and $\mathcal{D}_{i\ominus 1}$? We have not been able to construct examples of such os-

cillations without assuming some temporal ordering on each domain's route selection policies. Intuitively, if a period of the oscillation at \mathcal{D}_i includes routes from both $\mathcal{D}_{i\ominus 1}$ and $\mathcal{D}_{i\oplus 1}$, then varying route propagation delays can perturb the order of r -states within a period of the oscillation. For this reason, we believe that if D oscillates independent of route processing times and route propagation delays, the oscillations must either be clockwise or anti-clockwise.

We also believe that if a more general topology oscillates independent of topology changes, there must exist at least one cycle of domains that oscillates in a clockwise or anti-clockwise manner. As we have said before, in a more general topology, other domains may exhibit sympathetic oscillations.

To analytically examine route oscillations, we considered a constrained class of topologies. Are return graphs applicable in more general topologies? Obviously, our analysis applies to those sub-graphs of the more general topologies that satisfy the requirements for D . However, in D an r -state r 's return state was uniquely defined. In a general topology, more than one return state is possible for a given r at \mathcal{D}_i . Whether it is possible to derive conditions for the existence of an oscillation in these more general return graphs is left for future study.

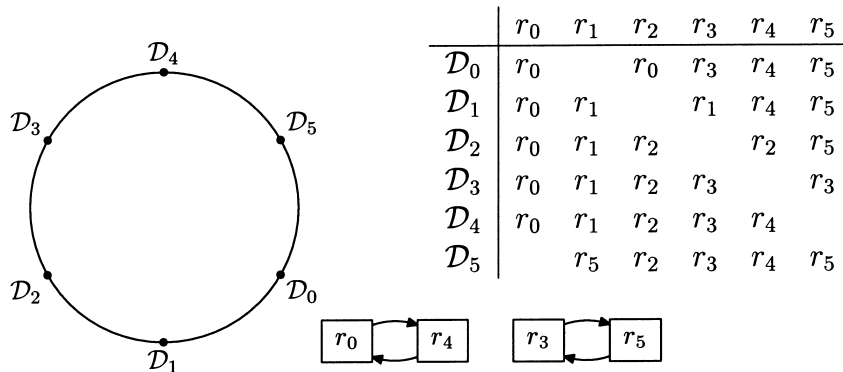


Fig. 7. **Different oscillation periods for different initial conditions.** A six domain topology in D is shown. There are two two-node cycles in G_0 . If r_0 and r_3 are initially advertised, the resulting oscillation has the following period at \mathcal{D}_0 : $\langle r_0, r_3, r_4, r_5 \rangle$. However, if r_0 and r_5 are initially advertised, the resulting oscillation has the following period at \mathcal{D}_0 : $\langle r_0, r_5, r_4, r_3 \rangle$.

4. Constraining PAIRS

In the previous section, we showed preference functions for which PAIRS can exhibit persistent route oscillations. We now consider constraining PAIRS to allow preference functions expressed in terms of a path attribute X . Such preference functions are *safe* if they do not cause oscillations in a general topology. We consider the question: Do there exist safe preference functions on X ? If there exists an X such that preference functions on X allow “interesting” policies, constraining PAIRS to these preference functions is an acceptable solution to the oscillation problem.

We believe that if preference functions on X are safe in D , then they are safe in more general topologies. Put differently, if preference functions based on X can result in a cycle of oscillating domains in a general topology, we can construct an oscillation in D caused by preference functions based on X . Intuitively, this construction simply “extracts” the cycle of domains and their preference functions from the more general topology.

To show that preference functions based on X are safe in D , it suffices to show that the equivalent return graphs can contain exactly one one-node cycle. As we have discussed earlier, routes in BGP and IDRP carry a `PATH` attribute – this is a sequence of domains that the route has traversed. In this section, we consider two possible preference functions based on the `PATH` attribute. We show that if all domains are constrained to selecting the shortest `PATH` route, oscillations cannot happen in D . We also show that if domains were allowed to independently select routes based on the first element of the `PATH` only (*next-hop*), multi-node cycles cannot form in a domain’s *return graph*.

