Descartes BGP:
A Conflict Detection and Response Framework for Inter-Domain Routing

***Abstract*—We present Descartes BGP (D-BGP), a fault detection and response framework that enhances the robustness, security, and manageability of inter-domain routing. D-BGP associates a state of "agreement," "conflict," or "persistent conflict" with each announced address prefix. When a D-BGP router receives a routing update in which a new AS claims to be an origin of a prefix, it alerts other D-BGP routers to collaboratively verify their ownership claim and resolve the potential conflict without reference to an oracle, such as a topology database server. If a conflict is "persistent," a black hole may have formed, pulling traffic destined to the prefix in conflict. When this happens, D-BGP logs useful diagnostic information to aid resolution by network administrators. In spite of the black hole, the D-BGP framework allows data traffic to reach critical network services located on or needed by the hosts within the prefix. We evaluate D-BGP with the Scaleable Simulation Framework NETwork (SSFNET) simulator and show that D-BGP resolves BGP faults and misconfigurations in real time, and mitigates a persistent conflict over the ownership of an IP prefix. We show that D-BGP provides path resilience quickly and with few messages. Using BGP update data obtained during an actual black hole event, we show that D-BGP's detection mechanism scales well.**

***Index Terms*— inter-domain routing, BGP, fault detection and response, multiple origin AS, black hole.**

«au contraire de cela, même que je pensais à douter de la vérité des autres choses,
il suivait très évidemment et très certainement que j'étais.»
"to the contrary, in the very act of thinking about doubting the truth of other things,
it very clearly and certainly followed that I existed."

- René Descartes (1596-1650), *Le Discours de la Méthode*, Quatrieme Partie

I. INTRODUCTION

Today's Internet consists of 21,000+ interconnected Autonomous Systems (ASes) [15]. These ASes create peering relationships among themselves as determined by business agreements. Once peering relationships are in place, the ASes use the Border Gateway Protocol (BGP) to establish routes to each of the Internet's 184,000+ IP prefixes. Higher-level services, such as domain name service (DNS) and public key infrastructure (PKI), rely on BGP's decentralized routing infrastructure to work properly. If BGP becomes unstable, all end-to-end transport-layer services become unstable too, if they do not simply fail. Indeed, a single BGP failure can cause a large portion of the Internet to become unreachable for a significant period of time [13].

Unfortunately, BGP is not robust against unexpected errors, failures, or malicious attacks [6,33,34,35,42]. A very small number of faulty ASes can have a significant impact on a much larger area of the Internet, as documented in many well-known failure instances [2,21,37,38,41]. While the framework proposed in this work can handle a variety of BGP failures and attacks, we focus on address prefix hijacking, *i.e.*, the *Multiple Origin AS* (MOAS) problem [47]. This problem occurs when two ASes announce that they are the origin for a prefix *P/n*, a network address in classless inter-domain routing format where *P* is the prefix of bits of an IP address that identify the network and $n = |P|$. If *P/n* is actually *not* multihomed, then one of the ASes must be making a false claim, inadvertently or maliciously. In so doing, this faulty AS becomes a *black hole*, diverts *P/n*'s traffic into itself, and prevents neighboring ASes from reaching any host within prefix *P/n.* As each neighboring AS rebroadcasts the new, shorter route, the black hole expands and pulls in more and more of *P/n*'s traffic.

When a black hole occurs, even non-faulty BGP router unwittingly aids the black hole's expansion by propagating incorrect routes to the affected prefix. This is due to an overly simplistic trust model: upon receiving a BGP message from one of its peers, a BGP router unconditionally accepts the message[1]. Encrypting and authenticating the communication between routers A and B can prevent man-in-the-middle attacks[7], but it cannot prevent B from being tricked if A misbehaves. Like other proposals for securing BGP, such as SoBGP [44],

1 Local router policy determines the best, or most preferred, route. By default, the best route is the shortest one.

this work is concerned with the correctness of the information that is distributed within the protocol itself. In other words, we must ensure that a router A cannot trick another router B by sending incorrect information. The proposal for the conceptual knowledge planealso addresses this problem [10]: a router, as an intelligent entity, should "reason about" the semantic meaning of the messages it receives. For example, such a router would scrutinize messages and consider whether a message might cause a black hole.

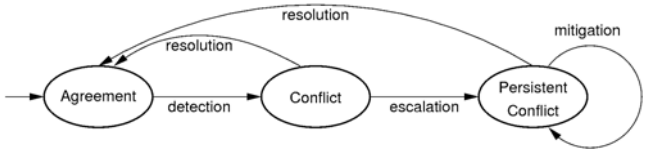

Figure 1: Prefix State in Descartes BGP.

To enhance the robustness, security, and manageability of inter-domain routing, we propose *Descartes BGP* (D-BGP), a conflict *detection and response* framework. For clarity of exposition, we assume that all routers run D-BGP, except in Section VI where we take up the question of D-BGP's deployability. A D-BGP router associates a state of "agreement," "conflict," or "persistent conflict" with each address prefix in its routing table, as shown in Figure 1. Normally, a prefix is in the *agreement* state. However, when a D-BGP router detects, in real-time, a potential origin conflict over a prefix, it changes the state of the affected prefix to the *conflict* state. This D-BGP router alerts the two D-BGP routers that are both claiming the prefix about their conflict. Using only local information, these routers each confirm whether they are, in fact, an origin of the prefix. If either router determines that it is not an origin of the prefix, it immediately issues a withdrawal, thereby ending the conflict. In this way, D-BGP resolves conflicts caused by transient faults and misconfigurations and returns the prefix involved to the *agreement* state.

When neither router withdraws its claim, the conflict over the prefix becomes *persistent*, indicating a serious fault – a potential black hole. D-BGP notifies network administrators and provides them with the identity of the routers involved in the conflict to aid them in resolving the conflict. Network administrators take action outside D-BGP, such as (1) reboot and reconfigure routers, (2) call other network administrators to collaboratively troubleshoot problems, sever peering relationships, or (3) manually override routing table entries. These actions control the administered system, but take place outside it. Collectively, we call the actions and interactions of network administrators, the *management plane*. Just as the BGP control plane establishes and maintains the data plane, the management plane establishes and maintains the control plane.

Pending resolution of the persistent conflict in the management plane, D-BGP mitigates the conflict and allows critical network services to reach the prefix in spite of its persistent conflict. Taking a page from King Solomon, D-BGP splits a prefix-in-conflict in two and hands each half to one of the two ASes claiming it. In other words, D-BGP tolerates the origin conflict over *P/n* by associating the deaggregations *P0/n+1* with the lower ordinal AS and *P1/n+1* with the higher ordinal AS. This then allows ASes not involved in the conflict to decide, by their selection of *P0* or *P1*, to which of the ASes in conflict to forward their data traffic bound for *P/n*. The obvious pressure this tactic places on *P/n*'s host address space is discussed in Section III.C. This ability to reach a prefix in conflict and D-BGP's skeptical trust model enable ASes to respectively maintain connectivity and avoid being dragged into black holes. Thus, each AS has an incentive to deploy D-BGP.

D-BGP makes all state transitions depicted in Figure 1 automatically without human intervention, apart from the resolution of persistent conflicts. We present the details of how D-BGP accomplishes these tasks and maintains its per-prefix state machine in Section III below.

We have evaluated D-BGP with the SSFNET simulator and shown that D-BGP resolves BGP misconfigurations in real time, allowing network traffic to continue throughout the event: results from simulations show that most network traffic in this case continues after a brief interruption. We show that D-BGP provides path resilience quickly and with few messages. In order to estimate D-BGP behavior in the internet, we analyze BGP update data obtained from an actual black hole event, and show that D-BGP's detection mechanism scales well.

By providing good detection capabilities and forensics information, D-BGP serves as a basis from which many different solutions can be implemented. In summary, D-BGP:

- Detects and resolves, in real time, inadvertent misconfigurations that result in conflicts over prefix origin;
- Detects and mitigates black holes by mapping deaggregations to each of two ASes in conflict over a prefix;
- Provides extensive logging of its activities for forensic analysis and troubleshooting of MOAS conflicts; and
- Uses only local information (no centralized oracle).

The rest of this paper is organized as follows. Section II describes related work and compares D-BGP to other proposals. Section III uses Figure 1 to describe D-BGP's architecture. It covers D-BGP's detection and handling of prefix origin conflict, escalation of that conflict to a persistent state, before presenting connectivity even under malicious attack. We also prove that in a D-BGP network there will be at least one detector. Section IV outlines

threats to D-BGP and D-BGP's countermeasures. In Section V, we simulate D-BGP and show that D-BGP scales well, since the number of detectors is small. Finally, Section VII considers future work and concludes.

II. Related Work

Security and robustness of network routing protocols have been studied for decades. For instance, the classical thesis of Radia Perlman [25] discussed how to support Byzantine robustness using packet flooding, in the context of link-state routing protocols. Due to the self-stabilization property [3] of protocols such as Open Shortest Path First (OSPF) [29], it has been shown [39,43] that, even without any extra security enhancements, link-state routing protocols survive many classes of insider attacks, so long as the self-stabilization mechanism is implemented correctly. Our paper is about inter-domain routing among a large number of autonomous systems, and a practical extension of BGP. The original motivation of D-BGP was, in fact, to add the self-stabilization property [1,4,5] into BGP in a scalable way.

