Verifying the Correctness of Wide-Area Internet Routing

Nick Feamster and Hari Balakrishnan
MIT Computer Science and Artificial Intelligence Laboratory
200 Technology Square, Cambridge, MA 02139
{feamster,hari}@csail.mit.edu

Abstract

Several studies have shown that wide-area Internet routing is fragile, with failures occurring for a variety of reasons. Routing fragility is largely due to the flexible and powerful ways in which BGP can be configured to perform various tasks, which range from implementing the policies of commercial relationships to configuring backup paths. Configuring routers in an AS is like writing a distributed program, and BGP's flexible configuration and today's relatively low-level configuration languages make the process error-prone. The primary method used by operators to determine whether their complex configurations are correct is to try them out in operation.

We believe that there is a need for a systematic approach to verifying router configurations before they are deployed. This paper develops a static analysis framework for configuration checking, and uses it in the design of rcc, a "router configuration checker". rcc takes as input a set of router configurations and flags anomalies and errors, based on a set of well-defined correctness conditions. We have used rcc to check BGP configurations from 9 operational networks, testing nearly 700 real-world router configurations in the process. Every network we analyzed had configuration errors, some of which were potentially serious and had previously gone unnoticed. Our analysis framework and results also suggest ways in which BGP and configuration languages should be improved. rcc has also been downloaded by 30 network operators to date.

1. Introduction

The Internet's interdomain routing infrastructure, based on Version 4 of the Border Gateway Protocol (BGP), is complex. Network operators rely on BGP to perform many tasks, including providing reachability to customers, configuring primary and backup access links [20], configuring transit and peering relationships [36], expressing routing policies for inbound and outbound routes [3], and performing traffic engineering and balancing traffic across multiple links [18]. It is a tribute to the designers and implementors of BGP that it can meet these important practical requirements while ensuring that competing Internet Service Providers (ISPs) can cooperate to achieve global connectivity.

Unfortunately, the complexity of Internet routing in real-world operation leads to a number of fragilities. Network operators are continually reporting serious problems that adversely affect the reachability of large parts of the Internet (see Section 2.3). The mechanism that gives BGP its much needed flexibility—*BGP configuration*—is largely to blame for these problems. BGP misconfigurations are common [3, 31] and lead to problems such as hijacked, leaked, or otherwise invalid routes; routing instability [30]; routing loops [12, 16]; and persistent oscillation [1, 24, 44].

BGP's behavior depends almost entirely on its configuration. In today's commercial routers, configuration is specified in a low-level, mechanistic fashion and involves complicated feature interactions both within one router and across multiple routers in the network. As a result, the process of specifying router configurations is error-prone.

There are at least three ways to improve this state of affairs. The first is to argue that BGP4 has outlived its purpose and to develop a new routing protocol. Of course, that protocol would have to be at least as flexible as BGP4, while being less error-prone. Unfortunately, it is not immediately obvious what we should change in BGP. A second approach would be to argue that errors arise because today's configuration languages are too "low-level" and are not well-designed, leading to programming errors. Again, it is not obvious what specific improvements should be made to configuration languages. A third approach, complementary to the previous two, would be to develop a framework for *analyzing* router configurations prior to deploying them.

This paper takes the third approach and presents several contributions. First, building on previous work [16], we derive a set of correctness constraints for BGP configuration. Second, we use these constraints to design and implement **rcc** ("router configuration checker"), a tool that analyzes router configurations and detects anomalies. rcc can help network operators debug their complex BGP configurations and correct errors *before* deploying them. Third, we use rcc to find errors in real-world configurations and present the findings of our experimental analysis. This analysis helps us understand *why* various routing problems occur and determine whether each problem is due to weaknesses in BGP or due to problems in specifying configuration. Finally, given an understanding of why configuration errors occur, we recommend specific changes both to BGP and to configuration languages. As the protocols and configuration languages for wide-area routing evolve, we believe that the ability to detect and fix errors in configuration before deployment will be invaluable to network operators.

Despite the need for tools and techniques for verifying BGP's correctness, there are significant challenges that have thus far prevented verification techniques from becoming used in practice in wide-area routing. First, defining what it means for BGP to be "correct" is not easy, because it is hard to define a "specification" for an operational BGP—its many modes of operation and many tunable parameters allow for a great deal of flexibility that is hard to specify. Additionally, BGP's configuration is distributed across many routers, and precisely defining how various aspects of BGP's configuration interact is challenging. We tackle these challenges by applying the correctness properties and rules from the "routing logic" [16] to each stage of BGP's operation and derive conditions under which BGP will (1) originate incorrect routes, (2) propagate incorrect routes, (3) fail to propagate routes when it should, (4) violate intended high-level policy, and (5) exhibit nondeterministic behavior. Our approach is akin to specifying invariants that should hold at each stage of the propagation of a BGP route through the network and deriving conditions under which those invariants will be violated.

With assistance from many network operators, we ran rcc on BGP configurations from 9 real-world networks. rcc found errors in every network. In most cases, the operators were unaware of these errors, which ranged from simple "single router" errors (*e.g.*, undefined variables) to more complex errors that involved interactions between multiple routers. We uncovered many serious errors, including the potential for network partitions caused by route propagation problems, propagation of invalid routes (usually due to improper or non-existent route filtering), and routers forwarding packets in ways that were inconsistent with high-level policy.

Our experimental analysis suggests that there are many reasons for configuration errors, but three reasons explain most errors. First, many errors arise from the complex, obscure mechanisms for propagating routes learned from BGP border routers within a network. The main techniques used to propagate these routes scalably within a network ("route reflection with clusters") are easily misconfigured. Second, even simple policy specifications (*e.g.*, treating a route as a backup) are specified using multiple levels of indirection in configuration files, making mistakes more likely. Finally, many errors reflect the fact that operators have no systematic process for configuring their networks; many of the errors we discover could be fixed with better configuration management tools.

The rest of this paper is organized as follows. Section 2 provides background on BGP and further motivation. Section 3 presents a framework for verifying BGP's correctness using static analysis. Section 4 describes the design and implementation of rcc, and Section 5 presents our experience with running rcc on operational networks. Section 6 surveys related work on protocol verification, and Section 7 concludes.

2. Motivation and Background

In this section, we explain why verifying BGP configuration is difficult and present an overview of BGP, paying particular attention to aspects of BGP that can cause incorrectness. We then discuss some empirical observations of routing errors from a previous study and our study of the North American Network Operators Group (NANOG) mailing list.

2.1 Challenges for BGP Verification

BGP configuration errors are difficult to diagnose because they often appear far from the actual source of the error and usually not immediately after the deployment of an erroneous configuration. Operators typically search for errors in a trial-and-error fashion because they have no way to systematically verify that a particular routing configuration is correct. They deploy a configuration, wait and see whether the configuration has any undesirable effects, and revert the configuration to a previous state if problems arise.

There are two further problems with verifying correctness:

1. *Defining correctness is difficult.* Operators configure BGP to achieve a variety of tasks. Defining "correctness" for a protocol that can behave in many ways is difficult. We use the *routing logic* [16] to help us define correctness properties and constraints.
2. *BGP configuration is distributed.* BGP's behavior depends on the interactions between multiple routers, both within a single network and across multiple networks. Verifying BGP configuration requires checking many such dependencies involving multiple routers.

We address these challenges by viewing BGP as a distributed program, rather than a protocol that conforms to a set specification.[1] We determine a set of invariants that should hold true independent of the details of configuration. Many of these invariants are verifiable using static analysis.

We determine which invariants require checking configurations at a single router, and which require checks across multiple routers. Fortunately, we find that most serious problems can be uncovered with checks that only require configurations from routers *within a single network*.[2] This finding allows each network to verify its configurations independently.

2.2 Border Gateway Protocol (BGP)

The Internet comprises over 17,000 independently operated networks, or autonomous systems (ASes), that exchange reachability information using the Border Gateway Protocol (BGP) [39]. BGP has three distinguishing components: (1) the *protocol* itself, which defines how routes are exchanged and the content of those routes; (2) the *decision process*, which defines how each router selects a best route from multiple options; and (3) the *protocol configuration*, which affects route attributes, as well as how (and whether) routes are exchanged within the AS and with other ASes.

2.2.1 Overview

BGP distributes routes to destination prefixes via incremental updates. Each router selects one "best" route to a destination, readvertises that route to neighboring routers, and sends updates to its neighbors when the best route changes. BGP is actually two distinct protocols: *external BGP (eBGP)*, which an AS uses to exchange routes with other ASes, and *internal BGP (iBGP)*, which an AS uses to distribute routes from other ASes to routers within its own AS.

Each BGP routing message contains a number of attributes. These include the *destination prefix* (or prefixes), the destinations associated with the route; the *AS path*, the sequence of ASes en route to the advertised destination; the *next-hop*, the IP address that the router should forward packets to in order to use the route; the *multi-exit discriminator (MED)*, which a neighboring AS can attach to a route to specify that one route should be preferred more (or less) than routes advertised by that AS at other routers; and the *community* value, which carries no explicit semantics, but is a way of labeling a route (*e.g.*, a router might label an incoming customer route so that when the route is readvertised internally, other routers can determine that it was learned from a customer).

2.2.2 BGP Operation and Configuration

BGP's protocol definition determines how routers select and propagate routes, but BGP's *configuration* affects (1) what (and whether) routes are originated and propagated, (2) how routes are modified as they propagate (which, in turn, affects route selection), and (3) how routes propagate between routers (*i.e.*, the routing topology).

We break BGP route propagation into a sequence of six steps, shown below. At each step, we note how BGP's configuration affects its behavior. We will refer to these steps throughout the paper.

