

2011-02-04

# Let the market drive deployment: a strategy for transitioning to BGP security

---

Gill, Phillipa; Schapira, Michael; Goldberg, Sharon. "Let the Market Drive Deployment: A Strategy for Transitioning to BGP Security", Technical Report BUCS-TR-2011-003, Computer Science Department, Boston University, February 4, 2011. [Available from: <http://hdl.handle.net/2144/11360>] <https://hdl.handle.net/2144/11360>

*"Downloaded from OpenBU. Boston University's institutional repository."*

# Let the Market Drive Deployment: A Strategy for Transitioning to BGP Security

Full version from February 10, 2011

Phillipa Gill\*  
University of Toronto

Michael Schapira  
Princeton University

Sharon Goldberg  
Boston University

## Abstract

With a cryptographic root-of-trust for Internet routing (RPKI [18]) on the horizon, we can finally start planning the deployment of one of the secure interdomain routing protocols proposed over a decade ago (Secure BGP [24], secure origin BGP [43]). However, if experience with IPv6 is any indicator, this will be no easy task. Security concerns alone seem unlikely to provide sufficient local incentive to drive the deployment process forward. Worse yet, the security benefits provided by the S\*BGP protocols do not even kick in until a large number of ASes have deployed them.

Instead, we appeal to ISPs' interest in increasing revenue-generating traffic. We propose a strategy that governments and industry groups can use to harness ISPs' local business objectives and drive global S\*BGP deployment. We evaluate our deployment strategy using theoretical analysis and large-scale simulations on empirical data. Our results give evidence that the market dynamics created by our proposal can transition the majority of the Internet to S\*BGP.

## 1. INTRODUCTION

The Border Gateway Protocol (BGP), that sets up routes from autonomous systems (ASes) to destinations on the Internet, is amazingly vulnerable to attack [7]. Every few years, a new failure makes the news; ranging from misconfigurations that cause an AS to become unreachable [37, 32], to possible attempts at traffic interception [11]. To remedy this, a number of widely-used stop-gap measures have been developed to *detect* attacks [22, 28]. The next step is to harden the system to point where attacks can be *prevented*. After many years of effort, we are finally seeing the initial deployment of the Resource Public Key Infrastructure (RPKI) [4, 38, 2, 29], a cryptographic root-of-trust for Internet routing that authoritatively maps ASes to their IP prefixes and public keys. With RPKI on the horizon, we can now realistically consider deploying the S\*BGP protocols, proposed a decade ago, to prevent routing failures by validating AS-level paths: Secure BGP (S-BGP) [24] and Secure Origin BGP (soBGP) [43].

### 1.1 Economic benefits for S\*BGP adoption.

While governments and industry groups may have an interest in S\*BGP deployment, ultimately, the Internet lacks a centralized authority that can mandate the deployment of a new secure routing protocol. Thus, a key hurdle for the transition to S\*BGP stems from the fact that each AS will make deployment decisions according to its own local business objectives.

**Lessons from IPv6?** Indeed, we have seen this problem before. While IPv6 has been ready for deployment since around 1998, the lack of tangible local incentive for IPv6 deployment means that we are only now starting to see the seeds of large-scale adoption. Conventional wisdom suggests that S\*BGP will suffer from a similar lack of local incentives for deployment. The problem is exacerbated by the fact that an AS cannot validate the correctness of an AS-level path unless all the ASes on the path deployed S\*BGP. Thus, the security benefits of S\*BGP only apply after a large fraction of ASes have already deployed the protocol.

**Economic incentives for adoption.** This paper side steps these issues by showing that global S\*BGP deployment is possible even if ASes local deployment decisions are *not* motivated by perceived improvements in security. To this end, we present a prescriptive strategy for S\*BGP deployment that relies solely on Internet Service Providers' (ISPs) local economic incentives to drive global deployment; namely, ISP's interest in attracting revenue-generating traffic to their networks.

### 1.2 A strategy for S\*BGP deployment.

Our strategy is prescriptive (Section 2). We propose guidelines for how (a) ASes should deploy S\*BGP in their networks, and (b) governments, industry groups, and other interested parties should invest their resources in order to drive S\*BGP deployment forward.

**1. Break ties in favor of secure paths.** First, we require ASes that deploy S\*BGP to actually use it to inform route selection. However, rather than requiring security be the first criteria that ASes use to select routes, we only require secure ASes to *break ties* between equally-good routes in favor of secure routes.

This way, we create incentives for ISPs to deploy S\*BGP so they can transit more revenue-generating customer traffic than their insecure competitors.

**2. Make it easy for stubs to adopt S\*BGP.** 85% of ASes in the Internet are *stubs* (*i.e.*, ASes with no customers) [9]. Because stubs earn no revenue from providing Internet service, we argue for driving down their deployment costs with a new idea from the Internet standards community: simplex (unidirectional) S\*BGP. We require secure ISPs to be responsible for deploying simplex S\*BGP at each of their stub customers. In practice, this means that simplex S\*BGP must either be extremely lightweight or heavily subsidized.

**3. Create market pressure via early adopters.** We propose that governments and industry groups concentrate their regulatory efforts, subsidies, or financial incentives, on convincing a small set of *early adopters* to deploy S\*BGP. We show that this set of early adopters can create sufficient market pressure to convince a large fraction of ASes to follow suit.

### 1.3 Evaluation: Model and simulations.

To evaluate our proposal, we needed a model of the S\*BGP deployment process.

**Inspiration from social networks?** At first glance, it seems that the literature on technology adoption in social networks would be applicable here [23, 42, 39, 44, 33, 19]. However, in social networks models, an entity’s decision to adopt a technology depends only on its immediate *neighbors* in the graph; in our setting, this depends on the number of secure *paths*. This complication means that many elegant results from this literature have no analogues in our setting (Section 9).

**Our model.** In contrast to earlier work that assumes that ASes deploy S\*BGP because they are concerned about security [8, 5], our model assumes that ISPs’ local deployment decisions are based solely on their interest in increasing customer traffic (Section 3).

We carefully designed our model to capture a few crucial issues, including the fact that (a) traffic transited by an ISP can include flows from any pair of source and destination ASes, (b) a large fraction of Internet traffic originates in a few large content provider ASes [27, 26], and (c) the cost of S\*BGP deployment can depend on the size of the ISP’s network. The vast array of parameters and empirical data relevant to such a model (Section 8) mean that our analysis is *not* meant to *predict* exactly how the S\*BGP deployment process will proceed in practice; instead, our goal was to evaluate the efficacy of our S\*BGP deployment strategy.

**Theorems, simulations and examples.** We explore S\*BGP deployment in our model using a combination of theoretical analysis and simulations on empirical AS-level graphs [9, 3] (Sections 5-7). Every example we

present comes directly from these simulations. Instead of artificially reducing algorithmic complexity by sub-sampling [25], we ran our simulations over the full AS graph (Section 4). Thus, our simulations ran in time  $O(N^3)$  with  $N = 36K$ , and we devoted significant effort to developing parallel algorithms that we ran on a 200-node DryadLINQ cluster [45].

### 1.4 Key insights and recommendations.

Our evaluation indicates that our strategy for S\*BGP deployment can drive a transition to S\*BGP (Section 5). While we cannot predict exactly how S\*BGP deployment will progress, a number of important themes emerge:

**1. Market pressure can drive deployment.** We found that when S\*BGP deployment costs are low, the vast majority of ISPs have incentives to deploy S\*BGP in order to differentiate themselves from, or keep up with, their competitors (Section 5). Moreover, our results show this holds even if 96% of routing decisions (across all source-destination AS pairs) are *not* influenced by security concerns (Section 6.6).

**2. Simplex S\*BGP is crucial.** When deployment costs are high, deployment is primarily driven by simplex S\*BGP (Section 6).

**3. Choose a few well-connected early adopters.** The set of early adopters cannot be random; it should include well-connected ASes like the Tier 1’s and content providers (Section 6). While we prove that it is NP-hard to even *approximate* the *optimal* set of early adopters (Section 6.1), our results show that even 5-10 early adopters suffice when deployment costs are low.

**4. Prepare for incentives to disable S\*BGP.** We show that ISPs can have incentives to *disable* S\*BGP (Section 7). Moreover, we prove that there could be deployment oscillations (where ASes endlessly turn S\*BGP on and off), and that it is computationally hard to even *determine* whether such oscillations exist.

**5. Minimize attacks during partial deployment.** Even when S\*BGP deployment progressed, 100% of ASes never become secure (Section 5, 6). As such, we expect that S\*BGP and BGP will coexist in the long term, suggesting that careful engineering is required to ensure that this does not introduce new vulnerabilities into the interdomain routing system.

**Paper organization.** Section 2 presents our proposed strategy for S\*BGP deployment. To evaluate the proposal, we present a model of the deployment process in Section 3. In Section 5-7 we explore this model using theoretical analysis and simulations, and present an in-depth discussion of our modeling assumptions in Section 8. Section 9 presents related work. The appendices contains implementation details for our simulations and proofs of all our theorems.

## 2. S\*BGP DEPLOYMENT STRATEGY

### 2.1 S\*BGP: Two possible solutions.

With RPKI providing an authoritative mapping from ASes to their cryptographic public keys, two main protocols have been proposed that prevent the propagation of bogus AS paths information:

**Secure BGP (S-BGP) [24].** S-BGP provides *path validation*, allowing an AS  $a_1$  that receives a BGP announcement  $a_1a_2\dots a_kd$  to validate that every AS  $a_j$  actually sent the BGP announcement in the path. With S-BGP, a router must cryptographically sign every routing message that it sends, and cryptographically verify every routing message that it receives.

**Secure Origin BGP (soBGP) [43].** soBGP provides a slightly weaker security guarantee called *topology validation*, that allows an AS to validate that a path it learns physically exists in the network. To do this, soBGP requires neighboring ASes to mutually authenticate a certificate for the existence of a link between them, and validate every path it learns from a BGP announcement against these cryptographic certificates.

Because our study is indifferent to attacks and adversaries, it applies equally to each of these protocols. We refer to them collectively as S\*BGP, and an AS that deploys them as *secure*.

### 2.2 How to standardize S\*BGP deployment.

To create local economic incentives for ISPs to deploy S\*BGP, we propose that Internet standards should require ASes to deploy S\*BGP as follows:

#### 2.2.1 Simplex S\*BGP for stubs.

About 85% of ASes in the Internet are *stubs*, ASes with no customers of their own, typically corporations, universities, and small residential providers that are Internet consumers [9]. For stubs, Internet access is a cost, rather than a revenue source, and it seems unlikely that security concerns alone will suffice to motivate stubs to undertake a costly S\*BGP deployment. However, because stubs have no customers of their own, they propagate only *outgoing* BGP announcements for *their own IP prefixes*. Thus, the Internet standards community has observed that stubs could deploy S\*BGP in a unidirectional (simplex) manner:

**Simplex S-BGP.** For S-BGP, this means that stubs need only sign outgoing BGP announcements for their own IP prefixes, but not validate incoming BGP announcements for other IP prefixes<sup>1</sup>. Thus, a stub need only store its own public key (rather than obtaining the

<sup>1</sup>A stub may even choose to delegate its cryptographic keys to its ISPs, and have them sign for him; while this might be a good first step on the path to deployment, ceding control of cryptographic keys comes at the cost of reduced security.

public keys of each AS on the Internet from the RPKI) and cryptographically sign only a tiny fraction of the BGP announcements it sees. Simplex S-BGP can significantly decrease the computational load on the stub, and can potentially be deployed as a software, rather than hardware, upgrade to its routers.

**Simplex soBGP.** For soBGP, this means that a stub need only create certificates for its links, but need not need validate the routing announcements it sees. Simplex soBGP is done offline; once a stub certifies his information in the soBGP database, its task is complete and no router upgrade is required.

The objective of simplex S\*BGP is to make it easy for stubs to become secure by lowering deployment costs and computational overhead. While we certainly allows for stubs (*e.g.*, banks, universities) with an interest in security to move from simplex S\*BGP to full S\*BGP, our proposal does not require them to do so.

**Impact on security.** With simplex S\*BGP, a stub lacks the ability to validate paths for prefixes other than its own. Since stubs constitute about 85% of ASes [9], a first glance suggests that simplex S\*BGP leads to significantly worse security in the global Internet.

We argue that this is not so. Observe that if a stub  $s$  has an immediate provider  $p$  that has deployed S\*BGP and is correctly validating paths, then no false announcements of fully secure paths can reach  $s$  from that provider, unless  $p$  *himself* maliciously (or mistakenly) announces false secure paths to  $s$ . Thus, in the event that stubs upgrade to simplex S\*BGP and all other ASes upgrade to full S\*BGP, the only open attack vector is for ISPs to announce false paths to their *own* stub customers. However, we observe the impact of a single misbehaving ISP is small, since 80% of ISPs have less than 7 stub customers, and only about 1% of ISPs have more than 100 stub customers [9]. Compare this to the insecure status quo, where an arbitrary misbehaving AS can impact about half of the ASes in the Internet (around 15K ASes) on average [15].

#### 2.2.2 Break ties in favor of fully secure paths.

In BGP, an AS chooses the path to a given destination AS  $d$  based on a *ranking* on the outgoing paths it learns from its neighbors (*e.g.*, Appendix A). Paths are first ranked according to interdomain considerations (local preference, AS path length). Ties are then broken according to intradomain considerations (geographic considerations, router ID, *etc.*).

**Secure paths.** We say that a path is secure iff *every* AS on that path is secure. We do this because an AS cannot validate a path unless every AS on the path signed the routing announcement (S-BGP) or issued certificates for the links on the path (soBGP).

**Security as part of route selection.** The next

part of our proposal suggests that once an AS has the ability to validate paths, it should actually use this information to inform its routing decisions. In principle, an AS might even modify its ranking on outgoing paths so that security is its highest priority. Fortunately, we need not go to such lengths. Instead, we only require secure ASes to *break ties* between equally good interdomain paths in favor of secure paths. This empowers secure ISPs to attract customer traffic away from their insecure competitors. To ensure that a newly-secure AS can *regain* lost customer traffic, we require that original tie-break criteria (*e.g.*, intradomain considerations) be employed in the case of equally good, *secure* interdomain paths. Thus, the size of the set of equally-good interdomain paths for a given source-destination pair (which we call the *tiebreak set*) gives a measure of available competition in the AS graph.

**Route selection at stubs.** For stubs running simplex S\*BGP, we consider both the case where they break ties in favor of secure paths (*i.e.*, because they trust their providers to verify paths for them) and the case where they ignore security altogether (*i.e.*, because they do not verify paths) (Section 6.7).

**Partially secure paths.** We do not allow ASes to prefer partially-secure paths over insecure paths, to avoid introducing *new* attack vectors that do exist even without S\*BGP (*e.g.*, attack in Appendix B).

We shall show that S\*BGP deployment progresses quite effectively even if stubs ignore security and tiebreak sets are very small (Section 6.7-6.6).

### 2.3 How third parties should drive deployment.

**Early adopters.** To kick off the process, we suggest that interested third parties (*e.g.*, governments, regulators, industry groups) focus regulation, subsidies, or external financial incentives on convincing a set of *early adopter* ASes to deploy S\*BGP. Once these ASes become secure, other ISPs should deploy S\*BGP as a result of market pressure, *i.e.*, to attract customer traffic.

**ISPs upgrade their stubs.** Next, we suggest that a secure ISP should be responsible for upgrading all its insecure stub customers to simplex S\*BGP. To achieve this, interested third parties should ensure that simplex S\*BGP is engineered to be as lightweight as possible, and potentially provide additional subsidies for ISPs that secure their stubs. (ISPs also have a local incentives to secure stubs, *i.e.*, to transit more revenue-generating traffic for multi-homed stubs (Section 5.1).)

## 3. MODELING S\*BGP DEPLOYMENT

We evaluate our proposal using a model of the S\*BGP deployment process. For brevity, we now present only the details of our model. Justification for our modeling decisions and possible extensions are in Section 8.

### 3.1 The Internet and entities.

**The AS graph.** The interdomain-routing system is modeled with a labeled AS graph  $G(V, E)$ . Each node  $n \in V$  represents an AS, and each edge represents a physical link between ASes. Per Figure 1, edges are annotated with the standard model for business relationships in the Internet [13]: *customer-provider* (where the customer pays the provider), and *peer-to-peer* (where two ASes agree to transit each other’s traffic at no cost). Each AS  $n$  is also assigned weight  $w_n$ , to model the volume of traffic that *originates* at each AS.

We distinguish three types of ASes:

**Content providers.** Content providers (CPs) are ASes whose revenue (*e.g.*, advertising) depends on reliably delivering their content to as many users as possible, rather than on providing Internet transit. While a disproportionately large volume of Internet traffic is known to originate at a few CPs, empirical data about Internet traffic volumes remains notoriously elusive. Thus, based on recent research [27, 26, 40] we picked five content providers: Google (AS 15169), Facebook (AS 32934), Microsoft (AS 8075), Akamai (AS 20940), and Limelight (AS 22822). Then, we assigned each CP weight  $w_{CP}$ , so that the five CPs originate an  $x$  fraction of Internet traffic (equally split between them), with the remaining  $1 - x$  split between the remaining ASes.

**Stubs.** Stubs are ASes that have no customers of their own and are not CPs. Every stub  $s$  has unit weight  $w_s = 1$ . In Figure 1, ASes 34376 and 31420 are stubs.

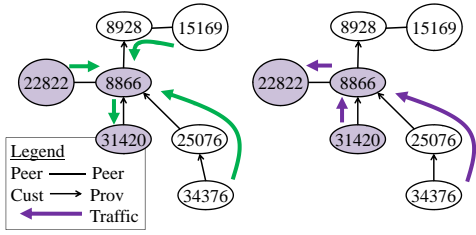
**ISPs.** The remaining ASes in the graph (that are not stubs or CPs) are ISPs. ISPs earn revenue by providing Internet service; because ISPs typically provide transit service, rather than originating traffic (content), we assume they have unit weight  $w_n = 1$ . In Figure 1, ASes 25076, 8866 and 8928 are ISPs.

