Detecting BGP Configuration Faults with Static Analysis

Nick Feamster and Hari Balakrishnan
MIT Computer Science and Artificial Intelligence Laboratory
{feamster, hari}@csail.mit.edu
http://nms.csail.mit.edu/bgp/rcc/

Abstract

The Internet is composed of many independent autonomous systems (ASes) that exchange reachability information to destinations using the Border Gateway Protocol (BGP). Network operators in each AS configure BGP routers to control the routes that the routers learn, select, and announce to other routers. Faults in BGP configuration can cause forwarding loops, packet loss, and unintended paths between hosts, all of which constitute failures of the Internet routing infrastructure.

This paper describes the design and implementation of rcc*, the* router configuration checker*, a tool that finds faults in the BGP configurations of routers in an AS using static analysis.* rcc *detects two broad classes of faults that affect network reachability: route validity faults, where routers may learn routes that do not correspond to usable paths, and path visibility faults, where routers may fail to learn routes for paths that exist in the network.* rcc *enables network operators to test and debug configurations before deploying them in an operational network, improving on the status quo where most faults are detected only during operation.* rcc *has been downloaded by more than sixty network operators to date, some of whom have shared their configurations with us. We analyze network-wide configurations from 17 different ASes to detect both faults that induce failures and faults that are benign.*

1 Introduction

This paper describes the design, implementation, and evaluation of the router configuration checker, **rcc**, a tool that uses static analysis to detect faults in Border Gateway Protocol (BGP) router configurations. rcc detects faults by checking conditions that are based on a high-level correctness specification. By finding faults over a distributed set of router configurations, rcc enables network operators to test and debug configurations before deploying them in an operational network, improving on the status quo of "stimulus-response" debugging where operators need to run configurations in an operational network before finding faults.

Router configuration languages offer considerable flexibility to network operators, who use them to provide reachability, express routing policy (*e.g.*, transit and peering relationships [25], inbound and outbound routes [3], etc.), configure primary and backup links [14], and perform traffic engineering across multiple links [11]. 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.

Configuring a network of BGP routers is like writing a distributed program where complex feature interactions occur both within one router and across multiple routers. This complex process is exacerbated by the number of lines of code (we find that a 500-router network typically has more than a million lines of configuration distributed across its routers), by the absence of useful high-level primitives in today's router configuration languages, by the diversity in vendor-specific configuration languages, and by the number of ways in which similar high-level functionality can be expressed in a configuration language. As a result, router configurations tend to have faults [3, 21].

Faults in router configurations can seriously affect end-to-end Internet connectivity, leading to failures whose symptoms are lost packets, forwarding loops, and unintended paths. Section 2.2 discusses the problems observed in operational networks in detail. Failure-inducing configuration faults include invalid routes (including hijacked and leaked routes); contract violations [10]; unstable routes [20]; routing loops [7, 9]; and persistently oscillating routes [1, 16, 31]. Many of these configuration faults can be detected by analyzing BGP configuration alone.

Detecting BGP configuration faults is crucial for preventing end-to-end routing failures, but doing so poses several challenges. First, defining a correctness specification for BGP is difficult: its many modes of operation and myriad tunable parameters permit a great deal of flexibility in both the design of a network and in how that design is implemented in the configuration itself. Second, BGP's configuration is distributed—analyzing how a network configuration behaves requires synthesizing distributed configuration fragments. This paper tackles these challenges and makes the following three contributions:

First, we define two high-level aspects of correctness—path visibility and route validity—and map this specification to conditions on the BGP configuration that must hold for those aspects of correctness to hold. Path visibility says that BGP will correctly propagate routes for existing, usable IP-layer paths; essentially, it states that the *control* path is propagating BGP routes correctly. Route validity says that, if routers attempt to send *data* packets on these routes, then packets will ultimately reach their intended destinations.

Second, we present the design and implementation of **rcc** ("router configuration checker"), a tool that analyzes BGP configuration of a single AS and detects possible violations of path visibility and route validity. Over sixty network operators have downloaded rcc. rcc focuses on detecting faults that have the potential to cause *persistent* routing failures. rcc is *not* concerned with correctness during convergence (since any distributed protocol will have transient inconsistencies during convergence). rcc's goal is to detect problems that may exist during steady state, even when the protocol converges; thus, it assumes that the protocol will converge to some outcome, given stable inputs.

Third, we use rcc to explore the extent of real-world BGP configurations faults; this paper presents the first direct analysis of BGP configuration faults in real-world ISPs. We have analyzed real-world, deployed configurations from 17 different ASes and detected more than 1,000 BGP configuration faults that had previously gone undetected by operators. These faults ranged from simple "single router" faults (*e.g.*, undefined variables) to complex, *network-wide* faults involving interactions between multiple routers. rcc discovered many faults that could potentially cause failures. These include: (1) faults that could have caused network partitions due to errors in how external BGP information was being propagated to routers inside an AS, (2) faults that cause invalid routes to propagate inside an AS, and (3) faults in policy expression that caused routers to advertise, and hence potentially forward packets, in a manner inconsistent with the AS's desired policies. The 17 network-wide configurations we analyzed with rcc had already been deployed (deployment being the current state of the art testing technique) on live, operational networks; rcc is actually intended to be used *before* configurations are actually deployed. If rcc were used before BGP configuration was deployed, we expect that rcc would be able to detect some of the immediately active faults, in addition to those we discovered in our evaluation.

Our analysis of real-world configurations suggests that most configuration faults stem from three main causes. First, the mechanisms for propagating routes within a network are overly complex. The main techniques used to propagate routes scalably within a network (*e.g.*, "route reflection with clusters") are easily misconfigured. Second, even simple policy specifications (*e.g.*, controlling route propagation) require configuration fragments on many (if not all) routers in the network, and each fragment often involves several layers of indirection (*e.g.*, "route maps", "community lists", etc.). Finally, operators lack a systematic process for configuring functions such as filtering.

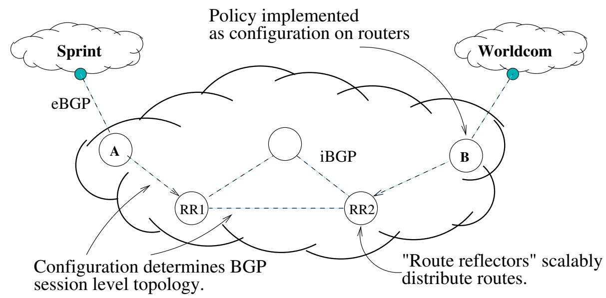

Figure 1. Overview of BGP configuration within an AS.

The rest of this paper proceeds as follows. Section 2 provides further motivation and background on BGP. Section 3 describes the design of rcc. Sections 4 and 5 discuss rcc's path visibility and route validity tests. Section 6 describes implementation details. Section 7 presents configuration errors and anomalies that rcc discovered in 17 operational networks. Section 8 addresses related work, and Section 9 concludes.

2 Background and Motivation

We briefly present an overview of today's interdomain routing infrastructure, focusing particularly on the aspects that are affected by configuration. Then, we study the extent and nature of interdomain routing misconfiguration as discussed on North American Network Operators Group (NANOG) mailing list.