There have been many proposals for enhancing the robustness of inter-domain routing. For instance, the router configuration checker (RCC) validates low-level BGP policy rules against a high-level correctness specification [12]. Unfortunately, it is hard to obtain a high-level correctness specification as the Internet is managed by different administrative domains with ad hoc coordination. The Inter-domain Route Validation (IRV) Protocol provides a router with a way to validate the correctness of a routing update [14]. It requires a network of IRV servers in the various ASes that routers can consult when they decide to check a route update message.

The Whisper protocol attempts to determine if routing paths are valid using a hash function at each router on the path. Each router adds a hash value to a path-announcement message, allowing a way to flag questionable route updates [36]. The Listen protocol describes a service that can be used to determine whether or not a route is working. This check could be used by D-BGP to assess the health of conflicting paths, and help D-BGP determine the correct path to a prefix.

The Routing Protocol Security (RPSEC) working group under IETF is defining the requirements for securing the routing plane as numerous prevention and validation approaches have been proposed to validate and authorize the route updates. For instance, SBGP [17] relies on a PKI [18] to check the integrity of prefix ownership, topology, and AS path information in a BGP update before using the update. In contrast, SoBGP uses the web-of-trust model, but still requires separately managed services for correct ownership and topology information. References [9,24] provide excellent sources of information for BGP and routing protocol security. Unlike SoBGP, psBGP [16,40] uses a centralized trust model for AS number authentication and a decentralized trust model for verifying the propriety of address prefix assignment.

One key challenge across these proposals is their need for "trusted servers" or oracles to validate the routes, while, at the same time, requiring a valid route in order to reach those services. In other words, they face a chicken-and-egg problem. SBGP requires a PKI infrastructure to manage and validate the public-key certificates for inter-domain routing information. On the other hand, D-BGP is a self-contained routing framework, designed to detect, analyze, and tolerate attacks/failures *without any centralized trust services*.

Inter-domain routing can be characterized by the fact that different ASes are managed independently, with each AS having its own routing policies and peering agreements. At the same time, no single AS has a global view of the Internet. No AS is able to locally determine which route it should use to reach a prefix it does not own, without receiving information from other ASes. In the management plane, network administrators *can* reason about which route to use. In BGP, information received from other ASes is assumed to always be correct. There is no screening of the information and no warning is given to the management plane if this assumption is violated and the information is not correct. In SoBGP, routers use a global topology database to validate the information they receive [44,45]. Clearly, an AS cannot rely on remote access to some database to bootstrap its routing. Unfortunately, this requires routers to obtain enough information to build the database locally. In SoBGP, the data needed to build the topology database has to be validated through the computationally intensive process of digital signature verification. The FIX incident [13], among others like it, suggests that transmitting large amounts of routing information and performing computationally intensive tasks should be avoided.

D-BGP does not present either of these problems. No remote access to a database is required and neither do we need to build a database locally. Unlike SoBGP, computationally demanding digital signatures are not necessary. D-BGP can handle every known MOAS problem. Furthermore, in the worst-case scenario of an attack by a malicious collusion of ASes, D-BGP relays the problem to the management plane and provides diagnostics that enables it to take action. D-BGP can be used in conjunction with the Listen and Whisper protocols. D-BGP also provides automatic resolution of transient problems and mitigation and escalation of persistent ones.

III. DESCARTES BGP ARCHITECTURE

Preliminaries: To simplify the exposition, we abstract an AS to a single BGP router. A *route* is a prefix combined with an AS path, a sequence of AS numbers: $<AS_0, AS_1, \cdots, AS_m>$. AS_m is *P/n*'s origin AS. A *BGP routing table* is a set of routes, or (*P/n*, AS path) pairs[2]. A *BGP update* message containing a route tells BGP routers that receive it that AS_0, the first AS on the AS path, prefers the specified path for traffic destined for *P/n*. A BGP update received by BGP router A is an *MOAS update* if it contains a route (*P/n*, $<AS_0, AS_1, \cdots, AS_m>$) where $\exists\, e \in$ A's routing table s.t. first(*e*) = *P/n* and second(*e*) = ($<AS_0, AS_1, \cdots, AS_r>$) where $AS_m \neq AS_r$.

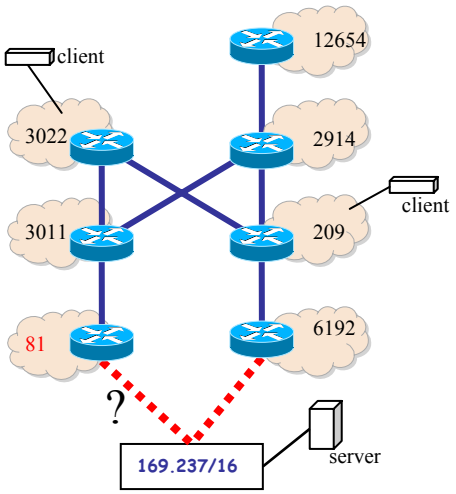

Figure 2: A Potential Routing Conflict.

When a BGP router receives a route update for the prefix *P/n*, it enters the new route into its routing table. If the new route becomes the current best route for *P/n*, the router forwards the route to its peers [28]. In Figure 2, AS81 announces that it is 169.237/16's origin AS when it sends (169.237/16, AS81) in an update to its neighbor AS3011. Previously, AS6192 was the undisputed origin of 169.237/16. The dashed line from AS81, labeled with a question mark, denotes AS81's new ownership claim. Since AS81's update advertises a shorter route to 169.237/16 than the route AS3011 previously held, AS3011 blindly accepts and forwards the update to its neighbors, AS3022 and AS2914. Since the route is the same length as their existing route for 169.237/16, these ASes, in turn, accept and forward the update to AS12654, which accepts the new route but cannot forward it (in this example because it has no other neighbors), and AS209, which neither accepts nor forwards it.

Any of these ASes could identify AS81's update as an MOAS update, since they already had an entry for 169.237/16 in their routing tables, but, under BGP, they do not. If 169.237/16 is not multihomed, then a black hole forms and subsumes all of the ASes in Figure 2, except AS209 and AS6192. In particular, a client in AS3022 can not reach a server in 169.237/16, while a client in AS209 is unaffected.

In contrast to BGP routers, D-BGP routers seek to distinguish legitimate MOAS updates from illegitimate MOAS updates. Failing that, D-BGP mitigates the impact of illegitimate MOAS updates. To this end, D-BGP augments BGP by adding three new roles to BGP routers:

1. *Detectors* notify checkers about potential route-update conflicts;
2. *Checkers* confirm their ownership of a prefix; and
3. *Enforcers,* in the event of a persistent conflict, stop the propagation of route updates deemed invalid under D-BGP.

A. *Conflict*

A *detector* is any D-BGP router that does not forward an MOAS update and instead asks the two ASes involved in a potential conflict to become checkers by sending them a *conflict* message. Figure 3 shows the detector's pseudocode. If a D-BGP router *cannot* or will *not* forward an MOAS update, it *must* become a detector, namely the detector of last resort. If a D-BGP router receives an MOAS update that it can forward, that D-BGP router may choose to become a detector. In this case, it delays forwarding the update, and sends a conflict message to AS81 and AS6192. In Figure 2, AS209 and AS12654 are the detectors of last resort; in the discussion that follows, we assume that they do become detectors in response to AS81's MOAS update.

[2] For clarity of exposition, we use routing table to refer to all three components of a BGP router's routing information base, or RIB.

```
IF MOAS_update(P/n) AND state(P/n) = agreement THEN
  state(P/n) = conflict
  send conflict(P/n, AS_new, AS_old)
  start conflict_timeout(P/n,AS_new,AS_old)
FI
IF state(P/n) = conflict
  AND withdrawal(P/n,AS_new) or withdrawal(P/n,AS_old)
THEN
  update routing table and forward update, if necessary
  state(P/n) = agreement
FI
IF conflict_timout(P/n,AS_new,AS_old) fires THEN
  state(P/n) = persistent conflict
  send persistent_conflict_message(P/n, AS_new, AS_old)
  Start persistent_conflict_timeout(P/n,AS_new,AS_old)
FI
IF persistent_conflict_timout(P/n,AS_new,AS_old) fires THEN
  IF update was held THEN
    update routing table
    forward update
  FI
  state(P/n) = agreement
FI
```

Figure 3: Detector Pseudocode.

Clearly, the ASes that simultaneously claim the same prefix are in a better position than a detector to know whether their claims are correct. For this reason, detectors react to an MOAS update by sending D-BGP's conflict message, over the BGP control plane, to each AS involved in the potential conflict. Table 1 depicts the payload of the conflict message. Its header is not shown, because its format is identical to the standard BGP header and differs only in its use of a new message type. Using the embedded AS path fields, D-BGP forwards the conflict message. The Timestamp field, which uses the format stated in RFC 1305, facilitates forensics (see Section III.E).

Prefix (variable)	Old Path Length (2 octets)	AS Path to AS_{old} (variable)	New Path Length (2 octets)	AS Path to AS_{new} (variable)	Timestamp (8 octets)

Table 1: D-BGP's Conflict Message.

After sending the conflict message, the detector waits for a route withdrawal for *P/n* from one of the two ASes. In Figure 2, both AS209 and AS12654 send conflict messages to both AS81 (AS_{new}) and AS6192 (AS_{old}).

