Rational ASes and Traffic Attraction: Incentives for honestly announcing paths in BGP

.

This is the full version from February 1, 2008

Sharon Goldberg
Princeton University

Shai Halevi
IBM T.J. Watson Research

ABSTRACT

Prior work on modeling interdomain routing assumed that autonomous systems (ASes) are interested only in obtaining the best possible *outgoing path* for their traffic to a destination. In reality, many more factors can influence the "rational behavior" of an AS; here we consider a natural model in which ASes are also interested in *attracting incoming traffic*, either out of greed (since other ASes pay it to carry their traffic), or malice (so it can drop, modify, or spoof packets). This model of rationality induces situations where an AS has an *incentive* to send Border Gateway Protocol (BGP) announcements that do not correspond to the AS-level paths that packets traverse in the data plane. In this work, we ask what enhancements to BGP and/or restricted classes of routing policies can ensure that ASes will have no incentive to lie about data-plane paths. We find that while protocols like Secure BGP [20] are necessary, they are in general not sufficient unless ASes are only interested in the next hop that their traffic takes to its destination. Our game-theoretic analysis highlights the high cost of ensuring that BGP path announcements match data-plane forwarding paths.

1. INTRODUCTION

Interdomain routing on the Internet consists of a *control plane*, where Autonomous Systems (ASes) discover and establish routes, and a *data plane*, where they actually forward packets along these routes. The control-plane protocol used in the Internet today is the Border Gateway Protocol (BGP). BGP is a path-vector protocol that allows ASes to discover routes through the Internet via path announcements from their neighboring ASes. In BGP, each AS has some private routing policies that may depend arbitrarily on economic, business, or performance considerations. Each AS applies its routing policies to the path announcements advertised by its neighbors, and uses them to select a neighbor to which it will forward traffic in the data plane.

Traditional work on securing interdomain routing (*e.g.,* Secure BGP [20], and the like [5, 15, 29]) has, with few exceptions [27], focused on the control plane; the goal of these works was loosely described as ensuring "correct operation of BGP" [20]. However, focusing on the control plane in isolation ignores the more important issue of how packets are actually forwarded in the data plane. While the basic goal of interdomain routing is reliable packet delivery, here we explicitly consider an additional goal that has been implicit in many previous works, *e.g.,* [15, 20, 27, 29]; namely, ensuring that the paths announced in the control plane match the AS-level forwarding paths that are used the data plane. This way, an AS can rely on BGP messages, say, to choose a high performance AS-level path for its traffic, or to avoid ASes that are perceived to be unreliable or adversarial [3, 18, 26].

In this paper we look for mechanisms that give *rational* ASes an *incentive* to honestly announce packet-forwarding paths in their BGP messages. One plausible way to do this is to deploy secure hop-by-hop data-plane protocols that verify and enforce AS-level paths, *e.g.,* [1, 22, 24, 30]. However, because secure data-plane protocols are expensive (Section 1.1), here we limit ourselves to mechanisms that run only in the control plane. We find that while protocols like Secure BGP [20] can help remove the incentive for ASes to lie about their forwarding paths, they are generally not sufficient; we *also* need to assume that routing policies of certain ASes depend *only* on the first AS that their traffic traverses on its route to its destination. Our results emphasize the high cost of ensuring that control- and data-plane paths match, even when we assume that every AS obeys a realistic, well-defined model of rationality.

Below we motivate our approach, overview our results, and discuss their implications; technical details are in Sections 2-5.

1.1 Matching the control and data planes.

One way to ensure that ASes honestly announce paths in the control plane is to deploy AS-path measurement and enforcement protocols that run in the data plane. However, determining AS-level paths in the data plane is a nontrivial task even in the absence of adversarial behavior (*e.g.,* [31] discusses the difficulty of determining AS-level paths from traceroute data). Moreover,

when ASes have an incentive to dishonestly announce forwarding paths in the control plane, we also need to ensure that these protocols cannot be 'gamed'. Thus, *secure* protocols must also make measurement packets indistinguishable from regular traffic *e.g.,* [22, 24, 30]. While this prevents ASes from hiding control- and data-plane mismatches (*e.g.,* by sending measurement packets over the path advertised in the control plane, while sending regular traffic over a different path), it also means that the overhead for these protocols is usually proportional to the amount of traffic sent in the data plane. Finally, while secure *end-to-end* data-plane protocols can robustly monitor performance and reachability *e.g.,* [2, 14, 27], these protocols do *not* trace the identities of the ASes on data-plane path; securely tracing AS paths requires the participation of every AS on the path [1, 22, 24, 30].

Since data-plane protocols are expensive, one can instead hope to ensure that control- and data-plane paths match by using Secure BGP [20] and the like [5]. These protocols, if ubiquitously deployed in the Internet, would ensure a property we call path verification; namely no AS can announce a path to its neighbors unless that path was announced to it by its neighbors. While path verification defends against many practical attacks (*e.g.,* announcement of paths that do not exist in the Internet topology [20]), a closer look reveals that by itself it can never ensure that control- and data-plane paths match. For example, even when Secure BGP is fully deployed, an AS a with two different paths announced by two different neighbors can lie in the control plane — announcing one path in the control plane, while sending traffic over the other path in the data plane.

While it is tempting to argue that ASes are unlikely to lie about their forwarding paths because they either fear getting caught or creating routing loops, this simplistic argument fails in many situations. Firstly, the hierarchy in the Internet topology often prevents such routing loops from forming (*e.g.,* if the lie is told to a stub AS, or see [4]). Also, empirical results indicate that catching such lies can be difficult, since even tracing the paths that packets take in the data plane is prone to error [31]. Finally, to minimize the likelihood of getting caught, an AS could lie only when it has a good idea about where its announcement will propagate.

1.2 Our game-theoretic approach.

In this paper we explore a space of control-plane enhancements to BGP, restrictions on routing policies and assumptions on AS behavior to understand the extent to which we can use *only control plane mechanisms* to find incentives for ASes to honestly announce data-plane paths in their BGP messages. To do this, we adopt the paradigm of distributed algorithmic mechanism design [6–10, 21, 23, 25], which is rooted in game theory. This paradigm assumes that ASes are *rational players*, rather than (potentially) arbitrarily malicious, and that they participate in interdomain routing because they derive utility from establishing routes and forwarding packets. Each AS will take whatever strategic actions it can to maximize its own utility. The task of mechanism-design is to develop mechanisms that ensure that rational players have no incentive to deviate from the behavior prescribed by the mechanism.

In our case, the mechanism that we consider is BGP (and the additional control plane mechanisms that we add to it, *e.g.,* Secure BGP [20]). The behavior prescribed by the mechanism is that each AS a should announce a path abP only if the path abP was announced to a by the neighbor b *to which AS* a *forwards traffic in the data plane.* If every AS adheres to this prescribed behavior, then control- and data-plane paths will match.[1] From now on, we follow the literature and say that a network is incentive compatible if, when all ASes are rational, then they all have no incentive to deviate from "correct operation of BGP". We remark that, in contrast to previous work on incentive compatibility, our model (Section 2) explicitly addresses the distinction between the control and data planes.

1.3 Modeling rationality & traffic attraction.

A recent result of Levin, Schapira and Zohar [21] showed that path verification (*e.g.,* Secure BGP) is sufficient for incentive compatibility even when ASes have arbitrary routing policies. Their encouraging result [21] improves on earlier works [7–10] that instead explored restricted classes of routing policies that can ensure incentive compatibility. Specifically, Feigenbaum et. al [8, 10] found that policy consistency is sufficient; policy consistency is a generalization of shortest-path routing that requires that the preferences of neighboring ASes regarding different paths always agree.

However, all prior results on incentive compatibility of BGP were obtained under a very restricted model of "AS rationality". Specifically, they assume that the utility that an AS derives *is uniquely determined by the outgoing path that traffic takes to the destination.* In reality, the utility of an AS is likely to be influenced by many other factors. For example, the utility of a commercial ISP may increase when it carries more traffic from its customers [19]. As another example, a nefarious AS might want to attract traffic so it can eavesdrop, degrade performance, or spoof packets [3, 18, 26].

Thus, in this paper we extend the notion of "ratio-

[1] We do not consider situations when the control and data plane do not match due to malfunction or misconfiguration; we consider this to be *irrational* behavior. We also do not consider control- and data-plane mismatches caused by route aggregation [31], since here, typically, only last hop of the (data-plane) AS-path is omitted from the BGP path announcement (See also the note in Section 2.1).

Control-Plane Integrity Checks	Model of 'rational' behavior at an AS		
	No traffic attraction	Attract customer traffic on a direct link	Attract any traffic
None	Policy consistency [8, 10]	No known restrictions on policy guarantee incentive compatibility	
Loop Verification	Policy consistency [8, 10]	Gao-Rexford conditions AND every AS uses next-hop policy with providers and peers	Next-hop policy
Path Verification	Arbitrary policy [21]	Gao-Rexford conditions AND every attractee uses next-hop policy with providers and peers	Next-hop policy

Table 1: Restrictions on routing policies[3] that, for a given model of AS rationality and type of control-plane integrity check, guarantee incentive compatibility of BGP.

nality", focusing in particular on the the effect of traffic attraction, where the utility of one AS increases when it transits *incoming traffic* from another AS (the attractee). We consider two models of traffic attraction; our first (generic) model encompasses all forms of traffic attraction; here we assume that the utility of an AS can increase if it attracts traffic from *any* AS in the network over any link (Section 2.2). Our second models captures a realistic form of economically-motivated traffic attraction. Because service contracts in the Internet are typically made between pairs of neighboring ASes [19], we assume that the utility of an AS will increase only if it attracts traffic from a neighboring customer AS that routes on the direct link between them (Section 3.3).[2]

Unfortunately, we find that even applying this second restricted form of traffic attraction means that previous results no longer apply; both path verification [21] and policy consistency [8, 10] are not sufficient for incentive compatibility. (See Figure 3.) This disappointing result motivates our search for new conditions that suffice to guarantee incentive compatibility.

1.4 Overview of our results.

On the bright side, we demonstrate that certain combinations of control-plane integrity checks and policy restrictions from the set below are sufficient to guarantee incentive compatibility. Table 1 sketches our results.[3]

In addition to path verification (*e.g.,* Secure BGP [20]), we introduce a weaker control-plane integrity check called loop verification, which roughly captures the setting where an AS falsely announces a routing loop, and is then caught and punished. Loop verification can be thought of a formalization of "the fear to get caught", and it may be easier to deploy than path verification (Section 4.4).

We consider policy restrictions (Section 3) that include policy consistency, as well as a more restricted class of routing policies called next-hop policy. Next-hop policy roughly requires that an AS selects routes to a destination based only on the immediate neighbor that advertises the route. We also consider the Gao-Rexford conditions [12]. These conditions, which are believed to reflect the economic landscape of the Internet [19], assume that neighboring nodes either have a customer-provider relationship (customer pays the provider for service) or peer-to-peer relationships (ASes freely transit each other's traffic), and that these business relationships induce restrictions on the policies at each AS.