### 3.2 The deployment process.

We model S\*BGP deployment as an infinite round process. Each round is represented with a state  $S$ , capturing the set of ASes that have deployed S\*BGP.

**Initial state.** Initially, the only ASes that are secure are (1) the ASes in the set of early adopters and (2) the direct customers of the early adopter ISPs that are stubs. (The stubs run simplex S\*BGP.) All other ASes are insecure. For example, in Figure 1, early adopters ISP 8866 and CP 22822 are secure, and stub 31420 runs simplex S\*BGP because its provider is secure.

**Each round.** In each round, *every* ISP chooses an action (deploy S\*BGP or not) that improves its utility relative to the current state. We discuss the myopic best-response strategy that ISPs use to choose their actions in Section 3.3. Once an ISP becomes secure, it deploys simplex S\*BGP at *all* its stub customers (Section 2.3). Because CPs do not earn revenues by pro-



**Figure 1: Destinations (left) 31420, (right) 22822.**

viding Internet service, some external incentive (*e.g.*, concern for security, subsidies) must motivate them to deploy S\*BGP. Thus, in our model, a CP may only deploy S\*BGP if it is in the set of early adopters.

Once ASes choose their actions, paths are established from every source AS  $i$  to every destination AS  $d$ , based on the local BGP *routing policies* of each AS and the state  $S$  of the AS graph. We use a standard model of BGP routing policies, based on business relationships and path length (see Appendix A). Per Section 2.3, we also assume that routing policies of secure ASes require them to break ties by preferring fully secure paths over insecure ones, so that the path to a given destination  $d$  depends on the state  $S$ . Paths to a destination  $d$  form a tree rooted at  $d$ , and we use the notation  $T_n(d, S)$  to represent the subtree of ASes routing through AS  $n$  to a destination  $d$  when the deployment process is in state  $S$ . Figure 1 (right) shows part of the routing tree for destination 22822; notice that  $T_{8866}(22822, S)$  contains ASes 31420, 25076, 34376.

**Termination.** We proceed until we reach a stable state, where no ISP wants to deploy (or disable) S\*BGP.

### 3.3 ISP utility and best response.

We model an ISP’s utility as related to the volume of traffic it transits for its customers; this captures the fact that many ISPs either bill their customers directly by volume, or indirectly through flat rates for fixed traffic capacities. Utility is a function of the paths chosen by each AS. Because path selection is a function of routing policies (Appendix A) and the state  $S$ , it follows that *the utility of each ISP is completely determined by the AS weights, AS graph topology, and the state  $S$ .*

We have two models of ISP utility that capture the ways in which an ISP can transit customer traffic:

**Outgoing utility.** ISP  $n$  can increase its utility by forwarding traffic *to* its customers. Thus, we define outgoing utility as the amount of traffic that ISP  $n$  routes to each destination  $d$  via a customer edge. Letting  $\hat{D}(n)$  be the set of such destinations, we have:

$$u_n(S) = \sum_{d \in \hat{D}(n)} \sum_{i \in T_n(d, S)} w_i \quad (1)$$

Let’s use Figure 1 to find the incoming utility of ISP  $n = 8866$  due to destinations 31420 and 22822. Destination 31420 is in  $\hat{D}(n)$  but destination 22822 is not. Thus, two CPs (Google AS 15169 and Limelight 22822), and 3 other ASes (*i.e.*, AS 8928, 25076, 34376) transit traffic through  $n = 8866$  to destination  $d = 31420$ , contributing a  $2w_{CP} + 3$  outgoing utility to  $n = 8866$ .

**Incoming utility.** An ISP  $n$  can increase its utility by forwarding traffic *from* its customers. Thus, we define incoming utility as the amount of traffic that ISP  $n$  receives via customer edges for each destination  $d$ . Thus, restrict the subtree  $T_n(d, S)$  to branches that are incident on  $n$  via customer edges to obtain the *customer subtree*  $\hat{T}_n(d, S) \subset T_n(d, S)$ , we have:

$$u_n(S) = \sum_d \sum_{i \in \hat{T}_n(d, S)} w_i \quad (2)$$

Let’s compute outgoing utility of  $n = 8866$  due to destinations 31420 and 22822 in Figure 1. For destination 31420, ASes 25076 and 34376 are part of the customer subtree  $\hat{T}_n(d, S)$ , but 15169, 8928 and 22822 are not. For destination  $d = 22822$ , ASes 31420, 25076, 34376 are part of the customer subtree. Thus, these ASes contribute  $2 + 3$  incoming utility to ISP  $n = 8866$ .

Realistically, ISP utility is some function of both of these models; to avoid introducing extra parameters into our model, we consider each separately.

**Myopic best response.** We use a standard game-theoretic update rule known as *myopic best response*, that produces the most favorable outcome for a node in the next round, taking other nodes’ strategies as given [17]. Let  $(\neg S_n, S_{-n})$  denote the state when  $n$  ‘flips’ to the opposite action that it used in state  $S$ , while other ASes maintain the same action they use in state  $S$ . ISP  $n$  changes its action in state  $S$  iff its *projected utility*  $u_n(\neg S_n, S_{-n})$  is sufficiently high, *i.e.*,

$$u_n(\neg S_n, S_{-n}) > (1 + \theta) \cdot u_n(S) \quad (3)$$

where  $\theta$  is a threshold denoting the increase in utility an ISP needs to see before it is willing to change its actions. Threshold  $\theta$  captures the cost of deploying BGP security; *e.g.*, an ISP might deploy S\*BGP in a given round if S\*BGP deployment costs do not exceed  $\theta = 5\%$  of the profit it earns from transiting customer traffic. Since  $\theta$  is multiplicative, it captures the idea that deployment costs are likely to be higher at ISPs that transit more traffic. The update rule is myopic, because it focuses on increasing ISP  $n$ ’s utility in the next round only. It is best-response because it does *not* require ISP  $n$  to speculate on other ASes’ actions in future rounds; instead,  $n$  takes these actions as given by the current state  $S$ .

**Discussion.** Our update rule requires ASes to predict their future utility. In our model, ASes have full infor-

mation of  $S$  and  $G$ , a common approach in game theory, which enables them to project their utility accurately. We discuss the consequences of our update rule, and the impact of partial information in Sections 8.1-8.2.

#### 4. SIMULATION FRAMEWORK

Computing utility  $u_n(S)$  and projected utility  $u_n(\neg S_n, S_{-n})$  requires us to determine the path from *every* source AS to *every* destination AS, for *every* ISP  $n$ 's unique projected state  $(\neg S_n, S_{-n})$ . Thus, our simulations had complexity  $O(|V|^3)$  on an AS graph  $G(V, E)$ . To accurately simulate our model, we chose *not* to ‘sample down’ the complexity of our simulations:

**Projecting utility for each ISP.** If we had computed the utility for only a few sampled ISPs, this would reduce the number of available secure paths and artificially prevent S\*BGP deployment from progressing.

**Simulations over the entire AS graph.** Our proposal is specifically designed to leverage the extreme skew in AS connectivity (*i.e.*, many stubs with no customers, few Tier 1s with many customers), to drive S\*BGP deployment. To faithfully capture the impact of this skew, we computed utility over traffic from *all* sources to *all* destination ASes. Furthermore, we ran our simulations on the full empirical AS graph [9], rather than a subsampled version [25], or a smaller synthetic topology (*e.g.*, BRITTE [30], GT-ITM [46]), as in prior work [8, 5]. We used the Cyclops AS graph from Dec 9, 2010 [9], augmented with an additional 16K peering edges discovered at Internet exchange points (IXPs) [3], as well as an additional augmented AS graph described in Section 6.8. (See also Appendix D.)

The AS graph  $G(V, E)$  had  $|V| = 36K$ ; to run  $O(|V|^3)$ -simulations at such a scale, we parallelized our algorithms on a 200-node DryadLINQ cluster [45] that could run through a single simulation in 1-12 hours. (Details of our implementation are in Appendix C.)

#### 5. CASE STUDY: S\*BGP DEPLOYMENT

We start by showing that even a small set of early adopters can create enough market pressure to transition the vast majority of ASes to S\*BGP.

**Case study overview.** We focus on a single simulation where the early adopters are the five CPs (Google, Facebook, Microsoft, Limelight, Akamai, see Section 3.1), and the top five Tier 1 ASes in terms of degree (Sprint (1239), Verizon (701), AT&T (7018), Level 3 (3356), Cogent (174)). Every ISP uses an update rule with a relatively low threshold  $\theta = 5\%$ , that the five CPs originate  $x = 10\%$  of the traffic in the Internet, and that stubs *do* break ties in favor of secure routes. We now show how even a small set of ten early adopters (accounting for less than 0.03% of the AS

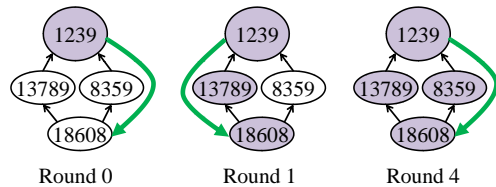


Figure 2: A Diamond: ISPs 13789 and 8359 compete for traffic from Tier 1 AS 1239.

Table 1: Occurrences of the Diamond scenario for early adopter ASes (sorted by degree).

Tier 1s	AS 174	878	CPs	AS 22822	175
	AS 3356	1,400		AS 15169	892
	AS 7018	340		AS 20940	178
	AS 701	706		AS 8075	1,149
	AS 1239	728		AS 32934	82

graph) can convince 85% of ASes to deploy S\*BGP, and secure 65% of all paths in the AS graph.

#### 5.1 Competition drives deployment.

We start by zooming in on S\*BGP deployment at two competing ISPs, in a scenario we call a DIAMOND.

**Figure 2:** Two ISPs, AS 8359 and AS 13789, compete for traffic from Sprint (AS 1239) to their stub customer, AS 18608. Sprint is an early adopter of S\*BGP, and initially the three other ASes are insecure. Both ISPs offer Sprint equally good two-hop customer paths to the stub, and AS 8359 is chosen to carry traffic by winning the tie break. In the first round, AS 13789 computes its projected utility, and realizes it can gain Sprint’s traffic by adopting S\*BGP and upgrading its stub to simplex S\*BGP. (See Section 8.2 for more discussion on how ISPs compute projected utility.) By the fourth round, AS 8359 has lost so much utility (due to traffic lost to ASes like 13789) that he decides to deploy S\*BGP.

Of course, Figure 2 is only a very small snapshot of the competition for traffic destined to a single stub AS 18608; utility for each ISPs is based on customer traffic transited to *all* destinations in the AS graph. Indeed, this DIAMOND scenario is quite common. In Table 1, we summarize the number of diamonds we counted, each involving at two ISPs, a stub and one of the early adopters.

#### 5.2 Global deployment dynamics.

Next, we look at deployment globally:

**Figure 3:** We show the number of ASes (*i.e.*, stubs, ISPs and CPs) and the number of ISPs that deploy S\*BGP at each round of the simulation. In the first round, 548 ISPs become secure; because each of these ISPs deploy simplex S\*BGP in their stubs, we see that over 5K ASes become secure by the end of the first round. In subsequent rounds, hundreds of ISPs deploy S\*BGP in each round; however, the number of newly

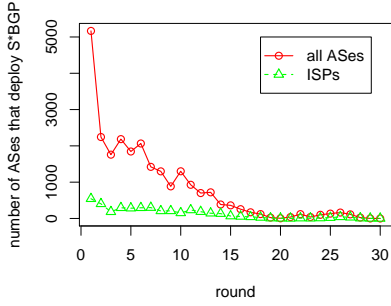


Figure 3: The number of ASes that deploy S\*BGP each round.

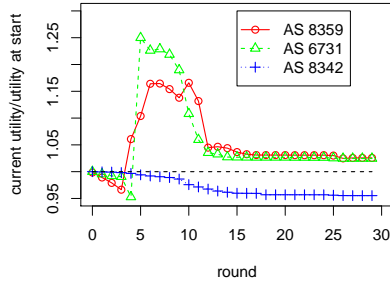


Figure 4: Normalized utility of ISPs in Fig. 2 and 7.

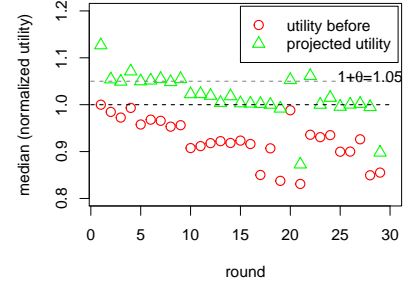


Figure 5: Projected and actual utility before deploying S\*BGP normalized by starting utility.

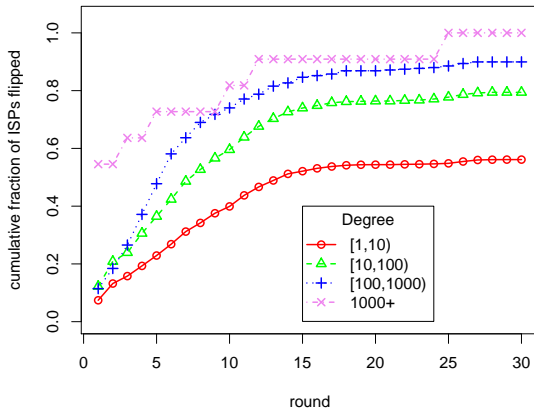


Figure 6: Cumulative fraction of ISPs that deploy S\*BGP by degree.

secure stubs drops dramatically, suggesting that many ISPs deploy S\*BGP to regain traffic lost when their stubs were secured by competitors. After the 17th iteration, the process tapers off, with fewer than 50 ASes becoming secure in each round. The final surge in deployment occurs in round 25, when a large AS, 6939, suddenly became secure, causing a total of 436 ASes to deploy S\*BGP in the remaining six rounds. When the process terminates, 85% of ASes are secure, including 80% of the 6K ISPs in the AS graph.

### 5.3 Impact of ISP degree on deployment.

The reader might be surprised to find that ISPs with high degree are more likely to deploy S\*BGP:

**Figure 6:** We consider the cumulative fraction of ISPs adopting S\*BGP in each round, separated by degree. Interestingly, ISPs with low degree ( $\leq 10$ ) are less likely to become secure. Indeed, we found a consistent set of about 1000 ISPs that *never* deploy S\*BGP in *any* of our simulations (not shown). These ISPs had average degree 6, and remained insecure because they never had to compete for customer traffic; indeed, they were usually providers to only single-homed stub customers.

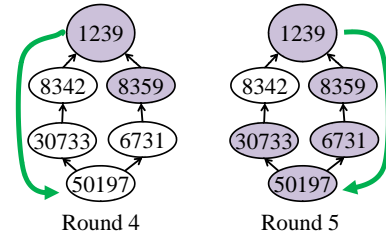


Figure 7: A newly created four-hop secure path.

### 5.4 Longer secure paths sustain deployment.

In Figure 3 we observed rapid, sustained deployment of S\*BGP in the first 17 iterations. This happens because longer secure paths are created as more ASes deploy S\*BGP, thus creating incentives for S\*BGP at ASes that are far away from the early adopters:

**Figure 7:** We once again encounter AS 8359 from Figure 2. We show how AS 8359's decision to deploy S\*BGP in round 4 allows a new ISP (AS 6731) to compete for traffic. In round 5 AS 6731 sees a large increase in utility by becoming secure. This occurs, in part, because AS 6731 can now entice six of the early adopters to route through him on a total of 69 newly-secure paths. Indeed, when AS 6731 becomes secure, he continues the chain reaction set in motion by AS 8359; for instance, in round 7 (not shown), AS 6731's neighbor AS 41209 becomes secure in order to offer Sprint a new, secure four-hop path to one of 41209's own stubs.

### 5.5 Keeping up with the competition.

Two behaviors drive S\*BGP deployment in a DIAMOND. First, an ISP becomes secure to steal traffic from a competitor, and then the competitor becomes secure in order to regain the lost traffic. We can watch this happening for the ISPs from Figure 2 and 7:

**Figure 4:** We show the utilities of ISPs 8359, 6731, and 8342 in each round, normalized by *starting utility*.

ity *i.e.*, the utility before the deployment process began (when all ASes, including the early adopters, were still insecure). As we saw in Figure 2, AS 8359 deploys S\*BGP in round 4 in order to regain traffic he lost to his secure competitors; here we see that in round 4, AS 8359 has lost 3% of his starting utility. Once AS 8359 deploys S\*BGP, his utility jumps up to more than 125% of his starting utility, but these gains in utility are only temporary, disappearing around round 15. The same is true in round 6 for AS 6371 from Figure 7. By round 15, 60% ISPs in the AS graph are already secure (Figure 3), and our ISPs can no longer use security to differentiate themselves, causing their utility to return to within 3% of their starting utility.

This is also true more generally:

**Figure 5:** For each round  $i$ , we show the median utility and median projected utility for ISPs that become secure in round  $i+1$ , each normalized by starting utility. (Recall from (3) that these ISPs have projected utility at least  $1+\theta$  times their utility in round  $i$ .) In the first 9 rounds, ISPs mainly deploy S\*BGP to steal traffic from competitors; that is, their projected utility in the round before they deploy S\*BGP is at least  $1+\theta = 105\%$  times their starting utility. However, as deployment progresses, ASes increasingly deploy S\*BGP in order to recover lost traffic and return to their starting utility; that is, in rounds 10-20 ISP utility drops to at least  $\theta = 5\%$  less than starting utility, while projected utility approaches starting utility ( $y=1$ ).

## 5.6 Is S\*BGP deployment a zero-sum game?

Our model of S\*BGP deployment is indeed a zero-sum game; we assume that ISPs compete over a fixed set of customer traffic. Thus, when the vast majority of ASes have deployed S\*BGP, ISPs can no longer use security to distinguish themselves from competitors (Figure 4). At the termination of this case study, only 8% of ISPs have an increase in utility of more than  $\theta = 5\%$  over their starting utility. On the other hand, 85% of ASes now benefit from a (mostly) secure Internet. Furthermore, like ASes 8359 and 6731 in Figure 4, many of these secure ASes enjoyed a prolonged period of increased utility that could potentially help defray the costs of deploying S\*BGP.

