You Can't Handle the Lie: Next-Hop Verification in BGP

Clay Thomas
claytont@cs.princeton.edu

Gavriel Hirsch
gbhirsch@cs.princeton.edu

January 8, 2019

Abstract

This paper presents a new protocol to be run by autonomous systems in a network alongside BGP, that reduces the set of contexts in which other AS's are incentivized to announce lies about the network. We show that by using this protocol to share information between different AS's, it is possible to detect a broader class of lies than would be possible by just doing path verification (such as in S-BGP), without having to deal with the complexity of extensive data-plane level monitoring. We also discuss the advantages and disadvantages of this new approach, as well as the contexts in which it still does not suffice.

1 Introduction

Routing on the Internet involves many distinct Autonomous Systems (AS's), each with its own data sources, destinations, and links; as well as its own preferences over how traffic is routed. An AS may prefer that the traffic it sends and receives be sent over the shortest path, in order to decrease latency; or it may prefer to send its traffic through or avoiding specific other AS's for economic incentives, due to contracts between AS's about routing costs; or it may prefer to avoid certain other AS's, if it is concerned about malicious activity. In the other direction, an AS may also prefer to attract or deter traffic from certain other AS's, again for economic incentives or perhaps even to spy on certain traffic.

These AS's typically use the Border Gateway Protocol (BGP) to announce routes to neighbors and learn routes from neighbors in the control plane, and to then choose how to actually route traffic in the data plane. However, BGP does not actually

enforce any requirement that an AS route traffic in a way that matches its announcements. Thus, due to all of the various (often conflicting) preferences that AS's have over how traffic is routed, these AS's can often have incentives to lie in the control plane about what they will actually do in the data plane.

We would like to be able to efficiently detect lies made by AS's in BGP announcements. However, verifying routes directly in the data plane typically involves a large overhead, such as in [PS03]. In addition, in general there will be at least one AS that knows what each other AS is truly doing, namely the one that directly receives traffic from it. As a result, if AS's are willing to collaborate then there is information that can be used to detect the existence of lies, without requiring overly extensive monitoring of traffic in the data plane. In this paper we present a new protocol that we call *Next-Hop Verification* which allows AS's to use this information to catch certain types of lies.

In the rest of Section 1 we discuss existing work, and we briefly outline our results and their limitations. Section 2 informally presents the model we use. Section 3 describes the next-hop verification protocol. Section 4 states and proves some theorems about the implications of using next-hop verification, and goes through examples of concrete scenarios to compare what would be possible with and without next-hop verification. Finally, Section 5 offers some conclusions and suggestions for follow-up work.

1.1 Previous work and our contributions

Much work has been done on analyzing BGP through game-theoretic models in which AS's may act strategically given their incentives. [LSZ08] shows that in a general set of contexts, a form of verification called *path verification*[1] ensures that no group of AS's can get strictly better routes for its traffic by telling lies if everyone else is telling the truth.

However, lying can potentially give other benefits beyond getting better routes for your own traffic. As mentioned before, an AS may also have incentives to attract or deter traffic by lying. [GHJ+08] analyzes BGP games in the presence of these types of incentives and shows that in many scenarios, even using path verification does not suffice to disincentivize lying. They also introduce another form of verification called *loop verification*, which is simpler but weaker, and describe conditions under which path and loop verification do disincentivize lying. However, they admit that many of these conditions are unreasonably strong, such as requiring that AS's always announce either all paths they are aware of or none at all to all of their neighbors.

[1] In the original paper they refer to it as route verification.

There is also much discussion of convergence in BGP. [LSZ08] argues that if we assume that the network infrastructure is not changing over time and that each AS makes BGP announcements based solely on a ranking of paths that is also constant over time, then subject to a condition called *No Dispute Wheels* the network will converge to a stable set of routes. In more general contexts though, convergence becomes very hard to reason about. Thus, in this paper we choose to focus on what happens after convergence. We show that if the network were to converge to a state that depends on lies, we would then be able to catch the lies and shame the liar. As a result, it should not be beneficial to lie in a way that leads to that state.