D-BGP depends on a detector springing into action. For every MOAS event, we must guarantee that there will be a detector. Recall that a tree is a connected graph with no cycles and a spanning tree of a graph G is a subgraph of G which is a tree and includes all the nodes of G.

Let AS_{new} be a misbehaving AS that sends out route update *U*. Traffic routed through AS_{new} to prefix *P/n* before update *U* is sent out is already at the mercy of AS_{new}. Thus, we will restrict our attention to route updates that are propagated to at least one router outside AS_{new}'s subtree and therefore can increase the size of AS_{new}'s subtree in the spanning tree defined by BGP for the prefix *P/n*. In other words, after the update, at least one additional AS will route its traffic through AS_{new}. Let D be the set of all detectors for update *U*, *i.e.,* the set of routers that receive the update *U* but do not forward it. Finally, let AS_{origin} be the undisputed origin of *P/n* prior to AS_{new}'s update.

Theorem: For any route update issued by AS_{new} that augments its subtree relative to the existing BGP spanning tree for prefix *P/n*, there exists at least one detector, *i.e.,* $|D| \geq 1$.

Proof: When a route update advertising a new path to *P/n* is sent out, it partitions the AS graph into two sets of ASes: (1) T, those ASes that use and forward the new route, and (2) N, those that do not. Note that $D \subseteq N$ and $N \neq \emptyset$, since AS_{origin} will not use the new route. Either AS_{origin} receives AS_{new}'s update or it does not. Since the AS graph is connected, if AS_{origin} does not receive the update, there exists an AS, AS_d, that receives, but does not propagate U toward AS_{origin}. $AS_{new} \neq AS_d$, since, by definition, AS_{new} seeks to increase the number of routers that route through it. Either AS_{origin} or $AS_d \in D$, so $|D| \geq 1$ ■

B. Verification

Upon receipt of a D-BGP conflict message for the prefix *P/n*, a *checker* verifies its ownership of *P/n*. See Figure 4 for the pseudo-code of a checker. First, a checker checks whether it is, in fact, configured to be *P/n*'s origin. If

not, it never issued a route update containing *P/n* and the misbehaving AS is somewhere between the checker and the detector. The checker logs its receipt of a conflict message for *P/n* for later forensics analysis. After this sanity check, the checker pings the host space of prefix *P/n*. If no host responds, the AS assumes that it is misconfigured. Alternately, the checker can consult an intra-AS authoritative database of prefixes hosted by the AS. If any of these tests fail, an honest checker issues a route withdrawal for *P/n*. In the event of a misconfiguration, such as an origin or export misconfiguration [19], these simple tests allow D-BGP to resolve most MOAS conflicts[3] in real time, without human intervention.

```
IF conflict_message(P/n,AS_new,AS_old)AND Self in {AS_new,AS_old}
THEN
  verify ownership of P/n
  IF not owner THEN
    issue withdrawal for P/n
  FI
FI
IF persistent_conflict_message(P/n,AS_new,AS_old)
    AND Self in {AS_new,AS_old}
THEN
  IF not owner THEN
    issue withdrawal(P/n,Self)
  ELSE
    issue persistent_conflict_deaggregation(Pb/n+1,Self)
  FI
FI
```

Figure 4: Checker Psuedocode.

If a checker still believes itself to own a prefix after these self-checks, it next checks whether *P/n* is multihomed. In other words, it checks whether the other AS involved in the conflict is also a legitimate host of *P/n*. D-BGP-compliant prefixes must register the fact that they are multihoming with each AS that hosts them, or provide a service that each hosting AS can query to learn the identity of a prefix' other origin ASes. The checker can then find out whether there is a true multihoming arrangement either by querying its local database or the service provided by *P/n*. Section VI explains how D-BGP handles non-compliant prefixes.

If, after all of these checks, a checker still believes itself, despite the conflict, to be an origin of *P/n*, it does nothing. In other words, D-BGP optimistically assumes, in line with BGP control plane data analyses, that most conflicts are transient [37,38].

When a detector times out while waiting to see a route withdrawal message from one of two ASes involved in a conflict, the detector notifies each AS that the conflict may have become persistent by sending D-BGP's persistent conflict message. The persistent conflict message is identical to the conflict message, except for a different message type field. The detector could have timed out because the checkers determined that *P/n* is multihomed and therefore neither sent out a withdrawal. When this happens, the checkers ignore the persistent conflict message. If the detector was *not* a detector of last resort and it therefore held an MOAS update, it must forward the update message that triggered it to become a detector. It does so if it times out without receiving a response to the persistent conflict message. A detector may spring into action due to an MOAS update issued by a malicious router that intentionally claims another router's prefix. When such a malicious update occurs, D-BGP can *mitigate*, but not resolve, the problem in real time, as presented next.

C. Persistent Conflict Mitigation

To resolve persistent conflict over *P/n*, network administrators must intervene since D-BGP cannot know whether a conflicting AS' claim is legitimate. So D-BGP first alerts network administrators (see Section III.E) that a persistent conflict over *P/n* between the ASes has occurred. While the conflict persists, D-BGP allows hosts within ASes uninvolved in the conflict to route their traffic to *P/n*, in spite of the conflict. The point of this data plane path resilience is to maintain connectivity, in the face of an adversary, for critical services, such as DNS and PKI, upon which recovery may depend. To this end, D-BGP splits *P/n* into *P0/n+1* and *P1/n+1*, and assigns *P0/n+1* to the AS in conflict with the lower ordinal and *P1/n+1* to the AS with the higher ordinal. Each uninvolved AS can then locally decide which of the two ASes conflicting over *P/n* it trusts more or whether to send its data traffic to both. How to best make this decision is an open problem for future research. Upon receipt of a persistent conflict message for a prefix *P/n* that is not multihomed, each AS in conflict over *P/n* issues the deaggregation *P0/n+1* or *P1/n+1* assigned to it. Figure 5 shows an example, where AS_{new} and AS_{old} are in a persistent conflict over prefix *P/n*, and have each announced the deaggregated prefix, assuming that AS_{new} is a lower-numbered AS. The possible

3 Indeed, router misconfiguration has caused all black hole events we have reviewed.

routes between the ASes are shown by the four solid lines. The short parallel lines across a route denote a route that is not used. AS1 decides locally to route all outgoing traffic addressed to *P/n* to AS_{old} using address enlargement, which is discussed next. AS2 makes the opposite decision, and decides to route *P/n* traffic to AS_{new}.

This deaggregation is always possible. If both P[0,1]/n+1 were claimed prefixes, there could be no conflict over *P/n*. If one of P[0,1]/n+1 were claimed, without loss of generality, *P0/n+1*, the owner of *P/n* acquiesced to that prior deaggregation. Since *P0/n+1* is more specific, the AS that owns *P/n*, when it acquiesced to *P0/n+1*'s announcement, effectively became the owner of *P1/n+1*. In this case, a persistent conflict over *P/n* actually concerns *P1/n+1*. If *P1/n+1* had been itself previously deaggregated, this analysis repeats until either we find that the actual conflict is over a prefix that D-BGP can deaggregate, or we run out of bits, in which case there could not have been a conflict.

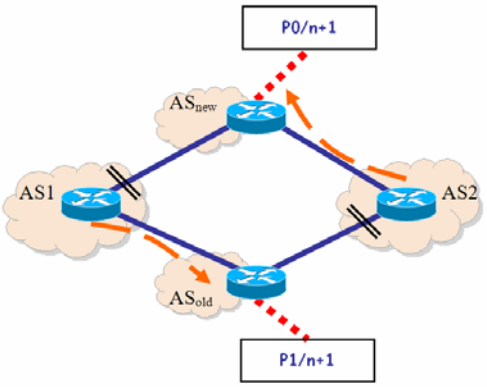

Figure 5: Local Decision During Persistent Conflict.

By default, D-BGP assigns *P0/n+1* to the lower-numbered AS and *P1/n+1* to the higher-numbered one. In Figure 2, AS81 would announce 169.237.0/17 and AS6192 would announce 169.237.128/17. The issuing ASes set the persistent conflict attribute of these deaggregation announcements. This transitive attribute activates D-BGP's enforcers (see Section III.D).

The deaggregated prefixes, namely 169.237.0/17 and 169.237.128/17 in this example, propagate to all ASes under BGP's default forwarding rules. At this point, each AS can locally decide how to handle the conflict. For example, an AS may choose to prefer the lower tier AS involved in a conflict or it may prefer the AS that hosted the prefix prior to the detection of the persistent conflict.

The immediate problem with using deaggregation to handle a persistent MOAS conflict over a prefix is the loss of half of that prefix' host address space. We propose two solutions: *address enlargement* and *DNS remapping*.

D-BGP routers can restore the lost half of the host address space. The strategy is to temporarily enlarge the address space. In Figure 6, AS_s locally decides, in a conflict over *P/n,* to trust the higher-numbered AS, AS_d. For traffic destined for *P/n* that originates within it, AS_s must rewrite the destination address in each packet's IP header from *P* to *P1*. AS_s cannot simply overwrite *b*, the most significant bit in the host portion of the destination address, so it first saves *b* elsewhere, in an address restoration bit. Normal BGP forwarding of data traffic then routes this traffic to AS_d. AS_d restores *b* from the address restoration bit, as shown, so AS_d can properly forward the packet to its final destination.