2.1 Overview of BGP Configuration

The Internet comprises over 17,000 independently operated ASes that exchange reachability information using BGP [28]. BGP distributes routes to destination prefixes via incremental updates. Each router selects one best route to a destination, announces that route to neighboring routers, and sends updates when the best route changes. Each BGP update contains several attributes. These include the *destination prefix* associated with the route; the *AS path*, the sequence of ASes that advertised the route; 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 use to specify that one route should be more (or less) preferred than routes advertised at

Property	Description	1995-1997	1998-2001	2002-2004	Total
Filtering	Filtering problems with private addresses	28 (40)	27 (50)	37 (73)	92 (163)
Leaked Routes	AS mistakenly announcing routes (*e.g.*, bad prefix, bad AS path, etc.)	13 (14)	23 (24)	20 (22)	56 (60)
Hijacked Routes	Observation of or complaint about stolen address space	5 (5)	14 (14)	3 (4)	22 (23)
Global Route Visibility	Filtering new address allocations	40 (53)	38 (57)	68 (95)	146 (205)
Oscillations	Persistent route oscillations	0 (0)	0 (2)	0 (3)	0 (5)
Routing Instability	Route instability (*i.e.*"flapping")	28 (32)	26 (34)	25 (32)	79 (98)
Attribute manipulation	BGP attribute manipulation (*e.g.*, for implementing backup)	9 (9)	15 (25)	7 (15)	31 (49)
iBGP-related	Apparently an iBGP problem	16 (17)	14 (22)	18 (30)	48 (69)
Routing Loops	Traceroute with loop	7 (7)	11 (12)	5 (7)	23 (26)
Blackholes	Traceroute with blackhole (or complaint)	8 (8)	25 (27)	83 (91)	116 (126)
Total		154 (185)	193 (267)	266 (372)	613 (824)

Table 1. Number of threads discussing BGP-related routing errors over the 10 years of the NANOG mailing list. Non-parenthesized numbers count threads addressing actual problems; parenthesized numbers also count generic discussion threads (*e.g.*, questions, documents) on the topic.

other routers of that AS; and the *community* value, which is a way of labeling a route.

BGP's *configuration* affects what (and whether) routes are originated and propagated, how routes are modified as they propagate (which, in turn, affects route selection), and how routes propagate between routers (*i.e.*, the session-level topology). To understand the importance of configuration in determining BGP's behavior, consider the AS shown in Figure 1. A single AS can have anywhere from two or three routers to many hundreds of routers. A single router's configuration can range from a few hundred lines to more than 10,000 lines.

Two of the more complicated configuration aspects determined by BGP's configuration are:

1. Session-level BGP topology. A router's configuration determines which other routers that router will exchange BGP routes with. A typical router has two types of BGP sessions: those to routers in its own AS (internal BGP, or "iBGP") and those to routers in other ASes (external BGP, or "eBGP"). A small AS with only two or three routers may have only 10 or 20 BGP sessions, but large backbone networks typically have more than 10,000 BGP sessions, more than half of which are iBGP sessions.

The session-level BGP topology determines how BGP routes propagate through the network. In small networks, iBGP is configured as a "full mesh" (every router connects to every other router). To improve scalability, larger networks typically use "route reflectors". A route reflector selects a single best route and announces that route to all of its "clients". Route reflectors can easily be misconfigured (we discuss iBGP misconfiguration in more detail in Section 4). Incorrect iBGP topology configuration can create persistent forwarding loops and oscillations [17].

2. Policy. Routers have import policies, which manipulate incoming routes; and export policies, which manipulate routes before the router advertises them to other routers. A router's policy can implement *filtering*: preventing a certain route from being accepted on inbound or readvertised on outbound. Policy can also manipulate route attributes. Common reasons for manipulating route attributes are: (1) indirectly controlling which route BGP selects as its "best" route, (2) controlling the "next hop" IP address for the advertised route, and (3) labeling a route with a "community", or a tag. In practice, a large backbone ISP may have more than a thousand different policies configured across hundreds of routers.

Policy configuration is complicated because global behavior depends on the configuration of individual routers. In Figure 1, router B may "tag" a route received from Worldcom with a certain community value, and router A's export policy may specify that routes bearing that community should not be exported to Sprint. Misconfiguration of either A or B can cause Worldcom's route to "leak" to Sprint.

2.2 Problems in Operational Networks

NANOG operates a mailing list where network operators report operational problems, discuss operational issues, etc. [24]. To gain a better understanding of 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 75,000 emails over the course of ten years, we first clustered the emails by thread and pruned threads based on a list of about fifteen keywords (*e.g.*, "BGP", "issue", "loop", "problem", "outage"). We then manually reviewed 1,600 email threads, 634 of which were relevant to BGP configuration issues. We classified each of these threads into one or more of the categories shown in Table 1.

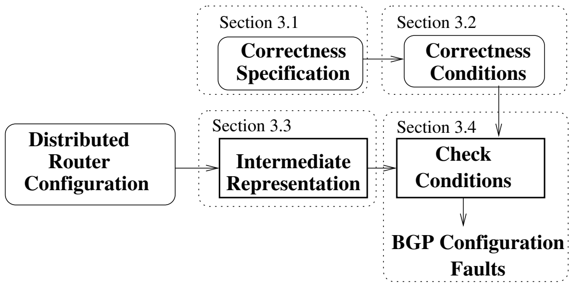

Figure 2. Overview of rcc.

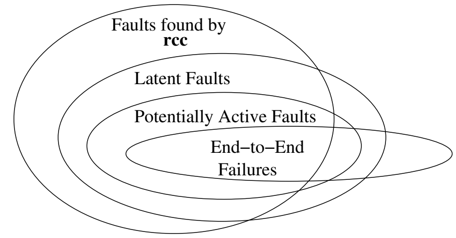

Figure 3. Relationships between faults and failures.

This informal study shows some clear trends. First, many interdomain routing problems are caused by configuration faults. Second, the state of affairs has not improved over the last ten years: the same types of errors and problems continually appear. Third, BGP configuration problems continually perplex even experienced network operators. A fault detection tool will clearly benefit network operators.

3 rcc Design

rcc analyzes both single-router and network-wide properties of BGP configuration and outputs a list of configuration faults. Figure 2 illustrates rcc's high-level architecture.

rcc's checks that the BGP configuration satisfies a set of correctness conditions, which are based on a correctness specification that we outline in Section 3.1. Section 3.2 explains how we map the correctness specification into conditions that rcc can check against the BGP configuration. Rather than operating directly on vendor-specific configuration, rcc converts the BGP configuration to a vendor-independent intermediate representation. It then checks this intermediate format for faults based on a set of correctness conditions. Section 3.3 details how rcc generates this intermediate representation, and Section 3.4 explains how rcc applies the correctness conditions to the intermediate representation to produce faults.

rcc does not operate on the correctness specification itself. Rather, we have used a high-level specification as a guide for designing the actual correctness conditions that rcc applies to the configuration. We envision that rcc has three classes of users: those that wish to run rcc with no modifications, those that wish to add new conditions that apply to the existing specification, and those that wish to augment the high-level specification. rcc's modular design allows users to specify other correctness conditions without changing the system internals. Some users may wish to extend the high-level specification to include other aspects of correctness (*e.g.*, safety [17]) and map those high-level specifications to conditions on the configuration.