[1] We explain in Section 6 why the absence of a set specification makes it infeasible to verify wide-area routing using approaches like model checking.
[2] Griffin and Wilfong's "safety" violation caused by incompatible policies in different networks is an exception [27]; see Section 3.6.

Property	Step	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	Total
Filtering	2	1 (1)	3 (4)	4 (9)	21 (27)	7 (14)	6 (9)	10 (23)	14 (27)	7 (19)	18 (29)	2 (2)	93 (164)
Leaked Routes	2	0 (0)	0 (0)	3 (3)	10 (11)	6 (7)	4 (4)	9 (9)	13 (13)	8 (9)	8 (8)	1 (1)	62 (65)
Hijacked Routes	2	0 (0)	1 (1)	1 (1)	3 (3)	5 (5)	4 (4)	1 (1)	5 (5)	2 (2)	1 (2)	0 (0)	23 (24)
Global Route Visibility	3	1 (1)	6 (7)	12 (17)	22 (29)	10 (14)	9 (12)	13 (16)	17 (30)	18 (27)	29 (37)	2 (2)	139 (192)
Oscillations	4	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (2)	0 (1)	0 (1)	0 (0)	0 (4)
Routing Instability	4	0 (1)	4 (6)	4 (4)	20 (22)	4 (5)	6 (7)	3 (4)	16 (22)	11 (12)	4 (5)	1 (1)	73 (89)
Attribute manip.	5	0 (0)	0 (0)	2 (2)	6 (7)	6 (9)	5 (5)	1 (1)	4 (11)	2 (6)	4 (8)	0 (0)	30 (49)
iBGP-related	6	0 (0)	2 (3)	4 (4)	9 (9)	4 (6)	2 (5)	3 (4)	5 (9)	6 (9)	1 (3)	0 (0)	36 (52)
Routing Loops	6	0 (0)	2 (2)	1 (1)	4 (4)	1 (1)	3 (3)	1 (1)	7 (8)	4 (4)	0 (2)	0 (0)	23 (26)
Blackholes	6	0 (0)	0 (0)	2 (2)	6 (6)	4 (4)	1 (1)	16 (16)	21 (23)	31 (32)	26 (26)	1 (1)	108 (111)
Total	—	2 (3)	18 (23)	33 (43)	101 (118)	47 (65)	40 (50)	57 (75)	102 (150)	89 (121)	91 (121)	7 (7)	587 (776)

Table 2: Number of threads discussing observed BGP-related routing errors over the 10 years of the NANOG mailing list. Numbers in parentheses denote the total number of threads on the topic, including generic discussion (*e.g.*, questions, anecdotes, documents, etc.).

.

1. Highest local preference
2. Shortest AS path length
3. Lowest MED (compared among routes with same next-hop AS only)
4. eBGP-learned over iBGP-learned
5. Lowest path cost to next hop (based on the interior routing protocol path)
6. Lowest router-id of BGP speaker (or oldest route)

Table 1: The BGP decision process used in step 4 (path selection). Obsolete steps (*e.g.*, "lowest origin type") are omitted.

1. **Origination.** A BGP-speaking router in some AS *originates* a route for a prefix. BGP configuration allows a router to advertise itself as the source AS for a prefix.[3]
2. **Export.** A router *exports* the route to a neighboring AS on an *eBGP* session. Each BGP session has an *export policy* that determines which routes are advertised to each BGP neighbor. A network operator uses export policy to control route propagation. For example, an AS should typically not advertise routes from one of its providers to another, because doing so would cause the network to transit packets between its two providers. Operators use export policy to implement bilateral agreements with other ASes.
3. **Import.** An eBGP-speaking router *imports* the route. At this stage, the router filters any routes it believes to be invalid. The router also discards routes whose next-hop attribute refers to an IP address not present in the router's forwarding table. Finally, the import policy may assign a *local preference* to the imported route.
4. **Selection.** The router *selects* the best route from the set of routes to the destination. The BGP decision process shown in Table 1 is a *de facto* standard that each vendor defines to behave in roughly the same way [8, 40]. The router applies the steps shown in the table to determine its best route to each destination. By altering the attributes of advertised routes in the router configuration, a network operator can exert great influence on an importing router's path choice. For example, the first step in the decision process is based on local preference, which operators can manipulate using configuration.
5. **Modification.** The router consults its configuration and *modifies* one or more of the best route's attributes (such as the "next-hop" field) before disseminating the best route to the other routers in its AS.
6. **Intra-AS Propagation.** The routers inside the AS use *iBGP* to propagate externally-learned routes, with each router selecting the best route among a set of choices. iBGP runs as an overlay over the interior routing protocol (the interior gateway protocol, or *"IGP"*). The traditional overlay topology is a "full-mesh", but this approach does not scale well. A common technique that improves the scalability of intra-AS route propagation is *route reflection*. The operator specifies the iBGP topology with route reflection in router configurations. This topology usually requires a smaller number of per-router BGP sessions than the full-mesh.

[3]In rare cases, non-BGP speaking routers can also originate routes.

At the end of Step 6, all eBGP-speaking routers proceed to Step 2 (export) and check if their best route to the destination should be exported to any other ASes.

2.3 Empirical Observations

In this section, we first discuss a previous BGP misconfiguration study that discovered that various types of BGP misconfigurations occur daily and explain how this work motivates our study. We then survey the archives of the NANOG mailing list, where operators have been discussing various network operations issues; our study, though informal, suggests that configuration has been causing problems *at every step* of BGP's route propagation over the past decade.

2.3.1 Transient misconfigurations

Mahajan *et al.* studied short-lived BGP misconfiguration by analyzing transient, globally-visible BGP announcements from an edge network [31]. They defined a "misconfiguration" as a transient route event that was followed by a return to normal behavior within a small amount of time (suggesting that the operator observed and fixed the problem). They found that many misconfiguration events could be traced to faulty route origination and incorrect filtering.

Mahajan *et al.*'s results suggest that operators commonly rely on "stimulus-response" reasoning. We extend their work by studying a complementary class of errors: those that are difficult to quickly locate and correct. Our work also helps operators detect the types of misconfigurations found by Mahajan *et al.* without stimulus-response testing.

2.3.2 Operator-observed problems

NANOG operates a mailing list where operators discuss observed problems [35]. The list has thousands of subscribers, and network operators regularly use the list to report network problems, ask questions regarding operational issues, etc.

To gain a better understanding for the types of errors that operators see in practice, we conducted a study of the list archives, which begin in mid-1994. Because the list has received about 68,000 emails over the course of 10 years, we first clustered the emails by thread and pruned threads based on a list of about 15 keywords (*e.g.*, "BGP", "issue", "loop", "problem", "outage", etc.). We then manually reviewed 1,500 *threads*, 491 of which were relevant to BGP configuration issues. We classified each of these 491 threads into one or more of the categories shown in Table 2. (The majority of the remaining threads pertaining to outages and problems were related to power outages and fiber cuts and not to BGP.) The categories can be classified according to the stages of BGP route propagation as follows:

- **Origination/Export.** *Prefix filtering* describes any problem related to misconfigured filters (either blocking prefixes that shouldn't be blocked or leaking prefixes, such as private addresses, that should be filtered). *Leaked routes* describes any thread relating to routes in the global routing table that should have been filtered; this includes leaked prefixes, as more general problems (*e.g.*, an AS leaking its provider's routes, etc.). *Hijacked routes* refers specifically to cases where an AS announces address space that belongs to another AS.
- **Import.** *Global route visibility* refers to cases where some prefix that should be globally visible is not. From the list, we observed that problems with global route visibility result when a routing registry allocates IP addresses that were previously private or reserved, and various ISPs fail to update their import filters.
- **Selection.** *Oscillations* refers to any thread related to persistent route oscillation. *Routing instability* describes any problem that involves routes being advertised and withdrawn rapidly.
- **Modification.** *Attribute manipulation* refers to any problems related to modification of routing attributes. Typically, these involve problems where the operator complains a router in a neighboring AS is not modifying a route attribute, such as local preference, as advertised.
- **Intra-AS Propagation.** *Routing loops* are specific cases where an operator posts a traceroute that contains a routing loop. *Blackholes* are specific cases where an operator either posts a traceroute that does not loop, but fails in the middle of the path, or where an operator specifically complains about such a blackhole. *iBGP-related* refers to any problems that seemed as though they were likely related to iBGP configuration (*i.e.*, loops and other types of intra-AS failures).

Although this analysis is informal, our observations suggest some clear trends. First, many interdomain routing problems are caused by configuration errors. Second, the state of affairs has not improved over the last 10 years: the same types of configuration errors and problems continually appear. Third, BGP configuration problems exist at every step of BGP's operation. Guaranteeing correct operation thus requires more than a few simple tweaks: network operators need a systematic framework for verifying BGP's correctness.

3. Verifying BGP Configuration Correctness

In this section, we describe our approach to verifying BGP configuration correctness. Using the routing logic and our model of BGP route propagation from Section 2.2.2 as a guide, we derive a set of correctness constraints for BGP configuration. We also determine which constraints can be analyzed by *static analysis*—parsing the set of router configurations from a single AS and applying the correctness constraints to reveal errors. Our analysis in this section provides the theoretical groundwork for the design of rcc, but it also presents several important insights on BGP configuration correctness. We identify aspects of correctness that cannot be detected using static analysis alone but require either runtime analysis or changes to either BGP or the configuration language.

Step	*Valid.*	*Visib.*	*Info Flow.*	*Det.*	*Safety*
1. Origination	•				
2. Export	•		•		
3. Import	•	•	•		
4. Selection				•	•
5. Modification	•		•		
6. Intra-AS Prop.	•	•		•	

Table 3: Applying correctness tests at each stage of BGP operation.