Where does the address restoration bit live within an IP packet? If D-BGP stored this bit in a packet's data payload, it would need to flag the packet for special processing, either with a flag in the IP header or by tagging the address restoration bit with a bit pattern. The former uses a bit in the IP header to move a bit out of the IP header, while the latter suffers from false positives and slow processing. Neither payload-based approach survives fragmentation by an intermediate, non-D-BGP router. Instead, there are three candidate unused bits that could be pressed into service within the IP header: Type of Service bits 6 and 7, as well as the high-order Flags bit.

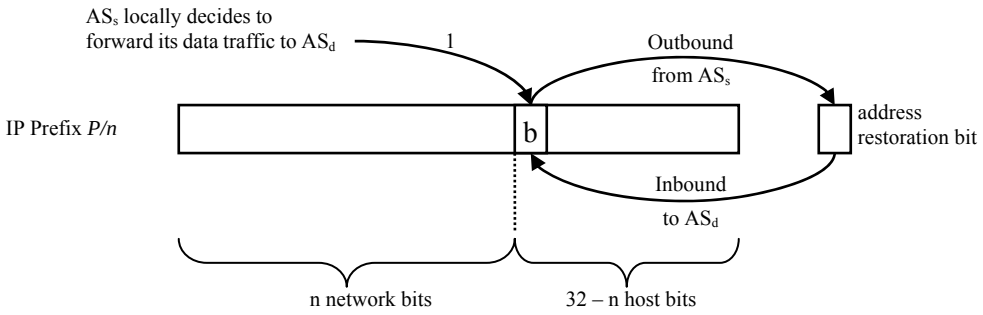

Figure 6: Address Enlargement.

If no bit in the IP header can be allocated for the address reservation bit, D-BGP can employ an alternate strategy. An AS in persistent conflict knows which of its servers are critical. To mitigate the loss of half of its address space, that AS can move these critical servers into that half of the prefix that it claimed when it issued its deaggregation message. Externally, it can rebind these servers in DNS to addresses in its half of the prefix in

conflict and employ network address translation to forward inbound traffic to a server's internal address.

D. Enforcers

All D-BGP routers are *enforcers*. Figure 7 contains their pseudocode. In their role as enforcers, D-BGP routers enforce the assignment of the deaggregated prefixes of *P/n* as specified by the ordinal ranking of the two ASes in conflict. For prefixes whose length is greater than a threshold defined in the management plane, enforcers also allow only one conflict deaggregation in the address space rooted at a given prefix, to prevent an adversary from trivially short-circuiting D-BGP's mitigation strategy by forcing repeated deaggregation until P's host space is exhausted. We do not know exactly to what value to set this threshold. It should be long enough to prevent an adversary from affecting large swaths of the Internet's address space before the adversary's neighbors, in their role as enforcers, start dropping its updates because it has issued too many deaggregations, but short enough to protect most prefixes.

```
IF MOAS_update from neighbor AND MOAS_rate(neighbor) > MOAS_threshold THEN
  drop MOAS_update
FI
IF persistent_conflict_deaggregation(P0/n+1,AS_low) AND AS_low < AS_high
  AND persistent_conflict_deaggregation(P1/n+1,AS_high)
THEN
  state(P/n) = persistent conflict(AS_low,AS_high)
FI
IF withdraw(Pb/n+1,ASx) AND state(P/n) = persistent_conflict(AS_new,AS_old)
    AND AS_x in {AS_new,AS_old}
THEN
  state(P/n) = agreement
FI
IF route update for Q/n' where Q is a subset of P/n
    AND |n'| > prefix_length_threshold AND state(P/n) != agreement
THEN
  drop route update
FI
```

Figure 7: Enforcer Pseudocode.

Enforcers also track the total number of prefix conflicts reported, by AS, and the total number of deaggregations issued, by AS. If the rate at which either of these events occurs exceeds a threshold, enforcers drop all further route updates from that AS until the rate falls below the threshold. Routers attempting to rapidly issue these types of events are throttled by setting a low threshold value. This technique is similar to BGP's route damping technique where routers ignore route updates issued by a router, if that router advertises and withdraws a route too often.

This heuristic not only curbs misconfiguration but also prevents a denial-of-service attack that seeks to destabilize BGP routers by flooding and overwhelming their routing tables. As has been observed in actual black hole events [2,13,21,37,38], if an AS is the source of a number of conflicts or issues numerous deaggregations, it is likely to be malfunctioning or malicious. Thus, all of its subsequent announcements are suspect. Note that enforcers only block control plane updates: data traffic flowing over existing routes to an AS in conflict is unaffected.

E. Management Plane

Network administrators resolve persistent conflicts in the management plane: they troubleshoot the problem, call each other, reconfigure routers, disconnect or replace physical interconnects. In short, their action is unconstrained by the routing infrastructure. This is true of BGP and remains true of D-BGP. Unlike BGP, D-BGP helps network managers in this task by providing superior forensics. In particular, every D-BGP router logs the MOAS updates it receives. In their role as a checker, every D-BGP router logs the receipt of a conflict message, the results of its verification activities triggered by that conflict message, and whether the conflict escalated and became persistent; in their roles as enforcers, they log the ASes whose updates they block and which thresholds those ASes exceeded and when.

With the information stored in these logs, D-BGP allows network administrators to isolate a faulty AS, even when that AS is not directly claiming to be the origin of a prefix, but rather is injecting forged MOAS updates into the control plane. In this case, the AS falsely claimed to be an origin in the forged MOAS update would log, upon receipt of the conflict message, that it was neither the origin nor had sent out the forged MOAS update. Armed with this knowledge, network administrators for the various ASes along a path from a detector to the impersonated AS can cooperate to track the receipt of the MOAS update backward along that path. The injector cannot send the forged MOAS update backwards along the path toward the impersonated AS because no D-BGP (or BGP) router,

using standard path acceptance rules, would accept the update from the injector. Thus, the injector is likely to be either the AS farthest from the detector that acknowledges receipt of the forged MOAS update, or the first that does not. The ability of D-BGP to locate the injector is similar in concept to the WATCHERS intranet router protocol [8]. However, the main difference between D-BGP and Fatih [22,23] is that Fatih is to decide which routers are malicious while D-BGP is merely to raise the flag of confliction among routers. Therefore, D-BGP provides a more practical (scalable) solution on MOAS problem.

IV. THREATS TO DESCARTES BGP

D-BGP extends BGP with new messages and state. In so doing, D-BGP exposes new avenues of attack to an adversary. The new messages are the MOAS update, conflict, persistent conflict, and persistent conflict deaggregation. In a sense, MOAS updates are not new, since they occur within BGP, but D-BGP distinguishes them and treats them differently: in this sense, D-BGP elevates them to new messages. A single, malicious AS, AS_m, can arbitrarily intercept and alter any traffic that traverses it. When issuing MOAS updates for the prefix *P/n*, AS_m can launch three different attacks. It could

1. modify the path in an MOAS update issued by another AS, while still leaving that origin AS unchanged, at the end of the AS path;
2. issue a false MOAS update with itself at the end of the AS path, thereby claiming to be an origin of *P/n*; or
3. forge an MOAS update whose AS path ends at some AS other than AS_m.

Let us consider each separately. First, if AS_m modifies an MOAS update for prefix *P/n* of another AS but does not change the origin AS, then AS_m can disrupt traffic, without causing an MOAS conflict. Although it does not aggravate the problem, D-BGP is designed to deal specifically with MOAS conflicts and, just as current BGP, it is susceptible to such manipulation.

Let us now consider the second attack. If AS_m issues a false MOAS route update claiming to be an origin of prefix *P/n*, AS_m will form a black hole and suck in traffic destined for *P/n*. Wherever AS_m lies within the black hole it forms, D-BGP works correctly so long as the honest AS involved in a conflict receives notification and issues its conflict deaggregation, as described in the Section III.C above. This, in turn, happens as long as: 1) there is at least one detector, and 2) AS_m cannot intercept messages between the detector and the true owner of *P/n*.

It should be clear from the definition of the detector that there is at least one path between the detector and the true owner of *P/n* that is not under the control of AS_m. Therefore, to show that D-BGP works correctly in this scenario, we need only show that there will always be at least one detector, which we proved in Section III.A.

Finally, if AS_m forges an MOAS update for prefix *P/n* "on behalf" of AS_b, causing a conflict with the true owner of the prefix AS_o, then AS_m is either on the path between the detector and AS_o or on the path between the detector and AS_b but not both. If AS_m lies between the true owner AS_o and the detector, then the detector's message will reach AS_b who will issue a route withdrawal and end the conflict. If, on the other hand, AS_m lies between the detector and AS_b then the detector's message will reach the true owner of the prefix who will issue the deaggregation as specified above.

In another line of attack, AS_m could send out false conflict and/or persistent conflict messages. A conflict message causes an AS to become a checker. The most expensive checker test is likely to be the prefix ping test. Thus, the adversary could use conflict messages to launch a denial of service (DoS) attack. As a simple countermeasure, a D-BGP router caches the result of each verification test, notably its ownership of a prefix *P/n*, for a short period of time. This prevents an AS from repeatedly resorting to expensive tests, such as a ping test, to verify that it is an origin of *P/n*. With this countermeasure in place, the handling of a sequence of conflict messages for a given prefix approaches the cost of handling a single message. This countermeasure also improves D-BGP's performance in general, since there may be many detectors for a single conflict sending many conflict messages to each AS in conflict.