Figure 3 shows the relationships between classes of configuration faults and the class of faults that rcc detects. *Latent faults* are faults that are not actively causing any problems but nonetheless violate the correctness conditions. A subset of latent faults are *potentially active* faults, for which there is at least one input sequence that is certain to trigger the fault. For example, an import policy that references an undefined filter on a BGP session to a neighboring AS is a potentially active fault, which will be triggered when that neighboring AS advertises a route that ought to have been filtered. When deployed, a potentially active fault will become *active* if the corresponding input sequence occurs. An active fault constitutes a routing failure for that AS.

Some active faults may ultimately appear as *end-to-end failures*. For example, if an AS advertises an invalid route (*e.g.*, a route for a prefix that it does not own) to a neighboring AS whose import policy references an undefined filter, then some end hosts may not be able to reach destinations within that prefix. Note that a potentially active fault may not always result in an end-to-end failure if no path between the sources and destinations traverses the routers in the faulty AS.

rcc detects a subset of latent (and hence, potentially active) faults. In addition, rcc may also report some false positives: faults that violate the conditions but are *benign* (*i.e.*, the violations would never cause a failure). Ideally, rcc would detect fewer benign faults if it could test the BGP configuration against an abstract specification. Unfortunately, producing such a specification requires additional work from operators, and operators may well write incorrect specifications. One of rcc's advantages is that it provides useful information about configuration faults without requiring any additional work on the part of operators.

3.1 Defining a Correctness Specification

Inspired by the routing logic [9], rcc checks two aspects of correctness: *path visibility* and *route validity*.[1] In the con-

[1] Previous work [9] presents three properties in addition to these two: information flow control (this property checks if routes "leak" in violation of policy), determinism (whether a router's preference for routes depends on the presence or absence of other routes), and safety (whether the protocol converges) [18]. Our work treats information flow control as a subset of validity. rcc does not check for faults related to determinism and safety because the former is not possible to check with static analysis alone, and the latter requires configurations from multiple ASes. We do not consider multi-AS checks in this paper.

text of BGP, a *path* is a sequence of IP hops (*i.e.*, routers) between two endpoints (*e.g.*, routers, end hosts, etc.). A *route* is a BGP message that advertises reachability to some destination via an associated path.

Path visibility: If there is a path to a destination that does not violate the policies of the routers along the path, *and* there is some valid route to the destination at a router, then the router is said to have a *visible path*. Paths may become invisible if the propagation of routes to destinations among routers in an AS is buggy. rcc detects configuration problems that may lead to paths becoming invisible (and hence, to some end hosts losing reachability to others even though usable paths between those end hosts are present).

Route validity: A route to a destination is *valid* if, and only if, the path advertised by that route:

1. Reaches the destination,
2. Corresponds to the path that packets take when using that route (*i.e.*, sending packets towards the next hop for that route), and
3. Conforms to the policies of routers on the path.

A route that is not valid may prevent packets that use that route from reaching the intended destination or cause packets to take a different path than the advertised route. rcc detects configuration faults that may lead to invalid routes.

rcc finds faults in BGP configuration only. To make general statements about path visibility and route validity, rcc assumes that the internal routing protocol (*i.e.*, interior gateway protocol, or "IGP"; OSPF is an example of an IGP) used to establish routes between any two routers within a AS is operating correctly. BGP requires the IGP to operate correctly because iBGP sessions may traverse multiple IGP hops and because the "next hop" for iBGP-learned routes is typically several IGP hops away.

The correctness specification that we have presented addresses *static* properties of BGP, *not* dynamic behavior (*i.e.*, its response to changing inputs, convergence time, etc.). BGP, like any distributed protocol, may experience periods of transient incorrectness in response to changing inputs. rcc detects faults that cause *persistent* failures. Previous work has studied sufficient conditions on the relationships between iBGP and IGP configuration that must be satisfied to guarantee that iBGP converges [17]; these conditions require parsing the IGP configuration, which rcc does not yet check. The correctness specifications and conditions in this paper assume that, given stable inputs, the routing protocol *eventually* converges to some steady state behavior.

Currently, rcc only detects faults in the BGP configuration of a *single AS* (usually, a network operator only has access to the BGP configuration from his own network). Because an AS's BGP configuration explicitly controls both session-level topology and policy, many types of misconfiguration, including partitions, route leaks, etc., are evident from the BGP configuration of a single AS.

PROBLEM	POSSIBLE ACTIVE FAULT
Path Visibility	
iBGP Signaling Problems	
Signaling partition: - of route reflectors - within a RR "cluster" - in a "full mesh"	Router may learn a suboptimal route or none at all.
Routers with duplicate: - loopback address - cluster ID	Routers may incorrectly drop routes as duplicates.
iBGP configured on one end	Routers won't exchange routes.
iBGP not to loopback	iBGP session fails when one interface fails.
Route Validity	
Policy Problems	
transit between peers	Network carries traffic "for free".
inconsistent export to peer	Violation of contract.
inconsistent import	Possible unintentional "cold potato" routing.
Advertisement Problems	
prepending with bogus AS	AS path is no longer valid.
originating unroutable dest.	Creates a blackhole.
incorrect next-hop	Other routers may be unable to reach the routes for a next-hop that is not in the IGP.
Miscellaneous	
Undefined References	
eBGP session: - w/no filters - w/undef. filter - w/undef. policy filter: - w/missing prefix policy: - w/undef. AS path - w/undef. community - w/undef. filter	• leaked internal routes • re-advertising bogus routes • accepting bogus routes from neighbors • unintentional transit between peers
Decision Process Problems	
nondeterministic MED age-based tiebreaking	Route selection depends on message order.

Table 2. BGP configuration problems that rcc detects, and their potentially active faults.

3.2 Mapping the Specification to Conditions

Because BGP configuration is flexible, defining precise conditions on BGP configuration that are equivalent to the high-level specification is extremely challenging. rcc's conditions are an attempt to map the path visibility and route validity specifications to conditions on BGP configuration that can be checked automatically. These conditions are neither complete (*i.e.*, they may not find all problematic configurations) nor sound (*i.e.*, they may report problems that are simply deviations from best common practice). However, static analysis techniques for program analysis are typically neither complete nor sound either [22].

Table 2 summarizes the correctness conditions that rcc checks. Path visibility conditions typically address potential problems with how iBGP routes propagate through the AS, or "iBGP signaling" (Section 4). rcc's route validity conditions test potential problems with policy configuration or how routes are advertised (Section 5). The "miscellaneous" conditions can affect both path visibility and route validity.

Ideally, operators would run rcc to detect configuration faults *before* they are deployed. Some of rcc's conditions detect faults that would most likely become active immediately upon deployment. For example, an operator would quickly notice that a router is advertising routes with an incorrect next-hop attribute, since any other router that tries to use those routes will be unable to route packets for those destinations. In this case, rcc can help the operator diagnose configuration faults and prevent them from introducing failures on the live network. Many of the conditions in Table 2 detect faults that could remain undetected even after the configuration has been deployed, until some sequence of messages triggers them. For example, an operator will likely not notice that a BGP session to a neighboring AS applies an undefined filter until that neighbor "leaks" an invalid route. In these cases, rcc can help operators find faults that could result in a serious failure.