3.1 Framework Overview and Assumptions

Any verification system requires a notion of "correctness." For wide-area Internet routing, as for any complex distributed program, specifying correctness properties is difficult. We build on the *routing logic* [16], which specifies five properties that routing protocols should attempt to satisfy. The routing logic defines five correctness properties:

1. *Validity* is the property that the existence of a route to a destination implies that there is a path based on that route in the network. This property guarantees that data traffic flows correctly.
2. *Visibility* is the property that the existence of some path to a destination implies the existence of a route to it. This property guarantees that routes propagate correctly.
3. *Information flow control* is the property that the flow of routes into, out of, and across an AS conforms to a specified policy. Information flow control is a unique property because it requires a policy model.
4. *Determinism* is the property that the best route selected by a router to a destination is (1) independent of the order in which the routes arrived, and (2) is unchanged if any suboptimal route is removed from consideration.
5. *Safety* is the property that the routing protocol arrives at a unique, stable route assignment. Most issues with safety involve policy interactions between ASes.

We motivate our verification framework by considering the steps involved in propagating a route from an originating AS through a sequence of routers and ASes (described in Section 2.2.2). At the end of each step, we can use the routing logic to determine whether the each of the above properties holds. In many cases, static analysis of router configurations can determine whether each of these properties holds. Table 3 summarizes which properties apply at each stage of BGP operation. The next five sections discuss each correctness property in terms of its effects on each stage of route propagation. Our framework is *extensible*, allowing users (*e.g.*, network operators) to easily add specific assertions or invariants that ought to hold in their configurations.

We assume that paths and routing protocols at lower layers are operating correctly. For example, we assume that if a destination is contained in the IGP, then routers will correctly forward packets to that destination. The correctness of protocols at lower layers should be verified separately.

3.2 Validity

If a routing protocol allows invalid routes to propagate, then packets may not always reach their intended destinations. BGP can violate validity in several different ways, most stemming from misconfiguration. In general, a router should only advertise a route if it is believes that there is a correct path from itself to the destination that can be used by any other router that hears the advertised router. To help achieve this goal, a BGP router only advertises a route if there is a corresponding entry in its forwarding table. However, this check alone is insufficient to ensure validity, because many other things can go wrong at each step of operation.

3.2.1 Incorrect origin AS (Step 1)

Misconfiguration can cause valid routes to be advertised from the wrong origin AS, rendering the advertised routes invalid. For example, in 1997, a misconfigured router in a small AS ("AS 7007") advertised itself as the origin AS for all destinations in the Internet, causing traffic for all Internet destinations to be routed through that AS [13].

The reason for the "AS 7007" problem was a router configuration feature used to redistribute IGP-learned routes into BGP and vice versa. The redistribution feature is used in networks where not all routers inside an AS support BGP, where the network operator has to rely on the IGP, rather than iBGP, to propagate eBGP-learned routes inside the AS (Step 6). The problem is that an erroneous configuration can cause all *IGP* routes to be redistributed into eBGP, in which case the externally-learned routes would appear as if they *originated* in the AS doing the IGP-to-eBGP redistribution. As the "AS 7007" problem showed, the result of such a misconfiguration is catastrophic. Static analysis can prevent this problem by analyzing the configuration and ensuring that eBGP-learned routes that are redistributed into the IGP are never redistributed back into eBGP.

3.2.2 Incorrect or missing filters (Steps 2 and 3)

A BGP router may export invalid routes for two reasons. The first is when the router detects a path failure to some prefix, but does not withdraw that prefix, preferring to maintain the aggregated route to which that prefix belongs in each BGP session. The second reason is due to errors in export filters (Step 2) and import filters (Step 3), which cause problems even when no failures occur.

The Team Cymru "bogon" project [10] regularly updates a list of prefixes from private, reserved, or unassigned IP addresses that should be filtered. Static analysis can determine whether import and export filters are up-to-date and being applied appropriately. Although bogon lists defend reasonably well against potentially incorrect routes, they present several problems. First, since new prefix blocks are continually being allocated, filters must be updated to ensure that an outdated bogon list does not filter routes for valid prefixes. Second, while filtering bogon prefixes can be a useful sanity check, it is not a panacea; a network can always advertise prefixes that have been legitimately allocated that it does not own, and filters based on bogon lists will not defend against such mishaps.

In general, checking that route origination is valid requires information about what prefixes that AS is authorized to originate. Ideally, checks for the validity of route advertisements should be built into the routing protocol itself. Although static analysis can prevent a network operator from mistakenly leaking routes and ensure that import and export filters are kept up-to-date, the inadequacy of filters, the subtleties of route origination, and the fact that invalid route origination is sometimes caused by malice all suggest that a protocol modification [29, 42] is the best way to ensure the validity of a route at Step 3.

3.2.3 Incorrect ASPATH attribute (Step 2)

BGP configuration allows a router to prepend an arbitrary AS number to the AS path attribute of a route before readvertising that route to other routers in Step 2. Network operators typically use

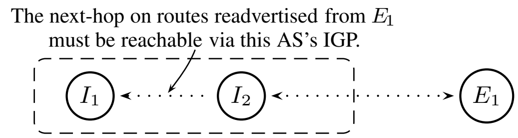

Figure 1: Valid iBGP routes require next-hop reachability.

AS path prepending to make one AS path look longer than another (if a path has more AS hops, other routers are less likely to choose that route). If a router configuration prepends an AS that is not its own, however, the AS path technically violates the validity property. Static analysis can verify that a router is not prepending arbitrary AS numbers to an AS path by analyzing the router configuration file and examining the AS numbers that the router prepends in its export policies.

3.2.4 Incorrect next-hop attribute (Step 5)

BGP configuration can also violate validity if the router incorrectly modifies (or fails to modify) the route's next-hop attribute in Step 5. Each BGP route contains a next-hop attribute that tells a router the IP address to forward packets to in order to use that route. BGP allows routers to either readvertise routes with the next-hop attribute unchanged or with the next-hop attribute as its own address. Most commonly, an eBGP-speaking router will set the next-hop to its "loopback address"[4] to ensure that the next-hop is reachable via the IGP, but there are good reasons not to do so [5]. If an eBGP-speaking router doesn't set the next-hop to itself, the next-hop from the neighboring AS's end of the eBGP session must be advertised via the IGP. For example, in Figure 1, when I_2 readvertises to I_1 routes that it learns from E_1, it must set the next-hop attribute for those routes to itself; if not, E_1's address must be reachable via the IGP.

Static analysis can verify the reachability of iBGP-advertised routes by determining how each router sets the next-hop attribute. If the next-hop is not set for an eBGP-speaking router, static analysis can check the IGP configuration to ensure that the IGP advertises a route for the next-hop associated with the opposite end of the BGP session.

3.2.5 Intra-AS inconsistencies (Step 6)

Every iBGP-speaking router selects a best route to each global destination. Each of these routes has a next-hop attribute; to use a route, a router forwards packets to the next-hop along the shortest *IGP* path. If iBGP-speaking routers along that IGP path have not selected the same best route to the destination (*i.e.*, they forward packets for that destination to a different next-hop), then packets will not follow their intended path, and forwarding loops can result. Previous work has explored these possibilities [12] and derived sufficient conditions for ensuring that these inconsistencies do not occur [16, 25]. Although we have not yet incorporated shortest IGP path computations into our implementation of rcc, static analysis of router configuration can derive the shortest IGP paths to each IGP destination (*i.e.*, anything that might be a BGP next-hop) and check these sufficient conditions.

3.3 Visibility

Visibility is an important property because it ensures that the network is not partitioned. Additionally, it ensures that each router can select its preferred route from all available choices. Visibility can be violated if routers fail to install valid routes or if an AS's iBGP

[4] A router's loopback address is the IP address for that router that can be reached via any interface. It is an IP address on the router that is always up.

topology is misconfigured. In this section, we discuss each of these problems.

3.3.1 Failure to install valid routes (Step 3)

Visibility can fail if a router hears a valid route to a destination but fails to install it. This event can occur as a result of a misconfigured filter, but it can also result from other subtleties. If a network has a router that does not participate in iBGP between two iBGP-speaking peers, then that router must obtain global routes some other way, such as from the IGP. To account for this, iBGP has an option known as "synchronization", which prevents the router from installing a route into its BGP table if a route for that prefix does not appear in the IGP. Some router vendors enable synchronization by default, which will commonly prevent routes from being installed into iBGP, since ASes almost never insert the global routing table into the IGP. Seeing a router with synchronization enabled is cause for suspicion.

3.3.2 iBGP signaling (Step 6)

At the intra-AS propagation step, routers inside an AS use iBGP to disseminate externally-learned routes to other routers in the AS. The traditional approach is to configure iBGP sessions between every pair of BGP-speaking routers within the AS, obtaining a "full-mesh" iBGP overlay topology. In this setup, each router selects a best route from among its available choices and, if that route was learned via eBGP, readvertises that route to every other BGP-speaking router.

Unfortunately, a full-mesh topology does not scale well. A commonly used method to improve scalability is *route reflection*. A subset of the routers are configured as *route reflectors*, with the configuration specifying a set of other routers as *route reflector clients*. A route reflector client may be served by multiple route reflectors for redundancy, in which case redundant reflectors must share all of their clients and belong to a single "cluster". A route reflector may itself be a client of another route reflector. Any router may also have other iBGP sessions with other routers.