4.1. Shortest `PATH`

In Fig. 2, at least one domain’s *r-states* contains a route with a `PATH` longer than its direct route. Denote by $l(r_0)$ the `PATH` length of r_0 at \mathcal{D}_0 . If each domain always selected its shortest path route, then $l(r_i) + 1 \leq l(r_{i \ominus 1})$, i.e., $l(r_i) < l(r_{i \ominus 1})$. Putting these inequalities together, we arrive at a contradiction.

This observation motivates considering shortest path route selection to realize safe policies. We can show that this preference function is always safe in D . If r_i is \mathcal{D}_i ’s shortest path route, then any route that \mathcal{D}_i selects and advertises will return to r_i . Therefore, there is only one one-node cycle at G_i , and by extension, there is only one one-node cycle in G_0 . Therefore, D will not oscillate if every domain uses shortest path route selection. We believe that shortest path route selection will not cause oscillations in other more general topologies.

This is not a new result. We know from [17,29] that a distance vector hop-by-hop algorithm augmented with a loop suppression mechanism always converges, i.e., never oscillates. This algorithm is similar to a PAIRS algorithm constrained to only select shortest `PATH` routes [8,27].

4.2. Next-hop

\mathcal{D}_2 in Fig. 2 advertises r_1 and r_2 to \mathcal{D}_0 . By looking at the entire `PATH` of those routes, \mathcal{D}_0 only selects r_2 and not r_1 . If \mathcal{D}_0 ’s preference functions are based only on the first element of the `PATH`, i.e., the next-hop, then \mathcal{D}_0 cannot assign different preferences to r_1 and r_2 .

This observation motivates considering next-hop-based preference functions to realize safe policies. However, domain preference functions in Fig. 6 are based on next-hop; we have shown that this topology is susceptible to a route oscillation for certain initial *r-states*.

Next-hop-based functions cannot result in multi-node cycles in D . Consider route preference functions expressed only on the next-hop. Each \mathcal{D}_i has two possible choices: it prefers no route advertised by $\mathcal{D}_{i \ominus 1}$, or it prefers every route advertised by $\mathcal{D}_{i \ominus 1}$. If any one \mathcal{D}_i always chooses r_i regardless of any route advertised by its neighbor, i.e., r_i returns to r_i in G_0 , G_0 contains exactly one one-node cycle. If all domains \mathcal{D}_i prefer routes advertised by their neighbors $\mathcal{D}_{i \ominus 1}$, G_0 has n one-node cycles, one corresponding to each r_i .

Next-hop-based preference functions have a possible stable assignment in D . In Section 4.3, we show how next-hop-based policies may be relatively safely realized when used in conjunction with other mechanisms.

Existing Internet provider policies are largely next-hop-based [3,4]. However, for next-hop-based preference functions to cause oscillations, there must exist a cycle of domains in which every \mathcal{D}_i prefers $\mathcal{D}_{i \ominus 1}$ over their fall-back route. The relatively small likelihood of this configuration probably explains why route oscillations have not been observed in the current Internet.

4.3. Discussion

Preference functions based on shortest PATH and next-hop restrict the kinds of policies that can be realized. A domain that has multiple routes to a given destination can choose any of those routes in PAIRS; but with shortest PATH route selection the domain can only choose among those routes that have the shortest path length. With next-hop-based preference functions, a domain cannot express policies about providers that are not directly adjacent; domains may desire such expressivity in a commercial Internet.

Other preference functions on the PATH are likely unsafe. All of our examples of topologies in earlier sections can be created using arbitrary preference functions on the PATH attribute. We have also found that preference functions on most other BGP and IDRP path attributes are unsafe (e.g., DIST_LIST_INCL). We conclude that in hop-by-hop inter-domain routing protocols, such as BGP/IDRP, constraining PAIRS to preference functions based on safe attributes allows only relatively “uninteresting” policies.