3.3 Generating the Intermediate Representation

rcc converts BGP configuration to an intermediate representation that it uses to check correctness conditions. As new configuration languages evolve, only the parser needs to be modified. rcc implements the intermediate representation as a set of relational database tables. Representing related information about routing configuration in tables makes it easier to ask questions about network-wide configuration, since all of the information related to the network's BGP configuration can be summarized in a handful of tables. A relational structure is natural because many sessions share common attributes (*e.g.*, all sessions to the same neighboring AS often have the same policies), and many policies have common clauses (*e.g.*, all eBGP sessions may have a filter that is defined in exactly the same way). Table 3 summarizes these tables.

Example of an intermediate representation. Figure 4 shows rcc's intermediate representation for a fragment of Cisco IOS. This Cisco configuration specifies a BGP session to a neighboring router with IP address `10.1.2.3` in AS 3. This statement is represented by a row in the *sessions* table. The second line of configuration specifies that the import policy (*i.e.*, "route map") for this session is defined as "`IMPORT_CUST`" elsewhere in the file; the intermediate representation represents the import policy specification as a pointer into a separate table that contains the import policies themselves. A single policy, such as `IMPORT_CUST` is represented as multiple rows in the *policies* table. Each row represents a single clause of the policy. In this example, `IMPORT_CUST` has two clauses: the first rejects all routes whose AS path matches the regular expression number "99" (specified as "`^65000`" elsewhere in the configuration), and the second clause accepts all routes that match AS path number "88" and community number "10"

Table	Description
global options	router, various global options (*e.g.*, router ID)
sessions	router, neighbor IP address, eBGP/iBGP, pointers to policy, options (*e.g.*RR client)
prefixes	router, prefix originated by this router
import/export filters	canonical representation of filter: IP range, mask range, permit or deny
import/export policies	canonicalized representation of policies
loopback address(es)	router, loopback IP address(es)
interfaces	router, interface IP address(es)
static routes	static routes for prefixes
Derived or External Information	
undefined references	summary of policies and filters that a BGP configuration referenced but did not define
bogon prefixes	prefixes that should always be filtered on eBGP sessions [6]

Table 3. Intermediate configuration representation schema.

and sets the "local preference" attribute on the route to a value of 80. Each of these clauses is represented as a row in the policy table; specifications for regular expressions for AS paths and communities are also stored in separate tables, as shown in Figure 4.

rcc converts a policy into the intermediate representation shown in Figure 4 through a process that we call *policy canonicalization*. rcc's intermediate representation stores nothing about the names of the policies themselves (*e.g.*, "`IMPORT_CUST`", AS regular expression number "88", etc.). Rather, the intermediate format only stores a description of what the route policy does (*e.g.*, "set the local preference value to 80 if the AS path matches regular expression `^3`"). Two policies may be written using entirely different names, regular expression numbers, etc., but if the policies operate exactly the same way, rcc will recognize that they are in fact the same policy.

3.4 Checking the Conditions

rcc uses the intermediate representation to detect BGP configuration faults in four ways:

1. Issuing simple queries against the relational tables.
2. Analyzing the BGP session-level topology.
3. Forming *beliefs* about the high-level policy that the BGP configuration is implementing.
4. Computing "closures" over policies: *i.e.*, determining which routes would (and would not) propagate between an AS's neighbors.

In all of these cases, rcc issues queries against the database, but the latter three require more complex logic. The rest of this section explains how rcc issues simple queries against the intermediate representation. Section 4 describes how rcc uses the BGP session-level topology to detect path visibility faults in iBGP. Section 5 explains how rcc uses beliefs and closures to detect route validity faults.

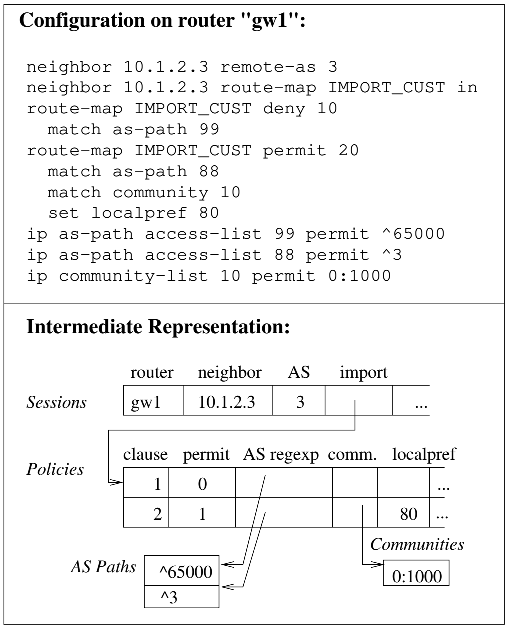

Figure 4. BGP configuration in intermediate format.

Checking conditions with simple queries. rcc checks many correctness conditions by executing simple queries against the intermediate representation. Checking conditions against the intermediate representation is simpler than analyzing distributed configuration files. Consider the test in Table 2 called "iBGP configured on one end"; this condition requires that, if a router's configuration specifies an iBGP session to some IP address, then (1) that IP address should be the loopback address of some other router in the AS, and (2) that other router should be configured with an iBGP session back to the first router's loopback address. rcc tests this condition as a single, simple "select" statement that "joins" the *loopbacks* and *sessions* tables.

As another example, to check that no routing policy in the AS prepends any AS number other than its own, rcc executes a "select" query on a join of the *sessions* and *policies* tables, which returns the ASes that each policy prepends (if any) and the routers where each policy is used. rcc then checks the *global* table to ensure that that for each router, the AS number configured on the router matches the ASes that any policy on that router prepends.

4 Path Visibility Faults

Recall that *path visibility* specifies that every router that has a path to a destination learns at least one valid route to that destination. It is an important property because it ensures that, if the network remains connected at lower layers, the routing protocol does not create any network partitions. Table 2 shows many conditions that rcc checks that affect path visibility; in this section, we focus on iBGP configuration faults that can violate path visibility and explain how rcc detects these faults.

Ensuring path visibility in a "full mesh" iBGP topology is reasonably straightforward; rcc checks that every router in the AS has an iBGP-session with every other router. If this condition is satisfied, every router in the AS will learn all eBGP-learned routes.

Because a "full mesh" iBGP topology scales poorly, operators often employ *route reflection* [2]. A subset of the routers are configured as *route reflectors*, with the configuration specifying a set of other routers as *route reflector clients*. Each route reflector readvertises its best route according to the following rules: (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. A router does *not* readvertise iBGP-learned routes over regular iBGP sessions, 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). If a route reflector client has multiple route reflectors, those 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. Reflector-client relationships among routers in an AS define a graph G, where each router is a node and each session is either a directed or undirected edge: a client-reflector session is a directed edge from client to reflector, and other iBGP sessions are undirected edges. An edge exists if and only if (1) the configuration of each router endpoint specifies the loopback address of the other endpoint[2] and (2) both routers agree on session options (*e.g.*, MD5 authentication parameters). G should also not have partitions at lower layers. We say that G is *acyclic* if G has no sequence of directed and undirected edges that form a cycle. For various reasons, including to ensure the existence of a stable path assignment, G should be acyclic.