As in the papers discussed above, our new *next-hop verification* protocol mostly runs in the control plane. However, it also uses minimal information from the data plane in order to verify the purported network equilibrium. It also allows us to catch lies even in the context where preferences involve traffic attraction and deterrence, without using path verification or relying on the strong assumptions of [GHJ+08]. Our general analysis of next-hop verification does however use the strong assumption that there is only a single lying agent. We also for simplicity and power focus on the situation in which everyone else participates fully. That said, the protocol would still provide some value even with only partial participation, and we will discuss this in Section 5. Finally, it is worth noting that next-hop verification is "bulkier" than loop verification, as it essentially has to distribute information across the whole collection of AS's, as well as do some minimal data-plane monitoring.

2 Model Details

Here we present an informal description of our model for the interactions between AS's.

We model the network of AS's as an undirected graph, with a node for each AS and an edge between any two AS's that can directly communicate with each other without going through a third AS. We assume that the graph is a single connected component, so any AS can in theory interact with any other AS (although in practice, say if an intermediate AS intentionally drops traffic, this may not always actually be possible).

There is a unique destination AS d. In practice, if N is the number of AS's then one could imagine having N distinct versions of the problem, each of which has a different destination AS. There also may be a unique malicious AS m.

2.1 BGP framework

In the BGP framework, AS's can announce the existence or removal of paths to each other. Each AS has an import policy that determines how it responds to path announcements from neighboring AS's. In this paper we ignore any concerns of storage for keeping track of all announcements from neighbors, so we assume that any newly observed path is added to a table, unless the receiving AS is already in the path. In this case it will either ignore the path announcement, or if the AS never announced the subpath containing itself it will raise an alarm that another AS has exported a false path. We also assume that on hearing an announcement, the AS can only take actions related to the full path, and not related to particular subpaths.

Each AS also has an export policy which determines how it will communicate the paths it is aware of to other AS's. In some settings, such as that of [GR01] in which for example customers will not route traffic between two of their providers, an AS may prefer not to announce all of the paths it knows about to all neighbors. Relatedly, an AS may make different announcements to different neighbors.

Finally, each AS has some preferences over how the actual traffic in the network flows. The AS's will choose a strategy, namely their import and export policies, based on these preferences.

We assume that the collection of strategies leads the network to converge to a stable solution. See [LSZ08, GHJ$^+$08, GSW99] for examples of more formal details about proving different types of convergence and what assumptions are necessary.

Definition 1. *We say that an AS* $\boldsymbol{m}$ *is* ***lying*** *if the stablility of the post-convergence equilibrium depends on a neighbor thinking that* $\boldsymbol{m}$ *is acting in a way that is inconsistent with the true equilibrium.*

We say that an AS is ***honest*** *if it is not lying.*

2.2 Verification

As mentioned before, various methods of catching lies have been suggested in the literature.

Definition 2. *In a network using* ***path verification*** *it is impossible for AS's to announce that they are using paths which were not already announced to them.*

Some extensions to BGP, such as S-BGP, can enforce path verification. However, it requires additional overhead as well as universal adoption [LGS13].

Definition 3. *In a network using* ***loop verification*** *no AS will use an export policy that involves not sending a path to a neighbor specifically because that neighbor is*

already in the path. In addition, if an AS ever sees a path containing itself that it did not announce, it will "raise an alarm", with the idea that the offender can be publicly shamed.

Note that if instead export policies did not send paths to neighbors who are already in them, the alarming in loop verification could not always be done.

Since loop verification is very minimal and easy to adopt we will assume that the network uses it.We do not assume that the network uses path verification, though we will show that *next-hop verification* actually handles a larger class of scenarios than path verification would.

maybe we don't want to do this

3 Next-Hop Verification Protocol

Once convergence has occurred, we propose the use of the following protocol by all of the AS's. Note that the malicious AS m may lie and not follow the protocol in order to avoid being caught, but since convergence has already occurred we assume that none of the other AS's have an incentive to lie or even to hide information they know.

Each node maintains a queue of queries which it needs to answer. A query is denoted $Q(a, b, d)$, representing a node announcing that a uses b as its next-hop in its path to destination d.