AS_m could use a "hole punching" attack to try to circumvent D-BGP. AS_m is interested in the data traffic of servers in *P/n*, whose origin is AS_o. Instead of issuing an MOAS update to claim P/n, it simply issues a deaggregation, such as $PS/n+|S|$ where $|S| > 1$, to claim that portion of *P/n*'s address space that contains those servers. $PS/n+|S|$ is more specific than *P/n*, so AS_o always receives the update and, under D-BGP, can reclaim $PS/n+|S|$ by issuing an MOAS update for it: D-BGP's path resilience mechanism will allow AS_o to restore service to the affected servers.

Since D-BGP enforcers lockdown a prefix in persistent conflict by ignoring deaggregations within that prefix, AS_m could engineer a persistent conflict just to prevent deaggregations within a prefix' address space. In particular, a persistent conflict over the 1 bit prefixes 0/1 and 1/1 in a D-BGP network would prevent rehoming, multihoming, and the delegation of new prefixes. This attack motivates the restriction that D-BGP enforcers only

lockdown prefixes whose length is greater than a threshold set in the management plane. Unlike BGP, D-BGP detect and forwards the persistent conflict event, along with the identities (AS numbers) of the perpetrators, to the management plane, where network administrators would be strongly motivated to fix the problem quickly.

The adversary can also send false persistent conflict messages to force an arbitrary number of D-BGP routers to issue conflict deaggregations, or directly send out the conflict deaggregation itself, subject to the limits imposed by D-BGP enforcers. When compared to BGP, as previously observed, D-BGP conflict deaggregations still allow even non-D-BGP routers to route at least half of their data traffic, given the assumption that host are uniformly distributed within a prefix' host address space. For D-BGP routers, this attack, in the worst case, reduces to all D-BGP routers incurring the overhead of one of the two data plane path resilience approaches proposed previously. Even while D-BGP routers are burdened with this overhead, which we intend to quantify in future work in terms of the end to end latency and CPU load per router it imposes, D-BGP maintains data plane connectivity for critical services, such as DNS.

V. Evaluation

We evaluated D-BGP in two ways. First, we used a network simulator to demonstrate D-BGP's effect on routing in the face of an MOAS event. Second, we used actual BGP updates from an MOAS event to estimate characteristics of D-BGP in such an environment.

A. *Simulation of D-BGP*

We used the scaleable simulation framework network (SSFNET) models [32] to verify the behavior of the D-BGP protocol in a simple network. Figure 6 depicts this network, which adds clients and servers to the network shown in Figure 2 to make the data traffic flow more interesting. We used transmission control protocol (TCP) connections to simulate the data flow. The long dashed lines ending in arrows represent the TCP data traffic flows between three clients and Server 1. The short dashed line ending with arrows shows the TCP data traffic flow between Client 3 and Server 2.

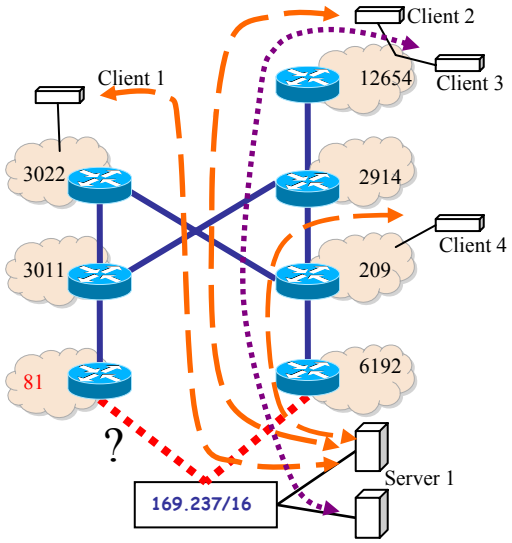

Figure 8: D-BGP Simulation Network Topology.

The graphs in Figures 9 (a), 9 (b), 10 (a), 10 (b), 11 (a), and 11 (b) show the throughput, measured by packet count vs. time, at the two servers in AS6192 — Server 1 and Server 2. The throughput is an indicator of the network's ability to route the data packets of the TCP connections. Figures 9 (c), 10 (c), and 11 (c) show timelines that correspond to the three simulations: MOAS with BGP (Figure 9 (c)), MOAS with D-BGP (Figure 10 (c)), and MOAS with D-BGP and persistent conflict (Figure 11 (c)). For example, at time 70.1 in Figure 9 (c), TCP starts up, and Figures 9 (a) and 9 (b) show the packet count begin to increase shortly after time 70. The events from time 70 up to time 100 are only shown in the first timeline, Figure 9 (c), but are common to all the simulations. Figures 9 (a) and 9 (b) show the baseline case: how BGP handles an MOAS event. Figure 9 (c) outlines the important events and illustrates why the TCP traffic to the servers is affected.

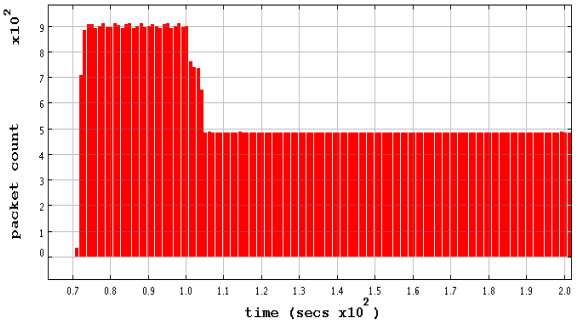

Figure 9 (a): Throughput at Server 1 Without D-BGP.

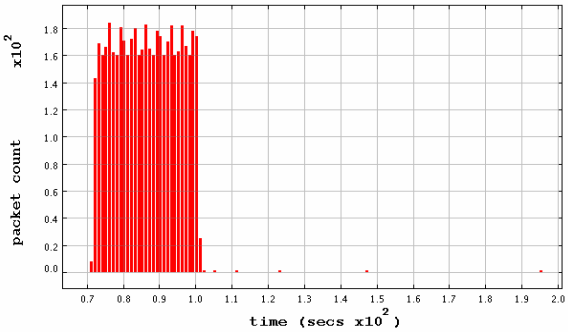

Figure 9 (b): Throughput at Server 2 Without D-BGP.

At an arbitrary simulator time of 100, AS81 claims 169.237/16 and issues an MOAS update. Assuming AS81 is faulty, its MOAS update forms a black hole that draws in all the routers in our network, except AS209 and AS6192, as described in Section III. As the routers accept the bad route and are consumed by the black hole, the TCP connections begin to fail, and the packet throughput to the servers drops off. The only TCP connection that is unaffected is the connection from Client 4 to Server 1, since AS209's route to AS6192 did not change. Note that the amount of throughput due to Client 4 is much higher than the other two clients, due to its proximity to Server 1. Figure 9 (a) shows how the data packets on this connection continue to flow unaffected until the simulation ends.

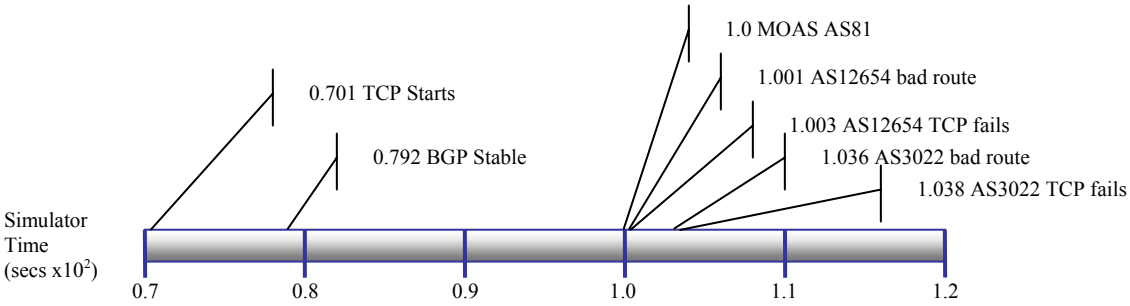

Figure 9 (c): Timeline of Simulation Events Without D-BGP.

Figures 10 (a),10 (b), and 10 (c) show the case where D-BGP is active and the conflict is a misconfiguration. The misconfigured router performs an immediate route withdrawal when notified of the error. The route withdrawal allows the routers to re-stabilize with valid routes. Figures 10 (a) and 10 (b) show how the TCP traffic is able to recover to the same level as before the MOAS event. Figure 10 (c) shows the main events that occur starting at simulator time 100 as D-BGP works to clear the MOAS event.

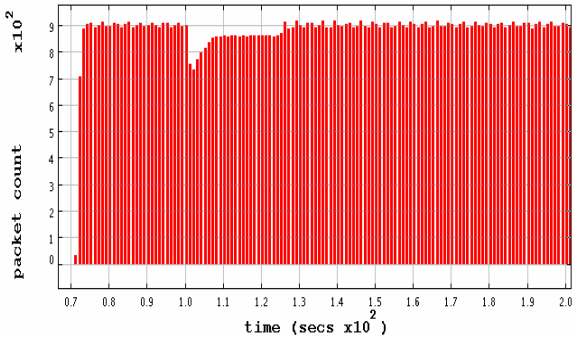

Figure 10 (a): Traffic at Server 1 with D-BGP.