Even a connected directed acyclic graph of iBGP sessions can violate path visibility. For example, in Figure 5, 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. Note that simply adding a regular iBGP session between routers W and Y would solve the problem.

This potentially active fault can prevent routers from learning any route to a destination, even though a path ex-

[2] 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.

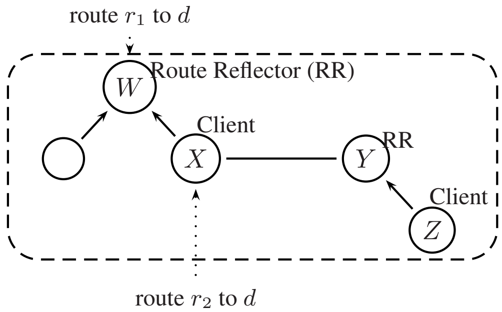

Figure 5. In this iBGP configuration, route r_2 will be distributed to all the routers in the AS, but r_1 will not. Y and Z will not learn of r_1, leading to a network partition that won't be resolved unless another route to the destination appears from elsewhere in the AS.

ists. In Figure 5, if a route to d is only learned at W, then routers Y and Z will not learn any route to the destination d. Even if Y or Z learned a route to d via eBGP, the route that these routers learn might be worse than the route learned at W. In this case, Y and Z would ultimately select a suboptimal route to the destination, an event that an operator would likely fail to notice, but which rcc can easily detect.

rcc checks the static iBGP configurations of the routers in an AS to determine whether such a situation can occur. More specifically, it determines 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 is the basis for a simple and efficient check.

Theorem 4.1 *Suppose that G is acyclic and connected at the IGP layer. Then, G does not have a signaling partition if, and only if, the BGP routers that are not route reflector clients form a full mesh.*

PROOF. Call the set of routers that are not reflector clients the "top-layer" of the iBGP graph G. 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, since no router readvertises an iBGP-learned route.

Conversely, if the top layer is a full mesh, observe that *if* a route reflector has a route to the destination, then all its clients have a route as well. Thus, if 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, from there to all the other top-layer routers, and from there to the other routers in the AS. □

rcc checks this condition by constructing the iBGP signaling graph G from the *sessions* table (Table 3). It assumes that the IGP graph is connected, then determines whether G is connected and acyclic and whether the routers at the top layer of G form a full mesh.

5 Route Validity Faults

BGP should satisfy *route validity*. Because BGP is a policy-based protocol, the configuration, not the protocol specification, affects which routes each router accepts, selects, and re-advertises. In this section, we focus on rcc's approach to detecting potential policy-related problems.

The biggest challenge for checking policy is that rcc operates without a specification of the intended policy. Requiring operators to provide a high-level policy specification would require designing a specification language and convincing operators to use it, and it provides no guarantees that the results would be more accurate, since errors may be introduced into the specification itself. Rather, rcc assumes that the network abides by some best common practices and constructs *beliefs* about a network operator's intended policy. rcc then searches for deviations from best common practices and beliefs about policy. We now highlight several aspects of policy that rcc checks:

1. A route that an AS learns from one of its "peers" should not be readvertised to another peer. Checking this condition requires determining how a route propagates across a network. Figure 6 illustrates how rcc performs this check, which proceeds as follows. Suppose that rcc is analyzing the configuration from AS X and needs to determine that no routes from Worldcom are exported to Sprint. First, rcc determines all routes that X exports to Sprint, typically a set of routes that satisfy certain constraints on their attributes. For example, router A may export to Sprint only routes that are "tagged" with the label "1000". (ASes often designate such labels to signify how a route was learned; *e.g.*, from a customer.) rcc then checks the *import* policies for all sessions to Worldcom, ensuring that no import policy will set route attributes on any incoming route that would place it in the set of routes that would be exported to Sprint.

2. An AS should advertise paths with equally good attributes to each peer at every peering point. An AS should not advertise routes with inconsistent attributes, since doing so may prevent its peer from implementing "hot potato" routing,[3] which typically violates peering agreements. Recent work has observed that this type of inconsistent route advertisement sometimes occurs in practice [10]. This violation can result from two configuration faults:

First, an AS may apply different export policies at dif-

[3]If ASes 1 and 2 are peers, then the export policies of the routers in AS 1 should export routes to AS 2 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)

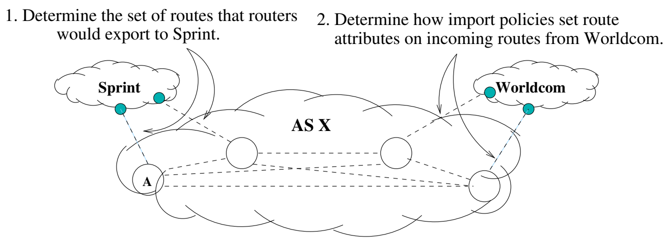

Figure 6. How rcc computes route propagation.

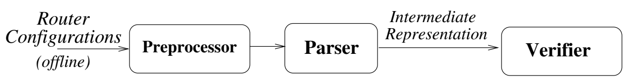

Figure 7. High-level overview of rcc implementation.

ferent routers to the same peer. Checking for consistent export involves comparing export policies on each router that has an eBGP session with a particular peer. Static analysis is useful because it can efficiently compare policies on many different routers. In practice, this comparison is not straightforward because differences in policy definitions are difficult to detect by visual inspection. rcc makes comparing export policies trivial by canonicalizing all of the export policies for an AS, as described in Section 3.3.

Second, an iBGP signaling partition can create inconsistent export policies because routes with equally good attributes may not propagate to all peering routes. For example, consider Figure 5 again. If routers W and Z both learn routes to some destination d, then route W may learn a "better" route to d, but routers Y and Z will continue to select the less attractive route. If routers X and Y re-advertise their routes to d to a peer, then the routes advertised by X and Y will not be equally good. Thus, rcc checks whether routers that advertise routes to the same peer are in the same iBGP signaling partition (as described in Section 4, rcc checks for *all* iBGP signaling partitions, but ones that cause inconsistent advertisement are particularly serious).

3. Import policies where sessions to a neighbor AS are the same except at one or two routers are most likely mistakes. This test relies on the *belief* that, if an AS exchanges routes with a neighboring AS on many sessions and most of those sessions have identical import policies, then the sessions with slightly different import policies may be misconfigurations. Of course, this test could result in many false positives because there are legitimate reasons for having slightly different import policies on sessions to the same neighboring AS (*e.g.*, outbound traffic engineering), but it is a useful sanity check nonetheless. rcc also detects inconsistent export policies.