5. Other approaches

The previous section indicates that there do not seem to exist path attributes that are simultaneously safe and interesting. To realize richer policy through independent route selection, yet avoid or minimize the impact of route oscillations, two other approaches are possible:

1. Require domains to coordinate among themselves for specifying policy. This coordination can allow interesting yet safe preference functions to be realized.

2. Allow domains to independently specify their policies, and deploy mechanisms to detect and suppress oscillations.

5.1. Coordination

Given global knowledge of the policies for all domains, it may be possible to analyze those policies for the likelihood of route oscillations. One or more domains could then modify their policies based on the results of this analysis.

One way of doing such an analysis may be to extend the return graph representation to more general topologies. We are considering this for future study. An alternative approach might be to simulate the effect of these policies off-line. Such a simulation would capture those oscillations that occur independent of initial conditions, e.g., Fig. 2. More extensive simulations might be necessary to capture those oscillations that depend on initial conditions, for example, the oscillations in Fig. 3.

For analysis to be possible, each domain’s policy must be available to all other domains at all times. One mechanism for making policies available is a route registry. Such a route registry currently exists in the Internet for inter-provider route co-ordination [1,3,4]. This seems to be a reasonable approach for safely realizing richer policies through hop-by-hop inter-domain routing in the Internet.

5.2. Detecting and suppressing oscillations

Global analysis detects the likelihood of oscillations a priori. It may be acceptable to allow domains to realize their policies independently and *suppress* oscillations when they occur. In order to suppress an oscillation in a cycle of domains, at least one of the domains in that cycle must modify its policies. In Fig. 2, if \mathcal{D}_0 modifies its preference function to assign a higher preference value to r_0 than r_2 , the oscillation will cease. This suggests the following general rule: when a domain detects that it is oscillating, it should assign the highest preference value to its fall-back route.

It is possible to conceive of a variety of detection schemes that indicate the likelihood of

oscillations. We describe two schemes that maintain some history of route transitions at a domain. The first scheme maintains all the r -states seen at a domain over some time period T . If this history contains a repeated sequence of routes and the domain is in a cycle of oscillating domains, then the rule described above will suppress the oscillation. To reliably detect oscillations, this scheme will need to keep a significant amount of history. Alternatively, a domain can maintain a time-decayed count of the route advertisements seen from each neighbor domain. If this “instability” count exceeds an empirically derived threshold, the domain may assume the likelihood of an oscillation. This scheme is currently deployed in the Internet [32] to suppress route advertisements caused by frequent topology changes.

The above detection schemes can generate false positives. In either scheme, the domain that maintains the history may not contribute to the oscillation, but may be sympathetically oscillating with some other domain that does. Instability counts cannot distinguish between oscillations and route advertisements caused by frequent topology changes. Therefore, it is desirable to apply our general rule to modify policies only temporarily.

Instability-based suppression [32] modifies policies temporarily. The original policies are restored after the instability count decays below another empirically derived threshold. Depending on the decay rate and the thresholds, this scheme may suppress oscillations in cases where a stable route assignment exists (for example, with next-hop-based preference functions, Fig. 3(c)). For other kinds of oscillations, such as in Fig. 2, domains do not oscillate when modified policies are in effect; but when the original policies are restored, they oscillate briefly until the instability-based suppression is re-established. This approach only reduces the impact of persistent route oscillations on the routing infrastructure.

Finally, other detection schemes are also possible. For example, in Fig. 2, \mathcal{D}_0 sees an oscillation with period $\langle r_0, r_2 \rangle$. The transition from r_2 to r_0 is a “negative transition” [24] because r_0 has a lower preference than r_2 at \mathcal{D}_0 . \mathcal{D}_0 's advertisement of r_0 causes \mathcal{D}_1 to make a positive transition. A negative

transition followed by a positive transition could be used to indicate the likelihood of an oscillation.