Denote the acting AS by n
function INITIALIZE
  for each hop (a, b) in n's path to d **do**
    Add the query $Q(a, b, d)$ to your query queue
function RESPOND($Q(a, b, d)$)
  if n previously responded to $Q(a, b, d)$ **then**
    return
  if $n = a$ **then**
    if n does not use b as its next hop for d **then**
      "raise the alarm"
    return
  if $n = b$ **then**
    if a does not use n as its next hop for d **then**
      "raise the alarm"
    send the query $Q(a, b, d)$ to all neighbors
  else (i.e. $n \neq a, b$)
    if a uses n as its next hop for d **then**

"raise the alarm"
else
send the query $Q(a, b, d)$ to all neighbors
function MAIN
for each query $Q(a, b, d)$ in *queue* **do**
RESPOND($Q(a, b, d)$)
clear the *queue*

3.1 Additional notes

- In the case where n is responding to a query $Q(a, n, d)$, it needs to check whether a actually forwards traffic directly to n for destination d, which must be done in the data plane. Accordingly, each AS should keep a flag for each other (neighboring AS) $\times$ (dest AS) pair, representing whether the first AS ever directly sends n traffic destined for the second AS.
- If an AS has too many neighbors and/or destinations and keeping all these flags becomes unmanageable, each AS can have a policy for determining which pairs it thinks are important to monitor for.

4 Results/examples

5 Conclusion

References

[GHJ+08] Sharon Goldberg, Shai Halevi, Aaron D. Jaggard, Vijay Ramachandran, and Rebecca N. Wright. Rationality and traffic attraction: Incentives for honest path announcements in bgp. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 267–278, New York, NY, USA, 2008. ACM.

[GR01] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, December 2001.

[GSW99] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. Policy disputes in path-vector protocols. In *Proceedings of the Seventh Annual International Conference on Network Protocols*, ICNP '99. IEEE, 1999.

[LGS13] Robert Lychev, Sharon Goldberg, and Michael Schapira. BGP security in partial deployment: Is the juice worth the squeeze? *CoRR*, abs/1307.2690, 2013.

[LSZ08] Hagay Levin, Michael Schapira, and Aviv Zohar. Interdomain routing and games. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 57–66, New York, NY, USA, 2008. ACM.

[PS03] Venkata N. Padmanabhan and Daniel R. Simon. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Comput. Commun. Rev.*, 33(1):77–82, January 2003.

A Definition of the Game

We model BGP via a two phase, asynchronous, infinite-round game, inspired by that of [LSZ08] and [GHJ+08]. The initial data is defined by a labeled graph $G = (N, L, \mathcal{V})$ which has:

- nodes N representing autonomous systems
- edges $L \subseteq N \times N$ representing communication channels between the autonomous systems
- valuation functions $\mathcal{V} = \{v_i\}_{i \in N}$ for each autonomous system. If T represents the final state of the game, then $v_i(T) \in \mathbb{R}_{\geq 0}$.

Each autonomous system corresponds to an agent in the game. During the game, each agent i keeps a tuple (t_i, h_i, q_i), where

- $t_i : N \to \mathcal{P}(N(i))$ is the (next-hop) forwarding table of agent i, where agents are able to choose multiple neighbors to send traffic to[2]
- h_i represents the *history* of agent i in the game thus far. We treat h as a sort of transcript without worrying much about its formal representation. In particular, h keeps track of the all route announcements and (next-hop verification) messages received by agent i
- q_i represents the *next-hop queries* that agent i has received. We treat q_i as a queue

Let the set of all such tuples for agent i be denote $State_i$, and let the set of states for all agents be denoted $State = \{(state_1, \ldots, state_k)\}$.

The strategy space of each agent i is given by (imp_i, exp_i, que_i), where:

- $imp_i : State_i \times N \times Path^{N(i)} \to \mathcal{P}(N(i))$ represents the import policy of i (for a destination $d \in N$, and paths announced by each neighbor $j \in N(i)$, adjust the forwarding table for destination d)
- $exp_i : State_i \times N \times N(i) \to Path$ represents the export policy of i (for a destination $d \in N$ and neighbor $j \in N(i)$, what path would you announce to j (if none, return an empty path))

[2]This allows manipulator agents to try to "fake" forwarding traffic as advertised, but actually send 99% of their traffic down a different path.