**It is better to deploy S\*BGP.** One might argue that a cynical ISP might preempt the process by *never* deploying S\*BGP. However, a closer look shows that its almost always in the ISPs interest to deploy S\*BGP. ISPs that deploy S\*BGP usually return to their starting utility or slightly above, whereas ISPs that do *not* deploy S\*BGP lose traffic in the long term. For instance, AS 8342 in Figure 7 never deploys S\*BGP. As shown in Figure 4, when the deployment process terminates, AS 8342 has lost 4% of its starting utility. Indeed, another look at the data (not shown) shows that the ISPs

that remain insecure when the process terminates lose on average 13% of their starting utility!

## 6. CHOOSING EARLY ADOPTERS

Next, we consider choosing the set of ASes that should be targeted to become early adopters of S\*BGP.

### 6.1 It's hard to choose early adopters.

Ideally, we would like to choose the *optimal* set of early adopters that could cause the maximum number of other ASes to deploy S\*BGP. We show that this is NP-hard, by presenting a reduction to the 'set cover' problem (proof in Appendix E):

**THEOREM 6.1.** *For an AS graph  $G(V, E)$  and a parameter  $1 \leq k \leq |V|$ , finding a set of early adopter ASes of size  $k$  that maximizes the number of ASes that are secure when the deployment process terminates is NP-hard. Approximating the solution within a constant factor is also NP-hard.*

As such, we use simulations of the deployment process to investigate heuristic approaches for choosing early adopters, including AS degree (*e.g.*, Tier 1s should be early adopters) and volume of traffic originated by an AS (*e.g.*, content providers should be early adopters).

### 6.2 The parameter space.

We consider how the choice of early adopters is impacted by assumptions on (1) whether or not stubs running simplex S\*BGP break ties based on security, (2) the AS graph, and (3) traffic volumes sourced by CPs.

**Outgoing utility.** Also, recall that we have two models of ISP utility (Section 3.3). In this section, we dive into the details of the *outgoing utility* model because it has the following very nice property:

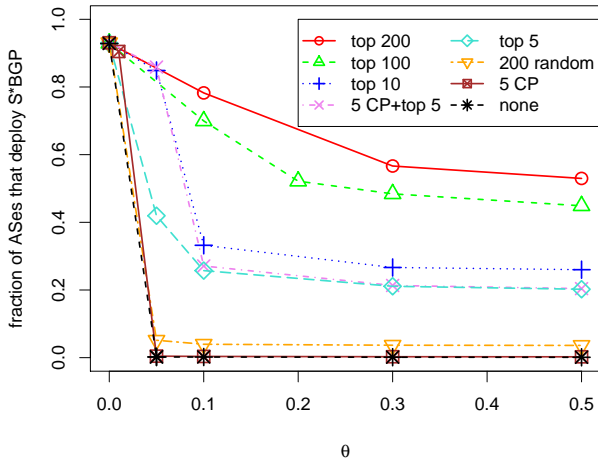
**THEOREM 6.2.** *In the outgoing utility model, a secure node will never have an incentive to turn off S\*BGP.*

As a consequence of this theorem (proof in Appendix H), it immediately follows that (a) every simulation must terminate, and (b) we can significantly reduce compute time by *not* computing projected utility for ISPs that are already secure. (We discuss complications that arise from the incoming utility model in Section 7.)

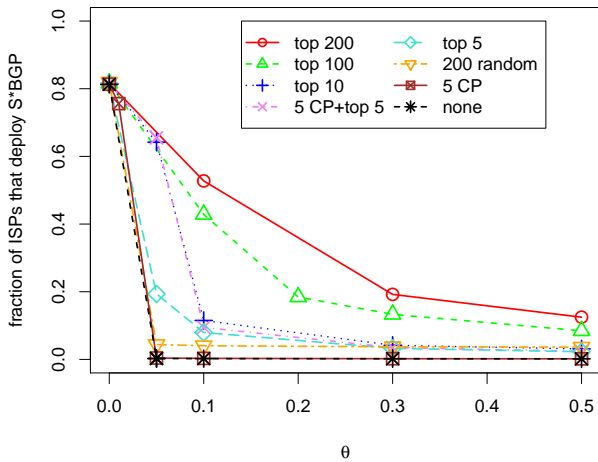
**Deployment threshold  $\theta$ .** Our update rule (3) is such that ISPs change their actions if they can increase utility by at least  $\theta$ . Thus, to gain insight into how 'difficult' it is to convince ISPs to deploy S\*BGP, we assume that each ISP uses the same threshold  $\theta$ , and sweep through different values of  $\theta$  (but see also Section 8.2).

### 6.3 Comparing sets of early adopters.

We next explore the influence of different early adopters:



(a) ASes



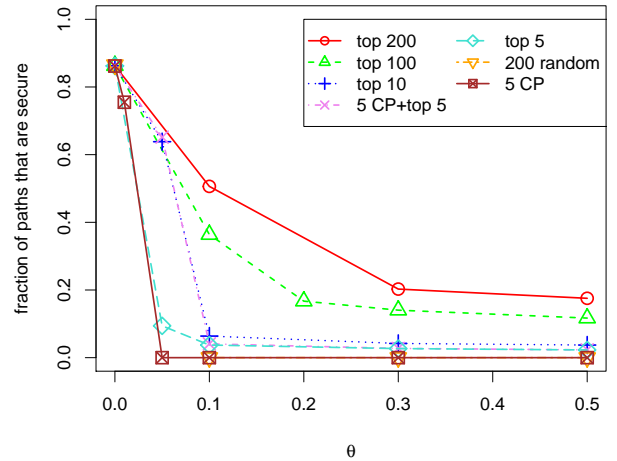
(b) ISPs

**Figure 8: Fraction of ASes (a) and ISPs (b) that deploy S\*BGP for varying  $\theta$  and early adopters.**

**Figure 8 (a):** We show the fraction of ASes that adopt S\*BGP for different values of  $\theta$ . We consider no early adopters, the top 5-200 ISPs in terms of degree, the five CPs, five CPs in combination with the top five ISPs, and 200 random ISPs.

**There are incentives to deploy S\*BGP.** For low values of  $\theta < 5\%$ , we observe that there is sufficient competition over customer traffic to transition 85% of ASes to S\*BGP. Moreover, this holds for almost every set of early adopters we considered. (Note that in the unrealistic case where  $\theta = 0$ , we see widespread S\*BGP deployment even with *no* early adopters, because we assume the stubs break ties in favor of secure paths. But see also Section 6.7.) Furthermore, we find that the five CPs have approximately the same amount of influence as the case where there are no early adopters; we investigate this in more detail in Section 6.8.

**Some ISPs always remain insecure.** We find



**Figure 9: Fraction of paths on the Internet that are secure.**

20% of the 6K ISPs in the AS graph [9, 3] never deploy S\*BGP, because they are never subject to competition for customer traffic. This highlights two important issues: (1) some ISPs may never become secure (*e.g.*, ASes whose customers are exclusively single-homed) (2) S\*BGP and BGP will coexist in the long term.

**Choice of early adopters is critical.** For higher values of  $\theta \geq 10\%$ , it becomes important to choose ISPs with high customer degree as early adopters. In fact, Figure 8 shows a set of 200 random ASes has significantly lower influence than a set containing only the five top ASes in terms of degree. For large values of  $\theta \geq 30\%$ , a larger set of high-degree early adopters is required, with the top 200 ASes in terms of degree causing 53% of the ASes to deploy S\*BGP for  $\theta = 50\%$ . However, to put this observation in some perspective, recall that  $\theta = 30\%$  suggests that the cost of S\*BGP deployment exceeds 30% of an ISP's profit margin from transiting customer traffic.

## 6.4 How much security do we get?

We count the number of secure paths at the termination of the deployment process, as a measure of the efficacy of the S\*BGP deployment process. (Of course, this is *not* a perfect measure of the AS graph's resiliency to attack; quantifying this requires approaches similar to [15, 8], an important direction for future work.)

**Figure 9:** We show the fraction of the  $(36K)^2$  paths between ASes that are secure, for the different sets of early adopters. As expected, we find that the fraction of secure path is only slightly lower than  $f^2$ , where  $f$  is the fraction of ASes that have deployed S\*BGP. (The  $f^2$  follows from the fact that for a path to be secure, both its source AS and its destination AS must be secure.) Indeed, the fact the number of secure paths is only 4% lower than  $f^2$  suggests that the majority of secure paths are likely to quite short.

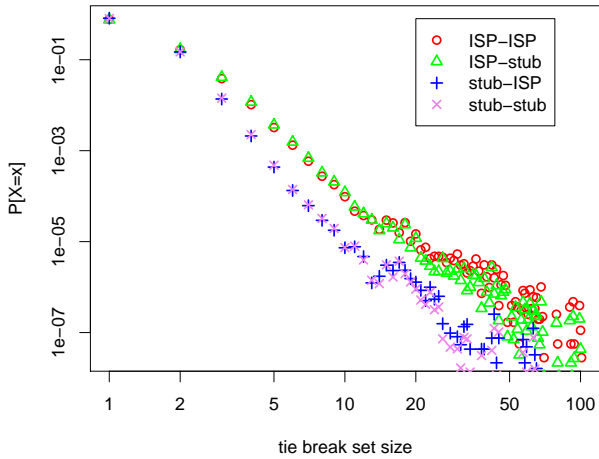


Figure 10: Probability density function of tie break set size in the AS graph [9, 3] for different source-destination pairs (log-log scale).

## 6.5 Market pressure vs. simple S\*BGP

The cause of for global S\*BGP deployment differs for low and high values of the deployment threshold  $\theta$ :

**Figure 8 (b):** We show the fraction of ISPs (not ASes) that deploy S\*BGP for the early adopter sets and varying values of  $\theta$ . For low values of  $\theta$ , market pressure drives a large fraction of ISPs to deploy S\*BGP. In contrast, for higher values of  $\theta$  very few ISPs deploy S\*BGP, even for large sets of well-connected early adopters. In these cases, most of the deployment is driven by ISPs upgrading their stub customers to simple S\*BGP. For example, for the top 200 ISPs, when  $\theta = 50\%$ , only a small fraction of secure ASes (4%) deploy S\*BGP because of market pressure, the vast majority (96%) are stubs running simple S\*BGP.

## 6.6 The source of competition: tie break sets.

Recall that the *tiebreak set* is the set of paths from which an source AS employs the security criterion to select paths to a destination AS (Section 2.2.2). A tiebreak set with multiple paths presents opportunities for ISPs to compete over traffic from the source AS.

**Figure 10:** We show distribution of tiebreak set size for all source-destination pairs of ASes. (This result holds for the AS graph [9, 3] under the assumption that ASes use the routing policies of Appendix A.) Noting that this graph has a log-log scale, observe that tiebreak sets are typically very small. ISPs have slightly larger tiebreak sets than stubs: an average of 1.30 for ISPs and 1.16 for stubs. Moreover, only 20% tiebreak sets contain more than a single path. This striking observation suggests that even a very limited amount of competition suffices to drive S\*BGP deployment for low  $\theta$ . Furthermore, we speculate that this might also explain why there is limited market pressure for S\*BGP deployment

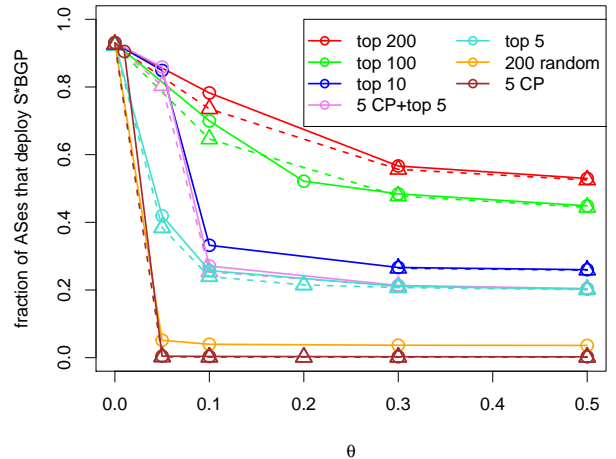


Figure 11: Fraction of ASes that deploy S\*BGP for different early adopters (dashed lines stubs do not prefer secure paths).

at ISPs when  $\theta > 10\%$ .

## 6.7 Stubs don't need to break ties on security.

So far, we have focused on the case where secure stubs break ties in favor of secure paths. Indeed, given that stubs typically make up the majority of secure ASes, one might expect that their routing decisions can have a major impact of the success of the S\*BGP deployment process. Surprisingly, we find that this is not the case. Indeed, our results are insensitive to this assumption, for  $\theta > 0$  and regardless of the choice of early adopters (Figure 11). We explain this by observing that stubs both (a) have small tiebreak sets, and (b) transit no traffic.

**Security need only effect a fraction of routing decisions!** Thus, only 15% of ASes (*i.e.*, the ISPs) need to break ties in favor of secure routes, and only 23% of ISP tiebreak sets contain more than one path. Combining these observations, we find that S\*BGP deployment can progress even if only  $0.15 \times 0.23 = 3.5\%$  of routing decisions are effected by security considerations!

## 6.8 Content providers vs. Tier 1s.

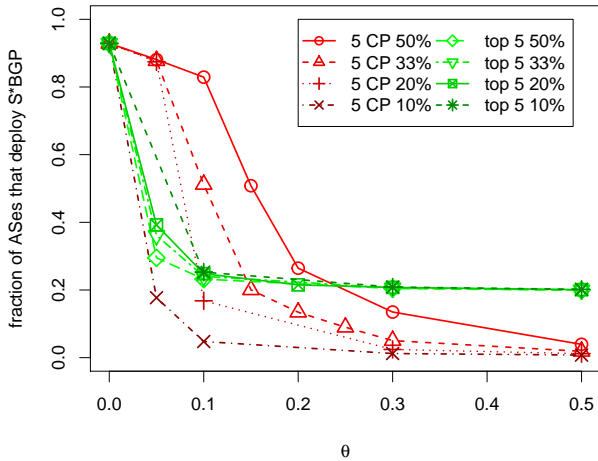
We compare the Tier 1s and CPs as early adopters.

### 6.8.1 Varying parameters.

To do this effectively, we varied a few parameters:

**1. Traffic volumes.** We swept through different values  $x = \{10\%, 20\%, 33\%, 50\%\}$  for the fraction of traffic originated by the five CPs (Section 3.1); recent work suggests a reasonable range is  $x = 10\text{-}20\%$  [27, 26].

**2. Connectivity of content providers.** Published AS-level topologies are known to have poor visibility into peering links at the edge of the AS-level topology [34]. This is particularly problematic for CPs,



**Figure 12: Fraction of ASes that deploy S\*BGP for the five content providers and five Tier 1s in the augmented topology.**

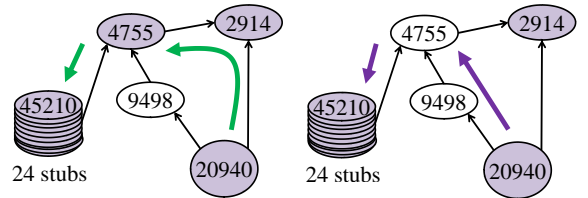
because they peer with many other ASes to cut down costs of delivering content [14]. Indeed, while the CPs known to have short path lengths [35], their average path length in our AS graph (with routing policies as in Appendix A) was 2.7 hops or more. Thus, for sensitivity analysis, we created an augmented AS graph with an additional 19.7K peering edges from the five CPs to 80% of ASes found to be present at IXPs [3]. In our new ‘augmented AS graph’, (described in Appendix D) the average path length of the CPs dropped to about 2, and their degree increased to be as high as the largest Tier 1 ISPs.

### 6.8.2 Impact of connectivity and traffic volumes.

**Figure 12:** We sweep through values of  $\theta$  to compare the five CPs and top five Tier 1s as early adopters for (a) different traffic volumes, (b) in our augmented AS graph vs. the original graph. We find the following:

- Traffic volumes vs. degree.* We were initially surprised to find that when the five CPs source  $x = 10\%$  of traffic, they are much less effective as early adopters than the top five Tier 1 ASes. Even though the Tier 1s and CPs have about equal degree, it turns out the dominant factor here is traffic volume; even though the CPs *originate* the majority ( $x = 10\%$ ) of traffic, the Tier 1s still *transit* 2-9X times more traffic. As  $x$  increases to 50%, the Tier 1s only transit 0.3-1.2X more traffic than is originated by the CPs. Thus, the CPs tend to have more influence for lower values of  $\theta \leq 10\%$ .

- Tier 1s deploy simplex S\*BGP at more stubs.* Figure 12 indicates the five Tier 1 consistently outperform the CPs as early adopters when  $\theta \geq 0.3$ . The explanation for this is simple; Tier 1s have a large number of stub customers that they immediately upgrade to simplex S\*BGP. This suggests that having CPs to up-



**Figure 13: AS 4755 incentives turn off S\*BGP.**

grade their *stub peers* to simplex S\*BGP could potentially drive S\*BGP deployment further.

## 6.9 Summary and recommendations.

We make two key observations regarding selection of early adopters. First, only a small number of ISPs suffice as early adopters when deployment thresholds  $\theta$  are small. Second, to withstand high  $\theta$ , Tier 1 ASes should be targeted. This is due to the high volumes of traffic they transit and the many stubs they upgrade to simplex S\*BGP. Finally, we note that our results hold even if more than 96% of routing decisions are insensitive to security considerations!

## 7. OTHER COMPLICATIONS

Intuition suggests that a secure ISP will observe increased utility because secure ASes transit traffic through it. While this is true in the *outgoing utility* model (Theorem 6.2), it turns out that this is *not* the case for the *incoming utility* model. We now discuss complications that might arise because we require S\*BGP to play a role in route selection.

### 7.1 Buyer’s Remorse: Turning off S\*BGP.

We present an example of a severe obstacle to S\*BGP deployment: an secure ISP that has incentive to turn *off* S\*BGP. The idea here is that when an ISP  $n$  becomes secure, some of  $n$ ’s incoming traffic might change its path, and enter  $n$ ’s network along peer/provider edges instead of customer edges, thus reducing  $n$ ’s utility.