When exhibiting a positive result for a certain set of conditions, we *prove* that BGP is incentive compatible in *every* network that satisfies these conditions. On the other hand, to exhibit a negative result, it suffices to show a counterexample network that satisfies all these conditions where BGP is not incentive compatible. Our results, detailed in Table 2, are 'tight' — for any positive result, weakening any condition (from the set we considered) allows us to find a counterexample.

1.5 Implications of our results.

Our results show that achieving incentive compatibility *using only control-plane mechanisms* require a very strong conditions: At the very least we need both (1) full deployment of either path or loop verification, and (2) next-hop policy between *all* pairs of nodes with certain business relationships (Section 5). And if we do not assume business relationships then we need stronger still conditions (Section 4). Furthermore, incentive compatibility is achieved only if these conditions are respected by *every node in the network* (Section 4.7). Thus, our results point to a negative answer to the question that we set out to investigate — practically speaking, it is unlikely that we could use only control-plane mechanisms to remove the incentives for ASes to dishonestly announce AS-paths in the BGP.

This leaves us with the choice of either employing expensive data-plane AS-path enforcement techniques [1, 22, 24, 30] when it is absolutely necessary to ensure that packets are forwarded on AS-level paths that match an AS's routing policies, or dismiss this idea altogether and instead content ourselves with some weaker set of goals for interdomain routing. It is certainly possible to formulate weaker but meaningful security goals and show that certain control-plane mechanisms or data-plane protocols meet these goals. However, doing this begs the question: if we are not interested in ensuring that AS paths announced in BGP are really used in the data plane, then why use a path-vector protocol at all?

[2]Other restricted models of traffic attraction may be possible; we leave analyzing those to future work.

[3]All the results in Table 1 also require an additional technical condition called 'no dispute wheel', see Appendix A.

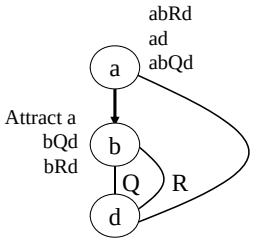


Figure 1: AS graph with traffic attraction.

2. OUR MODEL

We present our model of the BGP protocol here, in support of our results in Section 4-5. Because we consider traffic attraction, our model differs from the standard model (formulated by Griffin, Shepherd, and Wilfong in [16] and used by all subsequent works) in a few significant ways. Firstly, our model of interdomain routing in Section 2.1 makes explicit the separation between actions taken in the data plane (*i.e.,* choosing to forward packets to a neighbor) and the actions taken in the control plane (*i.e.,* sending and receiving path announcements). Furthermore, our work explicitly distinguishes between the preferences an AS uses in the BGP decision process, and the utility it obtains from a path assignment. Our new model of utility with traffic attraction is in Section 2.2. In Section 2.3 we discuss how to map between our notion of utility and the BGP decision process. Finally, we formally discuss our notion of incentive compatibility in Section 2.4.

2.1 Modeling interdomain routing with BGP.

The inter-domain routing system is modeled as an AS graph (see Figure 1). For simplicity, we model each AS as a single node, and edges represent direct (physical) communication links between ASes. Nodes with an edge between them are called neighbors. Below we denote nodes by lowercase letters, typically a, b, c, d, m and n. Because in practice BGP assigns routes to each destination separately, we follow [16] and assume that there is a unique *destination node* d to which all other nodes attempt to establish a path. (Thus, like most previous work, we ignore the issue of route aggregation [31]). Below we denote paths by upper-case letter, typically P, Q and R.

Each AS node in the graph has a set of permitted paths along which it is willing to route outgoing traffic to the destination. (In Figure 1 node b has permitted paths bQd and bRd.) Only simple paths are permitted. Paths with cycles are never permitted, since they would cause routing loops (where packets endlessly cycle between a set of nodes, and never reach the destination). For each permitted path P at each node n, node n has a set of neighbors from whom it is willing to accept incoming traffic and transit this traffic along P to the destination. (This models the fact that BGP-export policies at node n may be based on the outgoing path that n is using.) A path $P = a_0 a_1 \dots a_\ell$ in the AS graph is called a permissible path if for each i, the suffix $a_i \dots a_\ell$ is one of the permitted paths for node a_i, and moreover a_i is willing to transit the traffic of a_{i-1} over this path.

To model the BGP decision process, we assume that each node a has an algorithmic ranking function $r_a(\cdot)$ that ranks its permitted outgoing paths to the destination,[4] and acts on BGP path announcements. Then, in an execution of the BGP protocol, node a receives announcements of paths from its neighbors, chooses the permitted path that ranks highest by its ranking function, and "commits" to using that path in the data plane by announcing it back to all the neighbors for which it is willing to transit traffic over this chosen path. This sequence is repeated by all the nodes in a distributed manner.

When BGP is used to establish routes on the Internet, every node a is expected to send all its traffic to the destination along the first link in the path that it announced to its neighbors in the control plane. We call this first link the next hop for node a. (Thus, one of the many ways that a node can deviate from BGP is by sending data plane traffic over a different next hop than the one that is advertised in its path announcement, see Section 2.4.)

In this work we assume that ASes that follow the BGP protocol route all their traffic on a single AS path. (We leave the interesting question of modeling multipath routing to future work.) When modeling BGP, we imagine stopping the protocol at some point in time and considering the link that a node uses for routing in the data plane as the 'output link' of that node. Then the (data-plane) path assignment T that results from an execution of BGP consists of all the output links of all the nodes. Since each node only chooses a single output link to which it sends *all* its traffic, when all nodes behave honestly, the path assignment is necessarily a tree rooted at d. Consider a path P that was *announced* by node a in the control plane during the execution of BGP. We say that P is a true path if it is part of the (data-plane) path assignment T that results from this execution (so that the control plane matches the data plane for path P), and a false path otherwise.

In this work, we always assume that control plane messages sent between two neighbors on direct link cannot be tampered with (by a node not on the direct link). This can be enforced with the BGP TTL Security Hack [13] or via a pairwise security association between nodes using the TCP MD5 security options [17].

[4] As in prior work, we require a strict ranking: for any two paths with different next hops, say anP with next hop node n and $an'P'$ with next hop node n', if $n \neq n'$ then $r_a(anP) \neq r_a(an'P')$.

2.2 Utility, Valuation, and Attraction.

We model the ASes as "rational players" whose actions are dictated by a desire to maximize their utility from the resulting data-plane path assignment. Namely, each node a has a utility function $u_a(\cdot)$, and it tries to obtain a path assignment T such that $u_a(T)$ is as high as possible. Thus, while the algorithmic ranking function acts on control-plane messages, in our model, the utility function acts on the data-plane path assignment.

The utility functions that we consider in this work have two components: a valuation function that depends only on the path(s) from b to the destination, and an attraction function that depends only on the paths of nodes that route traffic via b. Namely, we assume that every node in the graph as a utility function of the form

$$u_b(T) = v_b(T) + \alpha_b(T) \quad (1)$$

where $v_b(T)$ depends only on simple paths from b to d in T, and $\alpha_b(T)$ depends only on simple paths from other nodes to b in T.[5] The valuation function $v_a(\cdot)$ is the same as was considered in previous work [6–10, 21, 23, 25], and it is meant to capture the intrinsic value of each outgoing path (as related, *e.g.,* to the cost of sending traffic on a particular path, to the reliability of the path, or to the presence of undesirable entities on that path). For example, in Figure 1, the valuation function of node b prefers the path bQd over path bRd. While we assume that all honest ASes choose a single outgoing link for all their traffic, a *misbehaving* node m might send its outgoing traffic on more than one outgoing link. In this case, we will assume that if m uses more than one path to d in T, then the valuation $v_m(T)$ is at most as high as the most valuable simple m-to-d path in the path assignment T.[6]

We add the attraction component $\alpha_b(T)$ in this work. Since in this work we are interested in situations where nodes want to attract traffic (and not deflect it), our most general form of the attraction function only requires that the α component does not increase when edges leading to b are removed from the path assignment. More formally, for a path assignment T and node b, let $T(b)$ be the set of edges along simple paths from other nodes to b in T (*e.g.,* if T is a tree then $T(b)$ is the subtree rooted at b). We assume that for every two path assignments T, T' and every node b, if $T'(b) \subseteq T(b)$ then $\alpha_b(T') \leq \alpha_b(T)$. Notice that this general condition covers many forms of traffic attraction, including when a nefarious AS wants to attract traffic so it can eavesdrop or tamper with packets [3].

We sometimes (see Section 3.3) consider more restricted forms of the attraction functions, where there is a specific subset of nodes from which a tried to attract traffic. When considering these restricted forms, we say that two nodes, a and b, have an *attraction relationship* if the attractor b increases its utility whenever the attractee a routes traffic through it (*e.g.,* as in Figure 1).

2.3 From utility to ranking.

In previous work, the utility of each AS was the same as its valuation function, and that utility directly determined the algorithmic ranking of paths the AS uses in the BGP decision process: the larger the valuation of a path, the highest its rank. In our model this direct translation does not necessarily hold: an outgoing path with low valuation could be preferred because it brings incoming traffic from attractees. (In Figure 1, node b's valuation function ranks path bQd over path bRd, but b has has higher utility when it routes on bRd, since in this case he attracts traffic from node a.)

Still, the BGP decision process as implemented in practice depends only on the outgoing paths. Hence, in our model, we assume that each node needs to "compile" its utility function into this ranking function. This means that we now have four functions associated with each node a: utility $u_a(\cdot)$, valuation $v_a(\cdot)$, and attraction $\alpha_a(\cdot)$ as described in Equation 1, and the algorithmic ranking $r_a(\cdot)$, which is the ordering of outgoing paths as installed in the routers of this AS. (Using this terminology, all prior work has $\alpha_a(\cdot) \equiv 0$ and $r_a(\cdot) \equiv u_a(\cdot) \equiv v_a(\cdot)$.)

The notion of "compilation" may model an ongoing process where an AS reacts to changes in network conditions, contractual agreements, new information that ASes learn about each other, and so on to set up its ranking function, in a way that maximizes its utility. (For example, an AS may periodically change its ranking in order to test what outgoing link draws more traffic from attractees.) However, since the time scale for compilation is usually much longer than the time scale for BGP itself (say, hours versus seconds), for the purpose of our analysis, we model the compilation process as being done "once and for all", and to analyze BGP with respect to a fixed algorithmic ranking function.