6 Implementation

rcc is implemented in about 6,500 lines of Perl and has been downloaded by about sixty network operators. The parser is roughly 60% of the code. Much of the parser's logic is dedicated to policy canonicalization. Figure 7 shows an overview of rcc, which takes as input the set of configuration files collected from routers in a single AS using a tool such as "rancid" [27]. rcc's functionality is decomposed into three distinct modules: (1) a preprocessor, which converts configuration into a more parsable version; (2) a parser, which generates the intermediate representation; and (3) a verifier, which executes the correctness checks.

The *preprocessor* adds scoping identifiers to configuration languages that do not have explicit scoping (*e.g.*, Cisco IOS) and expands macros (*e.g.*, Cisco's "peer group", "policy list" and "template" options). After the preprocessor performs some simple checks to determine whether the configuration file is a Cisco-like configuration or a Juniper configuration, it launches the appropriate parser. Many configurations (*e.g.*, Avici, Procket, Zebra, Quarry) resemble Cisco configuration; the preprocessor translates these configurations so that they more closely resemble Cisco syntax.

The *parser* generates the intermediate representation from the preprocessed configuration files. rcc parses both Cisco and Juniper configuration. The parser processes each configuration file independently. It makes a single pass over each configuration file, looking for keywords that help determine where in the configuration it is operating (*e.g.*, "`route-map`" in a Cisco configuration indicates that the parser is entering a policy declaration). The parser builds a table of canonicalized policies by dereferencing all filters and other references in the policy; if the reference is defined after it is referenced in the same file, the parser performs lazy evaluation. The parser flags any references that the it cannot resolve after reaching the end of the file as "undefined reference" errors. The parser proceeds file-by-file (taking care to consider that definitions are scoped by each file), keeping track of canonicalized policies and whether they have already appeared in other configurations.

We implemented each correctness condition in Table 2 by executing SQL queries against the intermediate format and analyzing the results of these queries in Perl. Many tests simply require querying the value of a single column from the *global options* table. Other tests, such as checking properties of the iBGP signaling graph, require reconstructing the iBGP signaling graph using the *sessions* table. The tests for iBGP signaling are about 700 lines of Perl, but most other tests are fewer than 30 lines of Perl each.

7 Evaluating Operational Networks with rcc

Our goal is to help operators move away from today's mode of stimulus-response reasoning by allowing them to check the correctness of their configurations *before* deploying them on a live network. rcc has helped network operators find faults in their deployed configurations, which we present in this section. 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 [21] (*i.e.*, those that quickly become apparent to operators when the configuration is deployed). If rcc were applied to BGP configurations before deployment, we expect that it could prevent more 75% of the "origin misconfiguration" incidents and more than 90% of the "export misconfiguration" incidents described in that study.[4]

7.1 Analyzing Real-World Configurations

We used rcc to evaluate the configurations from 17 real-world networks, including BGP configurations from every router in 12 ASes. We made rcc available to operators, hoping that they would run it on their configurations and report their results.

Working with network operators. Network operators are reluctant to share configuration files. Router configurations have proprietary information embedded in the policies. Also, many ISPs do not like researchers reporting on mistakes in their networks. (Previous efforts have enjoyed only limited success in gaining access to real-world configurations [30].) We learned that providing operators with a useful tool or service increases the likelihood of cooperation. When presented with rcc, many operators opted to provide us with configuration files, while others ran rcc on their configurations and sent us the output.

rcc detected over 1,000 configuration faults. The size of these networks ranged from two routers to more than 500 routers. Many operators insist that the details of their configurations be private, so we cannot report separate statistics for each network that we tested. Every network we tested had BGP configuration faults. Operators were usually unaware of the faults in their networks, which confirms our hypothesis that many configuration faults are not apparent.

7.2 Fault Classification and Summary

Table 4 summarizes the faults that rcc detected. rcc discovered potentially serious configuration faults as well as benign ones. The fact that rcc discovers benign faults underscores the difficulty in specifying correct behavior for BGP. Faults have various dimensions and levels of seriousness. For example, one iBGP partition indicates that rcc found one case where a *network* was partitioned, but one instance of unintentional transit means that rcc found two *sessions* that, together, caused the AS to carry traffic in violation of high-level policy. The absolute number of faults is not as important as noting that many of the faults occurred at least once.

PROBLEM	LATENT	BENIGN
Path Visibility		
iBGP Signaling Problems		
Signaling partition:		
- of route reflectors	4	1
- within a RR "cluster"	2	0
- in a "full mesh"	2	0
Routers with duplicate:		
- loopback address	13	120
- cluster ID		
iBGP configured on one end or not to loopback	420	0
Route Validity		
Policy Problems		
transit between peers	3	3
inconsistent export to peer	231	2
inconsistent import	105	12
Advertisement Problems		
prepending with bogus AS	0	1
originating unroutable dest.	22	2
incorrect next-hop	0	2
Miscellaneous		
Undefined References		
eBGP session:		
- w/no filters	21	—
- w/undef. filter	27	—
- w/undef. policy	2	—
filter:		
- w/missing prefix	196	—
policy:		
- w/undef. AS path	31	—
- w/undef. community	12	—
- w/undef. filter	18	—
Decision Process Problems		
nondeterministic MED	43	0
age-based tiebreaking	259	0

Table 4. BGP configuration faults in 17 ASes.

Figure 8 shows that many faults appeared in many different ASes. We did not observe any significant correlation between network complexity and prevalence of faults, but configurations from more ASes are needed to draw any strong conclusions. The rest of this section describes the extent of the configuration faults that we found with rcc.

7.3 Path Visibility Faults

The path visibility faults that rcc detected involve iBGP signaling and fall into three categories: problems with "full mesh" and route reflector configuration, problems configuring route reflector clusters, and incomplete iBGP session configuration. All of the conditions in this section required access to the BGP configuration for every router in the AS.

iBGP signaling partitions. iBGP signaling partitions appeared in one of two ways: (1) the top-layer of iBGP

[4] rcc detects the following classes of misconfiguration described by Mahajan *et al.*: reliance on upstream filtering, old configuration, community, forgotten filter, prefix-based config, bad ACL or route map, and typo.

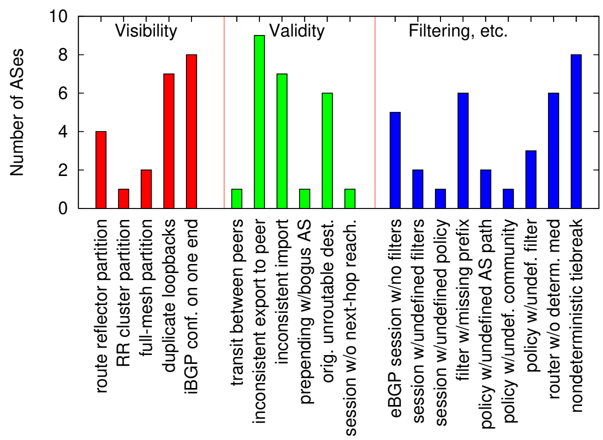

Figure 8. Number of ASes in which each type of fault occurred at least once.

routers was not a full mesh; or (2) 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. Together, these accounted for 9 iBGP signaling partitions in 5 distinct ASes, one of which was benign. While most partitions involved route reflection, we were surprised to find that even small networks had iBGP signaling partitions. In one network of only three routers, the operator had failed to configure a full mesh; he told us that he had "inadvertently removed an iBGP session". rcc also found two cases where routers in a cluster with multiple route reflectors did not have iBGP sessions to all route reflectors in those cluster.

rcc discovered one benign iBGP signaling partition. The network had a group of routers that did not exchange routes with the rest of the iBGP-speaking routers, but the routers that were partitioned introduced all of the routes that they learned from neighboring ASes into the IGP, rather than readvertising them via iBGP. The operator of this network told us that these routers were for voice-over-IP traffic; presumably, these routers injected all routes for this application into the IGP to achieve fast convergence after a failure or routing change. In cases such as these, BGP configuration cannot be checked in isolation from other routing protocols.