**Figure 13:** We show that AS 4755, a Telecom provider in India, has an incentive to turn off S\*BGP in its network. We assume content providers have  $w_{CP} = 821$  which corresponds to 10% of Internet traffic originating at the big five CPs (including Akamai’s AS 20940).

In the state  $S$  on the left, Akamai, AS 4755, and NTT (AS 2914) are secure, the stub customers of these two secure ISPs run simplex S\*BGP, and all other ASes are insecure. Here, AS 4755 transits traffic sourced by Akamai from his provider NTT AS 2914, to a collection of twenty-four of its stub customers (including AS 45210). Akamai’s traffic does *not* increase AS 4755’s utility because it arrives at AS 4755 along a provider edge.

In the state  $(\neg S_{4755}, S_{\neg 4755})$  on the right, AS 4755 turns S\*BGP off. If we assume that stubs running simplex S\*BGP do *not* break ties based on security, then the only ASes that could potentially change their routes

are the secure ASes 20940 and 2914. Notice that when AS 4755 turns S\*BGP off, Akamai’s AS 20940 has no secure route to AS 4755’s stub customers (including AS 45210). As such, Akamai will run his usual tie break algorithms, which in our simulation came up in favor of AS 9498, a customer of AS 4755. Because Akamai’s traffic is now enters AS 4755 on customer edges, AS 4755’s incoming utility *increases* by a factor of 205% per each of the 24 stub destinations.

**Turning off the entire network.** Our simulations confirmed that, apart from Akamai changing the path it uses to these twenty-four stubs, all other ASes use the same routes in state  $S$  and state  $(\neg S_{4755}, S_{-4755})$ . This means that AS 4755 has an incentive to turn off S\*BGP in his *entire network*; no routes other than those ones Akamai uses to reach the twenty-four stubs are impacted by his decision. Indeed, we found that the utility of AS 4755 increase by a total of 0.5% (over all destinations) when he turns off S\*BGP!

**Turning off a destination.** AS 4775 could just as well turn off S\*BGP on a *per destination* basis, *i.e.*, by refusing to propagate S\*BGP announcements for the twenty-four stubs in Figure 13, and sending insecure BGP messages for these destinations instead.

## 7.2 Turning off S\*BGP can cause oscillations.

To underscore the seriousness of an ISP turning off S\*BGP in his entire network, we now argue that a group of ISPs could *oscillate*, alternating between turning S\*BGP on and off, and never arriving at a stable state. In Appendix F, we exhibit an example AS graph and state  $S$  that proves that oscillations could exist. Worse yet, we show that it is hard to even *determine* whether or not the deployment process will oscillate!

**THEOREM 7.1.** *Given an AS graph and state  $S$ , it is PSPACE-complete to decide if the deployment process will terminate at a stable state in the incoming utility model.*

Our proof, in Appendix K is by reduction to the PSPACE-complete problem of determining whether a space-bounded Turing Machine will halt for a given input string. The complexity class PSPACE consists of all decisions problems that can be solved using only polynomial *space*, but in unbounded *time*. PSPACE-complete problems (intuitively, the hardest problems in PSPACE) are at least as hard as the NP-complete problems, and widely believed to be even harder.

## 7.3 How common are these examples?

At this point, the reader may be wondering how often an AS might have incentives to turn off S\*BGP.

**Turning off an entire network?** Figure 13 proves that cases where an ISP has an incentive to turn off

S\*BGP in its *entire network* do exist in realistic AS-level topologies [9]. However, we speculate that such examples will occur infrequently in practice. While we cannot provide any concrete evidence of this, our speculation follows from the fact that an ISP  $n$  obtains utility from many destinations. Thus, even if  $n$  has increased its utility by turning OFF S\*BGP for destinations that are part of subgraphs like Figure 13, he will usually obtain higher utility by turning ON S\*BGP for the other destinations that are not part of such subgraphs. (In Figure 13, this does not happen because the state  $S$  is such that only a very small group of ASes are secure; thus, no routes other than the ones pictured are effected by AS 4755’s decision to turn off S\*BGP.)

**Turning off a destination is likely.** On the other hand, it is quite easy to find examples of *specific destinations* for which an ISP might want to turn off S\*BGP. Indeed, a search through the empirical AS graph found that at least 10% of the 5,992 ISPs could find themselves in a state where they have incentives to turn off S\*BGP for at least one destination!

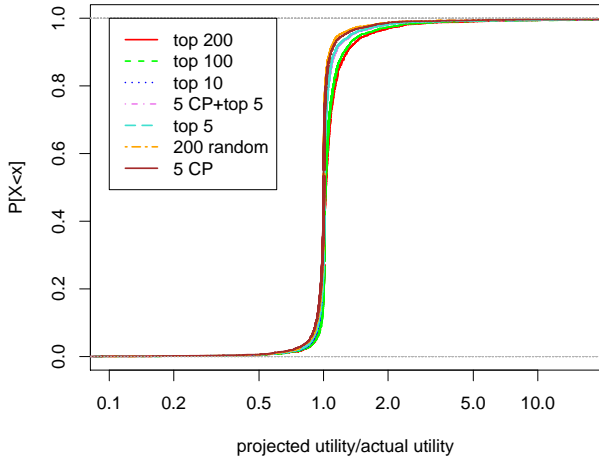
## 8. DISCUSSION FOR OUR MODEL

The wide range of parameters involved in modeling S\*BGP deployment means that our model (Section 3) cannot be *predictive* of S\*BGP deployment in practice. Instead, our model was designed to (a) capture a few of the most crucial issues that might drive S\*BGP deployment, while (b) taking the approach that simplicity is preferable to complexity.

### 8.1 Myopic best response.

For simplicity, we used a *myopic best-response* update rule that is standard in the game-theory literature [17]. In Section 5.6, we discussed the consequences of the fact that ISPs only act to improve their utility in the next round, rather than in long run. Another potential issue is that our update rule ignores the possibility that *multiple* ASes could deploy S\*BGP in the transition from a round  $i$  to round  $i + 1$ , resulting in the gap between the projected utility, and the actual utility in the subsequent round. Fortunately, our simulations show projected utility  $u_n(\neg S_n, S_{-n})$  is usually an excellent estimate of actual utility in the subsequent round. For example, in the case study of Section 5, 80% of ISPs overestimate their utility by less than 2%, 90% of ISPs overestimate by less than 6.7%. This observation also holds more generally across simulations:

**Figure 14:** We show the projected utility normalized by the utility nodes observed once they deployed S\*BGP for different sets of early adopters when  $\theta = 0$ . In most cases ASes projected utilities that are within a few percentage points of what they actually received in the next round.



**Figure 14: Projected utility normalized by actual utility after an AS deploys S\*BGP.**

## 8.2 Computing utility locally.

Because we lack information about interdomain traffic flows in the Internet, our model uses weighted counts of the subtrees of ASes routing through ISP  $n$  as a stand-in for traffic volumes, and thus ISP utility. While computing these subtrees in our model requires *global* information that would be unavailable to the average ISP (*e.g.*, the state  $S$ , the AS graph topology, routing policies), in practice, an ISP can just compute its utility by locally observing traffic flows through its network.

**Computing projected utility.** Computing projected utility  $u_n(\neg S_n, S_{-n})$  in practice is significantly more complex. While projected utility gives an accurate estimate of actual utility when it is computed using global information, ISPs may inaccurately estimate their projected utility when using only local information. Our model can accommodate these inaccuracies by rolling them into the deployment threshold  $\theta$ . (That is, if projected utility is off by a factor of  $\pm\epsilon$ , model this with threshold  $\theta \pm \epsilon$ .) Thus, while our approach was to sweep through a common value of  $\theta$  for every ISP (Section 6.2), extensions might capture inaccurate estimates of projected utility by randomizing  $\theta$ , or even by systematically modeling an ISP’s estimation process to obtain a measure for how it impacts  $\theta$ .

**Practical mechanisms for projecting future traffic patterns.** Because S\*BGP deployment can impact route selection, it is crucial to develop mechanisms that allow ISPs predict how security will impact traffic patterns through its network. Moreover, if ISPs could use such mechanisms to estimate projected utility, they would also be an important driver for S\*BGP deployment. For example, an ISP might set up a router that listens to S\*BGP messages from neighboring ASes, and then use these message to predict how becoming secure

might impact its neighbors’ route selections. A more sophisticated mechanism could use extended “shadow configurations” with neighboring ASes [1] to gain visibility into how traffic flows might change.

## 8.3 Alternate routing policies and actions.

**Routing policies.** Because our model of ISP utility depends on traffic volumes (Section 3.3), we need to a model for how traffic flows in the Internet. In practice, traffic flow is determined by the local routing policies used by each AS, which are arbitrary and not publicly known. Thus, we use a standard model of routing policies (Appendix A) based on business relationship and path length [15, 6].

Routing policies are likely to impact our results by determining (a) AS path lengths (longer AS paths mean it is harder to secure routes), and (b) tiebreak set size (Section 6.6). For example, we speculate that considering shortest path routing policy would lead to overly optimistic results; shortest-path routing certainly lead to shorter AS paths, and possibly also to larger tiebreak sets. On the other hand, if a large fraction of multihomed ASes always use one provider as primary and the other as backup (irrespective of the AS path lengths *etc.*) then our current analysis is likely to be overly optimistic. (Of course, modeling this is difficult given a dearth of empirical data on backup paths).

**Choosing routing policies.** An AS might cleverly choose its routing policies to maximize utility. However, the following suggests that this is intractable:

**THEOREM 8.1.** *When all other ASes’ routing policies are as in Appendix A, it is NP hard for any AS  $n$  to find the routing policy that maximizes its utility (in both the incoming and outgoing utility models). Moreover, approximating the optimal routing policy within any constant factor is also NP hard.*

The proof (in Appendix I) shows that this is NP-hard even if  $n$  has a single route to the destination, and must only choose the set of neighbors to which it announces the route. (Thus, the problem is tractable when the node’s neighbors set is of constant size.)

**Per-link S\*BGP deployment.** An ISP might be able to optimize its utility by turning ON S\*BGP on a *per link* basis, *i.e.*, with only a subset of its neighbors. (For instance, a second look at Figure 13 suggests that AS 4775 improve his utility by turning off S\*BGP on the link to his provider AS 2914.) Once again, this is intractable when an ISP has a large number of neighbors (Proof in Appendix J):

**THEOREM 8.2.** *Given an AS graph and state  $S$ , it is NP-hard to choose the set of neighbors for which ISP  $n$  should deploy S\*BGP so as to maximize its incoming*

utility. Moreover, approximating the optimum within any constant factor is also NP hard.

**Lying and cheating.** While it is well known that an AS can increase the amount of traffic it transits by manipulating its BGP messages [7], we avoided this issue because our focus is on technology adoption by economically-motivated ASes, not BGP manipulations by malicious or misconfigured ASes.

## 8.4 Other extensions to our model.

**Static AS graph.** Our model of interdomain routing assumes that the AS graph does not change. Because the time-scale of the deployment process can be quite large (*e.g.*, years), extensions to our model might also model the evolution of the AS graph with time, and possibly incorporate issues like the addition of new edges if secure ASes manage to sign up new customers.

**Mapping revenue to traffic volume.** Our model of ISP utility is based on the idea that revenue is related to the *total volume* of customer traffic the ISP transits. In practice, ISPs may use a variety of pricing policies, *e.g.*, by volume, flat rates based on discrete units of capacity. Thus, extensions might consider collecting empirical data on pricing policies to more accurately map revenue to traffic volumes.

## 9. RELATED WORK

**Social networks.** The diffusion of new technologies in social networks has been well studied in economics and game theory (*e.g.*, [23] and references therein). The idea that players will myopically best-respond if their utility exceeds some threshold is standard in this literature (*cf.*, our update rule (3)). However, in a social network, a player’s utility depends only on its immediate *neighbors*, while in our setting it depends on the set of secure *paths*. Thus, while [23] finds approximation algorithms for choosing an optimal set of early adopters, this is NP-hard in our setting (Theorem 6.1).

**Protocol adoption in the Internet.** The idea that competition over customer traffic can drive technology adoption in the Internet has appeared in many places in the literature [10, 36]. Ratnasamy *et al.* [36] suggest using competition for customer traffic to drive protocol deployment (*e.g.*, IPv6) at ISPs by creating new mechanisms for directing traffic to ASes with IPv6. Leveraging competition is much simpler with S\*BGP, since it directly influences routing decisions without requiring adoption of new mechanisms.

[21, 20, 41] study the role of converters (*e.g.*, IPv4-IPv6 gateways) on protocol deployment. While S\*BGP must certainly be backwards compatible with BGP, the fact that security guarantees only hold for fully-secure paths (Section 2.2.2) means that there is no reason to

convert BGP messages to S\*BGP messages. Thus, we do not expect converters to drive S\*BGP deployment.

**S\*BGP adoption.** Perhaps most relevant is Chang *et al.*’s comparative study on the adoptability of secure interdomain routing protocols [8]. Like [8], we also consider how early adopters create local incentives for other ASes to deploy S\*BGP. However, our study focuses on how S\*BGP deployment can be driven by (a) simplex S\*BGP deployment at stubs, and (b) the requirement that security plays a role in routing decisions. Furthermore, in [8] ISP utility depends on the security benefits offered by the partially-deployed protocol. Thus, the utility function in [8] depends on possible attacker strategies (*i.e.*, path shortening attacks) and attacker location (*i.e.*, random, or biased towards small ISPs). In contrast, our model of utility is based solely on economics (*i.e.*, customer traffic transited). Thus, we show that global S\*BGP deployment is possible even if ISPs’ local deployment decisions are *not* driven by security concerns. Also, complementary to our work is [5]’s forward-looking proposal that argues that extra mechanisms (*e.g.*, secure data-plane monitoring) can be added to S\*BGP to get around the problem of partially-secure paths (Appendix B). Finally, we note both our work and [5, 8] find that ensuring that Tier 1 ASes deploy S\*BGP is crucial, a fact that is not surprising in light of the highly-skewed degree distribution of the AS graph.

## 10. CONCLUSION

Our results indicate that there is hope for S\*BGP deployment. We have argued for (1) using simplex S\*BGP to secure stubs, (2) convincing but a small, but influential, set of ASes to become early adopters of S\*BGP, and (3) ensuring the S\*BGP influences traffic patterns by requiring ASes to (at minimum) break ties between equally-good paths based on security.

We have shown that, if deployment cost  $\theta$  is low, our proposal can successfully transition a majority of ASes to S\*BGP. The transition is driven through market pressure created when ISPs deploy S\*BGP in order to draw revenue-generating traffic into their networks. We also pointed out unexplored challenges that result from S\*BGP’s influence of route selection (*e.g.*, ISPs may have incentives to disable S\*BGP).

We hope that this work motivates the standardization and research communities to devote their efforts along three key lines. First, effort should be spent to engineer a lightweight simplex S\*BGP. Second, with security impacting route selection, ISPs will need tools to forecast how S\*BGP deployment will impact traffic patterns (*e.g.*, using “shadow configurations”, inspired by [1], with cooperative neighboring ASes) so they can provision their networks appropriately. Finally, our results suggest that S\*BGP and BGP will coexist in the long term. Thus, effort should be devoted to ensure that

S\*BGP and BGP can coexist without introducing new vulnerabilities into the interdomain routing system.

## Acknowledgments

This project was motivated by discussions with the members of the DHS S&T CSD Secure Routing project. We especially thank the group for the ideas about simplex S\*BGP, and Steve Bellovin for the example in Appendix B.

We are extremely grateful to Mihai Budiu, Frank McSherry and the rest of the group at Microsoft Research SVC for helping us get our code running on DryadLINQ. We also thank Edwin Guarin and Bill Wilder for helping us get our code running on Azure, and the Microsoft Research New England lab for supporting us on this project. We thank Azer Bestavros, John Byers, Mark Crovella, Jef Guarente, Vatche Ishakian, Isaac Keslassy, Eric Keller, Leo Reyzin, Jennifer Rexford, Rick Skowrya, Renata Texiera and Minlan Yu for comments on drafts of this work. This project was supported by NSF Grant S-1017907 and a gift from Cisco.