There are many conceivable ways of compiling the utility into ranking. It may make sense to set $r_a(T) = v_a(T)$ by default, and to deviate from this default only when such deviation demonstrates an advantage in terms of traffic attraction. For example, if there is a service-

[5] The only exception to Equation 1 is that if there is no path from a to the destination in T, then we assume that $u_a(T) = 0$, regardless of the attraction component $\alpha_a(T)$. Also, while we wrote the utility function as a *sum* of the valuation and attraction functions, all we require is that utility increases monotonically with both valuation and attraction.

[6] We remark that this assumption was implicitly used also in prior work on incentive compatibility, and it ensures that even for a manipulator m "the optimal strategy" is to also send its outgoing traffic over a single link. This is because the valuation of the path can only increase if it uses only the "best outgoing link" instead of using a few of them, and the attraction function does not depend on the outgoing links that m uses.

level agreement that obliges b to carry a's traffic via path bRd in return for monetary compensation α, then b might decide to set $r_b(bRd) = v_b(bRd) + \alpha$. In this work we mostly sidestep the question of how to compile the utility into ranking. Our counterexamples work "for any reasonable compilation" derived from the utility function, and our in positive results we state explicit conditions that the utility, valuation, and ranking functions should satisfy for the results to hold. In particular, all our theorems hold for the default $r_b(T) = v_b(T)$.

2.4 Defining incentive compatibility.

Incentive compatibility of a mechanism (with respect to some utility function) means that no participant can increase its utility by *unilaterally* deviating from the prescribed mechanism. That is, if a participant knows that everyone else is abiding by the rules, then it has no incentive to deviate from them.[7]

We assume that each AS knows the outgoing link on which it routes traffic (and the next AS at the end of that link), but may not know for certain the AS-path that the traffic takes further downstream. (We justified this assumption in Section 1.1.) This is a crucial assumption, since it implies that *an AS may try to lie to its upstream neighbors about the path on which it sends its traffic*. Similarly, we assume that ASes typically do not know the topology of the network except for their immediate neighborhood, and therefore must rely on path announcements by their neighbors for information about the full paths traversed by their traffic. This means that an AS may try to lie to its neighbors about the existence of paths in the network.

The "prescribed mechanism" that we consider for incentive compatibility is the usual BGP protocol, where, in a setup phase, each AS compiles its utility function into an algorithmic ranking function. After that, each AS repeatedly gets BGP path announcements from its immediate neighbors, uses its algorithmic ranking function to choose one of these paths, announces that path to its neighbors, and sends its outgoing (data-plane) traffic on the link to the next-hop neighbor on that path. However, in our model ASes are *rational*, so an AS may deviate from the "prescribed mechanism" above in order to increase its own utility. For example, a *manipulating AS* could announce a path with next-hop that is different from the one that it actually uses in the data plane, or announce different, arbitrary paths to each of its neighbors. Or, in Figure 1, manipulating AS b could deviate from BGP by announcing the path bRd in order to attract traffic from node a, while actually sending traffic over the path bQd.

The notion of incentive compatibility in our context means that if all the ASes but b follow the prescribed mechanism as above, then no strategy that b can use would give it a higher utility than following the same mechanism. Namely, there must exist a specific algorithmic ranking function (which can be derived from b's utility function), such that following that ranking function yields the highest achievable utility for b.

3. DEFINITIONS: ROUTING POLICY

This section presents the necessary formal definitions for the routing policies we consider in Section 4-5. We also define our restricted form of economically-motivated attraction in Section 3.3.

3.1 Policy consistency

This class of policies was defined in [8, 10] as a generalization of 'lowest-cost' or 'shortest-path' routing. Roughly, we say that two neighboring nodes a and b are *policy consistent* if there are two permissible paths $abQd, abRd$ from a to the destination d, such that if b prefers path bQd over bRd, then a also prefers $abQd$ over $abRd$. Referring to Figure 1, we can see that node a is *not* policy consistent with node b. We remark that policy inconsistencies can occur quite naturally in practice. For example, in Figure 1 AS a could prefer path R because it is more reliable, while AS b could prefer path Q because it is cheaper to use. (A complete definition of policy consistency is in Appendix B.)

3.2 Next-hop policy

Next-hop policy requires that a node only cares about the next hop that its traffic takes. This class of routing policies is more restrictive than policy consistency; *e.g.,* node c in Figure 3 is policy consistent but does *not* use next-hop policy with node n. Formally, we say that node a uses next-hop policy in the valuation with a neighbor node b, if for every two paths $abQd, abRd$ from a to the destination, it holds that b is willing to transit a's traffic on both paths, and moreover $v_a(abQd) = v_a(abRd)$. Next-hop in the ranking is defined similarly with $r(\cdot)$ replacing $v(\cdot)$. While next-hop policy is the most restrictive class of policies we consider here, empirical evidence [28] suggests that these policies are quite prevalent in the Internet.

3.3 The Gao-Rexford conditions and economically-motivated attractions.

In [12], Gao and Rexford gave a set of conditions, induced by business relationships between ASes that are widely believed to reflect the economic landscape of the current Internet [19]. These conditions, which much be followed by each node in the network, guarantee that the BGP protocol converges (more discussion is in Ap-

[7] In the mechanism-design literature, this notion is called an *ex-post Nash equilibrium*, and is weaker than *dominant strategy* (or *strategyproofness*) where participants have no incentive to deviate no matter what the others do. See [25] for more discussion.

Generic Network	Gao-Rexford & Attractions	Loop Verification	Path Verification	Policy Restrictions	**Incentive Compatible?**	Pointer to Result
✓	✓	**X**	**X**	policy-consistency	**No**	False Loop
✓	✓	**X**	**X**	next-hop policy	**No**	False Loop
✓	✓	✓	✓	policy-consistency	**No**	Bowtie
✓	✓	✓	✓	next-hop policy	**Yes**	Theorem 4.4
✓	✓	✓	✓	attractees use next-hop with their attractors	**No**	Little Dipper
✓		✓	✓	attractees use next-hop with all their neighbors	**No**	Big Dipper
	✓	**X**	✓	attractees use next-hop with peers and providers	**Yes**	Theorem 5.2
	✓	✓	**X**	attractees use next-hop with peers and providers	**No**	Result from [21] in Figure 7
	✓	✓	✓	all nodes use next-hop with peers and providers	**Yes**	Theorem 5.4
	✓	✓	✓	all nodes use next-hop with their providers	**No**	Little Dipper

Table 2: Summary of our results for incentive compatibility of BGP with traffic attraction. We require that our generic networks have 'no dispute wheel' (Appendix A).

pendix A). There are two kinds of edges in Gao-Rexford networks: customer-provider relationships (where typically the customer pays the provider for connectivity) and peer-to-peer relationships (where two nodes agree to transit each other's traffic for free). We restate the three Gao-Rexford conditions (GR1 - GR3) below.

GR1. Topology. There are no customer-provider cycles in the AS graph.

GR2. Transit. A node b provides transit to traffic from node a to node c only if at least one of nodes a and c are customers of node b.

GR3. Preferences. Node b prefers outgoing paths where the next hop is a customer over outgoing paths where the next hop is a peer or a provider, and prefers peer links over provider links.[8] GR3 can apply to either the valuation function or the algorithmic ranking.

Because the Gao-Rexford conditions are meant to model economic relationships, in this work we add a fourth condition that deals with traffic attraction by economically-motivated ASes (AT4) to our description of Gao-Rexford networks. This condition models the fact that in the Internet, service contracts are made between pairs of neighboring nodes such that the customer pays its provider for transit when it sends traffic over their shared link [19]. AT4 restricts the set of traffic attraction relationships that we allow in the AS graph, and thus does *not* model generic attraction relationships, *e.g.,* when an AS wants to attract (and say, snoop on) traffic from ASes that are a few hops away, or a peer wants to attract more traffic from its peer in order to pressure it to start paying for transit.

AT4. Attractions. A node b may only have an attraction relationship with its own direct customer. Furthermore, node b only increases its utility if the attractee customer node a routes through node a via the direct (a, b) link.

In the sequel, whenever we refer to a network that obeys Gao-Rexford, we always mean that it satisfies all four conditions, GR1-GR3 and AT4. When we draw Gao-Rexford networks we represent a customer-provider relationships by a directed edge with an arrow from customer to provider, and peer-to-peer relationships by an undirected edge. We represent an attraction relationship with a **bold** arrow from attractee to attractor.

4. RESULTS: GENERIC ATTRACTIONS

We now present our results for generic networks and a model of rationality that encompasses all possible attraction relationships, as we discussed in Section 2.2. The bottom line is that we prove that in networks with generic attraction relationships, BGP is incentive compatible if *all* nodes use next-hop policy and loop verification. (This result, presented in Section 4.5, also requires that the network have 'no dispute wheel' (a condition defined in Appendix A). We discuss loop verification in Section 4.4.)

To show the tightness our result, we present a series of counterexamples that show that weakening the (very strong) conditions described above means that BGP is no longer incentive compatible. While our focus here is on generic networks, our counterexamples will sometime be Gao-Rexford networks (see Section 3.3). This only strengthens our negative results, since if the negative results hold for Gao-Rexford networks, then they hold for generic (non-Gao-Rexford) networks as well.

[8]The original version [12] of the Gao-Rexford conditions does not require nodes to prefer peer links over provider links. To make our results as general as possible, we use this weaker version of GR3 in all our theorems, while our counterexamples do satisfy the stronger version of GR3.

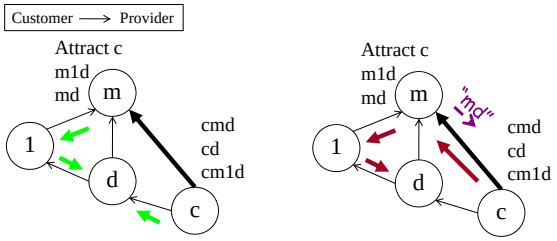


Figure 2: Inconsistent Policy

The theorems and counterexamples in this section are summarized in the first six rows of Table 2.

4.1 Path verification is not enough.

Recently, Levin, Schapira, and Zohar [21] found that path verification is sufficient for incentive compatibility when there is no traffic attraction.