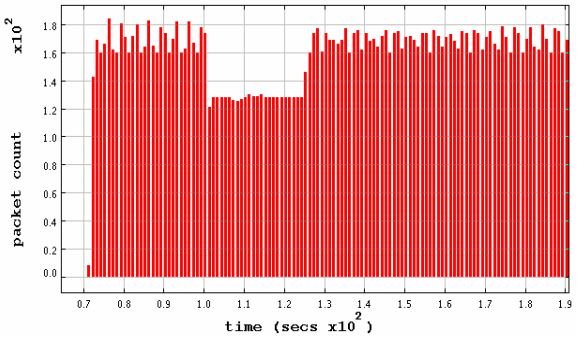

Figure10 (b): Traffic at Server 2 with D-BGP.

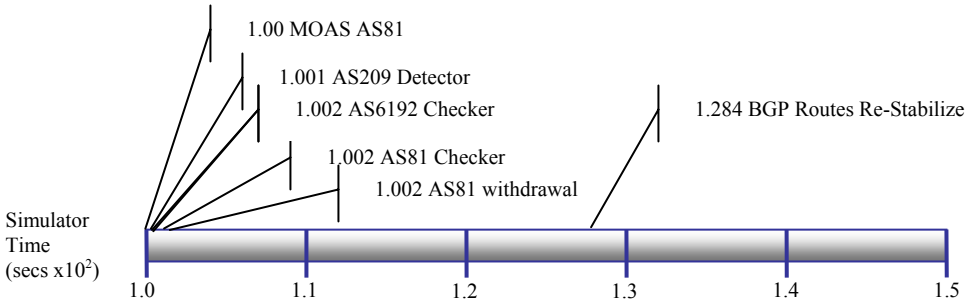

Figure 10 (c): Timeline of Simulation Events with D-BGP.

Figures 11 (a), 11 (b), and 11 (c) illustrate the case where D-BGP is active and a persistent conflict occurs. AS209 becomes a detector and notifies AS81 and AS6192 about the problem, but AS81 does not withdraw the route. AS209 times out, using an arbitrary timeout of 10 seconds, due to the lack of route withdrawal, and sends out a persistent conflict message to AS81 and AS6192. When AS6192 receives the persistent conflict message, it mitigates the MOAS event by deaggregating the IP prefix in conflict. The deaggregration update allows some of the TCP connections to restart transfer. Address enlargement would allow all of the TCP connections to restart, but address enlargement is not implemented in the current version of the simulator. Some of the TCP connections to

Server 1 are affected and cannot restart, which shows up as throughput that does not return to the pre-MOAS value in Figure 11 (a). All TCP connections to Server 2 restart, and the throughput at that server returns to the pre-MOAS value in Figure 11 (b). Figure 11 (c) shows the timeline of D-BGP events in this final case of persistent conflict, starting at simulator time 100.

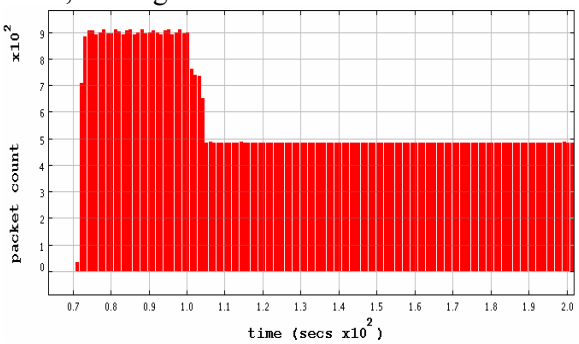

Figure 11 (a): Traffic at Server 1 in Persistent Conflict.

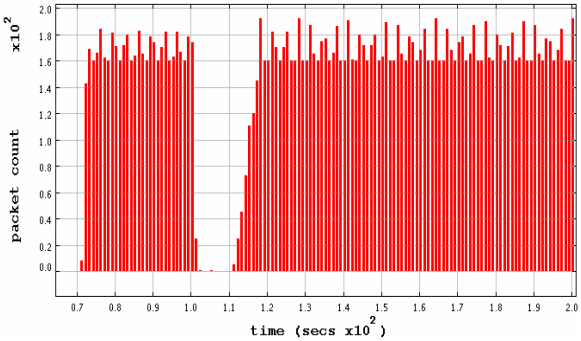

Figure 11 (b): Traffic at Server 2 in Persistent Conflict.

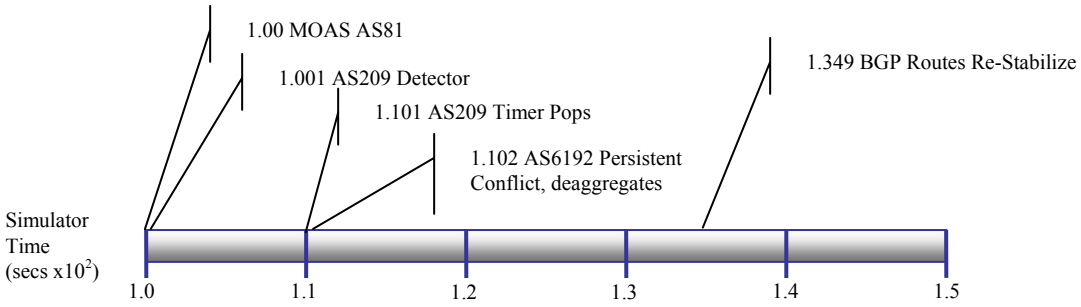

Figure 11 (c): Timeline of Simulation Events for Persistent Conflict.

One final comparison of the simulation cases can be made using the number of BGP and D-BGP messages sent during the simulation. Table 2 shows this comparison. D-BGP requires approximately 20% additional messages to re-stabilize the routers after the MOAS event in the case of misconfiguration. In the persistent conflict case, 28% more messages are required in order to mitigate the black hole, and restore some connectivity in the simulated network. Since D-BGP uses the same forwarding strategy as BGP, D-BGP overhead will scale linearly with the number of edges in the graph of ASes.

Event	BGP messages sent	D-BGP messages sent	Total messages sent	Result
MOAS with BGP (misconfiguration)	74	0	74	Black hole
MOAS with D-BGP (misconfiguration)	81	8	89	Black hole resolved
MOAS with D-BGP (persistent conflict)	84	11	95	Black hole mitigated

Table 2: Comparison of Message Count for the Three Simulations.

B. Detector Population

In Section III.B, we proved that there is at least one detector. For D-BGP to scale well, there cannot be too many detectors. We data-mine BGP traffic data in order to estimate the number of routers that would have had to become detectors if D-BGP had been deployed during an actual black hole event.

Definition: An *event horizon* is the set of edges that connect detectors to the routers that used and propagated a route update to the detector.

An event horizon partitions the AS graph into those ASes that did and those that did not change their route

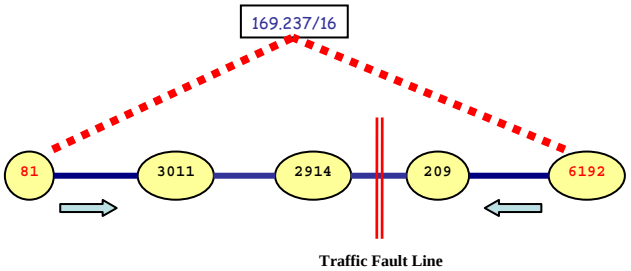

Figure 12: An Event Horizon.

after a route update message was broadcast. Since many routers can become detectors, if we project the AS graph onto a multi-dimensional space with the AS that issued an MOAS, we can view the event horizon as a surface. Figure 12 depicts an event horizon that forms when AS81 claims the ownership of prefix 169.237/16 that was previously owned by AS6192. AS2914 is the detector of last resort. We reduce the problem of estimating the number of detectors to finding and estimating the size of the event horizon.

The RIPE [26] and Oregon Route Views [31] projects have provided us with BGP traffic raw data for the past six years. These projects record BGP traffic at a number of BGP routers called observation points. When a newly received route update changes the current best path, a BGP router forwards that update. We are able to tell whether a false MOAS update became the current best route at each observation point. We can also tell whether an observation point is, or is not, inside the event horizon formed by that MOAS.

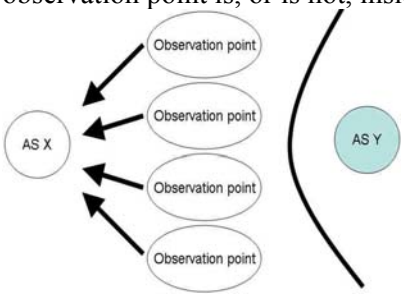

Figure 13 (a): Event Not Recorded.

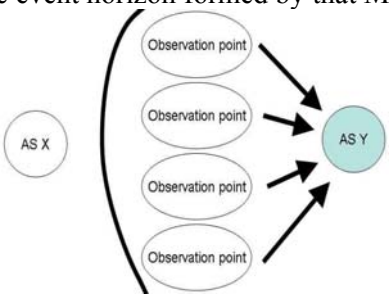

Figure 13 (b): All Observation Points Are Affected.