- $que_i : State_i \times \mathcal{P}(N(i)) \times Query \to \{True, False\} \cup (N(i) \times Query)^*$ is the query policy (given that the agents $A \subseteq N(i)$ forward to you for destination d (as given by the query), can you confirm/deny/forward the query).

Both phases of the game are controlled by an *activator* that schedules agents to act[3]. The activator must pick every agent infinitely often (this property is called being "fair"), but other than that we assume the activator is completely adversarial, i.e. our positive results must hold for every fair activation sequence. During phase one, the activator picks an agent i to act, and i updates t and h from $state_i = (t, h, q)$ by performing the following actions for each $d \in N$:

- For each $j \in N(i)$, let $e(j) = exp_j(state_j, d, i)$ denote the (possibly empty) path j wants to export to i for destination d
- Update $t(d) := imp_i(state_i, d, e)$
- Update h_i recording all observed exports and action chosen by i

The above is repeated until the following condition is reached: no node would change its forwarding table if activated. This is called reaching convergence. If this never occurs, phase two of the game never starts and all agents will receive $-\infty$ utility.

Up until now, everything in this model has been previously considered. We introduce phase two to capture the execution of next-hop verification, where nodes send around queries to try and detect lies, i.e. mismatches between the control and data plane. A query $Q_d(a, b) \in Query$ represents the claim that a uses b as its next-hop for destination d. Phase two starts with each node i starts with the query $Q_d(a, b)$ for every destination node d and every single hop $[a, b]$ on the route that i has installed for destination d. During each round of phase two, the activator picks a node i to act, and i updates h and q from $state_i = (t, h, q)$ by performing the following for each $query \in q$:

- If $que_i(state_i, fwds_i, query) = True$, do nothing
- If $que_i(state_i, fwds_i, query) = False$, then the "potential manipulator" of $query$ is "shamed". In this case, *every* agent is given utility $-\infty$[4]

[3]The activator captures the asynchronous nature of BGP. Previous work [GSW99, LSZ08] has gone even further than us and allowed message sending and receiving to happen in a scheduled manner, and allowing multiple agents to act simultaneously. We do not consider activation order in this level of detail.

[4]This suffices to model a strong negative consequence for the manipulator being caught lying. In practice, the node which queried would simply start using a different route.

- Otherwise, $que_i(state_i, fwds_i, query) \in (N(i) \times Query)^*$ gives a list of pairs $(j, query_j)$. For each such element of the list, add $query_j$ to the queue q_j of agent $j \in N(i)$.

Finally, update $q = []$ to the empty queue.

At the conclusion of the game, utilities are calculated as follows:

- If convergence is not reached, all nodes receive utility $-\infty$
- If any node was "shamed" during the next-hop phase, all nodes receive utility $-\infty$
- Otherwise, node i gets utility $v_i(State)$

B Definitions of Strategies and Classes of Instances

Let v denote the valuation function of a node i. Let T be the final state of a game, and let P be the path v gets to destination d. We assume v is of the form $v(T) = u(P) + \alpha(T)$, where α is the *attraction function* that indicates what i cares about other than getting a good path to d. We say that:

- i is attractionless if $v(T) = u(P)$, i.e. $\alpha(T) = 0$.
- i has volume attraction if $\alpha(T)$ is a function of the set of nodes whose path to d includes i
- i has next-hop attraction if $\alpha(T)$ is a function of the set of neighbors of i who route directly through i as their next-hop

Let $strat = (imp, exp, que)$ denote the strategy profile of a node i with valuation function v. We say that:

- *strat* is *honest* if *imp* simply selects the highest-ranked path according to v which is announced to i, and *exp* only announces the current favorite path to the destination that node i is currently using according to *imp*. Note that *exp* is allowed to arbitrarily filter paths and not export them as it pleases.