With route reflectors, route dissemination is slightly different than the full-mesh case [2]. Each route reflector readvertises its best route according to the following guidelines: (1) if the best route was learned from an iBGP peer, the route is readvertised to all route reflector clients; (2) if it was learned from a client or via an eBGP session, the route is readvertised on all iBGP sessions. Route reflector clients readvertise routes as usual: that is, they will only readvertise their best route on an iBGP session if the route was learned via eBGP. For non-client iBGP sessions, a router only readvertises eBGP-learned routes. It does *not* readvertise iBGP-learned routes to any non-client router, because such routes should have already been learned from a direct session with the router that first learned of the external route or from the appropriate route reflector(s).

Reflector-client relationships among routers in an AS define a *directed graph* in which each router is a node and each client-reflector session is represented with a directed edge. Each iBGP session that is not a reflector-client session is represented with an undirected edge. For various reasons, including to prevent routing information loops, the subgraph of this graph induced by the directed edges should by acyclic.

If misconfigured, even a connected directed acyclic graph of iBGP sessions can lead to visibility violations. For example, in Figure 2, routers Y and Z do not learn route r_1 to destination d (learned via eBGP by router W), because X will not readvertise routes learned from its iBGP session with W to other iBGP sessions. This is a visibility violation that we call an *iBGP signaling partition*; a path exists, but neither Y nor Z has a route for it.

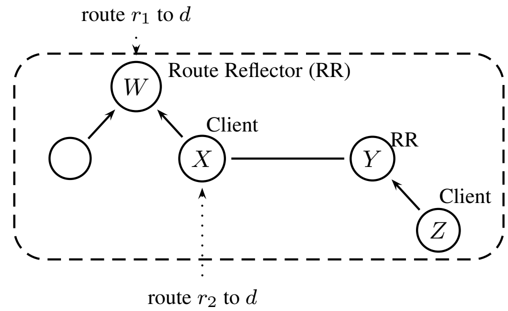

Figure 2: iBGP can have subtle visibility violations. If the iBGP sessions are configured as above, route r_2 will be distributed correctly to all the routers in the AS, but r_1 will not. Y and Z will not learn of r_1, leading to a network partition that won't be resolved unless another route to the destination appears from elsewhere in the AS.

We aim to check the static iBGP configurations of the routers in an AS to determine whether visibility violations are possible. More specifically, we want to determine if there is *any* combination of eBGP-learned routes such that at least one router in the AS will not learn at least one route to the destination. The following result provides the basis for a simple and efficient check.

THEOREM 3.1. *Suppose the iBGP reflector-client relationship graph contains no cycles. Then, the AS's configuration satisfies visibility if, and only if, the set of routers that are not route reflector clients forms a full mesh.*

PROOF. Call the set of routers that are not reflector clients the "top-layer" of the iBGP graph. If the top-layer is not a full mesh, then there are two routers X and Y with no iBGP session between them, such that no route learned using eBGP at X will ever be disseminated to Y. The reason is that no router re-advertises an iBGP-learned route.

If the top layer is a full mesh, observe that *if* a route reflector has a route to the destination, then all its clients are guaranteed to have a route as well. This implies that as long as every router in the top-layer has a route, all routers in the AS will have a route. If any router in the top-layer learns a route through eBGP, then all the top-layer routers will hear of the route (because the top-layer is a full-mesh). Alternatively, if no router at the top-layer hears an eBGP-learned route, but some other router in the AS does, then that route propagates up a chain of route reflectors (each client sends it to its reflector, and the reflector sends it on all its iBGP sessions) to the top-layer, and from there to all the other top-layer routers and to the other routers in the AS. ∎

The visibility test is now straightforward and requires only the router configurations:

1. Parse the router configurations to construct the iBGP graph of reflector-client and non-client relationships.
2. Check if the directed graph is acyclic. If it is not, then flag a cycle and report it as an anomaly. If it is acyclic, determine the "top-layer" of nodes that are not route reflector clients. There must be at least one such node.
3. Verify that the top-layer is a full-mesh. If not, report the missing iBGP sessions.

In addition to the above iBGP graph checks, we need to perform checks to confirm that the individual iBGP sessions are properly configured so they can correctly exchange routes. Sometimes, one

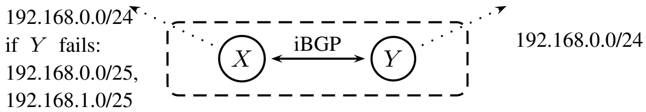

Figure 3: Conditional BGP advertisements are an example of a positive information flow control requirement. For example, an operator can specify that if the iBGP session between X and Y fails, that X should send announcements for more specific prefixes to its upstream.

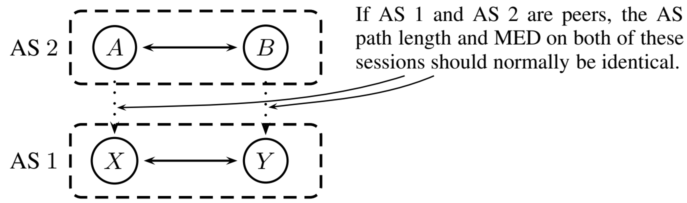

Figure 4: Peering agreements usually require consistent export (same AS path length and MED) across all peering points.

router may be configured to have an iBGP session with another, but the latter's configuration does not have such a session. Alternatively, both routers may have iBGP sessions with each other, but a mismatch in router options (*e.g.*, if one end has the MD5 password option enabled, but the other does not) may prevent the session from being established. This causes the iBGP session to be *incomplete*. If two routers in the same AS have the same loopback address, visibility can be violated if one of those routers hears a route from the other, and discards it, thinking it advertised the route itself. Although these problems are seemingly "trivial", we have found them to be surprisingly common.

An iBGP session can also become incomplete if it is established to an *interface* address of the router, rather than the loopback address. If a router establishes an iBGP session with a router's loopback address, then the iBGP session will remain active as long as that router is reachable via *any* IGP path between the two routers. If a router establishes an iBGP session with an interface address of another router, however, the iBGP session will go down if that interface fails, even if an IGP path exists between those routers. It is straightforward to check router configurations to verify that all these properties hold.

Although static analysis can ensure that iBGP configuration satisfies various correctness constraints, we believe that the configuration problems associated with iBGP mainly reflect the need for a better intra-AS route propagation protocol. The job of iBGP is fairly simple: consistent propagation of global routing information within the AS. There is no reason why intra-AS routing must behave like eBGP at all, and, in fact, a redesigned protocol that *enforces* consistency and visibility would likely be less prone to misconfiguration. We are exploring these possibilities as part of our ongoing work.

Theorem 3.1 suggests that the "cluster" mechanism for route reflection adds an additional layer of complexity to iBGP (and, hence, another potential source of misconfiguration) without necessarily improving redundancy. Operators should achieve redundancy by ensuring that the top-level of the route reflector hierarchy always maintains a full mesh.

3.4 Information-flow Control

Information flow control checks whether a network is propagating routes as it should. This property is important because route propagation determines traffic flow. If an AS is not being paid to transit traffic between two of its peers, it should ensure that its route advertisements do not cause it to do so. Similarly, if an AS purchases an expensive backup link, its route advertisements should ensure that this link is only preferred when the inexpensive primary link has failed. Information flow control concerns three aspects of route propagation:

1. Negative requirements. No route should be readvertised to an AS where policy specifies that it should not be advertised. For example, an operator may wish to implement a policy that prevents the routes from one of its neighbors from reaching another. Network operators commonly implement such policies to ensure they are not carrying traffic between two ASes with which they have a peering relationship.

2. Positive requirements. A policy may specify that a route *must* be readvertised to a certain AS or router. For example, an operator might specify that, if one router does not learn a certain route on one session, then it should advertise a more specific prefix on another. An operator might use such a policy to implement backup, as shown in Figure 3. Additionally, an AS may want to specify a policy that all routes to a neighboring AS be exported with identical route attributes.

3. Attribute manipulation requirements. Networks often allow neighboring ASes to use community values to specify how certain route attributes should be manipulated. For example, an AS might indicate to its neighbor that a certain route is a backup route and that it should therefore assign that route a lower local preference.

Information flow control requirements should be expressed in terms of a *policy model* that can be verified against the configuration. One possibility for expressing negative requirements in a policy model, as suggested in previous work [16], is Denning's lattice model [11]. Although this model expresses negative requirements quite well, it does not suggest how to specify positive requirements or attribute manipulation requirements. The best way to express these requirements remains an open question, but it is important: designing a concise way to express information flow requirements is a necessary step for designing higher-level configuration languages.

Verifying information flow control is difficult because it requires a policy model to verify against, which we don't have today. The best a configuration checking tool can do from the configurations alone is to infer what the policy model should be or make reasonable assumptions about standard practices. Ideally, a configuration checker would check the low-level BGP configuration against some high-level specification of operator intent, but even without this information, we can make some reasonable assumptions about information flow policies, which we describe in the rest of this section.

3.4.1 Controlled export (Step 2)

An AS's customers will sometimes advertise smaller prefixes to its upstream AS to load balance its inbound traffic, but it will tag those prefixes with an instruction to its upstream to not readvertise these prefixes [7]. The export policies on an AS's routers should always ensure that such a route is not readvertised to any neighbors. Network operators also control the export of routes between their peers and providers. Static analysis can check this property by analyzing the configuration files to verify that every eBGP session has a filter that matches routes that have this instruction and prevents them from being readvertised to other ASes.

3.4.2 Consistent export (Steps 2 and 5)

Barring unusual contractual arrangements, an AS should advertise paths with equally good attributes to each peer at every peering

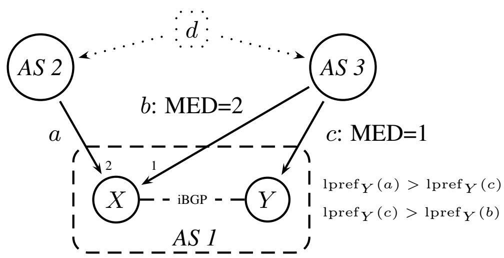

Figure 5: Nondeterminism causes oscillation. Adapted from [16]. The small numbers near router X indicate the router ID for each session.