Path Verification is the focus of most traditional work on securing BGP [5]; roughly, it ensures that nodes cannot announce paths that are not in the network. More formally, path verification is a control-plane integrity check that ensures that no node a in the network can announces a path abP to its neighbors, unless a neighbor node b announced the path bP to a. Path verification can be guaranteed when SBGP [20] or IRV [15] is fully deployed in the network; we note, however, that soBGP [29] does *not* provide path verification.[9]

We restate the result of Levin et. al. here, which also requires an additional restriction on routing policy called 'no dispute wheel' that we discuss in Appendix A:

Theorem 4.1 (From [21])**.** *Consider an AS graph* without traffic attraction *where there is path verification, and let m be a node such that there is no dispute wheel between the valuation of m and the ranking of all other nodes. Then it follows that the optimal strategy for m is to set its ranking to its valuation, and follow the BGP protocol without deviation.*

This encouraging result follows from the fact path verification restricts the set of false paths that can be announced by an AS a to the small set of paths that were announced to a by a's neighbors. Then, by restricting the rational behavior of AS a such that it is only interested in improving the *outgoing path* its traffic takes to the destination, [21] proves that AS a has no incentive to announce false paths from this small set. However, we now show that, even with path verification, traffic attraction can cause an AS to strategically announce a false path:

Figure 2: Inconsistent Policy is a network that obeys the Gao-Rexford conditions and where node m wants to attract traffic from node c. The unique stable assignment T_1 (where the ranking function is equal the valuation function at each node) when all nodes act honestly is shown on the left. The manipulated outcome T_2 is shown on the right. Here the manipulator m has an incentive announce a false path "md" to node c while actually using path $m1d$ in order attract c's traffic. Notice that this false path can be announced even if the network has path verification, since node 1 announced "$1d$" to m. Thus, Inconsistent Policy shows that path verification is not sufficient for incentive compatibility, even for restricted traffic attraction relationships (that obey AT4).

We remark that the situation in Inconsistent Policy could arise quite naturally in practice. As an example, while c is a customer of both m and d, c could have a service contracts with m and d such that usage-based billing on the m-c link is lower than billing on the d-c link. Then, c could prefer a route through m over the direct route to d as long as this route only increases AS-path length by a single hop. On the other hand, m could prefer to send traffic via 1 because 1 is geographically closer to m than d.

4.2 Policy consistency is not enough.

Notice that in Inconsistent Policy, node c has a policy inconsistency with node m. It is natural to ask if disallowing policy inconsistencies is sufficient for incentive compatibility. Indeed, Feigenbaum et. al. asked a similar question in [8,10], and obtained a positive result for the setting of no traffic attraction. We reproduce their result using our own terminology:

Theorem 4.2. *Consider an AS graph* without traffic attraction*, in which the ranking function at all nodes are policy consistent with all their neighbors. Assume further that m is a node satisfying (1) there is no dispute wheel between the valuation of m and the ranking of all other nodes, and (2) the valuation of m is policy consistent with the ranking of its neighbors. Then it follows that the optimal strategy for m is to set its ranking to its valuation, and follow the BGP protocol without deviation.*

They prove this by showing that the unique stable path assignment that results when every node behaves honestly and uses policy consistency, is locally optimal at every node (see Lemma B.3). The theorem follows since no node has an incentive to deviate from honest behavior (since honest behavior is already locally optimal).

However, we now show that in networks with traffic attraction, policy consistency at all nodes is still not sufficient for incentive compatibility, even if the network has path verification:

Figure 3: Bowtie is a Gao-Rexford network with policy consistency everywhere. The unique stable assignment T_1, when each node sets it algorithmic ranking

[9] soBGP only provides a topology of the AS graph, not information about the announcements made by nodes, and thus does not provide path verification.

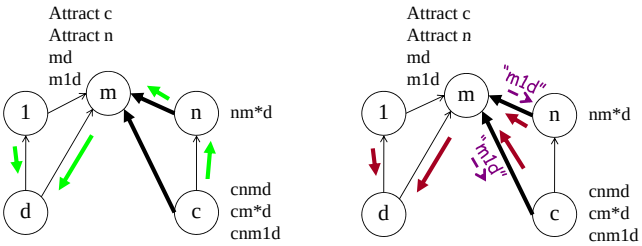


Figure 3: Bowtie

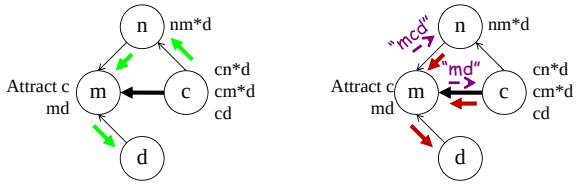


Figure 4: False Loop

equal to its valuation function, is shown on the left. The manipulated outcome T_2 is shown on the right, where each node except m is honest and sets its algorithmic ranking equal to its valuation function. Here m has an incentive to announce a false outgoing path "$m1d$" to all of its neighbors in order to attract traffic from the attractee c on the direct m-c link. Furthermore, m can announce this false path even in networks with path verification, since node 1 announced the path $1d$ to m.

We remark that even though c's traffic is routed via m in both T_1 and T_2, the manipulation in BOWTIE is quite reasonable in practice. For example, m might prefer the outcome in T_2 over the outcome in T_1 for load-balancing purposes, since in T_2 incoming traffic from c and n is spread over two links. As another example, suppose c is a small stub AS, n is a local ISP, and m is large Tier-1 ISP. Then, m might prefer the T_2 outcome because it has a usage-based billing contract with c on the m-c link and a lump-sum payment agreement on the m-n link. Or, perhaps m wants to put his competitor n out of business by attracting traffic from n's customers who are small stub ASes like c.

4.3 Next-hop alone is not enough.

From BOWTIE we learn that even requiring *both* path verification and policy consistency is not sufficient for incentive compatibility, even in Gao-Rexford networks where attraction relationships obey AT4. So we throw-up our hands and ask if we can at least guarantee incentive compatibility when everyone uses next-hop policy. Intuitively, it seems requiring next-hop policy at each should be sufficient; since each node cares only about the next hop that its traffic takes, it is tempting to conclude that lying about its outgoing path will not help an attractor convince its attractee to 'change its mind' and route through it in a manipulated outcome. (Notice that the manipulations in both INCONSISTENT POLICY and BOWTIE were of this form.)

Quite surprisingly, this intuition fails. We show that even requiring next-hop policy at every node is not sufficient. More precisely, we show that if the network does not have path verification, then even requiring the every node uses next-hop policy and follows the Gao-Rexford conditions is not sufficient for incentive compatibility.

Figure 4: FALSE LOOP is a Gao-Rexford network where all nodes use next-policy with all of their neighbors. The unique stable assignment T_1, when each node sets it algorithmic ranking equal to its valuation function, is on the left. The manipulated outcome T_2 is on the right, where each node except m is honest and set its algorithmic ranking equal to its valuation function. The manipulator m has an incentive announce a false outgoing path "mcd" to n in order to attract traffic from its attractee c. Notice that the outcome T_2 results whenever there is no path or loop verification, since the "false loop" will either cause node n not to announce any path to node c or cause node c to filter that path.

4.4 Introducing loop verification.

To deal with the manipulation in FALSE LOOP, we introduce a new control-plane integrity check that we call loop verification, that deals with detecting and preventing "false loops".

BGP uses two different approaches for detecting and preventing routing loops. One is *sender-side loop detection*, where a node a will not announce a path aR to node b if b happens to be on the path R, and the other is *receiver-side loop detection* where a will announce the path aR to all its neighbors, but b will detect the loop and discard that announcement. The advantage of using receiver-side loop detection, is that it allows a node b to hear announcements that *falsely* includes b in a path that b did not announce. Notice that for b to detect a 'false loop', b need only perform a *local* check to see if the path he receives matches the one that b actually announced. (This local check is much less onerous than the one that is required for path verification, which requires participation from all nodes on the AS path.)

At a high level, loop verification formalizes the idea that an AS will avoid lying in a BGP announcements because it fears getting caught by receiver-side loop detection. More formally, we require that if a node b receives an announcement of a path $P = QbR$, such that b never announced the route bR to its neighbors, then b "raises an alarm" and the first node who announced a path that includes bR will be punished, reducing its utility (say, to zero). This punishment process models the idea that b can catch and shame the AS that announced the false loop, *e.g.,* by broadcasting to the NANOG list.

The properties of loop verification are strictly weaker than those of path verification:

Theorem 4.3. *If a network has path verification, then no node will raise an alarm in loop verification.*

Proof. Follows immediately, since no node can announce a path that includes bR unless b announces bR. □

4.5 Next-hop & loop verification is enough!

Now that we have loop verification, we are ready to present the main result of this section. If we add loop verification to the network, we can eliminate the manipulation preformed by m in FALSE LOOP, and guarantee incentive compatibility if all nodes use next-hop policy. The following theorem applies even to networks that do not obey the Gao-Rexford conditions, even with the most general attraction functions as defined in Section 2.2. (The theorem also requires an additional condition called 'no dispute wheel' that we discuss in Appendix A.)

Theorem 4.4. *Consider an AS graph with loop verification, where the ranking function of each node uses next-hop policy with each of its neighbors. Further, let m be a node satisfying (1) the valuation of m and the ranking of all other nodes do not include a dispute wheel, and (2) the valuation of m is policy consistent with the ranking of its neighbors. Then it follows that the optimal strategy for m is to set its ranking to its valuation, and follow the BGP protocol without deviation.*

As a corollary of Theorem 4.4, if the valuations of all the nodes are next-hop (and have no dispute wheel), then the mechanism where every node sets its ranking function to equal its valuation function and follows BGP is incentive compatible. We remark here that since path verification is strictly stronger than loop verification, by Theorem 4.3, it follows that Theorem 4.4 also holds in networks with path verification.

The proof of Theorem 4.4 is quite technically involved, so we present it in Appendix B. Roughly, the proof amounts to showing that when all nodes use next-hop policy, the only strategically useful lie available to the manipulator is to announce a false loop. Then, we show that if the network has loop verification, some node detects the false loop and punishes the manipulator for its lie; thus, since the utility of the manipulator drops down to zero when it gets caught, it no longer has an incentive to announce a false loop, and incentive compatibility follows.

4.6 Cannot relax restrictions to attractees only.

The requirement in Theorem 4.4 that every node in the network uses a next-hop policy with all of its neighbors is very strong indeed. We would like to relax it somewhat. Ideally, we would have preferred to require that only attractees use next-hop policy with their attractors. Unfortunately, we argue below that this is not the case:

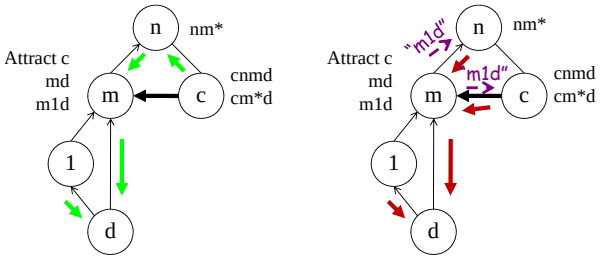


Figure 5: The Little Dipper.

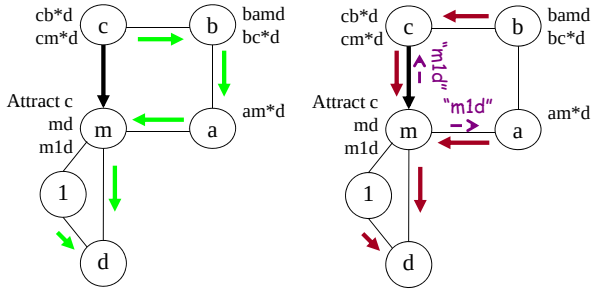


Figure 6: The Big Dipper.