6. Related work

IGRP [13] is a hop-by-hop protocol in which the metric is a weighted sum of a traffic sensitive component, and a distance sensitive component. Route oscillations can occur in IGRP [18]. When a domain chooses a route with a lower traffic sensitive component, and forwards traffic along that route, that route's metric increases. Intuitively, this “traffic feedback” causes route oscillations in IGRP.

The original ARPANET link state-based SPF algorithms [16,19] used delay as a metric. Route oscillations were observed on the ARPANET when portions of the network were heavily congested. These route oscillations are also due to similar traffic feedback [5]. Several solutions to delay-based oscillations have been proposed. These solutions use approaches such as constraining route selection [16], coordination [33], and explicit routing [6].

Path vector protocols were developed [27,28] to suppress counting-to-infinity problems that occur in distance vector algorithms. These protocols use the shortest path route selection function and are therefore not susceptible to oscillations. BGP [26] and IDRP [15,23] add independent route selection to path vector algorithms. In our paper, we have shown that eliminating the monotonically increasing metric can introduce route oscillations.

Link state protocols that use hop-by-hop forwarding [14,20] and explicit route forwarding mechanisms such as Viewservers [2], the unified routing architecture [10], and map state-based protocols, such as Nimrod [7], do not exhibit persistent route oscillations. The suitability of these protocols to routing in large Internets is discussed in [22].

7. Conclusions and future work

We have shown that independent route selection can result in persistent route oscillations

in hop-by-hop inter-domain routing. We believe that only shortest path route selection is provably safe. This significantly reduces the policies that can be realized using inter-domain routing.

Given the existence of a widely deployed commercial Internet infrastructure, a combination of policy analysis, and instability-based route suppression can be used to deal with route oscillations. The former can detect most route oscillations caused by inter-dependent policies. The latter mitigates the impact, on the infrastructure, of route oscillations not detected by analysis. Explicit routing can then be used to realize desired policies (i.e., make routes available) that hop by hop routing cannot safely advertise. The explicit routing component can then complement PAIRS-based hop-by-hop routing.

Anecdotal evidence suggests that the addition of configuration mechanisms affects protocol correctness in subtle ways. The mechanisms alter some of the original assumptions that were used to prove the protocol correct. It then becomes harder to detect the weaknesses in the protocol, or once the weaknesses are identified, to evaluate possible solutions. This paper has identified problems introduced by local policy configuration mechanisms in a distance vector algorithm. In a different context, basic link state algorithms are loop-free. Yet the addition of configuration mechanisms to address scalability issues in link state protocols introduces the likelihood of loops in link state protocols [25]. We are extending the systematic methods used in this paper to identify weaknesses in other protocols, and methodologically evaluate possible solutions to those weaknesses.

Further research is necessary to develop techniques to analytically determine the existence of route oscillations in more general topologies. Future work may also focus on simulation-based methodologies to determine the existence of route oscillations. Another promising area is the investigation of protocol mechanisms for detecting oscillations.

Acknowledgements

The authors would like to thank Cengiz Alaettinoğlu, Lee Breslau, Ram Gurumoorthy, Shai Herzog, Steve Hotz, Tony Li, Bill Manning, Yakov Rekhter, and Daniel Zappala, for their suggestions and contributions, both to the problem itself, and in their review of this paper.