## 11. REFERENCES

- [1] R. Alimi, Y. Wang, and Y. R. Yang. Shadow configuration as a network management primitive. In *Sigcomm*, 2008.
- [2] ARIN. ARIN resource certification. <https://www.arin.net/resources/rpki.html>.
- [3] B. Augustin, B. Krishnamurthy, and W. Willinger. IXPs: Mapped? In *IMC*, 2009.
- [4] R. Austein, G. Huston, S. Kent, and M. Lepinski. Secure inter-domain routing: Manifests for the resource public key infrastructure. draft-ietf-sidr-rpki-manifests-09.txt, 2010.
- [5] I. Avramopoulos, M. Suchara, and J. Rexford. How small groups can secure interdomain routing. Technical report, Princeton University Comp. Sci., 2007.
- [6] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the Internet. In *ACM SIGCOMM*, 2007.
- [7] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 2010.
- [8] H. Chang, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure BGP protocol. In *Sigcomm*, 2006.
- [9] Y.-J. Chi, R. Oliveira, and L. Zhang. Cyclops: The Internet AS-level observatory. *ACM SIGCOMM CCR*, 2008.
- [10] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's Internet. *Trans. on Networking*, 2005.
- [11] J. Cowie. Rensys blog: China's 18-minute mystery. <http://www.rensys.com/blog/2010/11/chinas-18-minute-mystery.shtml>.
- [12] A. Fabrikant and C. H. Papadimitriou. The complexity of game dynamics: BGP oscillations, sink equilibria, and beyond. In *SODA*, pages 844–853, 2008.
- [13] L. Gao and J. Rexford. Stable Internet routing without global coordination. *Trans. on Networking*, 2001.
- [14] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *PAM*, April 2008.
- [15] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How secure are secure interdomain routing protocols. In *Sigcomm*, 2010.
- [16] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *Trans. on Networking*, 2002.
- [17] S. Hart. Adaptive heuristics. *Econometrica*, 2005.
- [18] IETF. Secure interdomain routing (SIDR) working group. <http://datatracker.ietf.org/wg/sidr/charter/>.
- [19] M. Jackson and L. Yariv. Diffusion on social networks. In *Conférences des journées Louis-André Gérard-Varet Public Economy Theory Meeting*, 2005.
- [20] Y. Jin, S. Sen, R. Guerin, K. Hosanagar, and Z. Zhang. Dynamics of competition between incumbent and emerging network technologies. In *NetEcon*, 2008.
- [21] D. Joseph, N. Shetty, J. Chuang, and I. Stoica. Modeling the adoption of new network architectures. In *CoNEXT*, 2007.
- [22] J. Karlin, S. Forrest, and J. Rexford. Autonomous security for autonomous systems. *Computer Networks*, oct 2008.
- [23] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD*, 2003.
- [24] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *JSAC*, 2000.
- [25] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J.-H. Cui, and A. G. Percus. Sampling large internet topologies for simulation purposes. *Computer Networks (Elsevier)*, 51(15):4284–4302, 2007.
- [26] C. Labovitz. Arbor blog: Battle of the hyper giants. <http://asert.arbornetworks.com/2010/04/the-battle-of-the-hyper-giants-part-1-2/>.
- [27] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Sigcomm*, 2010.
- [28] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Shang. Phas: Prefix hijack alert system. In *Usenix Security*, 2006.
- [29] C. D. Marsan. U.S. plots major upgrade to Internet router security. *Network World*, 2009.
- [30] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal topology generation. In *MASCOTS*, 2001.
- [31] V. S. Mirrokni and A. Skopalik. On the complexity of Nash dynamics and sink equilibria. In *EC*, pages 1–10, 2009.
- [32] S. Misel. "Wow, AS7007!". Merit NANOG Archive, apr 1997. <http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html>.
- [33] S. Morris. Contagion. *Review of Economics Studies*, 2003.
- [34] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. Quantifying the completeness of the observed internet AS-level structure. *UCLA Computer Science Department - Technical Report TR-080026-2008*, Sept 2008.
- [35] F. Orbit. <http://www.fixedorbit.com/metrics.htm>.
- [36] S. Ratnasamy, S. Shenker, and S. McCanne. Towards an evolvable Internet architecture. In *Sigcomm*, 2005.
- [37] Rensys Blog. Pakistan hijacks YouTube. [http://www.rensys.com/blog/2008/02/pakistan\\_hijacks\\_youtube\\_1.shtml](http://www.rensys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml).
- [38] RIPE. RIPE NCC Resource Certification. <http://www.ripe.net/certification/>.
- [39] E. Rogers. *Diffusion of Innovations, 5th Edition*. Free Press, 2003.
- [40] Sandvine. Fall 2010 global internet phenomena, 2010.
- [41] S. Sen, Y. Jin, R. Guerin, and K. Hosanagar. Modeling the dynamics of network technology adoption and the role of converters. *Trans. on Networking*, 2010.
- [42] T. Valente. *Network Models of the Diffusion of Innovations (Quantitative Methods in Communication Subseries)*. Hampton Press, 1995.
- [43] R. White. Deployment considerations for secure origin BGP (soBGP). draft-white-sobgp-bgp-deployment-01.txt, June 2003, expired.
- [44] H. P. Young. *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, 2001.
- [45] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey. Dryadlinq: a system for general-purpose distributed data-parallel computing using a

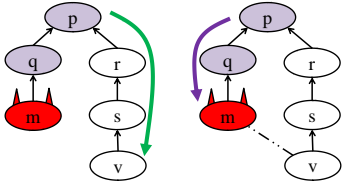


Figure 15: A new attack vector.

high-level language. In *Usenix OSDI*, 2008.

[46] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Infocom*, 1996.

## APPENDIX

### A. A MODEL OF ROUTING WITH BGP.

We follow [16] by assuming that each AS  $a$  computes paths to a given destination AS  $d$  based a *ranking* on outgoing paths, and an *export policy* specifying the set of neighbors to which a given path should be announced.

**Rankings.** AS  $a$  selects a path to  $d$  from the set of simple paths he learns from his neighbors as follows:

**LP Local Preference.** Prefer outgoing paths where the next hop is a customer over outgoing paths where the next hop is a peer over paths where the next hop is a provider.

**SP Shortest Paths.** Among the paths with the highest local preference, prefer the shortest ones.

**SecP Secure Paths.** If there are multiple such paths, and node  $a$  is secure, then prefer the secure paths.

**TB Tie Break.** If there are multiple such paths, node  $a$  breaks ties: if  $b$  is the next hop on the path, choose the path where hash,  $H(a, b)$  is the lowest.<sup>2</sup>

This standard model of local preference [13] captures the idea that an AS has incentives to prefer routing through a customer (that pays him) over a peer (no money is exchanged) over a provider (that he must pay).

**Export Policies.** This standard model of export policies captures the idea that an AS should only be willing to load his own network with transit traffic if his customer pays him to do so [13]:

**GR2** AS  $b$  announces a path via AS  $c$  to AS  $a$  iff at least one of  $a$  and  $c$  are customers of  $b$ .

### B. ATTACKS ON PARTIALLY SECURE PATHS

We show how preferring partially secure paths over insecure paths can introduce *new* attack vectors that do not exist even without S\*BGP:

**Figure 15:** Suppose that only ASes  $p$  and  $q$  are secure, and that malicious AS  $m$  falsely announces the path  $(m, v)$ , and suppose that  $p$ 's tiebreak algorithm

<sup>2</sup>In practice, this is done using the distance between routers and router IDs. Since we do not incorporate this information in our model we use a randomized tie break which prevents certain ASes from “always winning”.

prefers paths through  $r$  over paths through  $q$ . Then,  $p$  has a choice between two paths; a partially-secure false path  $(p, q, m, v)$ , and an insecure true path  $(p, r, s, v)$ . If no AS used S\*BGP,  $p$  would have chosen the true path (per his tiebreak algorithm); if  $p$  prefers partially secure paths, he will be fooled into routing to AS  $m$ .

## C. IMPLEMENTING OUR SIMULATIONS

We present implementation details for our simulations.

### C.1 On the scale of our simulations.

Our model introduces a number of scaling challenges:

**1. Aggregating utility for each destination.** Thus, to compute the utility function  $u_n(S)$  or  $u_n(\neg S_n, S_{-n})$ , we need to determine  $T_n(d, S)$ , the subtree of nodes routing through ISP  $n$  in state  $S$ , for *every destination* in the AS graph  $G(V, E)$  (see section 3.3). Thus, we need to do this  $|V| \approx 30K$ , since every AS can be a destination.

**2. Computing projected utility for each ISP.** Worse yet, notice that projected utility  $u_n(\neg S_n, S_{-n})$  is computed in a *unique state*  $(\neg S_n, S_{-n})$  for each ISP  $n$  in the AS graph. Since about 15% of ASes are ISPs, this results in  $0.15 \cdot |V| = 4.5K$  unique states  $(\neg S_n, S_{-n})$ .

**3. Computing the subtree  $T_n(d, S)$ .** Combining the above, it follows that we need to compute the subtree  $T_n(d, S)$  about  $0.15|V|^2 = 135M$  times in each round. To understand the scale of this computation, an algorithm that computes  $T_n(d, S)$  in 10ms would take 375 hours to run a single round of our simulations.

To effectively run simulations at such a scale, we heavily optimized the algorithms used to compute  $T_n(d, S)$ , and parallelized our simulations on a cluster of 200 machines running DryadLINQ, a recent research platform for distributed computing developed for the .NET (C#) framework.

### C.2 The fast routing tree algorithm.

Our optimized algorithm for computing  $T_n(d, S)$  is based on the following observation:

**OBSERVATION C.1.** *If ASes using the routing policies of Appendix A, the length and type (i.e., customer, peer, provider) of any node  $i$ 's path to a destination  $d$  is independent of the state  $S$  of the AS graph.*

(Discussion is in Appendix G.) [15] has an  $O(3|V|+|E|)$  algorithm that computes the *routing tree* of paths from every source AS in the AS graph  $G(V, E)$  to a given destination  $d$ . To further optimize this, we ran a modified version of [15]'s algorithm only *once* per destination  $d$  in order to obtain the following *per-destination information*: the (a) length (b) type (i.e., customer, peer, provider), and (c) *tiebreak set* of potential next hops for each node  $i$ 's path to destination  $d$ . Given

the per-destination information, we used the following algorithm to compute  $T_n(d, S)$ :

**Fast routing tree algorithm:** Since every node in  $i$ 's tiebreak set has a path to  $d$  that is one-hop shorter than node  $i$ 's path, we start at the destination  $d$  and proceed through each node  $i$  in ascending order of path length. For each node  $i$  in the AS graph, we determine (a) which AS in  $i$ 's tiebreak set that  $i$  chooses as its next hop to  $d$ , and (b) whether  $i$  has a fully-secure path to  $d$ , by checking if (1)  $i$  is secure in state  $S$ , and (2) there are nodes in  $i$ 's tiebreak set with a secure path to  $d$ .

The complexity of the fast routing tree algorithm is  $O(t|V|)$ , where  $t$  is the average size of the tiebreak sets in the AS graph. The average tiebreak set size is  $t = 1.18$  (across all source-destination AS pairs, see also Section 6.6), and for our graph  $|E| \approx 4|V|$ , so our optimized algorithm is almost 6 times faster than the  $O(3|V|+|E|)$  algorithm of [15]. After platform-specific optimizations, our C# implementation ran in about  $2ms$ .

### C.3 Parallelizing our simulations.

We parallelized our simulations using DryadLINQ, an approach similar to the Map-Reduce paradigm. That is, we parallelized (*i.e.*, mapped) our computation of the subtrees  $T_n(d, S)$  and  $T_n(d, (\neg S_n, S_{-n}))$  across destinations  $d$ , and then aggregated (*i.e.*, reduced) them to obtain utility  $u_n(S)$  and projected utility  $u_n(S)$ . We ran our code on a shared cluster of about 200 machines running DryadLINQ; by parallelizing across destinations, each machine was roughly assigned computation for about  $|V|/200 = 150$  destinations (but see [45] for details on how exactly this implemented in DryadLINQ).

**Initialization.** We initialize the simulation by computing the per-destination information for each destination in the AS graph. Thus, each machine runs a modified version of the  $O(3|V| + |E|)$  algorithm from [15] to obtain the per-destination information for each of the  $|V|/200 = 150$  destinations it is assigned. This step typically ran in under five minutes.

**Per-round computation.** For each round of the simulation, we start the ‘Map’ step of the computation by shipping out the state  $S$  and about  $|V|/200 = 150$  sets of per-destination information to each machine in the cluster. Then, for each destination, the each machine does the following:

- The fast routing tree algorithm is run once on state  $S$ . Then, a single pass is taken over the output to determine  $T_n(d, S)$  for each ISP  $n$  in the AS graph by counting the number of chosen paths that contain  $n$ .
- For each ISP  $n$ , the fast routing tree algorithm is run on state  $(\neg S_n, S_{-n})$ .  $T_n(d, (\neg S_n, S_{-n}))$  is

obtained by counting the number of paths in the output that contain  $n$ .

Thus, we require  $O(t|V| \times 0.15|V|)$  computation for each of the  $|V|$  destinations.

For the ‘Reduce’ step, we aggregate  $T_n(d, S)$  (*resp.*,  $T_n(d, (\neg S_n, S_{-n}))$ ) for each destination to obtain the utility  $u_n(S)$  (*resp.*, projected utility  $u_n(\neg S_n, S_{-n})$ ) for each ISP  $n$ . Finally, we use the utility and projected utility to determine whether an ISP  $n$  deploys S\*BGP or not, per the update rule in (3).

### C.4 Optimizing the computations.

In general, we need to run the fast routing tree algorithm  $|V|$  time for each destination in each round with state  $S$ . However, our simulations can be optimized with the following observations:

- If a destination  $d$  is not secure in state  $S$ , then there can never be any secure paths to that destination. Thus,  $T_n(d, S) = T_n(d, (\neg S_n, S_{-n}))$  for all ISPs  $n$ , we only run the fast routing tree algorithm once for destination  $d$  in that round.
- Theorem 6.2 shows that, in the outgoing utility model, a node never has an incentive to turn off S\*BGP. Thus, if ISP  $n$  is secure in a round with state  $S$ , we need not run the fast routing tree algorithm on state  $(\neg S_n, S_{-n})$  for all destinations in all subsequent rounds.
- Consider a round with state  $S$ , where ISP  $n$  does not have any nodes in its tiebreak set that have secure paths to destination  $d$ . It follows that ISP  $n$  cannot offer its neighbors any secure path in state  $(\neg S_n, S_{-n})$ , and thus will not change their tiebreak decisions relative to state  $S$ . Thus, we need not run the fast routing tree algorithm for state  $(\neg S_n, S_{-n})$  and destination  $d$ .

These optimizations significantly reduced our compute time. As a result of these optimization, our per-destination computations ran in a variable amount of time, depending on the state  $S$ .

### C.5 Putting it all together.

Overall, the complexity of our simulations is approximately  $O(|V| \cdot (3|V| + |E|)) \approx 6B$  for the initialization step, and  $O(0.15 \cdot t|V|^3) \approx 5000B$  per round, each of which was reduced by a factor of 200 by parallelizing with DryadLINQ. On an uncongested cluster, the initialization step ran in about five minutes, and one round typically completed in about 10 – 35 minutes, with congestion increasing running time by about 150%. Our simulations typically arrived at a stable state after 2-40 rounds, with simulations in where very few nodes became secure tended to terminate more quickly (Section C.4). Thus, we could run a simulation from start to finish in about 1-12 hours.

**Table 2: Summary of AS graphs**

Graph	ASes	peering	customer-provider
Cyclops+IXP [9, 3]	36,964	58,829	72,848
Augmented graph	36,966	77,380	72,848

**Table 3: Average path length from the five content providers to all other destinations**

AS	Cyclops	Augmented	Knodes
15169	2.7	2.1	2.2
8075	2.8	2.1	2.3
20940	3.6	2.2	2.2
22822	6.9	6.8	2.3
32934	3.5	2.1	2.4

## D. AS GRAPH SENSITIVITY ANALYSIS.

**Incompleteness of AS-level topologies.** It is widely reported that AS-level graphs of the Internet are incomplete [34], but a ground truth for AS-level connectivity remains elusive. This is especially problematic for large content providers that primarily peer with large numbers of ASes to drive down costs for both themselves and the networks they peer with. Since peering links are only exported to customers [34] and content providers do not generally have customers, this is potentially a large blind spot in the existing data. Indeed, we observed that average path lengths for the five CPs in the AS graph (according to the routing policies of Appendix A) were around 2.7-3.5 hops whereas they are reported to be much lower, around 2.2-2.4 hops [35]. The reported value we consider is the Knodes index [35], which uses public and private BGP data to measure the number of hops that must be traversed between IP addresses in a given network and all other IP addresses.

**Creating the augmented graph.** To understand how incompleteness of the AS-level topology impacts our results, we developed an augmented topology with particular focus on more accurate connectivity for the five CPs. Our strategy for augmenting the AS-level topology leveraged recent research on IXPs that finds that many CPs are joining IXPs and peering with a large fraction of their members [3].

We used the Cyclops AS graph from Dec. 9, 2010 with additional peering edges from [3]. In the Cyclops graph, the content providers have some customers, mainly owing to company acquisitions (e.g., YouTube’s AS 35361 is a customer of Google). Since the CPs do not generally provide transit to other ASes, we remove these 79 customer ASes from the graph. We summarize the resulting Cyclops+IXP graph in Table 2.

Starting with this graph, the five CPs were then connected randomly to ASes present at IXPs [3] until their average path length to *all destinations* on the Internet decreased to around 2.1-2.2 hops. The path lengths in the original and augmented topologies as well as the

**Table 4: Degree of five CPs in original and augmented graph with Tier 1s for comparison**

Cyclops+IXP	Cust.	Peer	Prov.	Total
Tier 1s				
174	2,928	377	12	3,317
3356	2,936	119	0	3,055
7018	2,407	135	0	2,542
701	2,069	72	0	2,141
1239	1,243	90	0	1,333
CPs				
15169	0	244	3	247
8075	0	90	2	92
32934	0	49	4	53
20940	0	81	31	112
22822	0	567	10	577
Augmented				
15169	0	3,931	3	3,934
8075	0	3,927	2	3,929
32934	0	3,922	4	3,926
20940	0	3,895	31	3,926
22822	0	3,917	10	3,927

reported Knodes index [35] (an approximation of path length) are shown in Table 3.

**Properties of the augmented graph.** In our augmented AS graph, the five CPs have higher degree than even the largest Tier 1s (summarized in Table 4). However, unlike the Tier 1s the five CPs edges are primarily peering edges. The five CPs also do not provide transit. Note also that the path lengths for LimeLight (AS 22822) are longer than paths observed by other ASes. This may be caused by the AS-level topology not being particularly complete for LimeLight or more likely, that LimeLight’s routing policies are very different from those of Appendix A.

## E. CHOOSING EARLY ADOPTERS IS HARD!

We now prove that finding the optimal set of early adopters is NP-hard.

**THEOREM E.1.** *For an AS graph  $G(V, E)$  and a parameter  $1 \leq k \leq |V|$ , finding a set of early adopter ASes of size  $k$  that maximizes the number of ASes that are secure when the deployment process terminates is NP-hard, both in the incoming and outgoing utility model. Approximating the solution within a constant factor is NP-hard as well.*

**PROOF.** We prove the theorem via reduction from the NP-complete SET-COVER. In SET-COVER we are given  $m$  subsets of a universe  $U$ ,  $S_1, \dots, S_m$ , and an integer  $k$ . The objective is to find a collection of  $k$  of the subsets that covers the most elements in  $U$ .

Given an instance of SET-COVER, our reduction to the problem of finding a set of early adopters is as follows. We create a network with a single destination node  $d$ , 2 nodes,  $s_{i1}$  and  $s_{i2}$ , for each  $S_i$ , and a node  $u$  for every element in  $U$  (plus some additional nodes,

as we now explain). Our construction is described in Figure 16.