Figure 5: THE LITTLE DIPPER is a Gao-Rexford network where all nodes use next-hop policy with their providers, and the only attraction relation is between c and its attractor m, over the direct link between them (notice that this relationship obeys AT4). In THE LITTLE DIPPER, node c does not use next-hop policy with its peer n by filtering the route $cnm1d$. The unique stable assignment T_1, when each node sets it algorithmic ranking equal to its valuation function, is shown on the left. The manipulated outcome T_2 is shown on the right, where each node except m is honest and set its algorithmic ranking equal to its valuation function. Here m has an incentive announce a false outgoing path "$m1d$" to all of its neighbors in order to attract traffic from the attractee c. Notice that this lie is possible even with path or loop verification, since node 1 announced the path "$1d$" to m.

Suppose instead that we required attractees to use next-hop policy with *all their neighbors*. It turns out that, in general, this is still not enough for incentive compatibility, as demonstrated in THE BIG DIPPER (Figure 6). (However, we show that this is enough in *Gao-Rexford* networks with path verification, see Theorem 5.2.)

Figure 6: THE BIG DIPPER is *not* a Gao-Rexford network. However, the network does not have a dispute wheel (see Appendix A) and all attractees (node c) use next-hop policy with all their neighbors. Notice also that all neighbors of the attractor also use next-hop policy with the attractor. Only node b — who is not a neighbor of the attractor — fails to use next hop policy by filtering the route $bam1d$. Again, the unique stable assignment T_1, when each node sets it algorithmic ranking equal to its valuation function, is shown on the left,

and the manipulated outcome T_2 is shown on the right, where each node except m is honest and set its algorithmic ranking equal to its valuation function. Again m has an incentive announce a false outgoing path "$m1d$" to all of its neighbors in order to attract traffic from the attractee c, and this false path can be announced even in networks with path or loop verification.

4.7 The need for ubiquitous participation.

BOWTIE, THE LITTLE DIPPER, and THE BIG DIPPER highlight another important point; namely, that even if a node follows the specified conditions for incentive compatibility, *e.g.,* next-hop policy, it is still possible for that node to learn a false path when some other node in the network fails to follow the specified conditions. For example, in BOWTIE (Figure 3), even though attractee node n uses next-hop policy with all its neighbors, n still learns a false path because node c fails to use next-hop policy. Similarly, in THE BIG DIPPER (Figure 6) only the 'distant' node b (who is not even a neighbor of the attractor) fails to use next-hop policy, and both nodes a and c, who use next hop policy learn false paths as a result. One can construct other counterexamples like THE BIG DIPPER where the single node that fails to use next-hop policy is arbitrarily far away from the nodes that learn false paths as a result. Thus, we emphasize that all the theorems in this paper only guarantee incentive compatibility if *every node* in the network follows the specified set of conditions. This need for ubiquitous participation mirrors the fact that the Gao-Rexford conditions only guarantee that BGP will converge if they are followed by every node [11], and the fact that SBGP only provides full path verification if it is deployed at each node.

5. RESULTS: GAO-REXFORD NETWORKS & ECONOMIC ATTRACTIONS

We now focus our attention on the setting where traffic attraction is economically-motivated, so that an attractor can only increase its utility when a customer routes on the link between them (per AT4 in Section 3.3). Since our focus here is on business relationships between nodes, we will also assume that the network obeys the Gao-Rexford conditions. It turns out that when we consider these more restricted networks (that obey GR1-GR3 and AT4, see Section 3.3) we can weaken next-hop policy restrictions required for incentive compatibility.

We already saw that even in this setting we must have either path or loop verification (FALSE LOOP in Figure 4), policy consistency is not enough (BOWTIE in Figure 3), and having next-hop between customers and providers is not enough (THE LITTLE DIPPER in Figure 5). However, we show below that there are two cases in which weaker notions of next-hop policy (in conjunction with some control plane integrity checks) suffice for incentive compatibility. Our results for this setting are summarized in the last four rows of Table 2.

5.1 It's best to be honest to your customers.

All of our results in this section hinge on the following lemma, which tells us that in Gao-Rexford networks with AT4 and either loop or path verification, where each attractee uses next-hop policy with all its providers and peers, then the optimal strategy to attract traffic is to behave honestly:

Lemma 5.1. *Consider an AS graph with path or loop verification, where the ranking functions of nodes obey the Gao-Rexford conditions GR1-GR3, and assume that each attractee in this graph uses next-hop policy with* all its providers and peers*. Let m be a node in the network, let T_1 be the unique stable assignment when m does not deviate from the BGP protocol, and let c be a customer attractee of m that* does not *route directly through m in T_1. Then there is no strategy that m can use to attract traffic from c over the direct link between them.*

The proof of this lemma, which is highly technical, is deferred to Appendix C. We prove this by contradiction. At a very high level, we use a series of arguments that use the Gao-Rexford conditions (GR1-GR3) to show that if m manages to attract traffic from c in T_2 but not in T_1, then it follows that the network must have a customer-provider loop.

5.2 With Gao-Rexford, next-hop at attractees can be enough!

When Gao-Rexford networks use path verification, it is sufficient for each attractee to have next-hop policy with all its providers and peers.

Theorem 5.2. *Consider an AS graph where there is path verification, and (1) the rankings of all nodes obey the Gao-Rexford conditions GR1-GR3, and (2) the rankings of every attractee are next-hop policy with* all its neighboring providers and peers*. Let m be any node with valuation and attraction functions that obey the Gao-Rexford conditions GR1-GR3 and AT4. Then it follows that the optimal strategy for m is to set its ranking to its valuation, and follow the BGP protocol without deviation.*

This theorem implies, as a corollary, that BGP is incentive compatible if each node sends its ranking function equal to its valuation function in a network that (1) uses path verification, (2) obeys the Gao-Rexford conditions (GR1-GR3) and the attraction condition AT4, and (3) where each attractee uses next-hop policy with his providers and peers.

Proof of Theorem 5.2. Let m be a "manipulator node", and let T_1 be the unique stable assignment when m sets

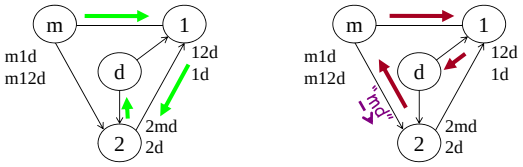


Figure 7: Counterexample from [21].

its ranking to equal its valuations (and all other nodes use their usual ranking as before).

Observe that the network satisfies the conditions of Levin, Schapira and Zohar's Theorem 4.1, since the valuation of m and ranking of the other nodes obey Gao-Rexford (and hence there is no dispute wheel) and the network uses path verification. Hence there is no strategy that m can use to get a path assignment T_2 with higher valuation than in T_1. On the other hand, our technical Lemma 5.1 says that m also cannot get any attractee that does not route through it in T_1. It follows that there is no strategy that m can use to get better utility than m had in T_1. □

5.3 All next-hop with peers & providers.

Unfortunately, replacing path verification with loop verification fails when only attractees use next-hop policy (*c.f.,* Theorem 5.2). To see why, consider the counterexample of Levin et al. [21] in Figure 7:[10]

Figure 7: In this Gao-Rexford network, there is no traffic attraction, and therefore no attractees. The unique stable assignment T_1 (where the ranking function is equal the valuation function at each node) is shown on the left. The manipulated outcome T_2 is shown on the right. Here the manipulator m has an incentive announce a false outgoing path "*md*" to node 1 in order to obtain the higher valued outgoing path $m1d$. Notice that m's manipulation would *not* cause an loop verification alarm, since it does not involve announcing a false loop.

Instead, we show below that when a Gao-Rexford network uses loop verification, it is sufficient for *every node* (not only attractees) to use next-hop policy with its providers and peers. To do this, we first need to prove a lemma similar to Theorem 4.1 and Theorem 4.2. This result, that tells us that *without traffic attraction*, Gao-Rexford networks with loop verification are incentive compatible when *all* nodes use next-hop policy with their provides and peers, may be somewhat interesting in its own right. The proof is in Appendix C.

Lemma 5.3. *Consider an AS graph* without traffic attraction*, in which the ranking functions at all nodes obey Gao-Rexford conditions GR1-GR3. Assume further that m is a node satisfying (1) the valuations of m obey GR1-GR3, and (2) m uses next-hop policy with all of m's neighboring peers and providers. Then it follows that the optimal strategy for m is to set its ranking to its valuation, and follow the BGP protocol without deviation.*

Then, we have the theorem below, which as a corollary implies BGP is incentive compatible when each node sets it ranking equal to its valuation:

Theorem 5.4. *Consider an AS graph with loop or path verification where the rankings of each node a (1) obey the Gao-Rexford conditions GR1-GR3 as well as the additional Attraction condition AT4, (2) uses next-hop policy with* all *a's* neighboring providers and peers*. Let m be any node with valuation and attraction functions that (1) obey the Gao-Rexford conditions GR1-GR3 and AT4, and (2) use next-hop policy with all m's neighboring peers and providers. Then it follows that the optimal strategy for m is to set its ranking to its valuation, and follow the BGP protocol without deviation.*

Proof. We can use Lemma 5.1 to argue that there is no strategy that m can use get traffic from an attractee that does not route through it in the unique stable assignment, and Lemma 5.3 to show that there is no strategy that m can use to get a path assignment with higher valuation than the unique stable path assignment. □

Next-hop at attractees, policy consistency everywhere else. We remark here that we can also combine Lemma 5.1 and the results of Feigenbaum et. al. [8,10] in Theorem 4.2. Then, we have that next-hop policy at all attractees and policy consistency everywhere else in a Gao-Rexford network with loop verification is sufficient for incentive compatibility.

6. CONCLUSIONS

In this work we considered control-plane mechanisms that provide incentives for rational ASes to honestly announce data-plane forwarding paths in their BGP messages (*i.e.,* incentive compatibility). We find that conditions that were previously shown to be sufficient to guarantee incentive compatibility no longer suffice if we assume that rational ASes can benefit by attracting incoming traffic from other ASes. We demonstrated that, in the absence of data-plane enforcement, incentive compatibility in the face of traffic attraction requires very strong restrictions on routing policy (*i.e.,* next-hop policy), as well as control-plane integrity checks (*i.e.,* loop verification or path verification protocols like Secure BGP [20]). Thus, our results suggest that in practice, it will be difficult to achieve incentive compatibility without resorting to expensive data-plane pro-

[10]Levin et al. used this counterexample to demonstrate that a network that obeys the Gao-Rexford conditions but does not have path verification is not necessarily incentive compatible.

tocols that verify and enforce AS-level paths. By highlighting the difficulty of matching the control- and data-planes, even under the assumption that ASes are rational (and not arbitrarily malicious), our results can also help inform decisions about whether security protocols should be deployed in the control-plane, in the data plane, or in both.