References

- [1] C. Alaettinoğlu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, M. Terpstra, Routing Policy Specification Language (RPSL), RFC 2622 edition, June 1999 (Obsoletes: RFC2280) (Status: Proposed Standard).
- [2] C. Alaettinoğlu, A.U. Shankar, The Viewserver hierarchy for inter-domain routing: protocols and evaluation, *IEEE Journal on Selected Areas in Communications* 13 (8) (1995) 1396–1410.
- [3] T. Bates, E. Gerich, L. Joncheray, J.M. Jouanigot, D. Karrenberg, M. Terpstra, J. Yu, Representation of IP Routing Policies in a Routing Registry, RIPE-181 edition, October 1994 (<ftp://ftp.ripe.net/ripe/docs>).
- [4] T. Bates, E. Gerich, L. Joncheray, J.M. Jounigot, D. Karrenberg, M. Terpstra, J. Yu, Representation of IP Routing Policies in a Routing Registry (ripe-81++), RFC 1786 edition, March 1995 (Status: Informational).
- [5] D.P. Bertsekas, Dynamic behavior of shortest path routing algorithms for communication networks, *IEEE Transactions on Automatic Control* AC-27 (1) (1982) 60–74.
- [6] L.M. Breslau, Adaptive source routing of real-time traffic in integrated services networks, Ph.D. thesis, University of Southern California, December 1995.
- [7] I. Castineyra, N. Chiappa, M. Steenstrup, The Nimrod Routing Architecture, RFC 1992 edition, August 1996 (Status: Informational).
- [8] C. Cheng, R. Riley, S.P.R. Kumar, J.J. Garcia Aceves, A loop-free extended Bellman–Ford routing protocol without bouncing effect, in: *Proceedings of the ACM SIGCOMM*, 1989, pp. 224–236.
- [9] D. Estrin, T. Li, Y. Rekhter, K. Varadhan, D. Zappala, Source Demand Routing: Packet Format and Forwarding Specification (Version 1), RFC 1940 edition, May 1996 (Status: Informational).
- [10] D. Estrin, Y. Rekhter, S. Hotz, Scalable inter-domain routing architecture, in: *Proceedings of the ACM SIGCOMM*, Baltimore, MD, August 1992, pp. 40–52.

- [11] P. Ford, Y. Rekhter, Explicit Routing Protocol (ERP) for IPv6, Internet Draft: SDR Working Group, January 1995, Work in progress.
- [12] J. Guckenheimer, P. Holmes, Non-linear Oscillations, Dynamic Systems, and Bifurcations of Vector Fields, Springer, New York, 1983, pp. 22–27 (Chapter 1).
- [13] C.L. Hedrick, An introduction to IGRP, Technical Report, Rutgers University, August 1991.
- [14] ISO, Information Processing Systems – Telecommunications and Information Exchange between Systems – Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for Use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473), ISO/IEC 10589 edition, 1992.
- [15] ISO, Protocol for Exchange of Inter-domain Routing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs, ISO/IEC/JTC1/SC6 IS10747 edition, 1993.
- [16] A. Khanna, J. Zinky, The revised ARPANET metric, in: Proceedings of the ACM SIGCOMM, September 1989, pp. 45–56.
- [17] L. Lamport, An assertional correctness proof of a distributed algorithm, Science of Computer Programming 2 (1982) 175–206.
- [18] S. Low, P. Varaiya, Stability of a class of dynamic routing protocols (IGRP), in: IEEE Proceedings of the INFOCOM, March 1993, vol. 2, pp. 610–616.
- [19] J. McQuillan, I. Richer, E.C. Rosen, The new routing algorithm for the ARPANET, IEEE Transactions on Communications COM-28 (5) (1980) 711–719.
- [20] J. Moy, OSPF Version 2, RFC 1583 edition, March 1994 (Obsoletes RFC1247) (Obsoleted by RFC2178) (Status: Draft Standard).
- [21] K. Ogata, Discrete-Time Control Systems, Prentice-Hall, Englewood Cliffs, NJ, 1987, pp. 351–353 (Chapter 4).
- [22] R. Perlman, Interconnections – Bridges and Routers, Addison-Wesley, Reading, MA, 1992.
- [23] Y. Rekhter, Inter-domain routing protocol (IDRP), Internetworking: Research and Experience 4 (1993) 61–80.
- [24] Y. Rekhter, Private communication, January 1996.
- [25] Y. Rekhter, S. Hotz, D. Estrin, Constraints on forming clusters with link-state hop-by-hop routing, Technical Report RC 19203 (83635) 10/6/93, IBM, IBM Research Division, T.J. Watson Research Center, YorkTown Heights, NY 10598, 1993.
- [26] Y. Rekhter, T.Li, A Border Gateway Protocol 4 (BGP-4), RFC 1771 edition (Obsoletes RFC1654) (Status: Draft Standard), March 1995.
- [27] K. Shin, M. Chen, Performance analysis of distributed routing strategies free of ping-pong-type looping, IEEE Transactions on Computers C-36 (2) (1987) 129–137.
- [28] K. Shin, M. Chen, Minimal order loop-free routing strategy, IEEE Transactions on Computers 39 (7) (1990) 870–881.
- [29] W.D. Tajibnapis, A correctness proof of a topology information maintenance protocol for a distributed computer network, Communications of the ACM 20 (7) (1977).
- [30] K. Varadhan, Protocol evaluation in the context of dynamic topologies, Ph.D. thesis, University of Southern California, August 1998 (<http://www.bell-labs.com/user/kvaradhan/doc/thesis/main.ps.gz>).
- [31] K. Varadhan, R. Govindan, D. Estrin, Persistent route oscillations in inter-domain routing, Technical Report USC CS TR 96-631, Department of Computer Science, University of Southern California, February 1996 (<ftp://usc.edu/pub/csinfo/tech-reports/papers/96-631.ps.Z>).
- [32] C. Villamizar, R. Chandra, R. Govindan, BGP Route Flap Dampening, RFC 2439 edition, November 1998 (Status: Proposed Standard).
- [33] Z. Wang, J. Crowcroft, Shortest path first with emergency exits, in: Proceedings of the ACM SIGCOMM, September 1990, pp. 166–176.