Step	*Description*
1	X learns routes (a, b) to d and prefers b (router ID).
2	X advertises b to Y.
3	Y learns routes (b, c) to d and prefers c (local preference).
4	Y advertises c to X.
5	X learns routes (a, b, c) to d and prefers a (MED, eBGP>iBGP).
6	X advertises a to Y (implicit withdrawal of b from Y).
7	Y learns routes (a, c) to d and prefers a (local preference).
8	Y withdraws c from X. Return to Step 1.

Figure 6: Steps in the BGP oscillation from Figure 5.

point. That is, an AS should not make one of its peering points look worse from the perspective of its peer. For example, in Figure 4, if ASes 1 and 2 are peers, then the export policies of the routers A and B should export routes to X and Y, respectively, that have equal AS path length and MED values. If not, router X could be forced to send traffic to AS 1 via router Y ("cold potato" routing), which typically violates peering agreements.

Checking for consistent export with static analysis involves comparing export policies on every router that has an eBGP session with a particular peer. In practice, this comparison is not straightforward because differences in policy definitions are difficult to detect by visual inspection; we discuss this problem in more detail in Section 4.2. Static analysis is particularly helpful because it can perform efficient consistency comparisons of policies on many different routers, which is intractable when done manually.

3.4.3 *Consistent import (Steps 3 and 5)*

Similarly, an AS may wish to arrange its import policies so that all routes received from a peer look equally good up to the IGP tiebreak step, thus allowing it to use nearest exit ("hot potato") routing with its peers. Of course, an AS cannot ensure that it receives AS paths of equal length at all peering points with its peer, but it can take precautions such as resetting the MED value on import and ensuring that the local preference value is the same everywhere. Static analysis can can also detect inconsistent import policies in the same way that it detects inconsistent export policies.

Although static analysis can help ensure that BGP configuration does not violate these types information flow control constraints, static analysis of existing configuration languages is a band-aid solution. First, information flow control requirements are currently specified by *mechanism*—for example, tagging routes with communities to affect how an import or export policy modifies route attributes and whether a route is filtered, specifying high-level policies in terms of low-level regular expressions, etc. This makes policy configuration unintuitive. Second, because checking information-flow control requirements require a policy model, it would make more sense for these types of requirements to be specified in terms of a high-level language.

3.5 Determinism

Determinism is an important property because a routing protocol can experience persistent oscillation if it is violated; determinism also makes BGP easier to model and debug.

3.5.1 *Timing dependencies (Step 4)*

If BGP-speaking routers select a best route to a destination from a set of routes independently of the order in which the routes in that set arrive, then protocol modeling and debugging becomes easier. For example, network operators can correctly emulate the BGP route selection process without having to simulate the order in which messages are passed between routers in an AS [18]; this makes common network tasks, such as offline optimization for traffic engineering, more tenable.

Previous work has observed that enabling two simple configuration options can achieve timing independence [17]. First, routers should compare any route that arrives for a destination with *all* available routes to a destination, rather than just the best route at that time [9]. Second, the final tiebreaking step in route selection should be based on the router ID value associated with each BGP session, rather than on the route that was learned first. These two configuration options ensure that the route that a router selects is independent of the arrival order of the routes in that set and does not affect the best route selection [17]. Static analysis can easily verify that the appropriate configuration options are enabled on each router by analyzing each router's configuration independently.

3.5.2 *No total ordering (Steps 4 and 6)*

The MED attribute allows an AS that exports routes to its neighbor to specify hints about the exit points it would prefer that neighboring AS select (at Step 4). MED adds the following two features to BGP; the first is fundamental, and the second is a positive side effect:

- *Exporting AS can control exit point from importing AS.* MED allows an AS to specify which exit a neighboring AS should prefer when sending packets to it.

- *Importing AS can respect MED hints from a neighboring AS but still express next-hop preferences and use nearest available exit.* The interaction between MED and the decision process allows an AS to prefer one AS over another but choose the nearest available exit if no route to the preferred AS is available. For example, in Figure 5, AS 1 can prefer routes via AS 3 unless there is a closer exit point via AS 2.

The interaction of MED with the decision process can cause persistent oscillation, as shown in Figure 5. When routes only routes a and b are advertised to router X, then X prefers route a (because the routes are equally good up to the router ID tiebreak, but a has a lower router ID). However, when X learns routes a, b, and c, then it prefers route b. This oscillation occurs because BGP violates *set monotonicity*: router X's preference of b over a reverses in the presence of c.

Static analysis is of no use in this case. Interactions between steps of the decision process cause set monotonicity to be violated. We should thus explore *protocol* and *decision process* modifications to BGP that satisfy determinism (*i.e.*, avoid persistent route oscillation) but still allow an exporting AS to provide hints about exit points to an importing AS.

By definition, if each router has a ranking over the set of all possible routes, then set monotonicity is satisfied. Every router in an AS could achieve a ranking over all received routes either by ignoring MEDs, selecting a best route prior to the step in the decision process that involves MED or by allowing MED to be compared

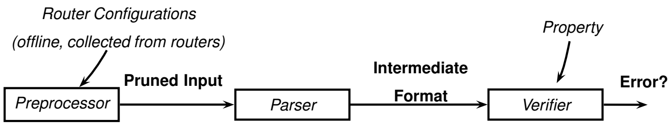

Figure 7: High-level design of rcc.

across all available routes. Unfortunately, these policy restrictions limit expressiveness: they do not allow a network operator to honor a neighboring network's request to perform cold potato routing.

One possibility is to modify the configuration language to require that each router express a fixed ranking over all eBGP *sessions*. We conjecture that an intra-AS routing protocol that assigns paths based on these rankings can prevent MED-induced routing oscillation without significantly limiting the configuration's expressiveness. We are exploring this possibility as part of our ongoing work.

3.6 Safety

Safety was originally defined in previous work on the Stable Paths Problem [24]; it is an important property because it concerns BGP's convergence to a stable routing. Most of the interesting properties related to safety involve coordination of interdomain policies: each AS must express preferences for paths such that, for a given path assignment, no AS wants to change its path assignment to another path that exists in that path assignment.

Griffin *et al.* showed that global static analysis of BGP configuration can detect sufficient conditions for safety to be satisfied [27], but these conditions can't be verified in practice because an operator only has access to his own BGP configuration. Previous work suggests that a suitable way to arrive at a BGP configuration that satisfies safety is to restrict the policies that network operators can express [23]. Gao *et al.* also recognize that as long as the inter-AS topology satisfies certain global properties (*e.g.*, no cyclic relationships between providers and customers) and each AS assigns preferences and export policies according to inter-AS relationships (*e.g.*, always prefer customer routes over peer and provider routes), then BGP will converge to a globally stable solution [21]. This analysis shows that BGP works well under certain conditions, but (1) verifying that preconditions about the inter-AS topology requires global knowledge about bilateral agreements between ASes, (2) safety is not guaranteed if either these preconditions do not hold, and (3) safety is not guaranteed if ASes violate recommended export and preference policies (recent work has shown that ASes do deviate from these recommended policies [46]).

Rather than restricting the expressiveness of policies, the interdomain routing protocol could be designed such that it is guaranteed to converge on fast timescales and allow more complicated policy negotiation on slower timescales. We are currently investigating this possibility.

4. rcc Architecture

In this section, we describe rcc, a tool that verifies BGP configuration correctness using static analysis. We have made rcc available to about 30 network operators. rcc parses a wide variety of configuration files—including Cisco, Juniper, Avici, Procket, Zebra, and Quarry—and its extensible design makes adding new correctness tests easy.

rcc comprises three modules: (1) a preprocessor, which converts router configuration into a more parsable version (*e.g.*, by expanding macros); (2) a parser that generates a vendor-independent representation of configuration state; and (3) a constraint checker, which executes correctness checks against the abstract configuration format. We first describe how rcc parses router configuration files into this intermediate format; then, we briefly explain how rcc verifies correctness.

Table Description	*Summary and Columns*
global router options	router, various global options (*e.g.*, router ID, synchronization, etc.)
eBGP and iBGP sessions	router, neighbor IP address, eBGP/iBGP, pointers to import/export policy, session options (*e.g.*, next-hop self), RR client?
import/export filters	canonical representation of prefix/distribute lists: IP range, mask range, permit or deny
import/export policies	normalized policies: AS/community regexps, localpref/MED, etc. settings, etc.
loopback address(es)	router, loopback IP address(es)
configured interfaces	router, interface IP address(es)
prefixes originated	router, prefix/mask
static routes	static routes for prefixes: router, prefix/mask
	Derived or External Information
undefined references	summary of route maps, community lists, etc. that the parser could not resolve: router, name and type of undefined reference
bogon prefixes	bogon prefixes [10]

Table 4: Summary of intermediate configuration format, which we represent in terms of a set of relational database tables.

4.1 Intermediate Configuration Format

Because of the wide variety of router configuration languages, rcc analyzes an abstract, vendor-independent representation of BGP configuration. An intermediate representation separates verification from parsing; as new configuration languages surface and syntax in existing configuration languages changes, the parser can be modified independently of the part of the tool that performs verification. Additionally, an intermediate format allows for *extensibility*; as configuration options proliferate, and as more correctness tests are requested, we can easily adapt the intermediate format.

While all vendor configuration languages have similar characteristics, they have slight differences in how they express certain behaviors. For example, in Cisco IOS, the next-hop attribute is assigned as a session-level option, whereas in JunOS, the next-hop is assigned in export policy. The intermediate configuration format must accurately represent configuration semantics in a vendor-independent fashion, but it must also account for vendor-specific subtleties.