Acknowledgments

We thank Jennifer Rexford, Michael Schapira and Boaz Barak for useful discussions, and Matthew Caesar, Andreas Haeberlen, Martin Suchara and Gordon Wilfong for comments that improved the presentation.

7. REFERENCES

[1] K. Argyraki, P. Maniatis, D. Cheriton, and S. Shenker. Providing packet obituaries. *ACM HotNets-III*, 2004.
[2] I. Avramopoulos and J. Rexford. Stealth probing: Data-plane security for IP routing. *USENIX*, 2006.
[3] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the internet. In *ACM SIGCOMM*, 2007.
[4] S. Balon and G. Leduc. Can forwarding loops appear when activating ibgp multipath load sharing? In *AINTEC*, 2007.
[5] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. Technical report, ATT Labs-Research, 2004.
[6] J. Feigenbaum, D. R. Karger, V. S.Mirrokni, and R. Sami. Subjective-cost policy routing. In X. Deng and Y. Ye, editors, *First Workshop on Internet and Network Economics*, 2005.
[7] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, 18(1), July 2005.
[8] J. Feigenbaum, V. Ramachandran, and M. Schapira. Incentive-compatible interdomain routing. In *Conference on Electronic Commerce*, page 130139, 2006.
[9] J. Feigenbaum, R. Sami, and S. Shenker. Mechanism design for policy routing. *Distributed Computing*, 18(4):293–305, 2006.
[10] J. Feigenbaum, M. Schapira, and S. Shenker. *Algorithmic Game Theory*, chapter Distributed Algorithmic Mechanism Design. Cambridge University Press, 2007.
[11] L. Gao, T. Griffin, and R. Rexford. Inherently safe backup routing with BGP. *IEEE Infocomm*, 2001.
[12] L. Gao and R. Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. on Network.*, 2001.
[13] V. Gill, J. Heasley, and D. Meyer. The generalized ttl security mechanism (gtsm). RFC 3682, 2004.
[14] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path quality monitoring in the presence of adversaries. In *SIGMETRICS*, June 2008.
[15] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Network and Distributed System Security Symposium*, 2003.
[16] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. on Network.*, April 2002.
[17] A. Heffernan. Protection of bgp sessions via the tcp md5 signature option. RFC 2385, 1998.
[18] K. J. Houle and G. M. Weaver. Trends in denial of service attack technology. Technical report, CERT Coordination Center, October 2001.
[19] G. Huston. Interconnection, peering, and settlements. In *Internet Global Summit (INET), The Internet Society*, June 1999.
[20] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *J. Selected Areas in Communications*, 18(4):582–592, April 2000.
[21] H. Levin, M. Schapira, and A. Zohar. Interdomain routing and games. In *ACM STOC*, May 2008.
[22] X. Liu, X. Yang, D. Wetherall, and T. Anderson. Efficient and secure source authentication with packet passports. In *SRUTI'06: Steps to Reducing Unwanted Traffic on the Internet*, pages 2–2. USENIX, 2006.
[23] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.
[24] V. Padmanabhan and D. Simon. Secure traceroute to detect faulty or malicious routing. *HotNets-I*, 2002.
[25] D. C. Parkes and J. Shneidman. Specification faithfulness in networks with rational nodes. In *Symposium on Principles of Distributed Computing*, 2004.
[26] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *ACM SIGCOMM*, 2006.
[27] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and Whisper: Security mechanisms for BGP. In *NSDI*, March 2004.
[28] F. Wang and L. Gao. On inferring and characterizing internet routing policies. In *ACM IMC '03*.
[29] R. White. Deployment considerations for secure origin bgp (sobgp), draft, internet engineering task force. draft-white-sobgp-bgp-deployment-01.txt, June 2003.
[30] E. L. Wong, P. Balasubramanian, L. Alvisi, M. G. Gouda, and V. Shmatikov. Truth in advertising: Lightweight verification of route integrity. In *PODC*, 2007.
[31] J. Z. Mao, J. Rexford and R. H. Katz. Towards an accurate AS-level traceroure tool. In *ACM SIGCOMM*, 2003.

APPENDIX

A. A CONDITION FOR STABILITY

Because BGP is a distributed protocol, earlier works [11, 12, 16] considered technical conditions that ensure that BGP converges to a *unique stable* path assignment. A path assignment T is *stable* if for every node a, the path that a uses to the destination in T is a's highest ranked path among all the paths that were advertised to a by its neighbors [16]. In our setting, we think of this "highest ranked path" as defined by a's algorithmic ranking function. Then, if ASes are 'honest' (*i.e.,* they follow the prescribed BGP mechanism), then once the protocol converges to a stable path assignment, this path assignment will be used by all nodes from then on.

Griffin et al. [16] described a global condition on the routing policies in the AS graph, called *'no dispute wheel'*, that ensures that a network has a unique stable path assignment, and moreover that BGP will converge to that path assignment. Roughly, a dispute wheel is essentially a set of nodes and their routing policies that can induce a routing anomalies like oscillations or non-convergence of BGP. While the exact definition of a dispute wheel is not important for the current work, we use several results from the literature [8, 10, 21] that only hold for networks with no dispute wheels. Thus, in this paper, *every network we consider always has 'no dispute wheel'*. Focusing on networks that have 'no dispute wheel' has practical advantages (instability can

wreck havoc on routing performance), and theoretical ones (we need to have a single path assignment in order to define the utility of a node). We can also feel comfortable with this assumption because the Gao-Rexford conditions are known to imply the no dispute wheel condition [11], and the Gao-Rexford conditions are believed to hold in most places in the current Internet [19, 28].

B. PROOFS: GENERIC ATTRACTIONS

We prove Theorem 4.4. Our proof uses the following lemma about unilateral manipulations in the AS graph. (We also use it later in the proofs of Lemma 5.1 and Lemma 5.3):

Lemma B.1 (False path lemma)**.** *Consider an AS graph where all the nodes in the graph except perhaps a single manipulator node m follow the BGP protocol, and let T be the unique path assignment that results. Suppose node n announces a false path P (*i.e., *P differs from the path that n uses in T). It follows that P must be of the form $P = nRmQd$ where nRm a true path and mQd is a false path.*

Proof. We first prove by contradiction that if n announces a path P that differs from the path that n uses in T, then m must be on the path that n uses in T. Let the path n uses in T be $n = a_r \rightarrow a_{r-1} \rightarrow ... \rightarrow a_1 \rightarrow a_0 = d$ and assume toward contradiction that m is not on this path. In this case, each node on the path n uses is honest, thus each node a_i announces to a_{i+1} a path $a_i a_{i-1} R$ where R is the path that a_{i-1} announced to a_i. Thus, a simple induction starting at node a_0 shows that node n announces a path $P = na_{r-1}...a_1 d$, which matches n's path in T. This contradicts the fact that n announces a false path, so we have that m must be on n's path in T. Suppose m occupies node a_j on n's path in T and announces a path Qd to node a_{j+1}. Since all the other nodes in the AS graph are honest, a similar induction starting at node m shows that n announces an path $P = na_{r-1}...a_{j+1}mQd$. Thus, letting path $R = a_{r-1} \rightarrow ... \rightarrow a_{j+1}$ we have that nRm is a true path. Since by assumption n announces a false path, it follows that path mQd must therefore be a false path. □

We also need a lemma of Feigenbaum et. al [8, 10], that says that policy-consistent networks are incentive compatible when there is no traffic attraction. Before we state their lemma, we first formally define policy consistency:

Definition B.2 (Policy consistency)**.** Two neighboring nodes a and b are said to have an *inconsistent policy* in their valuations if there are two permissible paths $abQd, abRd$ from a to the destination such that $v_b(bQd) \geq v_b(bRd)$ but $v_a(abQd) < v_a(abRd)$ (see *e.g.,* Figure 1). Then, nodes a and b are policy consistent in their valuation if they do not have an inconsistent policy, and moreover for every two paths $abQd, abRd$, if b is willing to transit a's traffic on $abQd$ and $v_b(bQd) \leq v_b(bRd)$ then b is also willing to transit a's traffic on $abRd$. (In other words, if $abQd$ and bRd are permissible and $v_b(bQd) \leq v_b(bRd)$ then also $abRd$ is permissible.)[11] The notion of policy consistency in the ranking function is defined similarly, with $r(\cdot)$ replacing $v(\cdot)$.

Lemma B.3 (From [10, Lemma 14.8])**.** *Consider an AS graph with no dispute wheel, in which the ranking functions at all nodes are policy consistent with all their neighbors. Then, the unique stable path assignment T_1 is locally optimal for all nodes. That is, for every node m in the AS graph, then $r_m(T_1) \geq r_m(T_2)$ for any path assignment $T_2 \neq T_1$.*

We are now ready to prove Theorem 4.4, that tells us a network with loop verification and (general) attraction relationships is incentive compatible if each node uses next-hop policy:

Proof of Theorem 4.4. First observe that if m sets its ranking to equal its valuation and every node honestly follows the protocol, then the network has a unique stable path assignment, which is a tree that we denote T_1.

Assume toward contradiction that node m is able to induce another assignment $T_2 \neq T_1$ (called the *manipulated assignment*) by unilaterally deviating from the protocol, and $u_m(T_1) < u_m(T_2)$.

By the discussion in Section 2.2 we can assume without loss of generality that m has a single outgoing link in T_2. We then prove that some other node b must have raised an alarm because it receives a path announcement of the form QbR where b did not announce the path R, and where m is on path Q. This contradicts either path verification (since b receive an announcement containing path through b that b did not announce) or loop verification (where the utility of m is set to zero when such an alarm is raised).

Since T_1 is the unique stable assignment that is obtained when m uses its valuation as ranking, since this valuation is policy consistent with the ranking of all other nodes, and since all other nodes use next-hop policy in their ranking (and thus are also policy consistent), then Lemma B.3 implies that $v_m(T_1) \geq v_m(T_2)$. But we know that m's utility is higher in T_2 than in T_1, so m must have gained utility from traffic attraction in T_2 that it did not have in T_1, namely $\alpha_m(T_2) > \alpha_m(T_1)$, which implies that $T_2(m) \not\subseteq T_1(m)$. (Recall from Sec-

[11] We remark that the definition of policy consistency from [10] does not explicitly include the transit condition stated here, but their proofs appear to implicitly rely on it. The "consistent filtering" condition from [8] is similar, but not exactly the same, as the one we use here.

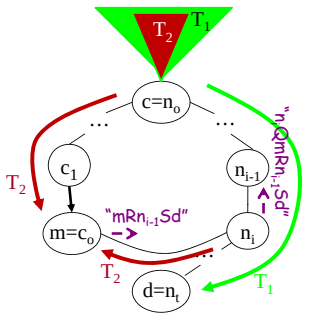


Figure 8: The proof of Theorem 4.4

tion 2.2 that $T(m)$ is the set of all edges in simple paths from other nodes to m in T.)