Route reflector cluster problems. In an iBGP configuration with route reflection, multiple route reflectors may serve the same set of clients. This group of route reflectors and its clients is called a "cluster"; each cluster should have a unique ID, and all routers in the cluster should be assigned the same cluster ID. If a router's BGP configuration does not specify a cluster ID, then typically a router's loopback address is used as the cluster ID. If two routers have the same loopback address, then one router may discard a route learned from the other, thinking that the route is one that it had announced itself. rcc found 13 instances of routers in distinct clusters with duplicate loopback addresses and no assigned cluster ID.

Different physical routers in the same AS may legitimately have identical loopback addresses. For example, routers in distinct IP-layer virtual private may route the same IPv4 address space over distinct VPNs.

Incomplete iBGP sessions. rcc discovered 420 incomplete 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). These faults are almost certainly benign. The most likely explanation for the large number of these is that network operators may disable sessions by removing the configuration from one end of the session without ever "cleaning up" the other end of the session.

7.4 Route Validity Faults

In this section, we discuss route validity faults. We first discuss policy-related faults; we classify faults as latent unless a network operator explicitly told us that the fault was benign. We then discuss some interesting faults related to route advertisement, all of which were benign.

7.4.1 Policy Problems

Decomposing policies across configurations on different routers can cause faults, even for simple policies such as controlling route export between peers. rcc discovered the following problems:

Transit between peers. rcc discovered three instances where routes learned from one peer or provider could be readvertised to another; typically, these faults 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.

Obsolete contractual arrangements can remain in configuration long after those arrangements expire. rcc discovered one AS that appeared to readvertise certain prefixes from one peer to another. Upon further investigation, we learned that the AS was actually 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!"

Inconsistent export to peer. We found 231 cases where an AS advertised routes that were not "equally good" at every peering point. It is hard to say whether these inconsistencies are benign without knowing the operator's intent, but nearly twenty of these inconsistencies were certainly accidental. For example, one inconsistency existed because of an undefined AS path regular expression referenced in the export policy; these types of inconsistencies have also been observed in previous measurement studies [10].

Inconsistent import policies. A recent measurement study observed that ASes often implement policies that result in late exit (or "cold potato") routing, where a router does not select the BGP route that provides the closest exit point from its own network [29]. rcc found 117 instances where an AS's import policies explicitly implemented cold potato routing, which supports this previous observation. In one network, rcc detected a different import policy for *every* session to each neighboring AS. In this case, the import policy was labeling routes according to the router at which the route was learned.

Inconsistent import and export policies were not always immediately apparent to us even after rcc detected them: the two sessions applied policies with the same name, and both policies were defined with verbatim configuration fragments. 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).

7.4.2 Advertisement Problems

We describe configuration faults involving route attributes. rcc found only benign faults in this case.

Unorthodox 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*.[5]

iBGP sessions with "next-hop self". We found two cases of iBGP sessions that violated common rules for setting the next-hop attribute, both of which were benign. First, rcc detected route reflectors that appeared to be setting the "next hop" attribute. Although this practice is not likely to create active faults, it seemed unusual, since the AS's exit routers typically set the next hop attribute, and route reflectors typically do not modify route attributes. Upon further investigation, we learned that some router vendors do not allow a route reflector to reset the next-hop attribute. Even though the configuration specified that the session would reset the next-hop attribute, the configuration statement had no effect because the software was designed to ignore it. The operator who wrote the configuration specified that the next-hop attribute be reset on these sessions to make the configuration appear more uniform. Second, routers sometimes reset the next-hop on iBGP sessions to themselves on sessions to a route monitoring server to allow the operator to distinguish which router sent each route to the monitor.

[5]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.

7.5 Miscellaneous Tests

We used rcc to perform tests that checked for undefined references to policies and filters, as well as several other minor router configuration options. Some of these faults, while simple to check, could have serious consequences (*e.g.*, leaked routes), if rcc had not caught them and they had been activated.

Undefined references in policy definitions. Several large networks had router configurations that referenced undefined variables and BGP sessions that referenced undefined filters. These faults can sometimes result in unintentional transit or inconsistent export to peers or even potential invalid route advertisements. In one network, rcc found four routers with undefined filters that would have allowed a large ISP to accept and readvertise *any* route to the rest of the Internet; this potentially active fault could have been catastrophic if a customer had (unintentionally or intentionally) announced invalid routes, since ASes typically do not filter routes coming from large ISPs. This misconfiguration occurred despite the fact that the router configurations were being written with scripts; an operator had apparently made a mistake specifying inputs to the scripts. Operators can detect such faults using rcc.

Non-existent or inadequate filtering. Filtering can go wrong in several ways: (1) no filters are used whatsoever, (2) a filter is specified but not defined, or (3) filters are defined but are missing prefixes or otherwise out-of-date (*i.e.*, they are not current with respect to the list of private and unallocated IP address space [6]).

Every network that rcc analyzed had faults in filter configuration. Some of these faults would have caused an AS to readvertise *any* route learned from a neighboring AS. In one case, policy misconfiguration caused an AS to transit traffic between two of its peers. Table 4 and Figure 8 show that these faults were extremely common: rcc found 21 eBGP sessions in 5 distinct ASes with no filters whatsoever and 27 eBGP sessions in 2 ASes that referenced undefined filters. Every AS had partially incorrect filter configuration, and most of the smaller ASes we analyzed either had minimal or no filtering. Only a handful of the ASes we analyzed appeared to maintain rigorous, up-to-date filters for private and allocated IP address space.

The problem with filtering seems to stem from the lack of a process for installing and updating filters. One operator told us that he would be willing to apply more rigorous filters if he knew a good way of doing so. Another ISP runs sanity checks on its filters and was surprised to find that many sessions were referring to undefined filters. Even a well-defined process can go horribly wrong: one ISP intended to use a feed of unallocated prefixes to automatically install its filters, but instead ended up readvertising them.

Because there is a set of prefixes that every AS should always filter, some prefixes should be filtered by default.

Nondeterministic route selection. rcc discovered more than two hundred 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 fault, 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." Non-deterministic features should be disabled by default.