In addition to facilitating verification, an intermediate format helps us understand the aspects of configuration that affects BGP's behavior. The intermediate format is a concise representation of configuration semantics that allows an operator to see the entire configuration in an abstract form at a glance. Table 4 summarizes this intermediate format.

4.2 Preprocessing and parsing

The first step in generating the intermediate configuration format from vendor specific configuration files, *preprocessing*, (1) adds scoping identifiers to configuration languages that do not have explicit scoping (*e.g.*, Cisco IOS) and (2) expands macros (*e.g.*, Cisco's "peer group" option) to regularize configuration syntax.

Parsing generates the intermediate configuration format from the preprocessed configuration files. This process is fairly straightforward. Because each router has its own configuration, the parser can process each configuration file independently. The *global options*, *sessions*, and *prefixes* tables are populated from statements within the BGP configuration section of the configuration files. Other tables are generated by parsing the appropriate sections of the configuration.

The most complicated part of parsing is generating the *policies*, *filters*, and *sessions* tables. Because we are interested in comparing policies *across* routers, we must generate a normalized policy specification that specifies a step-by-step description of exactly what that policy does. This process generates a normalized "pseudocode" representation of each policy so that policies can be compared across routers. This process requires dereferencing every variable that is specified in the policy (*i.e.*, community-lists, AS path lists, etc.). The parser inserts every variable that a policy references that it cannot resolve into the *undefined references* table and joins these normalized policies with the *sessions* table; thus, in rcc's intermediate configuration format, distinguishing route maps that are the same or different across multiple routers becomes trivial.[5] Filters are normalized in the same fashion.

4.3 Verification

Once the parser generates the intermediate configuration format, rcc checks correctness constraints that test for specific configuration properties. We implemented the checks that we discussed in Section 3 as SQL queries. Some of the checks, such as those that check whether the "deterministic MED" or "no synchronization" options are enabled, are as simple as querying a single column from the *global options* table. Others, such as checking properties of the iBGP signaling graph, require running multiple queries against the *sessions* table to determine whether the top level of the route reflector hierarchy forms a clique. Certain queries involve comparing the entries from two tables. For example, rcc checks whether every route that is originated by BGP has a corresponding route; this requires comparing entries in the *networks* table against those from the *routes* table.

We have not yet implemented every correctness check from Section 3; in particular, several of the checks for iBGP forwarding correctness and determinism from previous work [1, 25] require knowledge about shortest IGP paths through the network, which rcc does not yet have (since it does not yet parse IGP configuration). However, rcc's extensible design facilitates adding these checks in the future.

5. Analyzing BGP Configuration in Practice

We now describe our experience using rcc to test real-world BGP configurations. After describing our experiences obtaining configurations, we summarize the errors and anomalies that rcc found in BGP configurations of real-world networks.

5.1 Obtaining Real-world Configuration

We aim to study real-world BGP configuration to better understand (1) how BGP is commonly configured today and (2) what types of configuration problems commonly occur in practice. Unfortunately, most network operators are not eager to share BGP configuration files. Router configuration files have proprietary information, such as details about relationships with neighboring ASes (implicit in how the neighbors' routes are preferred, filtered, etc.). Additionally, many ISPs may not like researchers pointing out that their networks have mistakes. Finally, network operators are continually busy tackling immediate problems, and crises seem to be non-stop (at a recent routing workshop, a network operator suddenly left the room mid-session to fix his network configuration). Despite previous efforts [15, 43], the research community has not been particularly successful at gaining access to real-world configurations.

[5]Recognizing policies that are similar or different simply by "eyeballing" multiple router configurations is not easy. Policies that have the same name on two different routers may do entirely different things, and vice versa. We have observed many instances of this type of obfuscation in practice.

Problem	*Property (Section)*	*Fix*	**Errors**	**Anom.**
First-class Errors: Serious				
eBGP session w/no filters	Validity (3.2.2)	P	21	0
session w/undefined filters	Validity (3.2.2)	P	26	0
missing prefix in filters	Validity (3.2.2)	P	143	0
non-RR iBGP partition	Visibility (3.3.2)	P	2	0
route reflector partition	Visibility (3.3.2)	P	3	0
RR cluster partition	Visibility (3.3.2)	P	2	0
duplicate loopbacks	Visibility (3.3.2)	L	3	1
unintentional transit	Info. Flow (3.4.1)	L,P	3	3
Second-class Errors: Annoyances				
inconsistent export to peer	Info. Flow (3.4.2)	L,P	3	3
router w/o determ. med	Deteterm. (3.5.1)	D	31	0
nondeterministic tiebreak	Deteterm. (3.5.1)	D	63	0
router w/synchronization	Visibility (3.3.1)	P	3	0
Third-class Errors: Cleanup				
session w/undefined policy	—	L	2	0
policy w/undefined AS path	—	L	6	0
policy w/undef. community	—	L	12	0
policy w/undefined ACL	—	L	12	0
incomplete iBGP sessions	Visibility (3.3.2)	P	76	0
prefix adv. w/o route	Visibility (3.3.1)	L	324	2
Curiosities				
session w/foreign AS prepend	Validity (3.2.3)	L,P	0	1
session w/o next-hop reach.	Validity (3.2.4)	P	0	2
cold potato import policy	Info. Flow (3.4.3)	L,P	0	23

Table 5: Configuration errors and anomalies in practice. (Recommended fixes are P: Protocol; L: Language; D: Decision process)

Despite these challenges, we were able to run rcc on the configurations of about 700 routers in 9 ASes, including BGP configurations from every router in 5 ASes. The size of these networks ranged from 2 routers to several hundred routers. We made rcc available to operators, hoping that they would run it on their configurations and report results; in response, some operators ran rcc on their configurations and reported the results, but most sent us their router configurations and had us run rcc for them. We did not have the complete BGP configuration for 4 of the 9 ASes we tested; for these, we performed single-router correctness checks (*e.g.*, filter configuration).

Because many operators are adamant that their router configurations be kept private (*i.e.*, both the fact that their router configurations might have mistakes and the proprietary nature of network operations in general), we cannot report separate statistics for each network that we tested. Nevertheless, *every network we tested had BGP configuration errors or anomalies*. When we discovered configuration errors in a network and subsequently raised them to an operator's attention, the operator was usually unaware (though often not surprised) that the errors existed. This confirms our hypothesis that many BGP configuration errors that occur in practice and affect correctness may not be apparent at configuration time.

5.2 Real-world BGP Configuration Analysis

Table 5 summarizes the errors and anomalies we discovered using rcc. We classified any aspect of configuration that turned out to be a mistake as an error, and anything that rcc incorrectly flagged as an error as an anomaly (*i.e.*, an operator told us that the "error" was in fact a special case). Figure 8 shows that most of these errors appeared in the configurations of more than one AS.

The results have several characteristics worth noting. First, er-

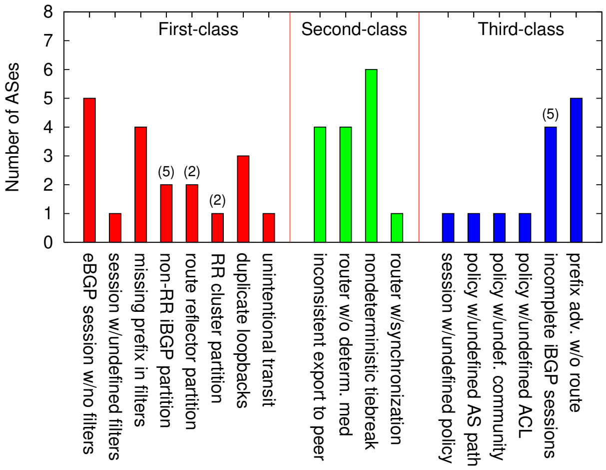

Figure 8: Number of ASes in which each type of error or anomaly occurred at least once. Unless noted in parentheses, each test was run on all 9 ASes.

rors have various dimensions. For example, one iBGP partition indicates that we found one case where a *network* was partitioned, but one instance of unintentional transit means that rcc found two *sessions* that appeared to be transiting traffic in violation of high-level policy. Thus, the absolute number of errors is not as important as noting that many of the errors, especially the more serious ones, occurred at least once (and, as shown in Figure 8, in more than one AS). Second, the errors vary in seriousness; one iBGP signaling partition is more serious than a handful of incomplete iBGP sessions that don't create any partitions. In this section, we classify errors according to three levels of seriousness.

Because we used rcc to test configurations that were *already deployed* in live networks, we did not expect rcc to find many of the types of transient misconfigurations that Mahajan *et al.* found in their study [31] (*i.e.*, those that quickly become apparent to operators when the configuration is deployed). We believe that, in reality, operators could apply rcc to router configurations *before* deploying router configurations, and, thus use static analysis to discover these types of errors as well.

The rest of this section describes these errors in more detail. We also shed light on *why* we think these errors are occurring and recommend possible changes to the protocol and configuration languages to reduce the likelihood of these errors.

5.2.1 First-class errors: Serious errors

1. Non-existent or inadequate filtering. As we described in Section 3.2.2, correct filtering practices do not guarantee validity, but they certainly prevent obviously invalid routes from propagating. Missing or incorrect import filters can result in invalid routes being propagated within an AS and advertised to other ASes. Filtering can go wrong in several ways: (1) no filters are used whatsoever, (2) a filter is specified in a session, but not defined, or (3) filters are defined but are out-of-date (*i.e.*, they are not current with respect to the list of "bogon" prefixes [10]). Table 5 shows that these types of errors were common. No AS we analyzed had completely correct filter configuration, and many of the smaller ASes we analyzed either had minimal filtering or none at all.