Hence there must exist some node that routes through m in T_2 and uses a different outgoing link in T_2 than in T_1. Denoting $m = c_0$, we continue to find nodes $c_i (i \geq 1)$ as follows: For each node c_i, if there are nodes that route through c_i in T_2 and use a different outgoing link in T_2 than in T_1, then we let c_{i+1} be the "closest to c_i among all these nodes. Namely, c_{i+1} uses a different outgoing link in T_2 than in T_1, but all the other nodes on the path from c_{i+1} to c_i in T_2 (if any) use the same outgoing link in T_1 and T_2.

We repeat this process until we reach a "last node" c, such that every other node that routes through it in T_2 (if any) uses the same outgoing edge in T_1 and T_2. Observe that since we must reach such "last node" since otherwise we will eventually repeat a node, say node c_r. But, if we repeat a node, then it must follow that c_r is a part of routing loop in T_2. Since by construction c_r routes through m, it follows that m is part of a routing loop as well. However, if m is part of a routing loop in T_2, it follows that $u_m(T_2) = 0$ which contradicts our assumption that $u_m(T_2) > u_m(T_1) > 0$ (where the second inequality follows because m has a path to d in T_1).)

Let c be the 'last node' in the sequence above. That is, c is a node that (1) routes through m in T_2, (2) uses a different outgoing edge in T_2 than in T_1, and (3) every node that routes through c in T_2 uses the same outgoing link in T_1 and T_2 (and therefore must also routes through c in T_1). We now again invoke Lemma B.3 to conclude that $r_c(T_1) > r_c(T_2) > 0$ (where we know that $r_c(T_2) > 0$ because by construction c has a path to the destination through m in T_2), which in particular implies that c can reach the destination in T_1. Denote the path of c in T_1 by $c = n_0 \to n_1 \to \cdots \to n_t = d$. Let n_i be the node closest to c on this path (but not c itself) that has any path available to the destination in T_2. (We know that such a n_i exists, because in particular node n_{t-1} is one hop away from d.)

Now we observe that n_i does not route in T_2 through n_{i-1}, since:

(i) If $i = 1$ (so $n_{i-1} = c$) then by construction n_1 does not route through c in T_1, and so it also does not route through in in T_2 (because of the way we chose c).

(ii) If $i > 1$, then n_i has a path to the destination in T_2 but n_{i-1} does not, so in particular n_i does not route through n_{i-1} in T_2.

Denote the path of n_i to d in T_1 by $n_i P_1 d$, and its path to d in T_2 by $n_i P_2 d$. We know that neither path goes through n_{i-1}, so the next-hop assumption implies that $r_{n_{i-1}}(n_{i-1} n_i P_2 d) = r_{n_{i-1}}(n_{i-1} n_i P_1 d) = r_{n_{i-1}}(T_1)$, and that n_i is willing to transit n_{i-1}'s traffic over the path $n_i P_2 d$. On the other hand, Lemma B.3 implies that $r_{n_{i-1}}(T_2) < r_{n_{i-1}}(T_1) = r_{n_{i-1}}(n_{i-1} n_i P_2 d)$. Namely, n_{i-1} has an available path through n_i in T_2 with ranking higher than what n_{i-1} actually uses in T_2. But n_{i-1} does not use the link to n_i in T_2, so it must be that the path that n_i announces to n_{i-1} does go through n_{i-1} (even though the actual path that n_i uses does not).

Since all nodes but m are honest, it follows from the false path lemma (Lemma B.1) that the path that n_i announces has the form $P = n_i Q m R n_{i-1} S d$ (for some paths Q, R, S). But n_{i-1} could not have announced the path $n_{i-1} S d$, since:

(i) If $i = 1$ (so $n_{i-1} = c$) then c routes through m, and therefore it must announce a path that includes m (since m is the only manipulator in the graph), which would make the path P include a loop, contradicting the fact that n_i chose it.

(ii) If $i > 1$, then n_{i-1} has no path to the destination in T_2, so in particular it did not announce the path $n_{i-1} S d$.

Hence n_{i-1} would raise an alarm for a path containing m, which is what we needed to prove. □

C. PROOFS: GAO-REXFORD NETWORKS

In this section, we prove Lemma 5.1 (Appendix C.1) and Lemma 5.3 (Appendix C.2) for Gao-Rexford networks with customer attractions. But first, we need the following useful concept:

Transitive customers. We refer to a node c as a *strict transitive customer* of node a if node c is connected to node a via a path consisting of only customer-provider links as in the right half of Figure 9. We also restate here a simple, useful lemma of the Gao-Rexford conditions proved by Gao, Griffin and Rexford in [11].

Lemma C.1 (Transitive customers [11, Theorem VII.4]). *If either the path $P = abRc$ or the path $P' = cR'ba$ is permissible, and if node a is not a customer of node b, then node c is a strict transitive customer of node b over the permissible path.*

We also need to prove the following helper lemma that we shall use to derive contradictions in several places below:

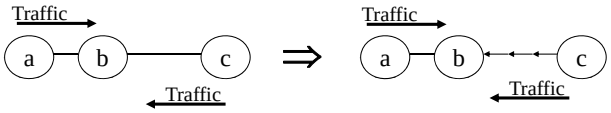


Figure 9: Lemma C.1.

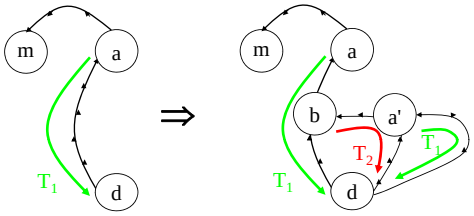


Figure 10: Proof of Lemma C.2

Lemma C.2. *Consider an AS graph where the ranking functions of nodes obey the Gao-Rexford conditions (GR1-GR3), and assume that all the nodes in the graph except perhaps a single manipulator node m follow the BGP protocol. Let T_1 be the unique stable assignment when m follows the BGP protocol, and let T_2 be a path assignment that results from some other arbitrary strategy of m.*

If there is a node a in the network such that (1) a is a strict transitive customer of the manipulator m, (2) a uses a different path in T_2 than in T_1, and (3) the destination d is a strict transitive customer of a along a's path in T_1. Then there is a different node $a' \neq a$ which is a strict transitive customer of a, such that a' also satisfies the conditions (1)-(3).

Proof. Since a is a strict transitive customer of m, and the destination d is a strict transitive customer of a on a's T_1 path, then the Topology condition GR1 implies that m cannot be on the path of a in T_1. Denote by b the node closest to the destination along a_i's path in T_1 that uses a different path in T_2 than in T_1 (we know that such a b exists since in particular node a is such a node), and denote the paths of b in T_1 and T_2 by bQ_1d and bQ_2d, respectively. It follows that the path Q_1d is available to b in T_2, and from the Topology condition GR1 we know that m is not on that path.

Since all the nodes on the path Q_1d are honest and they all use that path in T_2, then b must have received an announcement Q_1d from the first hop on that path, and yet it chose a different path in T_2. We conclude that b's ranking has $r_b(T_2) > r_b(T_1)$. And since b's next hop in T_1 is a customer, the Preferences condition GR3 implies that b's next hop in T_2 must also be a customer, and applying Lemma C.1 we get that the destination is a strict transitive customer of b along the path bQ_2d.

Let node a' be the node closest to the destination along the path bQ_2d that uses a different path in T_2 than in T_1, and denote the paths of a' in T_1 and T_2 by $a'R_1d$ and $a'R_2d$, respectively. It follows that the path R_2d is available to a' also in T_1. Notice that a' is also a strict transitive customer of the manipulator m, and that that destination d is a strict transitive customer of a' along the path R_2d. Since all the nodes on the path R_2d uses it also in T_1, then a' must have received an announcement R_2d from the first hop on that path, and yet it chose a different path in T_1. We conclude that the ranking of a' has $r_{a'}(T_1) > r_b(T_2)$, which also implies that $a' \neq b$.

Since $r_{a'}(T_1) > r_{a'}(T_2)$ and since the next hop after a' on the path $a'R_2d$ in T_2 is a customer of a', the Preferences condition GR3 implies that the next hop after a' on the path $a'R_1d$ in T_1 must also be a customer. Then, we can apply Lemma C.1 to find that the destination is a strict transitive customer of a' along the path $a'R_1d$ in T_1.

We established that a' satisfies the conditions (1)-(3), and we also know that b is a transitive customer of a (or a itself), a' is a strict transitive customer of b, and $a' \neq b$. It follows that $a' \neq a$, since otherwise we would have a customer-provider loop in the graph. □

C.1 It's best to be honest to your customers.

We now prove Lemma 5.1, that tells us that the optimal strategy to attract an attractee that uses next-hop policy with *all its providers and peers* in a Gao-Rexford network is to behave honestly.

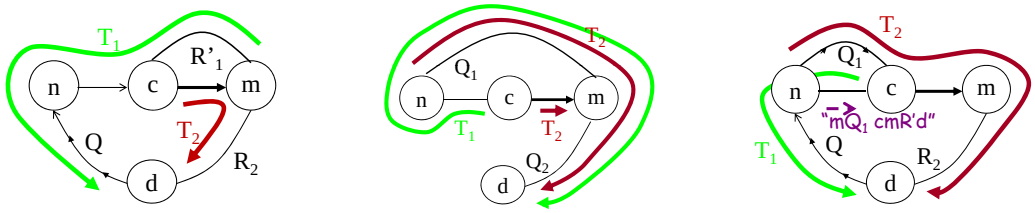
Proof of Lemma 5.1. Let m, c and T_1 be as in the lemma statement. Assume toward contradiction that there is some strategy that the "manipulator" m can use, that results in a different path assignment T_2 in which c routes directly through m. We derive a contradiction by proving a sequence of claims that together imply that the conditions of Lemma C.2 must hold in this graph, and then repeatedly apply Lemma C.2 to show that the graph contains a customer-provider cycle, violating the Topology condition GR1.

Denote the paths of m to the destination in T_1 and T_2 by mR_1d and mR_2d, respectively. Also denote the path of c to the destination in T_1 by $cnQd$ (where n is the first node after c on that path, and we know that $n \neq m$).

Claim C.3. *The ranking of c satisfies $r_c(T_1) > r_c(T_2)$.*

(We comment that this claim *does not follow* from Lemma B.3, since we do not assume to have policy consistency everywhere.)