$d$  is a stub customer of all  $s_{i1}$ 's, and each  $s_{i1}$  is a customer of  $s_{i2}$ . Each  $s_{i2}$  is a provider of every stub  $u$  that corresponds to an element in the universe  $U$  (in SET-COVER) that belongs to  $S_i$ . Observe that each node  $u$  has a 4-hop provider route to  $d$  through every node  $s_{i2}$  such that  $u \in S_i$ . We shall assume that each node  $u$  has another 4-hop provider route that does not go through any of the aforementioned nodes, and that this route is preferable to all other routes. We also assume that all of these additional routes are disjoint. For simplicity, we do not include these routes in the figure.

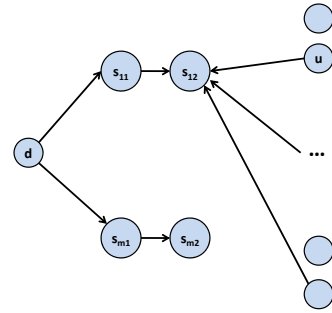


Figure 16: Proof of Theorem E.1.

We restrict our attention to the case that there exist a collection  $C$  of  $k$  sets in  $S_1, \dots, S_m$  (in the SET-COVER instance) that together cover *all* elements in  $U$ . (The proofs establishing the NP-completeness of SET-COVER actually show that even when guaranteed that  $k$  such sets exist finding them is NP-hard.) We observe that in this case, the optimal choice of early set of adopters of size  $k$  (in both the incoming and outgoing utility models) is to select each  $s_{i1}$  such that  $S_i \in C$  to be an early adopter. Observe that following this choice of early adopters, at the first time step all their  $s_{i2}$  providers will deploy S\*BGP and consequently all the  $u$  nodes will upgrade to S\*BGP. Observe also that every other choice of  $k$  nodes to secure leads to worse end results.

Hence, an optimal set of early adopters implies an optimal solution to SET-COVER. SET-COVER is not only NP-hard to solve exactly but also NP-hard to approximate within any constant factor. That is, even finding a collection of sets of size  $\alpha k$ , where  $\alpha$  is a constant, that covers all elements in  $U$  is NP hard. We make the following important observation. Our reduction is such that a constant approximation solution to our problem guarantees a constant approximation to SET-COVER. To see this, observe that every choice of  $\alpha k$   $s_{i1}$ 's leads the corresponding  $\alpha k$   $s_{i2}$ 's and their customers (that correspond to the covered elements in SET-COVER) to deploy in the next time step. Thus, eventually, the number of nodes to deploy is  $2\alpha k$  plus the number of covered elements in  $U$ . Hence, a constant factor approximation for our problem implies a constant factor approximation to SET-COVER. The theorem follows.  $\square$

## F. OSCILLATIONS EXIST

We prove that oscillations exist in the incoming utility model. Our proof is an scenario we call the OSCILLATOR; a group of nodes that alternates between turning S\*BGP ON and OFF, and never arrives at a stable

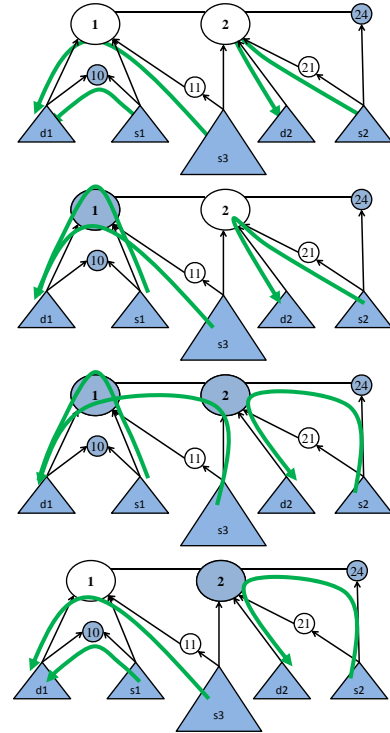


Figure 17: Phases of the Oscillator.

state:<sup>3</sup>

**Figure 17** There are three source of traffic  $s1$ ,  $s2$ ,  $s3$ , trees of nodes connected by customer-provider edges (so that all nodes in the customer tree are transitive customers of the root). There are two stub destinations  $d1$  and  $d2$ . For simplicity, assume that  $s1$  and  $s3$  send traffic to  $d1$  via the nodes shown, and that they have some better path to  $d2$  (not shown). Similarly suppose  $s2$  sends traffic to  $d2$  via the nodes shown. Suppose that  $s3$  contains many more nodes than  $s1$ , and that each of  $s1, s2, s3$  has a tiebreak algorithm that prefers paths through nodes with the lowest AS number. We'll also assume that nodes 11, 21 are stuck in an insecure state, while nodes 24 is stuck in a secure state, and show an oscillation involving nodes 1 and 2:

<sup>3</sup>This example was *not* taken from our simulations.

**Phase 1:** Initially, both 1 and 2 are insecure. Here, ISP 1’s incoming utility increases from (customer) traffic originating at  $s3$  and node 2 obtains incoming utility from customer traffic originating at  $s2$ .

**Phase 2:** Node 1 now becomes secure in order to attract customer traffic from  $s1$ . Note that node 2 has now lost incoming utility, because node 1 has stolen away the customer traffic from  $s3$ .

**Phase 3:** Node 2 now becomes secure in order in order to get back the traffic from  $s3$ . In doing this, node 2 loses incoming utility from nodes in  $s2$  (since their traffic now enters  $s3$  along a peer edge) but since  $s3$  has more nodes than  $s2$ , node 2 is still better off. Notice that node 1 has now lost the customer traffic from  $s3$ .

**Phase 4:** Now, node 1 turns on S\*BGP in order to get back the customer traffic from  $s3$ ; (by becoming insecure,  $s3$  choosing paths according to his tiebreak algorithm, that comes up in favor of node 11, *cf.*, Figure 13). Notice that node 2 has lost the customer traffic from  $s3$ ; furthermore, traffic from  $s2$  has no impact on his utility, because he receives it via a peering edge.

**Phase 5:** Finally, node 2 increases his incoming utility by turning off S\*BGP (again *cf.*, Figure 13). This follows because he convinces  $s2$  to route to him along a customer edge (rather than a peer edge, as in Phase 3), and we are back in Phase 1!

## G. A NOTE ON BGP CONVERGENCE

In this section, we make a number of observations about route selection when ASes use the routing policies of Appendix A. Namely, we show that when ASes use these routing policies, (1) it follows that BGP converges to a stable state (*i.e.*, no AS will want to change its path selection in order to obtain a path that is more preferred according to its ranking function [16]), and (2) we discuss Observation C.1, that was the basis of our optimized simulation algorithms.

**Preliminaries.** Since BGP sets up routes to each destination independently, we focus on routing to a unique destination  $d$ . We say that a route  $(v_0, v_1, \dots, d)$  is *exportable* if for every  $i \geq 0$  it holds that  $v_{i+1}$  announcing the route  $(v_{i+1}, \dots, d)$  to  $v_i$  does not violate **GR2**. We say that a route  $(v_0, v_1, \dots, d)$  is a *customer route* if  $v_1$  is a customer of  $v_0$ . We define *peer routes* and *provider routes* analogously.

**Discussion for Observation C.1.** Given a deployment state  $S$  (*i.e.*, the set of secure nodes), for every node  $i$  we define the  $BEST - ROUTES(i, S, d)$  to be the set of all exportable routes from  $i$  to  $d$  that are preferred by  $i$  over all other exportable routes. Observe that all routes in  $BEST - ROUTES(i, S, d)$  must belong to the same type—customer routes, peer routes, or provider routes—and either all be (entirely) secure

or all be insecure. Moreover, all routes in  $BEST - ROUTES(i, S, d)$  must be of the same length. We define  $NEXT(i, S, d)$  to be the set that contains every node  $j$  such that  $j$  is  $i$ ’s next-hop node on some route in  $BEST - ROUTES(i, S, d)$ . The following lemma shows that we can capture the routing decisions of every AS  $i$  in the graph by maintaining information about (a)  $NEXT(i, S, d)$ , (b) length of paths in  $BEST - ROUTES(i, S, d)$ , and (c) type of path in  $BEST - ROUTES(i, S, d)$ . Furthermore, it argues that BGP converges to a stable state:

**LEMMA G.1.** *In our routing model, BGP is guaranteed to converge to a stable state where each node  $i$ ’s next-hop is the node in  $NEXT(i, S, d)$  that  $i$  breaks ties in favor of.*

**PROOF.** The lemma follows from the three following propositions:

**PROPOSITION G.2.** *In our routing model BGP is guaranteed to converge to a stable state where for each node  $i$  whose  $BEST - ROUTES(i, S, d)$  consists of customer routes it holds that  $i$ ’s next-hop is in  $NEXT(i, S, d)$ .*

**PROOF.** We prove this by induction on the path length. Consider the case that  $BEST - ROUTES(i, S, d)$  consists of a customer route of length 1. Clearly, in this case  $i$  will select the direct route to  $d$  (that is its most preferred route), and so its next-hop will indeed trivially be as in the statement of the theorem. Next, consider the case that  $BEST - ROUTES(i, S, d)$  consists of customer routes of length 2. Observe that in this case each node in  $NEXT(i, S, d)$  has a customer route of length 1 and thus will, for some moment onwards, converge to that route (as we have established). Thus, from some point in time forth  $i$  shall have all routes in  $BEST - ROUTES(i, S, d)$  available to it and shall hence have a node in  $NEXT(i, S, d)$  as a next-hop, and specifically the node that  $i$  breaks ties in favor of. And so on.  $\square$

Similar proofs establish the following.

**PROPOSITION G.3.** *In our routing model BGP is guaranteed to converge to a stable state where for each node  $i$  whose  $BEST - ROUTES(i, S, d)$  consists of peer routes it holds that  $i$ ’s next-hop is in  $NEXT(i, S, d)$ .*

To see why Proposition G.3 holds, observe that if a node  $i$  has an exportable peer route in  $BEST - ROUTES(i, S, d)$ , then the next-hop on that route has an exportable customer route in its  $BEST - ROUTES$  set. Therefore, by Proposition G.2, the next-hop node’s route will, from some moment in time forth, stabilize. From that moment onwards node  $i$  will have have a route in  $BEST - ROUTES(i, S, d)$  available to it.

PROPOSITION G.4. *In our routing model BGP is guaranteed to converge to a stable state where for each node  $i$  whose  $BEST - ROUTES(i, S, d)$  consists of provider routes it holds that  $i$ 's next-hop is in  $NEXT(i, S, d)$ .*

The proof of Proposition G.4 is similar to that of the previous propositions. Propositions G.2 and G.3 establish that the route of every node  $i$  whose  $BEST - ROUTES(i, S, d)$  set only consists of customer or peer routes will eventually stabilize. We can now use induction (as in the proof of Proposition G.2) to show that the routes whose  $BEST - ROUTES$  sets consist of provider routes will also eventually stabilize). The induction now is on the number of customer-provider edges on the route.

## H. ISPS NEVER TURN OFF S\*BGP IN THE OUTGOING UTILITY MODEL

THEOREM H.1. *In the outgoing utility model, a secure node will never have an incentive to turn OFF S\*BGP.*

PROOF. Consider a deployment state  $S$  where a node  $i$  is OFF and the deployment state  $\bar{S}$  that is identical to state  $S$  with the exception that node  $i$  is ON in  $\bar{S}$ . We now prove that  $i$ 's outgoing utility in  $\bar{S}$  (that is, its utility from the routing state BGP converges to when the deployment state is  $\bar{S}$  cannot possibly be less than its outgoing utility in  $S$ . Because we establish that this is true for every node  $i$  and every two such states  $S$  and  $\bar{S}$ , it follows that a node can never gain from deactivating S\*BGP.

The above follows from Lemma G.1. To see this, consider a single destination  $d$ . We have shown that when the deployment state is  $S$ , BGP is guaranteed to converge to a stable state where each node  $i$ 's next-hop is the node in  $NEXT(i, S, d)$  that  $i$  breaks ties in favor of (see Appendix G). Recall that it must hold, for each node  $i$ , that either all routes in  $BEST - ROUTES(i, S, d)$  are secure or all routes in  $BEST - ROUTES(i, S, d)$  are insecure.

Now, consider two states  $S$  and  $\bar{S}$  as above, that differ only in that a node  $i$  becomes secure in  $\bar{S}$ . Consider a node  $j \neq i$  in the network. We handle three cases:

1. **Case I: Both  $BEST - ROUTES(j, S, d)$  and  $BEST - ROUTES(j, \bar{S}, d)$  consist only of insecure routes.** That is, the fact that  $i$  turned ON in  $\bar{S}$  did not create new “best routes” for node  $j$ . Therefore,  $BEST - ROUTES(j, S, d) = BEST - ROUTES(j, \bar{S}, d)$ . Lemma G.1 implies that in both  $S$  and  $\bar{S}$   $j$  will select the same next-hop en route to  $d$ .
2. **Case II:  $BEST - ROUTES(j, S, d)$  consists only of insecure routes and  $BEST - ROUTES(j, \bar{S}, d)$**

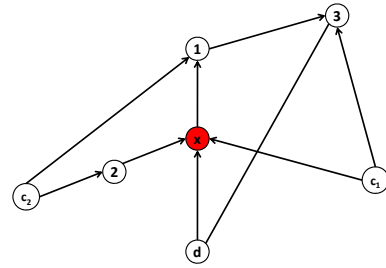


Figure 18: The DILEMMA network:  $x$  cannot attract both  $c_1$  and  $c_2$ 's traffic simultaneously along customer edges.

**consists only of secure routes.** Observe that in this case all routes in  $BEST - ROUTES(j, \bar{S}, d)$  must traverse  $i$  (as previously  $j$  had no secure routes).

3. **Case III: Both  $BEST - ROUTES(j, S, d)$  and  $BEST - ROUTES(j, \bar{S}, d)$  consist only of secure routes.** Observe that in this case  $BEST - ROUTES(j, S, d) \subseteq BEST - ROUTES(j, \bar{S}, d)$  ( $i$ 's transition to S\*BGP could not have made a secure route insecure, only add to the set of secure routes).

Observe that the above three cases imply that if  $i$  did not create new secure “best routes” for another node  $j$  then  $j$ 's choice of next-hop en route to  $d$  remains the same. Otherwise,  $j$  might choose a next-hop that goes through  $i$ . Thus,  $i$  can never lose traffic from transitioning to S\*BGP. This holds for every destination  $d$ . The theorem follows.  $\square$

## I. FINDING THE OPTIMAL ROUTING POLICY IS HARD

We now show that it is NP-hard for a node to find a routing policy (ranking function and export policy) that is optimal with respect to traffic attraction. [15] shows that this statement is correct in the outgoing utility model. (Specifically, Theorem F.7 and Theorem F.8 work if the so called “manipulator” is a customer of the destination node. This implies that when all other ASes' routing policies are as in Appendix A, it is NP hard for any AS  $n$  to find the routing policy that maximizes its outgoing utility.) We now extend the result in [15] to the incoming utility model.

THEOREM I.1. *When all other ASes' routing policies are as in Appendix A, it is NP hard for any AS  $n$  to find the routing policy that maximizes its incoming utility. Moreover, approximating the optimal routing policy within any constant factor is also NP hard.*

PROOF. Our proof follows the proof in [15], that shows that attracting traffic is NP-hard when the node has no economic considerations, but merely wants as much traffic as possible to flow through it. [15] shows that it is even hard to approximate the optimum within

any constant factor. We show that this hardness result continues to hold in our context. The key ingredient in the proof in [15] is showing the existence of a construction called the DILEMMA network. This construction is then used in a reduction from the NP-hard INDEPENDENT-SET problem. To prove our NP-hardness results we show that a DILEMMA network can be constructed for our context as well. The rest of our proof then proceeds as in [15]. The reader is referred to [15] for an overview of the complete proof argument.

In the DILEMMA network, there exist a node  $x$  that wishes to attract the traffic of two other nodes in the network,  $c_1$  and  $c_2$ . However, while  $x$  can attract the traffic of  $c_1$  alone, or of  $c_2$  alone, it is unable to attract the traffic of both nodes simultaneously. We now show how a network can be constructed in our context.

Consider the network in Figure 18. Observe that node  $x$  is directly connected to the (single) destination  $d$ , and so its ranking function is trivial. Thus, node  $x$ 's choice of routing policy boils down to the choice of export policy (*i.e.*, which nodes to announce the route  $(x, d)$  to). Observe also that  $x$  must clearly announce  $(x, d)$  to its customers (for otherwise it will definitely fail in attracting traffic from  $c_1$  or  $c_2$ ). Hence, the only decision that  $x$  must make is whether or not to announce  $(x, d)$  to node 1.

We assume that nodes' tie-breaking rules are such that they never break ties in favor of routes that have  $x$  as a next hop. We make the following observations. If  $x$  does not announce the route  $(x, d)$  to node 1, then  $x$  will attract  $c_2$ 's traffic along a customer edge (as  $c_2$  will have no other route to  $d$ ), but will lose  $c_1$ 's traffic (as 3 will then route directly to  $d$  and so  $c_1$  shall route through 3). If, on the other hand,  $x$  does announce the route  $(x, d)$  to 1 then  $x$  attracts  $c_1$ 's traffic along a customer edge (as 3 will then select the long customer route through  $x$ ), but  $c_2$ 's traffic shall then reach  $x$  along the edge  $(1, x)$  (a provider edge).

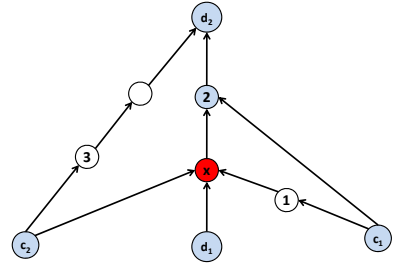
The proof now follows from the arguments in [15].  $\square$

## J. PER-LINK DEPLOYMENT IS HARD

We consider the case that a node can deploy S\*BGP on a per-link basis. That is, instead of just turning itself ON or OFF, the node can decide to sign/verify routes for a specific neighbor, yet not to do so for another neighbor. We note, that we focus on the case that deployment entails both signing and verification, and ignore the scenario that a node might decide only to verify signed routes from a neighbor, but not to sign to routes for that neighbor.