Kanna Varadhan is a member of the technical staff at Bell Laboratories, Lucent Technologies in Murray Hill, New Jersey. He has a Ph.D. in Computer Science from the University of Southern California (1998), an M.S. in Computer Science from the Ohio State University (1988), and a B.Tech. in Electrical Engineering from the Indian Institute of Technology, Madras (1986). Between his M.S. and Ph.D., he got his hands dirty for some number of years as Network Engineer for the Ohio Academic Resources Network. While at USC, he was a member of the VINT project, and a significant member of the development effort of ns-2, the widely used network simulator. Currently, he is exploring the robustness of protocol mechanisms to improve micro-mobility in wireless data networks.

Ramesh Govindan is a Project Leader of ISI's NSF-sponsored Routing Arbiter project and a Research Assistant Professor of Computer Science at the University of Southern California. Dr. Govindan received his Ph.D. (1992) and M.S. (1989) in Computer Science from the University of California at Berkeley, and his B.Tech. (1987) in Computer Science and Engineering from the Indian Institute of Technology at Madras, India. He also worked for two years at Bell Communications Research, Morristown, NJ. While at Bellcore, he was co-PI on the DARPA funded Pip project, and participated actively in the IPng standardization efforts within the IETF.



Deborah Estrin is a Professor of Computer Science at the University of Southern California in Los Angeles where she joined the faculty in 1986. Estrin received her Ph.D. (1985) and M.S. (1982) from the Massachusetts Institute of Technology and her B.S. (1980) from U.C. Berkeley. In 1987, Estrin received the National Science Foundation, Presidential Young Investigator Award for her research in network interconnection and security. Estrin is a co-PI on the DARPA Virtual Internet Testbed (VINT) project

and the NSF Routing Arbiter project at USC's Information Sciences Institute where she spends much of her time supervising doctoral student research. While she continues her research related to protocol scaling and multicast, most recently she has begun to focus on problems related to networking and coordination among very large numbers of physically-embedded devices (sensors, actuators).