Proof. Assume toward contradiction that $r_c(cmR_2d) > r_c(cnQd)$. Since c is an attractee of m (and therefore its customer), then c must use next-hop policy with m. It follows that the path R_1 must go through c, or else we would have $r_c(cmR_1d) = r_c(cmR_2d) > r_c(cnQd)$



(a) Claim C.3 (b) Claim C.4 (c) Claim C.5

Figure 11: Pictorial representation of the proof of Lemma 5.1

(where the equality follows from the fact that c uses next-hop policy with m) and c would have routed through m also in T_1. Hence we can re-write the path that m takes in T_1 as $R_1 = R'_1 cnQ$, as depicted in Figure 11(a).

Since c is a customer of m, it follows from the Topology condition GR1 that m cannot be a strict transitive customer of c along the path mR'_1c. Hence there are adjacent nodes between m and c on the path R'_1 (call them a, b) such that a is not a customer of b. Since the path mR'_1cnQd is permissible (because it is actually used in T_1), we can apply Lemma C.1 to conclude that d is a transitive customer of b along this path. In particular it means that n is a customer of c. Notice that this is true even if $n = d$. But this violates the Preferences condition GR3, since we assumed that $r_c(cmR_2d) > r_c(cnQd)$ where m is a provider of c and n is its customer. □

From Claim C.3 we can also conclude that $n \neq d$ (otherwise, c would have a path available through d in T_2, and since $r_c(cd) = r_c(T_1) > r_c(T_2) = r_c(cmR_2d)$, c would not route through m in T_2.)

Claim C.4. *The node n uses a different path for its traffic in T_2 than in T_1.*

Proof. Assume toward contradiction that n uses the same path nQd from T_1 also in T_2.

From Claim C.3 we know that $r_c(cmR_2d) < r_c(cnQd)$, so if n would have announced to c its true path to the destination then c would have routed through n also in in T_2. Hence n must have announced a false path in T_2. On the other hand we know that n is honest (since $n \neq m$), and so from the false path lemma (Lemma B.1) we can write path that n uses in T_2 as $Q = Q_1 mQ_2$ and the false path that n announces to c as $nQ'd$ for $Q' = Q_1 mQ'_2$, where $Q'_2 \neq Q_2$. (Note that both Q and Q' begin with the true path nQ_1m.) We now distinguish between two cases: either n is a customer of c, or it is not.

1. If n is a customer of c, then applying Lemma C.1 to the permissible path $cnQd = cnQ_1mQ_2d$, we conclude that d must be a strict transitive customer of c on this path, and therefore also m must be a strict transitive customer of c. But c is a customer of m, so we have a customer-provider loop in the graph, violating the Topology condition GR1.
2. If n is not a customer of c, then c must use next-hop policy with n. (This follows since c is an attractee of m, so it must use next-hop policy with *all* its providers and peers, including with n.)

   Therefore the false path $nQ_1mQ'_2d$ that n announces to c must include c itself, or else we would have $r_c(cnQ'd) = r_c(cnQd) > r_c(cmR_2d)$ (where the equality follows from the fact that c uses next hop policy with n and the inequality from Claim C.3) and c would not have routed on cmR_2d in T_2. But the path nQ_1mQ_2d that n uses in T_2 does not go through c (because c uses the path cnQ_1mQ_2d in T_1, and if c was on either Q_1 or Q_2 we would have a routing loop in T_1). It follows that the false path mQ'_2d that n announces to c in T_2 must include c. We now re-write the path $nQ'd$ as $nQ_1mQ''_2cQ''_3d$. (Note that Q''_3 cannot include the node m, or else this path would have included a routing loop and n would not have chosen it in T_2.)

   However, c could not have announced the path cQ''_3d in T_2, since c routes through m in T_2 and Q''_3 does not include the node m. This contradicts our assumption that the network uses path or loop verification: if we had path verification then no node could have announced a path that includes cQ''_3d, and if we only have loop verification then c would have seen this announcement from n and raised an alarm, and in particular it would not have routed through m.

With both these cases leading to contradictions, we conclude that the node n cannot use the path nQd in T_2. □

Claim C.5. *The node n is a strict transitive customer of m, and the destination d is a strict transitive customer of n over the T_1 path nQd.*

Proof. If n is a direct customer of c then the first part of the lemma follow trivially (since c is a customer of m),

and the second part follows by applying Lemma C.1 to the permissible path $cnQd$.

If n is not a customer of c, then c must use next hop policy with n. Let $nQ'd$ be that path that n announces to c in the manipulated outcome. Then it must be that c is somewhere on the path Q', or else we would have $r_c(cnQd) = r_c(cnQ'd) > r_c(cmR_2d)$ (where the equality due to the fact that c uses next-hop policy with n and the last inequality due to Claim C.3), and c would not have chosen to route through m in T_2.

Since c routes through m in T_2, then it also announces a path that goes through m. Denote by $cmR'd$ the path announced by c. As before, we use path or loop verification at c to conclude that the path $nQ'd$ that is announced by n to c must have $cmR'd$ as a suffix (otherwise c would raise an alarm when it receives the announcement $nQ'd$ from n). Therefore re-write the path $nQ'd$ as $nQ'_1cmR'd$. We know that Q'_1 does not include m (or else n wouldn't have chosen this path since it would contain a routing loop through m). Hence the partial path nQ'_1cm must be the one that is actually used to route traffic in T_2 (and in particular it must be a permissible path). Applying Lemma C.1 to this path, we conclude that n is a strict transitive customer of c (and therefore also of m).

Moreover, since n is a strict transitive customer of c then the Topology condition GR1 says that it cannot be a provider of c. We assumed that n is also not a customer of c, so they must be peers. We can now apply Lemma C.1 to the permissible path $cnQd$, to conclude that the destination d is a strict transitive customer of n over this path. □

Claims C.4 and C.5 established the existence of a node n which is (1) a strict transitive customer of the manipulator m, and where (2) n uses a different path in T_2 than in T_1, and (3) the destination d is a strict transitive customer of n along n's path in T_1. Lemma C.2 asserts that there must be another node $n_1 \neq n$ which is a strict transitive customer of n, where n_1 also satisfies the conditions (1)-(3). Repeated applications of this lemma thus give us a sequence of nodes $n_1, n_2, \ldots$ such that for all i $n_i \neq n_{i-1}$ and n_i is a strict transitive customer of n_{i-1} (and they all satisfy the same conditions). Since there are a finite number of nodes in the AS graph, eventually one of the nodes in the sequence will repeat, resulting in a customer-provider cycle and violating the Topology condition GR1.

We see that our assumption that c routes through m in T_2 leads to a contradiction, this concluding the proof of Lemma 5.1. □

C.2 All next-hop with peers & providers.

We now prove Lemma 5.3, that tells us that networks without traffic attraction where all nodes use next-hop policy with *all their providers and peers* are incentive compatible.

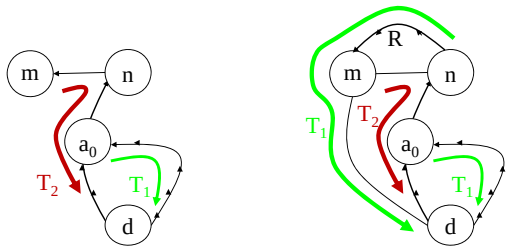


Figure 12: The proof of Lemma 5.3. Left, n is a customer of m. Right, n is not.

Proof of Lemma 5.3. Let m be a "manipulator node", and let T_1 be the unique path assignment when m sets its ranking to equal its valuations (and all other nodes use their ranking as before).

Assume to the contrary, that m can bring about a different path assignment T_2 such that $v_m(T_2) > v_m(T_1)$. We use Lemma C.2 to show that this assumption implies a customer-provider loop, violating the Topology condition GR1. Denote the T_2 path of m by $mnQd$ (i.e., n is the next hop in the T_2 path of m). We distinguish two cases: either n is a customer of m, or it is not:

1. If n is a customer of m — that provides transit to m's traffic along the path $mnQd$ — then we can apply Lemma C.1 to see that the destination must be a strict transitive customer of m (and n) along this path. Since $v_m(T_2) > v_m(T_1)$ and in T_1 we know that m sets its algorithmic ranking to equal its valuation function, then it must be that n uses a different paths in T_2 and T_1 (or else m would have chosen n also in T_1, which means that it was using the same path in both T_1 and T_2).
2. If n not is a customer of m, then m must use next hop policy with n in its valuation function. This means that for every path $nQ'd$ that does not go through m, we have $v_m(mnQ'd) = v_m(mnQd) = v_m(T_2) > v_m(T_1)$. So it must be that n routes through m in T_1 (or else m would have routed thought n in T_1 and we would get $v_m(T_2) = v_m(T_1)$). Denote the path that n takes in T_1 by $nRmR'd$. Since n is not a customer of m and since $v_m(T_2) = v_m(mnQd) > v_m(mR'd) = v_m(T_1)$, the Preferences condition GR3 implies that m's next hop on the path $mR'd$ in T_1 cannot be m's customer. But since m transits n's traffic on the path $nRmR'd$ in T_1, then applying Lemma C.1 to this path (upto m's next hop on $mR'd$) we get that n must be a strict transitive customer of m along R. This, in turn, implies that n cannot be a provider of m (because of the Topology condition GR1), so it must

be a peer of m.

Since n is a peer of m, then applying Lemma C.1 to the path $mnQd$ in T_2 implies that the destination d is a strict transitive customer of n along path Q.

In both cases we concluded that the node n satisfies (i) n is a strict transitive customer of m, (ii) n uses a different path in T_1 and in T_2, and (iii) the destination d is a strict transitive customer of n along n's T_2 path.

From items 1 and 3, it follows that node m cannot be on the path nQd in T_2 (otherwise we would have a customer-provider cycle). Since all nodes except the manipulator act honestly, it follows that all nodes on the path nQd in T_2 announce this true path to their neighbors.

Let a_0 be the node closest to the destination on the path nQd that uses a different path in T_2 than in T_1 (we know that such a node exists since in particular n is such a node). It follows that a_0's T_2 path is available to a_0 in T_1, and that a_0 receives the announcement of this path in both T_1 and T_2. Since a_0 chooses a different path in T_1, it follows that $r_{a_0}(T_1) > r_{a_0}(T_2)$.

Now, if we apply the Preferences condition GR3 to a_0's algorithmic ranking, we must have that a_0's next hop in T_1 is a customer. Then, applying Lemma C.1 to a_0's T_1 path, we have that the destination d must be a strict transitive customer of a_0 along a_0's T_1 path.

We established the existence of a node a_0 such that (1) a_0 is a strict transitive customer of n (and therefore also of m), (2) a_0 uses a different path in T_1 than in T_2, and (3) the destination d is a strict transitive customer of a_0 along a_0's T_1 path. These are the conditions needed for Lemma C.2, and we can now repeatedly apply this lemma to obtain a sequence of nodes $a_0, a_1, a_2, ...$ connected by customer-provider edges. Since there are a finite number of nodes in the AS graph, eventually one of the nodes in the sequence will repeat, resulting in a customer-provider cycle and violating the Topology condition GR1). □