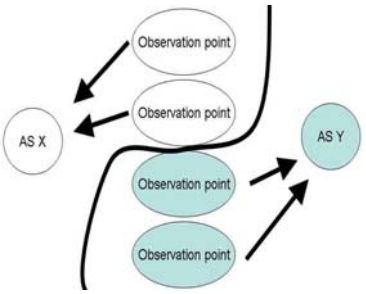

Figure 13(c): Some of the Observation Points Are Affected.

1) Location of Event Horizon

When an MOAS event for the prefix *P/n* occurs: none, all, or some of the observation points could change their current best route to *P/n*. When none change (Figure 13 (a)), the MOAS event is not recorded. When all change (Figure 13 (b)), the MOAS event is recorded and we can infer the location of the event horizon. When some change (Figure 13 (c)) the MOAS event is also captured and the event horizon lies somewhere between the observation points.

We used two approaches to analyze BGP raw data: *per-day* and *per-update* analysis. *Per-day* analysis focuses on the BGP routing table changes between consecutive days, while *per-update* analysis focuses on the BGP routing table changes caused by each BGP update message. When processing raw BGP data, we start with *per-day* analysis. Once we have found an interval in which a significant MOAS event occurred, we switch to *per-update* analysis to provide finer granularity.

Our related research [37], showed that AS15412 caused an MOAS storm on April 6 2001. Therefore, we applied *per-update* analysis to find event horizons in the raw BGP traffic from RIPE of April 2001. The results of this analysis showed the propagation of the falsely announced BGP update messages precisely.

On April 6, 2001 at 5:21pm (GMT time), AS15412 began sending many BGP update messages that falsely claimed that AS15412 was the origin AS of a large set of prefixes. These updates triggered a sequence of MOAS events for the claimed prefixes. Each affected prefix *P/n* has an event. Along the event horizon are detectors of last resort. During the black hole event, the fourteen observation points in RIPE recorded that AS15412 announced 30093 prefixes. This is a lower bound: AS15412 could have announced more than the 30093 prefixes that RIPE captured (Figure 13 (a)). AS15412 previously owned only 5 of the 30093 prefixes it claimed and thus caused 30088 MOAS events.

Among the 30088 prefixes falsely claimed by AS15412, the event horizons for 29016 prefixes (97%) fell between two sets of the 14 observation points (Figure 13 (c)) while, for the remaining 1072 prefixes (3%), each prefix' event horizon lay between the legitimate origin AS and 14 observation points (Figure 13 (b)). In summary, we have verified that we can find event horizons and estimate their location for the April 6, 2001 AS15412 case.

2) Size of Event Horizon

Let *P/n* be a prefix, AS_X and AS_{OP} be ASes where AS_{OP} is in particular one of the observation points in BGP data, and α and β be arbitrary sequences of ASes. Then ASPath (*P/n*) = "$AS_{OP} - \alpha - AS_X$" denotes the AS path from the origin AS_X to the observation point AS_{OP} for the prefix *P/n*. An MOAS occurs whenever AS_{OP} receives ASPath (*P/n*) = "$AS_{OP} - \beta - AS_Y$", and $AS_X \neq AS_Y$.

In order to estimate the number of detectors at the edge of an event horizon formed by a black hole conflict for prefix *P/n*, we need at least one route from an observation point to each of the conflicting origin ASes. Given these two AS paths, we can construct the path "$AS_X - \alpha - AS_{OP} - \beta - AS_Y$". If any router AS_Z appears twice along the path, there is an alternative path from AS_{OP} to AS_Z and the path from AS_X to AS_Y is not optimal. Hence, to

calculate the optimal route from AS_X to AS_Y, we excise the routers on the path between two appearances of Z until no router appears twice in the path. Consider ASPath (140.113.0.0/16) = "AS4608 – AS1221 – AS16779 – AS1 – AS7018 – AS1659 - AS9916" and ASPath (140.113.0.0/16) = "AS4608 - AS1221 - AS1 - AS3561 - AS15412." These paths share AS1, so the optimal path from AS9916 to AS15412 is "AS9916 - AS1659 - AS7018 - AS1 - AS3561 - AS15412." Assuming all BGP routers along the paths employ the default route selection policy, which prefers the shortest route, AS7018 would become the detector in this example.

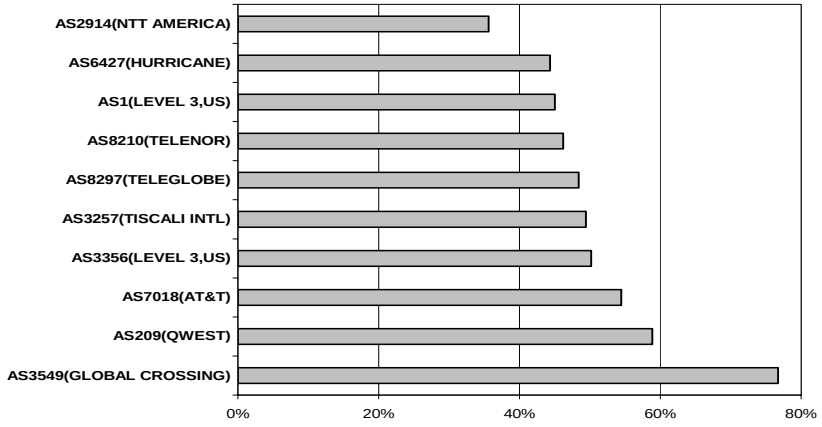

Figure 14 – Top 10 ASes by detected MOAS events

In Section V.B.1, we enumerated the three ways an MOAS event for a prefix *P/n* manifests itself in our data set. In the case depicted by Figure 13 (a), the MOAS event is not captured and we can estimate neither the event horizon nor the number of detectors that compose it. When every observation point changes its best route to *P/n*, as shown in Figure 13 (b), every observer has two routes to the prefix in conflict *P/n.* Therefore, we can estimate the edge of the event horizon. Figure 13 (c) depicts the case where only some of the observation points change their best route to *P/n*, those observers that change had paths to each AS in conflict and, again, we can calculate the edge of the event horizon. From those observation points that did not capture the event, we estimate the path from observation point to the newly announced conflicting origin AS_Y as follows: we extract all current available routes to AS_Y from the observation point's routing table. Among those routes, we flag the shortest route as optimal and use it as the second input for estimating the edge of the event horizon.

As shown in Section V.B.1, 29016 prefixes fell into the third case and we had to extract the route from the observation point to AS15412. Fortunately, AS15412 previously owned 5 prefixes: 62.216.128.0/20, 62.216.144.0/24, 62.216.148.0/24, 62.216.149.0/24 and 62.216.151.0/24. We used this information to estimate the edge of the event horizon for all the routers that were not captured.

If the network had been running D-BGP on April 6, 2001, 933 detectors would have detected all 30088 MOAS events that occurred on that day. In Figure 14, the Y-axis ranks the top ten ASes in order of the number of conflicts they detected. AS3549 was able to detect 77% of the conflicts during April 2001. Moreover, for any prefix *P/n*, there are 8.88 detectors available on average with a standard deviation of 2.33.

In summary, we conclude that a small group of detectors are able to detect a large number of falsely announced prefixes.

VI. DEPLOYABILITY

ASes that use D-BGP can avoid being pulled into a black hole by becoming detectors. Furthermore, D-BGP aware ASes can locally decide where to send traffic to in the case of a conflict. Even if an AS "chooses not to decide" or is not D-BGP aware, at least half of the conflicting prefix' address space is still reachable[4].Thus, there is an incentive for D-BGP adoption.

From the point of view of a standard BGP router in a network partially composed of D-BGP routers, a D-BGP router is functionally indistinguishable from another BGP router, since D-BGP is a superset of BGP. Normal BGP operation is not disturbed by the adoption of D-BGP.

In an incremental deployment scenario, D-BGP could create a D-BGP overlay on the existing BGP routing protocol. In order to achieve an overlay, the IP address of the neighbor D-BGP routers needs to be determined, and once determined the IP address can be used to create a multi-hop D-BGP connection. The process of discovering the neighboring D-BGP routers consists of two parts: 1) D-BGP routers add or update a new transitive attribute containing an IP address into route updates, and 2) D-BGP routers must monitor route updates looking for the new

[4] The whole address space can still be reached if the more elaborate path resiliency countermeasures proposed are adopted.

attribute. The attribute would contain the AS number, as well as the IP address of the next hop D-BGP router. Each D-BGP router would then know the closest D-BGP capable router on each AS path that is received. The D-BGP next hop information is used to create a multi-hop D-BGP connection to the closest D-BGP router on each path. A multi-hop D-BGP connection can only form if a pre-existing peering agreement exists between the AS detecting a D-BGP router and the AS managing the detected D-BGP router.

In an incremental deployment setting, we cannot guarantee that a detector will flag every MOAS event, simply because the detectors of last resort could be BGP, not D-BGP, routers. However, the fact that any D-BGP router that could be pulled into a black hole has the option and an incentive to become a detector itself mitigates this problem. Using the D-BGP next hop information it gathers, a D-BGP router can use heuristics to help determine if it should become the detector. When deciding whether to forward a route update a detector, it knows if the next hop router is a non D-BGP router. In addition, it can determine the number of hops to the next D-BGP capable router. If its neighbors are all only BGP capable, the D-BGP router has more incentive to become the detector, since each hop that is not D-BGP capable raises the risk that there will be no detector.