We show that in the incoming utility model, choosing which links to deploy S-BGP on is NP-hard.

**THEOREM J.1.** *Given an AS graph and state  $S$ , it is NP-hard to choose the set of neighbors for which ISP  $n$  should deploy S\*BGP so as to maximize its incoming*



**Figure 19: The DILEMMA network:  $x$  cannot attract both  $c_1$  and  $c_2$ 's traffic simultaneously along customer edges.**

*utility. Moreover, approximating the optimum within any constant factor is also NP hard.*

**PROOF.** Our proof, similarly to the proof of Theorem I.1, follows the proof in [15]. Thus, the key is showing the existence of a DILEMMA network construction, where a node  $x$  can attract the traffic of node  $c_1$  alone (along a customer edge), or of a node  $c_2$  alone (along a customer edge), but is unable to attract the traffic of both nodes simultaneously.

We present such a DILEMMA network construction in Figure 18. Nodes  $c_1$  and  $c_2$  wish to sent traffic to destination nodes  $d_1$  and  $d_2$ , respectively. We assume that nodes' tie-breaking rules are such that they never break ties in favor of routes that have  $x$  as a next hop. Observe that if node  $x$  secures the link to node 2 then it gains  $c_2$ 's traffic (to which it can now offer a secure route to  $d_2$ ), yet loses  $c_1$ 's traffic (as  $c_2$  would prefer the secure route to  $d_1$  through 2). However, if node  $x$  does not secure the link to node 2, then it gains  $c_1$ 's traffic (because of tie-breaking), but loses  $c_2$ 's traffic (also because of tie-breaking). Thus,  $x$  can indeed attract traffic (along a customer edge) of either  $c_1$  or  $c_2$ , but cannot attract both nodes' traffic simultaneously.

The proof now follows from the arguments in [15].  $\square$

We prove that in the outgoing utility model finding the utility maximizing per-link deployment can be done in a computationally-efficient manner.

**THEOREM J.2.** *Given an AS graph and state  $S$ , deploying S\*BGP for all neighbors maximizes the outgoing utility for every node in the network.*

The proof of theorem J.2 uses the exact same arguments as in the proof of Theorem H.1. (Only now we apply these arguments on a per-link basis.)

## K. DETECTING OSCILLATIONS IS HARD!

The complexity class PSPACE consists of all decisions problems that can be solved using only polynomial space (no limit is set on the amount of *time* it takes to

compute the solution). Intuitively, PSPACE-complete problems can be viewed as the hardest problems that lie within PSPACE. These problems are at least as hard as the notoriously NP-complete SAT, TRAVELING-SALESMAN, and CLIQUE problems, and are widely believed to be even harder. Hence, if a decision problem is PSPACE-complete then it is impossible to solve in a computationally-efficient manner unless the most fundamental premises of contemporary complexity theory (including  $P \neq NP$ ) collapse.

We present the following computational problem, that we call “S\*BGP ADOPTION”. In S\*BGP ADOPTION, the input is a network with  $n$  nodes (as described in Section 3), and a string  $S = (S_1, \dots, S_n) \in \{0, 1\}^n$ . The goal in S\*BGP ADOPTION is to determine whether, when starting at a state in which each node  $i$  is using S-BGP iff  $S_i = 1$ , the deployment of S-BGP can enter indefinite oscillations. S\*BGP ADOPTION can easily be shown to be in PSPACE. This is because with polynomial space it is possible to simply go over all possible starting states and check, for each one, whether the deployment dynamics converge to a stable state. We prove that S\*BGP ADOPTION is, in fact, PSPACE-complete, thus establishing the intractability of predicting the network evolution from a given state.

**THEOREM K.1.** *Given an AS graph and state  $S$ , it is PSPACE-complete to decide if the deployment process will terminate at a stable state in the incoming utility model.*

In the remainder of this section we prove Theorem K.1, that is, establish that S\*BGP ADOPTION is PSPACE-complete in the incoming utility model.

### K.1 Definition of STATIC-MODE.

We present a reduction to S\*BGP ADOPTION from the following problem, that we term “STATIC-MODE” and show is PSPACE-complete. In STATIC-MODE the input is a *space-bounded* Turing machine and the goal is to determine, for a given string that is fed into the Turing machine, whether the Turing machine eventually enters a fixed configuration. We now present STATIC-MODE in detail.

#### Input:

- a specification of a space-bounded Turing machine  $M$  in the form of an 8-tuple  $(Q, \Gamma, b, \Sigma, q_0, F, r, \delta)$ , where
  - $Q$  is a finite, non-empty set of *machine-states*.  $|Q| = q$ .
  - $\Gamma$  is a finite, non-empty set of the tape *alphabet/symbols*.  $|\Gamma| = \gamma$ .
  - $b \in \Gamma$  is the unique *blank symbol*.
  - $\Sigma \subseteq \Gamma$  is the set of *input symbols*.
  - $q_0$  is the *initial machine-state*.

- $F \subseteq Q$  is the set of *final machine-states*.
- $r > 0$  is an integer bound on the length of  $M$ 's tape.
- $\delta : \{1, \dots, r\} \times Q \times \Gamma \rightarrow \{1, \dots, r\} \times Q \times \Gamma$  is the *transition function*.  $\delta$  specifies how  $M$  moves from one *configuration* of  $M$ —a location of  $M$ 's head  $h \in \{1, \dots, r\}$ , a state  $s \in Q$  and a string of symbols  $g \in \Gamma^r$  that specifies the cells' contents—to another. We restrict  $\delta$  so that if  $\delta(h, s, g) = (h', s', g')$ , then  $h' \in \{h-1, h, h+1\}$  (that is, the head can move at most one cell at each time).

- a string  $x \in \Sigma^r$  that is  $M$ 's *input string* (the input to be fed into  $M$ ).

**Goal:** We call a configuration  $c = (h, s, g)$  of the Turing machine  $M$  *static* if  $\delta(c) = c$ . The objective in STATIC-MODE is to determine whether  $M$ 's execution for the input string  $x$  reaches a static configuration.

STATIC-MODE is closely related to the problem of determining whether a space-bounded Turing machine  $M$  (as above) will *halt* for a given input string, that is known to be PSPACE-complete (see, *e.g.*, [12, 31]). Indeed, a simple reduction from the latter (essentially guaranteeing via easy tweaks that  $M$  enter a static mode instead of halting) establishes that STATIC-MODE too is PSPACE-complete.

**PROPOSITION K.2.** *STATIC-MODE is PSPACE-complete.*

### K.2 High-level overview.

We now give a high-level overview of our proof that S\*BGP ADOPTION is PSPACE-complete. We present a polynomial-time reduction from STATIC-MODE to S\*BGP ADOPTION, thus establishing that the latter is also PSPACE-complete. We first translate the input to STATIC-MODE to an input in S\*BGP ADOPTION. We then show that the deployments dynamics in S\*BGP ADOPTION simulate the execution of the Turing machine  $M$  in STATIC-MODE and prove that convergence to a stable state in S\*BGP ADOPTION is achieved iff  $M$  enters a static configuration in STATIC-MODE.

Given an input to STATIC-MODE, we construct a network in S\*BGP ADOPTION. We now present some key elements in our construction. We discuss the details in Section K.10. We create the following sets of nodes (we also create additional nodes that are added to guarantee traits discussed below):

- $k$  *head nodes*,  $h_1, \dots, h_r$ , where each node  $h_i$  represents a possible location of Turing machine  $M$ 's head.
- $q$  *machine-state nodes*,  $\{s_\beta\}_{\beta \in Q}$ , where each node  $s_\beta$  represents the state  $\beta \in Q$  of STATIC-MODE.

- $k$  cell clusters,  $C_1, \dots, C_r$ , where each cell cluster  $C_i$  consists of  $\gamma$  symbol nodes  $\{g_{i,\sigma}\}_{\sigma \in \Gamma}$ , each representing a possible symbol in cell  $i$ .

We call a state of the network in S\*BGP ADOPTION “clean” if only a single head node  $h$  is using S-BGP (is ON), only a single machine-state node  $s$  is ON, only a single node  $g_i$  in every cell-cluster  $C_i$  is ON. Observe that every clean state in S\*BGP ADOPTION captures a configuration of the Turing machine  $M$  is STATIC-MODE (in which the index of the single head node using S-BGP is the location of  $M$ ’s head, and so on).

We set the initial state of the network in S\*BGP ADOPTION to be the clean state in which the single head node ON is  $h_1$ ; the single state node ON is the node that represents  $q_0$ ; in each cell cluster  $C_i$ , the single symbol node ON is  $g_{i,x_i}$ . Observe that this clean state in S\*BGP ADOPTION captures the initial configuration of  $M$  (the head is pointing to the first cell, that contains  $x_1$ , and the machine-state is the initial machine-state  $q_0$ ).

We prove that, from a clean state  $\alpha$  in S\*BGP ADOPTION that represents a configuration  $c = (h, s, g)$  in STATIC-MODE, the network evolves (within a constant number of time steps) to another clean state that represents the configuration that immediately follows  $c$  in STATIC-MODE (after applying  $\delta$ ). Thus, we show that when starting at an initial state of the network as described above, the network evolution essentially mimics the execution of  $M$  for the input string  $x$ . In addition, we prove that if  $M$  enters a static mode when executed on  $x$ , then the network will converge to a stable state in S\*BGP ADOPTION, and vice versa. This reduction hence establishes that S\*BGP ADOPTION is indeed PSPACE-complete.

### K.3 The AND and CHICKEN gadgets

We now present two gadgets that play a crucial role in the proof: the AND GADGET and the CHICKEN GADGET. We first make the following general remarks about our constructions below.

**Fixed nodes.** In all of our gadgets there are some nodes that can sometimes be ON and sometimes be OFF, and some nodes, that we call “fixed nodes”, whose status is fixed, *i.e.*, they are either ON all the time or OFF all the time. There are many simple gadgets that we could construct to ensure that a particular node remains stuck in a certain state regardless of the state of the other nodes in the network. To reduce clutter we omit these.

**Routing policies.** Nodes’ routing policies in all our constructions are as in Appendix A.

**Tie-breaking.** In our constructions below we shall assume that nodes break ties between equally good routes in favor of the route with the next hop that has the low-

est AS number.

**Traffic flows.** To create the flows of traffic in our gadgets, we specify a source of traffic, and a destination of traffic. Traffic sources are trees of nodes connected by customer-provider edges (so that all nodes in the customer tree are transitive customers of the root). We represent such trees as pyramids in our figures, and indicate the number of nodes in a customer tree with a label, *e.g.*, *Cross 1* in Figure 21 has size  $m$ . Similarly, we depict traffic *destinations* as pyramids. To create a traffic flow of a particular magnitude in our gadgets, a customer tree will send traffic to destination, *e.g.*, in Figure 21 customer tree *Cross 1* sends traffic to destination  $d_2$ , creating a traffic flow of size  $m$ . The nodes in the gadgets will then (potentially) obtain utility by attracting this traffic flow.

**Getting rid of non-designated traffic.** Our gadgets will rely on the fact that only designated traffic flows affect the decisions made by nodes in the gadget. We will always state (in the text) exactly which flows are designated, *e.g.*, *Cross 1* sending traffic to  $d_2$  in Figure 21. However, we also need a way to ensure that non-designated traffic does not flow into the gadget and affect the decisions made by the nodes. In the two simple two gadgets below, this can easily be achieved. We later discuss how this is done for more complex constructions.

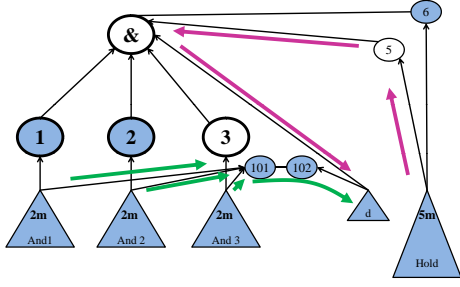
### K.4 The AND Gadget.

We present the AND GADGET, shown in Figure 20. In this gadget there are three “input-nodes”, 1, 2, 3, and a single “output node”  $\&$ . The AND GADGET is such that node  $\&$  turns ON iff nodes 1, 2, 3 turn on (and thus, node  $\&$  can be regarded as the AND operator for inputs 1, 2, 3). To achieve this we create additional fixed nodes: four nodes 101, 102, 5, 6, the destination  $d$  and customer trees *And i* for  $i = 1, 2, 3$  and *Hold*. All fixed nodes are constantly ON, with the exception of node 5, that is OFF.

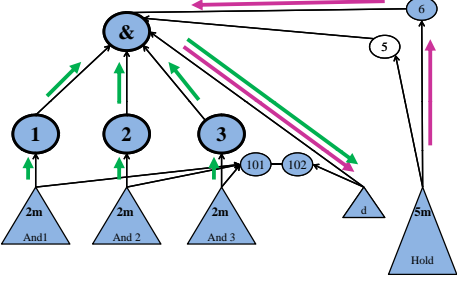
PROPOSITION K.3. *Node  $\&$  turns ON iff nodes 1, 2, 3 turn ON.*

PROOF. We make the following observations.

- **And traffic.** For each  $i \in \{1, 2, 3\}$ , the *And i* customer tree has two available routes to  $d$ : a secure route (*And i*, 101, 102,  $d$ ) and the route (*And i*,  $i$ ,  $\&$ ,  $d$ ). Observe that the latter route is chosen iff both  $i$  and  $\&$  are ON (since the customer tree prefers secure routes over insecure routes, and  $i$  has a lower AS number than 101.) Observe also that if the latter route is used by customer tree *And i* then  $\&$  obtains  $2m$ -units of utility.
- **Hold traffic.** Customer tree *Hold* also has two available routes to  $d$ , route (*Hold*, 5,  $\&$ ,  $d$ ) and route



(a)  $\&$  is OFF, it prefers  $5m$ -units of utility from Hold traffic



(b)  $\&$  is ON, it prefers  $6m$ -units of utility from And traffic

**Figure 20: States of the And Gadget .**

(Hold, 6,  $\&$ ,  $d$ ). Observe that the latter route is chosen iff both  $i$  and  $\&$  are ON (since if  $\&$  is on, the hold traffic has a secure route to  $d$ ). Observe also that if the former route is chosen,  $\&$ 's utility is  $5m$  (since it receives the traffic from a customer), while if the latter router is chosen,  $\&$  obtains no utility (since it receives the traffic from a peer).

Hence,  $\&$  will turn ON iff  $\&$  can obtain more than  $5m$ -units of utility (which is what  $\&$  receives when it is OFF, from the Hold traffic). This occurs iff 1, 2 and 3 are all turned ON so that  $\&$  attracts  $2m+2m+2m = 6m$  units of And traffic.  $\square$

## K.5 The CHICKEN Gadget.

We present the CHICKEN GADGET, shown in Figure 21. CHICKEN GADGET emulates the famous two-player game of chicken. In this gadget, there are two nodes, 10 and 20, and some additional fixed nodes. CHICKEN GADGET is such that (1) whenever both 10 and 20 are ON, or both 10 and 20 are OFF, both nodes want to change to the other action; and (2) if one node in  $\{10, 20\}$  is ON, but the other is OFF, then both nodes are content with their actions (and hence the state of the network is stable).

Consider the CHICKEN GADGET construction, shown in Figure 21. Observe that the CHICKEN GADGET is asymmetrical, as node 20 is a provider of node 10. CHICKEN GADGET also consists of the fixed nodes 1, 2, 3, 4, 5, 6, destinations  $d_1, d_2$ , and four customer trees. Customer tree *Local 1* wants to reach destination  $d_1$ , customer

tree *Local 2* wants to reach destination  $d_2$ , customer tree *Cross 1* wants to reach destination  $d_2$ , and customer tree *Cross 2* wants to reach destination  $d_1$ . The fixed Nodes 1, 2, 4 and 5 are always OFF, and nodes 3, 6, 1000, 20 and all of the destinations and customer trees are always ON. We choose  $\epsilon$  and  $m$  so that  $\epsilon \ll m$ .

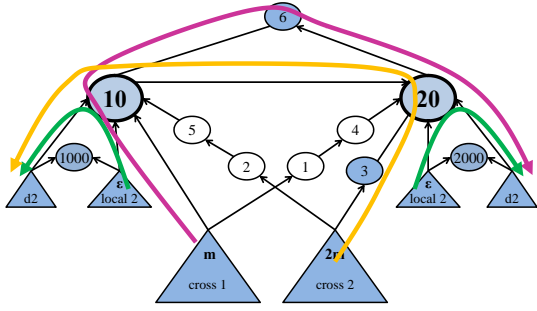
LEMMA K.4. CHICKEN GADGET is such that

1. if both 10 and 20 are ON then both nodes wish to be turned OFF;
2. if both 10 and 20 are OFF then both nodes wish to be turned ON;
3. if 10 is ON and 20 is OFF, then both nodes do not wish to select another action;
4. if 10 is OFF and 20 is ON, then both nodes do not wish to select another action.

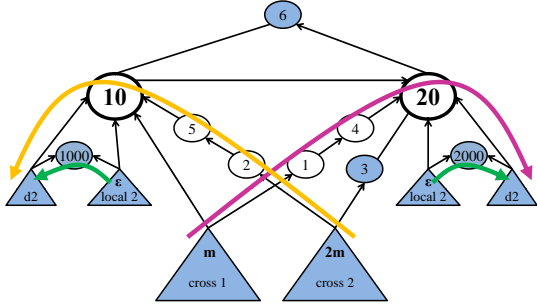
PROOF. Figure 21 depicts all four states of the CHICKEN GADGET. We make the following observations.