It is reasonable to wonder why ASes are so delinquent with respect to filtering. One operator told us that he would certainly apply more rigorous filters if he knew a good way of doing so. Another ISP runs sanity checks on its filters but was surprised to find that many sessions were referring to undefined filters. The problem with filtering seems to be associated with the lack of a systematic *process* for installing and updating filters. Because there is a set of prefixes that every AS should always filter, filter configuration should probably be automated and driven from a common registry. Another possibility is to build validity checks into BGP itself [29, 42].

2. iBGP signaling partitions. Every network configuration we checked with rcc that used route reflection had at least one signaling partition. These signaling partitions appeared in one of four ways: (1) the top-layer of iBGP routers was not a full mesh; (2) if route-reflectors were used, the top-layer of the route reflector hierarchy was not a full mesh; (3) a route reflector cluster had two or more route reflectors, but at least one client in the cluster did not have an iBGP session with every route reflector in the cluster; (4) two routers were assigned identical loopback addresses. Together, these accounted for 10 iBGP signaling partitions in 4 distinct ASes. While the most interesting partitions involved route reflection, we were surprised to find that even small networks had iBGP signaling partitions. In one network of only three routers, rcc discovered that the operator had failed to configure a full mesh; he told us that he had been experimenting and had inadvertently removed an iBGP session.

iBGP's goal is simple (*i.e.*, scalable, consistent route propagation within an AS), but its mechanisms are so complex and obscure that making mistakes is easy. Additionally, researchers are still discovering subtleties that fundamentally affect correctness (such as those from Section 3.3.2). A replacement for iBGP would eliminate most of these problems.

3. Unintentional transit. rcc discovered three instances where routes learned from one peer or provider could be readvertised to another; typically, these errors occurred because an export policy for a session was intended to filter routes that had a certain community value, but the export policy instead referenced an undefined community.

These errors seem to occur because today's BGP configuration language provides no way to specify simple policies. Operators make subtle errors because they are forced to specify these policies in terms of complex mechanisms. Interdomain routing would benefit greatly from a language that allowed operators to easily express simple policies would reduce the likelihood of these types of errors.

5.2.2 Second-class errors: Annoyances

1. Inconsistent export to peer. We found three cases where an AS was potentially violating peering agreements by advertising routes that were not "equally good" at every peering point. When rcc detected inconsistent export policies to a certain peer, we would refer back to the configurations to further analyze the inconsistencies. In many cases, the inconsistency appeared to be accidental; for example, one inconsistency existed because of an undefined AS path regular expression referenced in the export policy.

Sometimes, the inconsistency between two policies was not immediately apparent to us, even after rcc had pointed out the existence of an inconsistency: the two sessions used policies with the same name, and the definition of the policies would also look the same. The difference resulted from the fact that the difference in policies was three levels of indirection deep. For example, one inconsistency occurred because of a difference in the definition for an AS path regular expression that the export policy referenced (which, in turn, was referenced by the session parameters). A high-level policy specification could also help in this respect.

2. Nondeterminism. rcc discovered 34 routers that were configured such that the arrival order of routes affected the outcome of the route selection process (*i.e.*these routers had either one or both of the two configuration settings that cause nondeterminism). Although there are occasionally reasonably good reasons for introducing ordering dependencies (*e.g.*, preferring the "most stable" route; that is, the one that was advertised first), operators did not offer good reasons for why these options were disabled. In response to our pointing out this error, one operator told us,"That's a good point, but my network isn't big enough that I've had to worry about that yet." Many operators either are unaware of or indifferent to the benefits of determinism. Still, as nobody offered good reasons for having non-deterministic features enabled, they should probably be disabled by default.

3. Synchronization. Because most ASes only have routers that participate in iBGP, the configuration option that requires iBGP to be "synchronized" with the IGP (Section 3.3.1) is usually disabled. We discovered three routers that enabled synchronization; we were unable to contact the operator to find out why (a vendor provided us with anonymized configurations of one if its customers), but they appeared to be mistakes. The vendor speculated that their automatic configuration tool may at one time have been configured to enable synchronization by default.

5.2.3 Third-class errors: Cleanup issues

1. Undefined references. Several networks, including some large networks, had router configurations with basic parse errors, such as policies that referenced undefined variables and BGP sessions that referenced undefined filters. These types of errors can sometimes result in unintentional transit or inconsistent export to peers, as described above, but we also found many undefined references that, while not fundamentally affecting correctness, were likely not achieving the desired effect (*e.g.*, load balance, etc.). These problems mostly seem to surface in configurations where operators said they were frequently changing settings. Operators should take care to fix these parse errors (rcc is helpful for doing so), but BGP implementations should ideally check for undefined references at configuration time.

2. Incomplete iBGP sessions. When analyzing BGP configurations for complete ASes, rcc discovered hundreds of "half" iBGP sessions (*i.e.*, a configuration statement on one router indicated the presence of an iBGP session to another router, but the other router did not have an iBGP session in the reverse direction). In some cases, iBGP sessions (or parts of iBGP sessions) were "shutdown" but remained in the router configuration files. rcc can help operators clean configuration files by highlighting these types of inconsistencies (which often result from obsolescence).

3. Attempt to advertise prefixes with no route. rcc detected hundreds of instances where a router would attempt to originate a prefix via eBGP but would have no route to that prefix. Because BGP will not advertise any prefix for which it does not have a route, the configuration lines that advertised these prefixes have no effect and should be removed.

5.2.4 Anomalies and Curiosities

Because the ways that operators can configure BGP are so diverse, rcc sometimes incorrectly flagged errors that were in fact configuration "tricks". The fact that rcc discovers configuration aspects that violated our correctness rules but were nonetheless "correct" underscores the difficulty in defining correctness for BGP. We now survey some of the more interesting anomalies that rcc discovered.

Special AS relationships. When checking whether an AS was providing unintentional transit, rcc discovered one AS that appeared to readvertise certain prefixes from one peer to another. Upon further investigation, we learned that the AS that was carrying traffic between these peers was a previous owner of one of the peers. When we notified the operator that his AS was providing transit between these two peers, he told us, "Historically, we had a relationship between them. I don't know the status of that relationship is these days. Perhaps it is still active—at least in the configs!"

Creative AS path prepending practices. An AS will often prepend its own AS number to the AS path on certain outbound advertisements to affect inbound traffic. However, we found one AS that prepended a neighbor's AS on *inbound advertisements* in an apparent attempt to influence *outbound traffic*. One network operator also mentioned that ASes sometimes prepend the AS number of a network that they want to prevent from seeing a certain route (*i.e.*, by making that AS discard the route due to loop detection), effectively "poisoning" the route. We did not witness this poisoning in any of the configurations we analyzed.

iBGP sessions with next-hop self. We found two types of iBGP sessions that violate common rules for setting the next-hop attribute. First, we saw routers with route-reflector sessions that appeared to be resetting the next-hop attribute. In fact, some vendors disable the ability to reset the next-hop attribute on sessions to route reflector clients (*i.e.*, configuring the router to reset the next-hop attribute has no effect). The configuration specified that the next-hop attribute be reset because it made iBGP configuration more uniform. We also noticed that routers reset the next-hop attribute on iBGP sessions to route servers, which allows an operator to determine which router advertised a route to the route server.

Cold-potato routing. Previous work used traceroutes to observe that ASes often implement policies that cause traffic to exit their network at a point other than the nearest exit [41]. rcc found more than 20 instances where an AS's policies import policies explicitly implemented cold potato routing, which supports this previous observation.

5.2.5 Effects of Network Size

To explore how complexity affects the prevalence of errors in router configuration, we examined whether the configuration errors in a network are relatively more prevalent in larger networks. One might expect that ASes with larger configuration files, more routers, or more sessions might have proportionally more errors. We did not observe any significant correlation between network complexity and prevalence of errors, but configurations from more ASes are needed to draw any strong conclusions.

6. Related Work

We discuss previous work in three related areas: analysis of BGP behavior, verification of network protocols, and improving router configuration.

6.1 Analysis of BGP Behavior

Previous work introduced the notion of eBGP convergence and stated sufficient conditions for convergence [24, 27, 44]. Gao and Rexford also state sufficient conditions for eBGP convergence that can be checked locally and observe that typical policy configurations satisfy these conditions [19].

More recent efforts have addressed iBGP correctness. Griffin *et al.* examine two aspects of iBGP correctness: non-convergence (the iBGP analog to the eBGP problem) and "deflections", whereby packets do not follow their intended path [25]. This work astutely observes that the conditions for convergence under iBGP with route reflection are analogous to those specified by Gao and Rexford for eBGP.

Convergence is an important aspect of correctness, but it is by no means the only aspect. In this paper, we examine other aspects of correctness as well, with the goal of verifying real-world BGP configuration. In the process, we derive several complementary theoretical results.

Labovitz observed that BGP can take as long as 15 minutes to converge [30], and Mao discovered that "damping" routes that oscillate can delay convergence further [32]. Other work has simulated the effects of BGP's timer settings on convergence time [26]. This work focuses on *whether BGP will operate correctly at all*, regardless of timing and faults.

6.2 Protocol Verification

Model checking has been reasonably successful in verifying the correctness of programs [22] and other network protocols [4, 28, 34]. Previous work explains the difficulties of applying a model checking approach to BGP [14]. In short, model checking is not appropriate for verifying BGP configuration because its effectiveness heavily depends on exhausting the state-space within an appropriately-defined environment [33]. A fundamental problem with model checking BGP is that *policy hides states*: the behavior of an AS's BGP configuration depends on routes that arrive from other ASes, some of which, such as backup paths, cannot be known in advance.