During a conflict, the D-BGP checker role may go unfilled, since both routers involved in a conflict may not be D-BGP capable. In this case, the nearest D-BGP router to a BGP checker will simply drop the route conflict message. Of course the D-BGP router can log this event for later forensic use. If the second router involved in the conflict is a D-BGP router, then D-BGP can still work to provide access to half the affected address space via conflict deaggregation. Even ASes that do not run D-BGP will still route half of their traffic correctly, assuming a uniform distribution of hosts within a prefix.

Ideally, D-BGP compliant prefixes that enable checking as described previously should be used. But, if not, D-BGP has a safe default: In the absence of conclusive information that a prefix *P/n* is multihomed, D-BGP assumes that *P/n* is not, and, in the event of a persistent conflict, falls back on one of its data plane path resilience mechanisms which maintain connectivity to *P/n*, at the cost of some overhead. Note that this improves on BGP by allowing connectivity even from AS that would be within a black hole. A similar problem may occur if one of the routers in an MOAS event is a BGP router and cannot become a checker. In particular, an adversary can selectively attack such routers. Under such attacks, D-BGP performs no worse than BGP and still provides partial connectivity.

VII. Conclusion And Future Work

We have presented the D-BGP framework, which can detect conflicts over address prefix ownership in BGP, and resolve the conflicts in real time. SSFNET simulations demonstrated the ability of D-BGP to handle both conflict caused by misconfiguration as well as a black hole.

D-BGP imposes new roles and responsibilities on BGP routers. We are working on quantifying this overhead both during the normal operation of D-BGP and when D-BGP is itself under attack. We have proposed two mechanisms for maintaining path resilience in the data plane during a black hole – by rebinding critical servers into the conflict deaggregation prefix assigned to an AS and by restoring the host bit lost by the conflict deaggregation. We plan to integrate D-BGP's functionality with BGP and deploy it on DETER [11] to measure the performance and overhead of D-BGP, similar to the our work in [38,46].

References

[1] A. Agarwal and J.W. Atwood, "A Unified Approach to Fault-Tolerance in Communication Protocols Based on Recovery Procedures," IEEE/ACM Transactions on Networking, Vol. 4, No. 5, October 1996.

[2] "Anatomy of a leak: AS9121," NANOG 34, 2005.

[3] A. Arora and M. Gouda, "Distributed Reset," IEEE Transactions on Computers, Vol. 43, No. 9, September 1994.

[4] A. Arora and M. Gouda, "Closure and Convergence: A Foundation of Fault-Tolerant Computing," IEEE Transactions on Software Engineering, Vol. 19, No. 11, November 1993.

[5] A. Arora and S.S. Kulkarni, "Detectors and Correctors: A Theory of Fault-Tolerance Components," International Conference on Distributed Computing Systems (ICDCS), 1998.

[6] S. Bellovin, J. Ioannidis, and R. Bush. "Position Paper: Operational Requirements for Secured BGP" DHS Secure Routing Workshop, March 2005.

[7] S. Bellovin and A. Zinin, "RFC 4278, Standards Maturity Variance Regarding the TCP MD5 Signature Option (RFC 2385) and the BGP-4 Specification ", January 2006.

[8] K. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. Olsson, “Detecting Disruptive Routers: A Distributed Network Monitoring Approach,” IEEE Symposium on Security and Privacy, pages 115--124, 1998.
[9] K. Butler, T. Farley, P. McDaniel and J. Rexford, “A Survey of BGP Security Issues and Solutions,” Technical Report TD-5UGJ33, AT&T Labs - Research, Florham Park, NJ, February 2004. (Revised June 2004).
[10] D. Clark, C. Partridge, J.C. Ramming, and J.T. Wroclawski, “A Knowledge Plane for the Internet,” ACM SIGCOMM’2003.
[11] Cyber Defense Technology Experimental Research project, http://www.isi.deterlab.net.
[12] N. Feamster and H. Balakrishman, “Detecting BGP configuration Faults with Static Analysis,” Proc. Networked Systems Design and Implementation, May 2005.
[13] “The Day the Internet Died - Courtesy of the Florida Internet Exchange,” http://www.flix.net/
[14] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. “Working around BGP: An incremental approach to improving security and accuracy in interdomain routing,” In Proceedings of Symposium on Network and Distributed System Security (NDSS’03), February 2003.
[15] G. Huston. BGP Routing Table Analysis Reports. http://bgp.potaroo.net/, 2006.
[16] E. Kranakis, P.C. van Oorschot, T. Wan. "On Inter-domain Routing Security and Pretty Secure BGP (psBGP)," Carleton University, School of Computer Science, Technical Report TR-05-08, September 2005.
[17] S. Kent, C. Lynn, J. Mikkelson, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," Proceedings of ISOC Network and Distributed Systems Security Symposium, February 2000.
[18] S. Kent and L. Millett, Ed., "Who Goes There? Authentication Through the Lens of Privacy," National Academies Press, 2003.
[19] R. Mahajan, D. Wetherall, and T. Anderson, “Understanding BGP misconfiguration,” In Proceedings of the ACM SIGCOMM Conference, Aug. 2002.
[20] A. Mankin, D. Massey, C. L. Wu, S. F. Wu, L. Zhang, “On Design and Evaluation of Intention-Driven ICMP Traceback,” ICCCN’2001.
[21] S. Misel, “Wow, AS7007!” http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html
[22] A Mizrak, Y. Cheng, K. Marzullo and S. Savage, ”Fatih: Detecting and Isolating Malicious Routers via Traffic Validation,“ IEEE Transactions on Dependable and Secure Computing, 3(3), July 2006.
[23] A. Mizrak, Y. Cheng, K. Marzullo and S. Savage, “Fatih: Detecting and Isolating Malicious Routers,” Proceedings of the IEEE Conference on Dependable Systems and Networks (DSN), June 2005.
[24] D. Pei, L. Zhang, and D. Massey, “A Framework for Resilient Internet Routing Protocols,” IEEE Network, March 2004.
[25] R. Perlman, "Network Layer Protocols with Byzantine Robustness," MIT/LCS/TR 429, October 1988.
[26] Réseaux IP Européens Network Coordination Centre, http://www.ripe.net/
[27] RFC – 1305 “Network Time Protocol (Version 3) Specification, Implementation and Analysis,” IETF.
[28] RFC – 1771 “A Border Gateway Protocol 4 (BGP-4),” IETF.
[29] RFC – 2328 “OSPF version 2,” IETF.
[30] RFC – 4080, “Next Steps in Signaling (NSIS): Framework,” IETF.
[31] University of Oregon Route Views Project , http://www.routeviews.org/
[32] Scalable Simulation Framework, http://www.ssfnet.org/
[33] F. Schneider, S. Bellovin, Ed., "Trust in Cyberspace," National Academy Press, 1999.
[34] G. Siganos and M. Faloutsos, “A Blueprint for Improving the Robustness of Internet Routing,” 2005
[35] G. Siganos and M. Faloutsos, “Detection of BGP routing misbehavior against Cyber-Terrorism,” Military Communications Conference (MILCOM), 2005.
[36] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, “Listen and Whisper - Security mechanisms for BGP,” Proc. Symposium on Networked Systems Design and Implementation (NSDI'04), March 2004.
[37] S. Teoh, K. Ma, S. F. Wu, D. Pei, L. Wang, L. Zhang, D. Massey, and X. Zhao, “Visual-based Anomaly Detection for BGP Origin AS Change (OASC) Events,” DSOM, 2003.
[38] S. Teoh, K. Zhang, S. M. Tseng, K. Ma, and S. F. Wu, “Combining Visual and Automated Data Mining for Near-Real-Time Anomaly Detection and Analysis in BGP,” ACM VizSec/DMSEC 2004 workshop, conjunction with ACM CCS, October 2004.
[39] B. Vetter, F. Wang, and S. F. Wu, “An experimental study of insider attacks for the OSPF routing protocol,” ICNP’1997.

[40] T. Wan, E. Kranakis, P.C. van Oorschot. "Pretty Secure BGP," Network and Distributed System Security Symposium (NDSS'05), Feb. 2005.
[41] T. Wan and P.C. van Oorschot. "Analysis of BGP Prefix Origins During Google's May 2005 Outage," the 2nd International Workshop on Security in Systems and Networks Apr. 2006.
[42] T. Wan, P.C. van Oorschot, E. Kranakis. "A Selective Introduction to Border Gateway Protocol (BGP) Security Issues," NATO Advanced Studies Institute on Network Security and Intrusion Detection, IOS Press, 2006.
[43] F. Wang "Vulnerability analysis, intrusion prevention and detection for link state routing protocols," Ph.D. dissertation, NCSU, 2000.
[44] R. White "Deployment considerations for secure origin BGP (soBGP)," Internet Draft, 2002.
[45] R. White, D. McPherson, S. Sangli, "Practical BGP," Addison-Wesley, 2005.
[46] K. Zhang, S. Teoh, S. M. Tseng, R. Limprasittipom, C. N. Chuah, K. Ma, and S. F. Wu, "Performing BGP Experiments on a Semi-Realistic Internet Testbed Environment," Accepted by The 2nd International Workshop on Security in Distributed Computing Systems.
[47] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. "An Analysis of BGP Multiple Origin AS (MOAS) Conflicts," ACM SIGCOMM Internet Measurement Workshop, Nov. 2001.