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

Clay Thomas
claytont@cs.princeton.edu

Gavriel Hirsch
gbhirsch@cs.princeton.edu

January 6, 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 control-plane 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 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 announcement. However, verifying routes directly in the data plane typically involves a large overhead (CITE EXAMPLES FOR THIS, E.G. THE ONES FROM GOLDBERG). In addition, many types of lies that could be beneficial to AS's do not involve making the same lies to everyone, but rather telling distinct lies to different neighbors. As a result, if AS's are willing to collaborate then there is often information directly in the control plane that can be used to detect the existence of lies, without requiring that we monitor 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.

This isn't accurate. Our idea is to corroborate lies lies in the control plain against the truth in the data plane. Few examples show effectiveness of checking "different lies" (i.e. most examples from Goldberg have the manipulator telling the same lie to every "victim node"). See the google doc

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, and Appendix A provides a more formal model. Section 3 describes the next-hop verification protocol, and Appendices B and C contain pieces of our implementation and proofs of theorems. Section 4 describes our results and gives examples of scenarios in which next-hop verification catches lies, whereas path and loop verification are insufficient on their own. Finally, Section 5 offers some conclusions.

1.1 Previous work

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

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

these conditions are unreasonably strong, such as requiring that AS's only consider the immediate next hop in the path in their preferences.

1.2 Overview of our results

As in the papers discussed above, our new *next-hop verification* protocol runs in the control plane rather than in the data plane. 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, though as we will discuss the protocol would still provide value even in partial deployment. In addition, next-hop verification is "bulkier" than loop verification, as it essentially has to distribute information across the whole collection of AS's.

2 Model Details

Here we present an informal description of our model for the interactions between AS's. See Appendix B for a more formalized model.

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. The goal is to

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 (CITE GAO REXFORD) 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. As in [GHJ+08] we assume that every AS can compile their preferences into a ranking only over paths and an export policy, both of which will be static throughout the entire game. This brings us to our notion of what it means to be honest or lie.

Definition 1. *We say that an AS* ***interim lies*** *if it announces a path that it does not plan to use, whether that path actually exists or not, or if it fails to announce the removal of a path that it had previously announced.*

We say that an AS is ***interim honest*** *if it never lies.*

Note that under this definition an AS can hide information from neighbors while still being interim-honest.

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 paths that include subpaths which have not already been announced to them.*

"impossible to announce paths which weren't announced to them" should suffices... The subpath notion doesn't make sense to me.

Some extensions to BGP, such as S-BGP, can enforce path verification. However, it requires additional overhead as well as universal adoption (CITE SOMETHING).

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.*

As mentioned before, when an AS receives a path announcement that already includes it, the AS can raise an alarm if it never announced the corresponding subpath.

While the "export policy" thing is true, I think the "shaming" should be the highlight of definition 3

If instead export policies did not send paths to neighbors who are already in them, this detection 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.

Next-hop verif doesn't detect a superset of path verif, but I think it should detect a superset of loop verif.

2.3 Convergence

We also need to guarantee that given the rankings and export policies chosen by the AS's, BGP will converge to a stable routing graph. In the literature this is typically done via an assumption called *No Dispute Wheel*, and we adopt the same assumption. See Appendix B for more details.

It is possible that some interim lies will be hard to detect. Instead, next-hop verification will focus on *ex-post* honesty and lying. We often leave out the phrase "ex-post".

I don't think we need this section/considerations

Definition 4. *We say that an AS is* ***(ex-post) lying*** *if the stablility of the post-convergence equilibrium depends on a neighbor thinking that the AS is acting in a way that is inconsistent with the true equilibrium.*

We say that an AS is ***(ex-post) honest*** *otherwise.*

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 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 $Q(a, b)$, representing a node announcing that a uses b as its next-hop for the destination d.

function INITIALIZE
  for each link (a, b) in your path to d **do**
    Add the query $Q(a, b)$ to your query queue

function RESPOND($Q(a, b)$)
  Let the acting AS be denoted by n
  if You have previously responded to the query $Q(a, b)$ OR if $n = a$ **then**

```
        return
    if n = b then
        if a forwards to you for destination d then
            return
        else
            "raise the alarm"
    else
        if a forwards to you for destination d then
            "raise the alarm"
        else
            send the query Q(a, b) to all neighbors (other than a)
function MAIN
    for each query Q(a, b) in queue do
        RESPOND(Q(a, b))
    clear the queue
```

4 Results/examples

5 Conclusion

References

[FSS07] Joan Feigenbaum, Michael Schapira, and Scott Shenker. *Algorithmic Game Theory, chapter Distributed Algorithmic Mechanism Design.* Cambridge University Press, 2007.

[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.

[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.

[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.

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.

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 $Q(m, r, d) \in Query$ contains the following data:

- A potential manipulator m
- An announced next-hop r
- A destination d

Phase two starts after reaching convergence (formally, something) after which each node i starts with the query $Q(m, r, d)$ for every destination node d and every single hop $[m, r]$ on the route that i has installed for destination d. During 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

[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.

- If $que_i(state_i, fwds_i, query) = False$, then the "potential manipulator" m of $query$ is "shamed" and given utility $-\infty$
- 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 a node was "shamed" during the next-hop phase, it receives 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
- *strat* is a *next hop participant* if *que* implements next-hop verification fully and honestly, as described in previous sections

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

Definition 5. *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

((TODO: be better))

First, we start with a lemma that shows why next-hop verification is so powerful.

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. Suppose that m announces a next-hop of r to node n, but actually uses $p \neq r$ in the data plane. Furthermore, suppose there exists a path from n to p not containing m. Then m with be caught by next-hop verification, and will receive utility $-\infty$.*

Proof. The query goes along the path. □

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 an attractionless BGP instance with next-hop verification. Let $S = (S_1, \ldots, S_n)$ be the set of all honest, next hop participating strategies. Then S is a set ex-post Nash equilibrium.*

Proof. If not, then every path from *victim* to $realNextHop$ goes through m. However, this means no simple path from m to d can contain *victim*, or any of the nodes that *victim* could directly effect without routing through m. Thus, m could not have actually gotten a better path through the lie. □

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 BGP instance with next-hop verification and traffic volume attraction. Let $S = (S_1, \ldots, S_n)$ be the set of all honest, next hop participating strategies. Then S is a set ex-post Nash equilibrium.*

Proof. As shown before, m cannot get a better path. We show further that m cannot attract more traffic (in volume). Suppose m did manage to attract more traffic from a victim v. Then, if P denotes the path v originally took to d, then P does not contain m. Because v didn't use P in the honest situation, the node n which m lied to must be connected to P I guess. Thus, $realNextHopOfM$ must be connected to d with a path not containing m, and d is in turn connected to the guy who heard the lie. I guess. □