Recent work proposes the "routing logic" [16], which makes verifying the correctness of a routing protocol more tractable and suggests that the logic could be useful for designing correctness checks, new configuration languages, and protocol modifications.

6.3 Improving Router Configuration

Mahajan *et al.* suggest that high-level languages and configuration checking could avert many instances of BGP misconfiguration [31]. In this work, we explore the types of BGP configuration errors that can be averted with configuration checking, and we devise a systematic framework that helps us outline which BGP configuration problems should be fixed by configuration language redesign versus protocol redesign.

Several products and research projects focus on configuration management. Commercial tools, such as IPAT [45] and NetDoctor [37], analyze and summarize network configuration, and highlight rudimentary configuration errors. AT&T is developing a configuration management tool with similar functionality; their project focuses on moving towards automating network configuration for enterprise networks [6]. These projects analyze the configuration of all network protocols, not just BGP, but typically only search for syntactic configuration errors (*e.g.*, a reference to an undefined route map, etc.), without a higher level correctness framework. Our work is the first to apply a high-level model for routing protocol correctness to configuration checking.

Many network operators use configuration management tools such as "rancid" [38], which periodically archive router configuration and allow operators to track changes to router configuration. When a network catastrophe coincides with the configuration change that caused it, these tools can help operators revert to an older configuration. Unfortunately, a configuration change may induce an error that does not immediately appear, and these tools do not tell operators whether the network configuration is correct in the first place.

7. Conclusion

BGP's correct operation requires that the protocol be correctly configured; despite the fact that BGP4 is almost 10 years old, our understanding of what it means for BGP to behave "correctly" has been minimal, and operators are continually making the same configuration mistakes. To improve the state of the art in router configuration, this paper offers several contributions. First, we present a framework for verifying BGP's correctness. Second, we apply this framework to the design and implementation of rcc, a tool that uses static analysis to verify BGP configuration correctness. Our tool has helped operators debug real-world BGP configuration. Third, we present our findings on BGP configuration errors and anomalies from static analysis of real-world BGP configurations. This paper is the first to explore BGP configuration errors using analysis of real-world BGP configuration files. Finally, we suggest concrete ways to improve BGP's correctness, distinguishing problems that should be fixed with protocol modifications from those that should be fixed with a better configuration language.

8. References

[1] BASU, A., ET AL. Route oscillations in IBGP with route reflection. In *Proc. ACM SIGCOMM* (Pittsburgh, PA, August 2002).

[2] BATES, T., CHANDRA, R., AND CHEN, E. *BGP Route Reflection - An Alternative to Full Mesh IBGP*. Internet Engineering Task Force, Apr. 2000. RFC 2796.

[3] BEIJNUM, I. V. *BGP*. O'Reilly and Associates, September 2002.

[4] BHARGAVAN, K., OBRADOVIC, D., AND GUNTER, C. A. Formal verification of standards for distance vector routing protocols. *Journal of the ACM (JACM) 49*, 4 (July 2002), 538–576.

[5] BICKNELL, L. Re: transit across the ixs. `http://www.merit.edu/mail.archives/nanog/1999-02/msg00192.html`, February 1999.

[6] CALDWELL, D., ET AL. The cutting EDGE of IP router configuration. In *Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)* (Cambridge, MA, November 2003).

[7] CHANDRA, R., AND TRAINA, P. *BGP Communities Attribute*. Internet Engineering Task Force, August 1996. RFC 1997.

[8] Cisco BGP Best Path Selection Algorithm. `http://www.cisco.com/warp/public/459/25.shtml`.

[9] Endless BGP Convergence Problem in Cisco IOS Software Releases. `http://cisco.com/warp/public/770/fn12942.html`.

[10] Team Cymru bogon route server project. `http://www.cymru.com/BGP/bogon-rs.html`.

[11] DENNING, D. E. A lattice model of secure information flow. *Communications of the ACM 19*, 5 (May 1976), 236–243.

[12] DUBE, R. A comparison of scaling techniques for BGP. *ACM Computer Communications Review 29*, 3 (July 1999), 44–46.

[13] FARROW, R. Routing instability on the Internet. *Network Magazine* (March 4, 2002). `http://www.networkmagazine.com/article/NMG20020304S0007/2`.

[14] FEAMSTER, N. Practical verification techniques for wide-area routing. In *Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)* (Cambridge, MA, November 2003).

[15] FEAMSTER, N., AND BALAKRISHNAN, H. A systematic approach to BGP configuration checking. `http://www.nanog.org/mtg-0310/feamster.html`, October 2003. NANOG 29.

[16] FEAMSTER, N., AND BALAKRISHNAN, H. Towards a logic for wide-area Internet routing. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture* (Karlsruhe, Germany, Aug. 2003).

[17] FEAMSTER, N., BORKENHAGEN, J., AND REXFORD, J. Techniques for interdomain traffic engineering. *Computer Communications Review 33*, 5 (October 2003).

[18] FEAMSTER, N., WINICK, J., AND REXFORD, J. A model of BGP routing for network engineering. In *Proc. ACM SIGMETRICS* (New York, NY, June 2004).

[19] GAO, L. On inferring automonous system relationships in the Internet. *IEEE/ACM Transactions on Networking 9*, 6 (December 2001), 733–745.

[20] GAO, L., GRIFFIN, T. G., AND REXFORD, J. Inherently safe backup routing with BGP. In *Proc. Infocom* (Anchorage, AK, April 2001).

[21] GAO, L., AND REXFORD, J. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking* (December 2001), 681–692.

[22] GODEFROID, P. Model Checking for Programming Languages using VeriSoft. In *Proc. ACM Symposium on Principles of Programming Languages* (1997).

[23] GRIFFIN, T., JAGGARD, A., AND RAMACANDRAN, V. Design principles of policy languages for path vector protocols. In *Proc. ACM SIGCOMM* (Karlsruhe, Germany, August 2003).

[24] GRIFFIN, T., AND WILFONG, G. An analysis of BGP convergence properties. In *Proc. ACM SIGCOMM* (Cambridge, MA, August 1999).

[25] GRIFFIN, T., AND WILFONG, G. On the correctness of IBGP configuration. In *Proc. ACM SIGCOMM* (Pittsburgh, PA, August 2002).

[26] GRIFFIN, T. G., AND PREMORE, B. J. An experimental analysis of BGP convergence time. In *Proc. ICNP* (Riverside, CA, November 2001).

[27] GRIFFIN, T. G., SHEPHERD, F. B., , AND WILFONG, G. The stable paths problem and interdomain routing. *IEEE Transactions on Networking 10*, 1 (2002), 232–243.

[28] HAJEK, J. Automatically verified data transfer protocols. In *Proc. ICCC* (1978), pp. 749–756.

[29] KENT, S., LYNN, C., MIKKELSON, J., AND SEO, K. Secure border gateway protocol (S-BGP) - real world performance and deployment issues. In *Proc. NDSS 2000* (2000).

[30] LABOVITZ, C., AHUJA, A., BOSE, A., AND JAHANIAN, F. Delayed Internet Routing Convergence. *IEEE/ACM Transactions on Networking 9*, 3 (June 2001), 293–306.

[31] MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM* (Aug. 2002), pp. 3–17.

[32] MAO, Z. M., GOVINDAN, R., VARGHESE, G., AND KATZ, R. Route Flap Damping Exacerbates Internet Routing Convergence. In *Prof. ACM SIGCOMM 2002* (Pittsburgh, PA, August 2002).

[33] MUSUVATHI, M., AND ENGLER, D. Some lessons from using static analysis and software model checking for bug finding. In *Workshop on Software Model Checking* (Boulder, CO, July 2003).

[34] MUSUVATHI, M., AND ENGLER, D. A framework for model checking network protocols. In *Proc. First Symposium on Networked Systems Design and Implementation* (San Francisco, CA, March 2004).

[35] The North American Network Operators' Group mailing list archive. `http://www.cctec.com/maillists/nanog/`.

[36] NORTON, W. Internet service providers and peering. `http://www.equinix.com/press/whtppr.htm`.

[37] Opnet NetDoctor. `http://opnet.com/products/modules/netdoctor.htm`.

[38] Really Awesome New Cisco ConfIg Differ (RANCID). `http://www.shrubbery.net/rancid/`, 2004.

[39] REKHTER, Y., AND LI, T. *A Border Gateway Protocol 4 (BGP-4)*. Internet Engineering Task Force, 1995. RFC 1771.

[40] REKHTER, Y., LI, T., AND HARES, S. *A Border Gateway Protocol 4 (BGP-4)*. Internet Engineering Task Force, Nov. 2003. Internet Draft draft-ietf-idr-bgp4-23.txt; work in progress.

[41] SPRING, N., MAHAJAN, R., AND ANDERSON, T. Quantifying the causes of path inflation. In *Proc. ACM SIGCOMM* (Aug. 2003).

[42] SUBRAMANIAN, L., ROTH, V., STOICA, I., SHENKER, S., AND KATZ, R. Listen and whisper: Security mechanisms for bgp. In *Proc. First Symposium on Networked Systems Design and Implementation* (San Francisco, CA, March 2004).

[43] BGP config donation. `http://www.cs.washington.edu/research/networking/policy-inference/donation.html`.

[44] VARADHAN, K., GOVINDAN, R., AND ESTRIN, D. Persistent route oscillations in inter-domain routing. *Computer Networks 32*, 1 (2000), 1–16.

[45] WANDL IPAT. `http://wandl.com/html/ipat/IPAT_new.cfm`, 2003.

[46] WANG, F., AND GAO, L. On inferring and characterizing internet routing policies. In *Proc. Internet Measurement Conference* (Miami, FL, October 2003).