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

Clay Thomas
claytont@cs.princeton.edu

Gavriel Hirsch
gbhirsch@cs.princeton.edu

December 27, 2018

Abstract

The main goal of this work is to provide evidence for the following claim: corroborating facts from the data plane and control plane can improve the incentive properties of BGP.

In [LSZ08], the authors take a popular and well-studied extension of BGP, and study how it interacts with incentive properties. Their central claim of the authors is that BGP with path verification (such as S-BGP) has simple incentive properties: assuming all other nodes tell the truth, no node (or group of nodes) can lie in order to get strictly better routes. However, in [GHJ+08] it is demonstrated that in realistic models nodes have incentives to lie in order to *attract traffic*, e.g. ISP attracting traffic from customers. One idea of [GHJ+08] is to find the simplest set of criterion needed for incentive-compatibility to hold, including the simplest form of verification required. They find that in some settings, a simple form of verification known as *loop verification* suffices. Our work is motivated by the question: *What is the simplest form of verification needed to provide good incentive properties of BGP?* We call the form of verification we came up with Next-Hop verification.

1 Introduction

2 Previous Work

3 Next-Hop Verification

(a) Informal Model and Assumptions

Mention how in reality, nodes would inspect the dest field of the IP packets. Etc. Etc.

(b) The Protocol

One subtlety to note: the ASes should maybe start out by asking *themselves* whatever next-hop queries they have. E.g. in GrandMa, node a can immediately tell that m isn't telling the truth, because m says he forwards traffic to a, but doesn't.

4 Our Implementation Sketch

Well, the formal model below ends up basically being our implementation soooooo....

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[1]
- 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))
- $que_i : State_i \times \mathcal{P}(N(i)) \times Query \to \{True, False\} \cup (N(i) \times Query)^*$ 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[2]. The activator must pick every agent infinitely often (this property is called being "fair"), but

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

[2] 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.

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
- 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 1. *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. □