- *strat* is a *next hop participant* if *que* implements next-hop verification fully and honestly, as described in previous sections. That is,

$$que(state, fwds, Q(a,i)) = \begin{cases} True & a \in fwds \\ False & a \notin fwds \end{cases}$$

$$que(state, fwds, Q(a,b)) = \begin{cases} False & b \neq i \wedge a \in fwds \\ N(i) \times \{Q(a,b)\} & b \neq i \end{cases}$$

Like [GHJ+08] ((add the random lavi-nisan paper)), the correct "solution concept" for us is that of *Set ex-post Nash equilibrium.*

Definition 4. *Let* $(S_1, \ldots, S_n)$ *denote sets of strategies for players* $1, \ldots, n$*, i.e. each* $s \in S_i$ *is a function from the private information of* i *to a strategy. Then* $(S_1, \ldots, S_n)$ *is a* Set ex-post Nash equilibrium *if for each player* i *and each* $\vec{s}_{-i} \in \vec{S}_{-i}$*, there exists some* $s^* \in S_i$ *such that for* all arbitrary strategies t*, we have*

$$v_i(g_{act}(s^*(v_i), s_{-i}(v_{-i}))) \geq v_i(g_{act}(t, s_{-i}(v_{-i})))$$

for every set of valuations $\vec{v}$ *and every fair activation sequence act.*

C Proofs

The following lemma is the key to our positive results.

Lemma 1. *Let* G *be a BGP instance with the next-hop verification phase. Suppose all nodes except one manipulator* m *are next-hop participants. Assume that* m *announces to node* v *a hop* (a, b)*, where in the data plane the hop* (a, c) *is used for some* $b \neq c$*. (Note that both* b *and* c *may be used if* $a = m$ *and* m *is "faking traffic"). Furthermore, suppose there exists a path from* v *to* c *not containing* m*. Then* m *with be caught by next-hop verification, and will receive utility* $-\infty$*.*

Proof. Let P denote the path from v to c not containing m. Because the activation sequence is fair, every node along the path with be activated, in the proper order going from v to c. Node v starts with the query $Q(a, b)$, and thus it will eventually travel to b, which will "raise the alarm" and give m utility $-\infty$. □

Note: should probably get rid of the protocol's ability to "drop queries" in view of a false-negative

Our first result formalizes "when nodes are attractionless, there is no incentive to deviate from honest BGP with next-hop verification".

Theorem 1. *Let G be a stable outcome of an attractionless BGP instance. Suppose there is a single manipulator m, and all other nodes honestly participate in BGP and in next-hop verification. If m lies in order to get a better path to d, then next-hop verification will catch that lie and shame m.*

Proof. Suppose m announces a hop (a, b) to v, where (a, c) is actually in a route from m to d for some $c \neq a$. For contradiction, assume that every path from c to v includes m (so m can drop the next-hop queries and its lie won't be caught). Now, the route from m to d which includes (a, c) must be a simple path, so it cannot include m. If there was a path from v to d which did not contain m, then the path from c to d would give a simple path from v to c. Thus, every route from v to d includes m.

Because all non-m nodes are honest, m must get v to change its route to d in order to get a different path by lying to v. However, v's route cannot effect m's route to d, because every route from v to d includes m. This contradicts the assumption that m got a better path by lying, and shows that there exists a path from c to v not including m. By the previous lemma, this means m will be caught and shamed. □

The following theorem formalizes our claim "nodes have no incentive to deviate from honest BGP when next-hop verification is used, even when nodes have volume attraction".

Theorem 2. *Let G be a stable outcome of a BGP instance with traffic volume attraction. Suppose there is a single manipulator m, and all other nodes honestly participate in BGP and in next-hop verification. If m lies in order to attract traffic from some node u, then next-hop verification will catch that lie and shame m.*

Proof. Suppose m did manage to attract more traffic from a victim u. Let P denotes the path u originally took to d, and let Q denote the path u takes in the manipulated outcome G. Note that $m \in Q$ but $m \notin P$. Let v denote a node that m lied to, and suppose v is told that hop (a, b) is used while (a, c) is actually used for $c \neq b$.

Because the lie told to v must effect the path chosen by u, there exists a path R from u to v not including m. Furthermore, because m uses the route S from c to d (and profits from it) we know $m \notin S$. Thus, by eliminating any possible loops from RPS, we get a simple path from v to c which does not include m. Thus, by the lemma, m will be caught and shamed. □

However, ((add the counterexample of bowtie for generic attraction)).