7.6 Higher-level Lessons

Our evaluation of real-world BGP configurations from operational networks suggests four higher-level lessons about the nature of today's configuration process. First, operational networks—even large, well-known, and well-managed ones—have faults. Even the most competent of operators find it difficult to manage BGP configuration. Moreover, iBGP is misconfigured often; in fact, in the absence of a guideline such as Theorem 4.1, it is hard for a network operator to know what properties the iBGP signaling graph should have. Second, although operators use tools that automate some aspects of configuration, these tools are not a panacea. In fact, we found that the incorrect use of these tools can cause configuration faults to occur. Third, maintaining network-wide policy consistency appears to be hard; invariably, in most ASes there are routers whose configuration appears to contradict the AS's desired policy. Finally, we found that route filters are poorly maintained. Routes that should never be seen on the global Internet (*e.g.*, routes for private addresses) are rarely filtered, and the filters that are used are often misconfigured and outdated.

8 Related Work

We discuss work in three areas: router configuration, BGP convergence, and verification in other settings.

Router configuration. Mahajan *et al.* studied short-lived BGP misconfiguration by analyzing transient, globally-visible BGP announcements from an edge network [21]. They defined a "misconfiguration" as a transient BGP announcement that was followed by a withdrawal within a small amount of time (suggesting that the operator observed and fixed the problem). They found that many misconfigurations are caused by faulty route origination and incorrect filtering. rcc can help operators find these faults; it can also detect faults that are difficult to quickly locate and correct. rcc also helps operators detect the types of misconfigurations found by Mahajan *et al. before* deployment.

Some commercial tools analyze network configuration and highlight rudimentary errors [26]. Previous work has proposed tools that analyze intradomain routing configuration [12] and automate enterprise network configuration [5]. These tools detect router and session-level syntax errors only (*e.g.*, undefined filters), a subset of the faults that rcc detects. rcc is the first tool to check *network-wide properties* using a vendor-independent configuration representation and the first tool that applies a high-level specification of routing protocol correctness.

Many network operators use configuration management tools such as "rancid" [27], which periodically archive router configuration and provide revision tracking. When a network problem coincides with the configuration change that caused it, these tools can help operators revert to an older configuration. Unfortunately, a configuration change may induce a latent or potentially active fault, and these tools do not detect whether the configuration has these types of faults in the first place.

Analysis of BGP "safety" and stability. Previous work has analyzed problems where eBGP and iBGP may not converge to a stable path assignment, introducing the notion of a stable path assignment and stating sufficient conditions to guarantee that BGP will arrive at such an assignment [16, 18, 31]. This property is called *safety*. Gao and Rexford state sufficient conditions for safety in eBGP and observe that typical policy configurations satisfy these conditions [13]; Griffin *et al.* note that analogous sufficient conditions apply to iBGP with route reflection [17]. Safety is an important aspect of correctness, but it is by no means the only aspect. Our work examines aspects of BGP configuration that may be incorrect, even if the protocol converges to a stable path assignment.

Verification in other settings. Model checking has been successful in verifying the correctness of programs [15] and other network protocols [4, 19, 23]. Previous work explains the difficulties of applying a model checking approach to BGP [8]. In short, model checking is not appropriate for verifying BGP configuration because model checking depends heavily on exhausting the state-space within an appropriately-defined environment [22]. A fundamental problem with model checking BGP is that 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.

9 Discussion and Conclusion

In recent years, much work has been done to understand BGP's behavior, and much has been written about the wide range of problems it has. Some argue that BGP has outlived its purpose and should be replaced; others argue that faults arise because today's configuration languages are not well-designed. We believe that our evaluation of faults in today's

BGP configuration has provided a better understanding of the types of errors that appear in today's BGP configuration and the problems in today's configuration languages. Our findings should help inform the design of wide-area routing systems in the future.

Despite the fact that BGP is almost 10 years old, operators continually make the same mistakes as they did during BGP's infancy, and, regrettably, our understanding of what it means for BGP to behave "correctly" is still rudimentary. This paper takes a step towards improving this state of affairs by making the following contributions:

- We define a high-level correctness specification for BGP and map that specification to conditions that can be tested with static analysis.
- We use this specification to design and implement rcc, a static analysis tool that detects faults by analyzing the BGP configuration across a single AS. With rcc, network operators can find many faults *before* deploying configurations in an operational network. rcc has been downloaded by over 60 network operators.
- We use rcc to explore the extent of real-world BGP misconfigurations. We have analyzed real-world, deployed configurations from 17 different ASes and detected more than 1,000 BGP configuration faults that had previously gone undetected by operators.

In light of our findings, we suggest two ways to make interdomain routing less prone to configuration faults. First, protocol improvements protocol itself, particularly in intra-AS route dissemination (iBGP), could avert many BGP configuration faults. The current approach to scaling iBGP should be replaced. Route reflection serves a single, relatively simple purpose, but it is the source of many faults, many of which can't be checked with static analysis of BGP configuration alone [17]. The protocol that disseminates BGP routes within an AS should enforce path visibility and route validity.

Second, BGP should be configured with a higher-level specification language. BGP configuration is too low-level: it affords flexibility in the *mechanisms* that implement high-level policy, but this flexibility introduces complexity, obscurity, and opportunities for misconfiguration rather than design flexibility or expressiveness. For example, rcc detects many faults in implementation of some high-level policies in low-level configuration; these faults arise because there are many ways to implement the same high-level policy, and the low-level configuration is unintuitive. Ideally, a network operator would never touch low-level mechanisms (*e.g.*, the community attribute) in the common case: instead, an operator could configure interdomain routing using a language that more directly reflects high-level policies and goals.

References

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

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

[5] CALDWELL, D., GILBERT, A., GOTTLIEB, J., GREENBERG, A., HJALMTYSSON, G., AND REXCORD, J. The cutting EDGE of IP router configuration. In *Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)* (Cambridge, MA, Nov. 2003).

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

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

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

[9] 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).

[10] FEAMSTER, N., MAO, Z. M., AND REXFORD, J. BorderGuard: Detecting cold potatoes from peers. In *Proc. ACM SIGCOMM Internet Measurement Conference* (Taormina, Sicily, Italy, Oct. 2004).

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

[12] FELDMANN, A., AND REXFORD, J. IP network configuration for intradomain traffic engineering. *IEEE Network* (Sept. 2001).

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

[14] GAO, L., GRIFFIN, T. G., AND REXFORD, J. Inherently safe backup routing with BGP. In *Proc. IEEE INFOCOM* (Anchorage, AK, Apr. 2001).

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

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

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

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

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

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

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

[22] 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).

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

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

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

[26] Opnet NetDoctor. http://opnet.com/products/modules/netdoctor.htm.
[27] Really Awesome New Cisco ConfIg Differ (RANCID). http://www.shrubbery.net/rancid/, 2004.
[28] REKHTER, Y., AND LI, T. *A Border Gateway Protocol 4 (BGP-4)*. Internet Engineering Task Force, Mar. 1995. RFC 1771.
[29] SPRING, N., MAHAJAN, R., AND ANDERSON, T. Quantifying the causes of path inflation. In *Proc. ACM SIGCOMM* (Karlsruhe, Germany, Aug. 2003).
[30] BGP config donation. http://www.cs.washington.edu/research/networking/policy-inference/donation.html.
[31] VARADHAN, K., GOVINDAN, R., AND ESTRIN, D. Persistent route oscillations in inter-domain routing. *Computer Networks 32*, 1 (2000), 1–16.