- **Local Traffic.** Traffic from *Local 1* has two available routes to  $d_1$ : a secure provider route (*Local 1*, 1000,  $d_1$ ) and an equal-length provider route through node 10 (*Local 1*, 10,  $d_1$ ). Observe that the former route is chosen if 10 is OFF, since in this case, it is the only secure route, (e.g., Figure 21(b), 21(d)). Observe also that the latter route is chosen if node 10 turns ON, since in this case both available routes are secure, and 10 has lower AS number than 1000 (e.g., Figure 21(a), 21(c)). Node 10's utility increases by a small amount,  $\epsilon$ -units, if the latter route is chosen. The same statement is true for *Local 2* and node 20.
- **Cross 1 Traffic.** Consider the  $m$ -units of traffic from *Cross 1* to  $d_2$ . To understand what routes are available to *Cross 1*, we first need to look at the route 10 chooses to get  $d_2$ ; namely, observe that 10 chooses a longer peer route (10, 6, 20,  $d_2$ ) over the shorter provider route (10, 20,  $d_2$ ) (see e.g., Figure 21(a)). Now we can see that *Cross 1* has two equidistant provider routes to  $d_2$ , route (*Cross 1*, 10, 6, 20,  $d_2$ ), and insecure route (*Cross 1*, 1, 4, 20,  $d_2$ ). We have two cases:

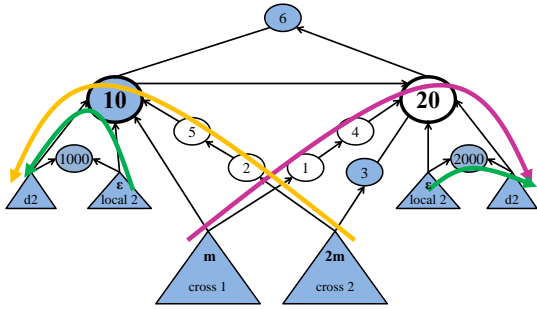
- If both nodes 10 and 20 are turned ON, the former route becomes secure, and is chosen by *Cross 2* (Figure 21(a)). In this case, the utility of node 10 increases by  $m$ -units, as it receives traffic from the directly connected *Cross 1* customer tree. Meanwhile, the utility of node 20 is unchanged, because 20 receives this traffic from its provider node 6.
- If at least one of nodes 10 and 20 are turned OFF, then the former route is *insecure*; thus, *Cross 1* will make its route choice based on AS



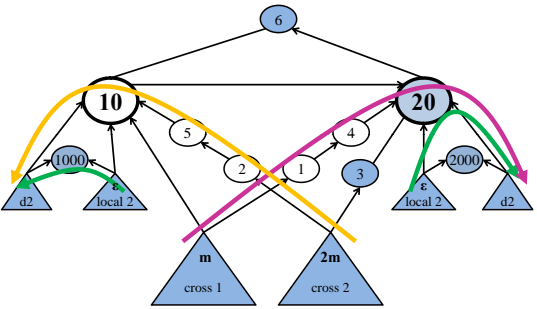
(a) (ON,ON) Undesirable cross traffic



(b) (OFF,OFF) Desirable cross traffic



(c) (ON,OFF) Desirable cross traffic



(d) (OFF,ON) Desirable cross traffic

**Figure 21: States of the Chicken Gadget.**

number, and choose the former route through node 1 (*e.g.*, Figure 21(b), 21(c), or 21(d)). In this case, node 10 receives no utility (because it is not on the route the traffic takes), while node 20 receives  $m$ -units of utility, as it receives this traffic from its customer 4.

- **Cross 2 Traffic.** Consider the  $2m$ -units of traffic from *Cross 2* to  $d_1$ . This time, we start by looking at the routes available from 20 to  $d_1$ ; namely, observe that 20 chooses the customer route (20, 10,  $d_1$ ) over the provider route (20, 6, 10,  $d_1$ ) (see *e.g.*, Figure 21(a)). Now, *Cross 2* has two equidistant provider routes to  $d_2$ , route (*Cross 2*, 3, 20, 10,  $d_1$ ), and insecure route (*Cross 2*, 2, 4, 10,  $d_1$ ). We again have two cases:

- If both routes 10 and 20 are turned ON, the former route becomes secure, and is chosen by *Cross 2* (Figure 21(a)). In this case, the utility of node 10 is unaffected (because it receives this traffic from its provider 20). The utility of node 20 is also unchanged (as it receives its peer 3).
- If at least one of nodes 10 and 20 are turned OFF, then the former route is *insecure*; thus, *Cross 2* will make its route choice based on AS number, and choose the former route through node 2 (*e.g.*, Figure 21(b), 21(c), or 21(d)). In this case, route 10 receives  $2m$ -units of utility (because it receives the traffic from its customer 5), while node 20 receives no utility (as it is not on the route taken by this traffic).

$\epsilon \ll m$  and so nodes 10 and 20 value Cross traffic significantly more than Local traffic. From the discussion above it is clear that when both 10 and 20 are turned ON, the Cross traffic provides them with utility  $(m, 0)$  (*i.e.*, 10 has utility  $m$ , 20 has utility 0). When at least one of these nodes is OFF, the Cross traffic provides them with utility  $(2m, m)$ . Clearly, the latter case is more desirable, and so we refer to this as *desirable Cross traffic*. The former case is called *undesirable cross traffic*.

Based Figure 21, and the discussion above, we can construct a bi-matrix for the nodes 10 and 20 in CHICKEN GADGET . The bi-matrix is shown in Figure 5. From the bi-matrix, it is clear that the CHICKEN GADGET has two stable states, where 10 and 20's actions are (ON, OFF) and (OFF, ON), and that it emulates (an asymmetric version of) the classic chicken game. Notice also that when the nodes are attracting desirable Cross traffic (*i.e.*, least one of them is turned OFF), each node obtains at least  $m - \epsilon$ -units more utility than when it attracts undesirable cross traffic (*i.e.*, both of them are ON).

□

	10 ON	10 OFF
20 ON	$m + \epsilon, \epsilon$	$2m, m + \epsilon$
20 OFF	$2m + \epsilon, m$	$2m, m$

Table 5: Chicken Gadget matrix. Assume  $m \gg \epsilon$ .

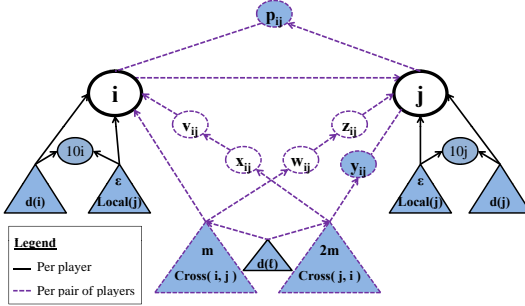


Figure 22: Chicken Gadget between nodes  $i < j$  of Selector Gadget

## K.6 The $k$ -SELECTOR Gadget.

The SELECTOR GADGET, shown in Figure 22, is a generalization of the CHICKEN GADGET. In the  $k$ -SELECTOR GADGET there are  $k$  nodes  $1, \dots, k$  and, for each node  $i \in \{1, \dots, k\}$ , there is an additional node  $10i$ , a destination  $d(i)$  and a customer tree  $Local(i)$  (as shown with solid lines in Figure 22).  $\forall i, j \in \{1, \dots, k\}$  such that  $i < j$ ,  $k$ -SELECTOR GADGET contains a CHICKEN GADGET that includes the nodes  $p_{ij}, x_{ij}, y_{ij}, v_{ij}, w_{ij}, z_{ij}$  and  $m$ -units of cross traffic from each of the  $(k-j)$  CHICKEN GADGET with nodes  $j < i$ . Thus, each node  $j$  has utility  $m(k-1) + m(j-1)$ . However, this state is unstable; if a node in  $\{1, \dots, k\}$  turns ON unilaterally, it increases its utility by  $\epsilon$ -units by attracting local traffic while still retaining desirable cross traffic.

Intuitively,  $k$ -SELECTOR GADGET generalizes CHICKEN GADGET in that all nodes are incentivized to be at a state in which exactly a single node in  $1, \dots, k$  is ON and all other nodes in  $1, \dots, k$  are OFF.

LEMMA K.5.  $k$ -SELECTOR GADGET is such that

1. there are  $k$  stable states, each where one of the nodes in  $\{1, \dots, k\}$  is ON and all other nodes in  $\{1, \dots, k\}$  are OFF.
2. in states in which more than one node in  $\{1, \dots, k\}$  is ON, all nodes wish to turn OFF.

PROOF. To construct the SELECTOR GADGET, we use a clique of CHICKEN GADGET. Figure 22 shows the CHICKEN GADGET that is used as building block in the SELECTOR GADGET. It is easy to see that this gadget works identically to that of Figure 21, if we ensure that:

- Node  $x_{ij}$  has lower AS number than node  $y_{ij}$ .
- Node  $w_{ij}$  has lower AS number than node  $i$ .

Before proving the correctness of the lemma, we discuss how to handle non-designated traffic. We want to ensure that traffic from one CHICKEN GADGET does not flow into another CHICKEN GADGET, e.g., that traffic from  $Cross(i, j)$  to some destination  $d_\ell$ ,  $\ell \neq j, k$ , does not flow through the gadgets. We have a simple fix; simply

connect the offending pair (e.g.,  $Cross(i, j)$  to  $d_\ell$ ) with a peer-to-peer edge. Clearly, doing this cannot introduce new customer-provider loops into the network; furthermore, this peer route will be preferred by the pair over the provider route through the nodes in the gadget. Finally, this ensures that customer trees and destination nodes may only have peers and providers; as such they cannot be used to transit other traffic (i.e., one of the offending pair, e.g.,  $d_\ell$ , cannot carry traffic from some node or customer tree  $a$  to some other node or customer tree  $b$ ) and potentially create problems for our gadgets.<sup>4</sup>

We say the the SELECTOR GADGET is set to state  $\ell \in \{1, \dots, k\}$ , if only node  $\ell$  is turned ON, and all other nodes in  $\{1, \dots, k\}$  are turned OFF. The basic idea of the proof is that each node wants to retain desirable cross traffic at all costs; thus, if more than one node turns ON, pairs of these ON nodes will be connected by a CHICKEN GADGET in which they lose the desirable cross traffic, and thus want to turn OFF. We now argue that each state  $\ell$  is stable, and all other states are unstable. We handle three cases:

- **All nodes are OFF.** Here, every CHICKEN GADGET is in the (OFF, OFF) state, so that every node in  $\{1, \dots, k\}$  receives desirable cross traffic from each CHICKEN GADGET. That is, each node  $j \in \{1, \dots, k\}$  gets  $2m$ -units of cross traffic from each of the  $(j-1)$  CHICKEN GADGET with nodes  $j < i$ . Thus, each node  $j$  has utility  $m(k-1) + m(j-1)$ . However, this state is unstable; if a node in  $\{1, \dots, k\}$  turns ON unilaterally, it increases its utility by  $\epsilon$ -units by attracting local traffic while still retaining desirable cross traffic.
- **Only node  $\ell$  is ON.** In this case, there is at least one OFF node in each CHICKEN GADGET, so all nodes in  $\{1, \dots, k\}$  receive desirable cross traffic. The utility of all nodes in  $\{1, \dots, k\}$  except  $\ell$  remains as in the ‘All nodes OFF’ state, except node

<sup>4</sup>We observe that our construction does not introduce any customer-provider loops (thus does not violate the Gao-Rexford conditions [13]). Since the CHICKEN GADGET is inherently asymmetrical, we need to show that the connecting CHICKEN GADGET between every pair nodes  $(i, j)$  does not create customer-provider loops that violate GR1. First, each individual CHICKEN GADGET has no customer-provider loops. Thus, it remains to consider only customer-provider loops involving interconnections between nodes  $i, j, p_{ij}$  in different chicken gadgets. Since each  $p_{ij}$  only has two edges, one of which is a peer-peer edge, it cannot be involved in any customer-provider loops. This, we need only concern ourselves with the customer-provider edges between  $i$  and  $j$ . However, observe that for every two nodes  $i < j$  that appear in a CHICKEN GADGET, node  $i$  is always a customer of node  $j$ . Thus, the existence of a customer-provider loop implies that there must be some nodes  $i \neq j$  such that both  $i > j$  and  $i < j$  (—a contradiction!).

$\ell$  that obtains additional local traffic, slightly increasing its utility to  $\epsilon + m(k - 1) + m(\ell - 1)$ .

- **More than one node is ON.** Here, there is at least one pair of nodes  $(i, j)$  in  $\{1, \dots, k\}$  connected via CHICKEN GADGET in the (ON,ON) state, thus receiving undesirable cross traffic. This is an unstable state, since the utility of every such  $i$  and  $j$  will increase by at least  $m - \epsilon$  if they turn OFF.

□

## K.7 The TRANSITION Gadget.

Our next building block is the  $k$ -TRANSITION GADGET, shown in Figure 23. Intuitively, TRANSITION GADGET resets an  $k$ -SELECTOR GADGET from the stable state  $i$  (all nodes in  $\{1, \dots, k\}$  are OFF except node  $i$ ) to the stable state  $j$ . We do not use the TRANSITION GADGET directly in our reduction, but a modified version of this gadget, called a TRIPLE TRANSITION GADGET. For expository purposes, we start with a detailed explanation of the TRANSITION GADGET.

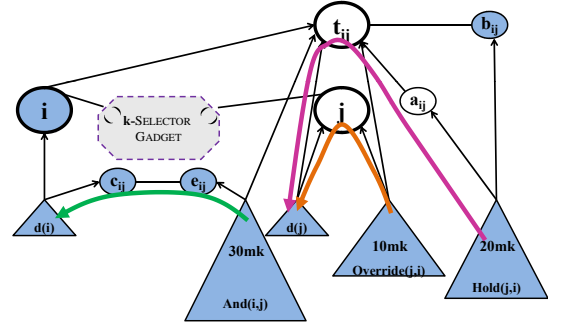
The TRANSITION GADGET shown in Figure 23 has three non-fixed nodes: the two selector nodes  $i, j$ , that are attached to an  $k$ -SELECTOR GADGET, and a selector-transition node  $t_{ij}$ . In addition,  $k$ -SELECTOR GADGET includes the new fixed nodes  $a_{ij}, b_{ij}, c_{ij}, e_{ij}$ , such that  $t_{ij}$  has lower AS number than  $e_{ij}$ , and  $a_{ij}$  has lower AS number than  $b_{ij}$ . Finally, there are three new customer trees. *Override*( $i, j$ ), and *Hold*( $x_i, j$ ), each wanting to reach destination  $d_j$ , and *And*( $i, j$ ), that wants to reach destination  $d_i$ . All destinations, customer trees, and other fixed nodes except  $a_{ij}$  are ON; only the fixed node  $a_{ij}$  is OFF.

The TRANSITION GADGET guarantees the following.

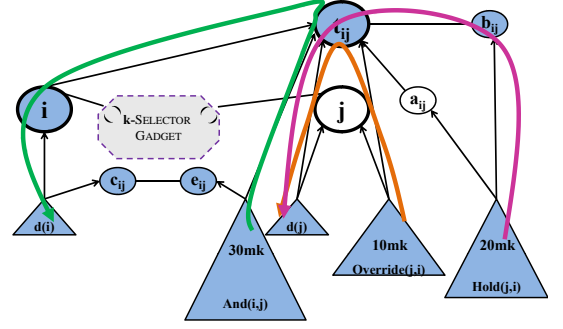
LEMMA K.6. *When the  $k$ -TRANSITION GADGET is at stable state  $i$ , the network evolves to stable state  $j$  within a constant number of time steps.*

PROOF. We start by discussing the routes available for each relevant flow of traffic in the TRANSITION GADGET:

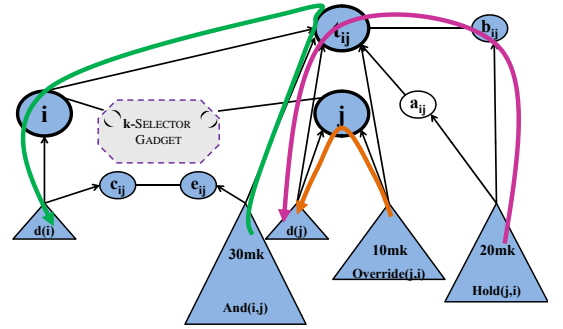
- **And Traffic.** Traffic from *And*( $i, j$ ) to  $d_i$  has a secure provider route (*And*( $i, j$ ),  $e_{ij}, c_{ij}, d_i$ ) and another route (*And*( $i, j$ ),  $t_{ij}, i, d_i$ ). Since  $t_{ij}$  has lower AS number than  $e_{ij}$ , the latter route is chosen iff it is secure, *i.e.*, both  $t_{ij}$  and  $i$  are ON. When the former route is used, neither the selector-transition node nor the selector node obtain any utility (since they are not on the route). When the latter route is used, the selector-transition node  $t_{ij}$  obtains  $30mk$ -units of utility (since it receives *And* traffic along a provider edge), and the selector node  $i$  obtains no utility (since it receives the *And* traffic from a provider).



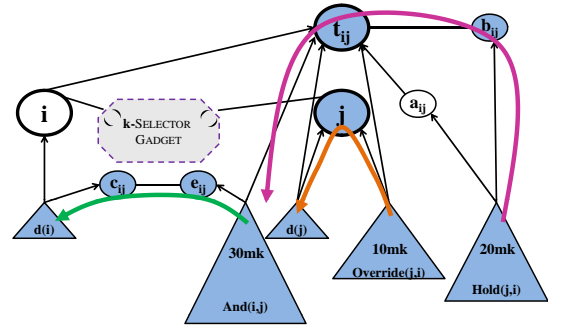
(a) (ON,OFF,OFF)  $t_{ij}$  unstable.



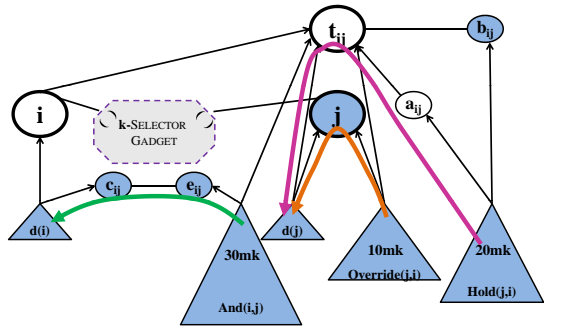
(b) (ON,OFF,ON)  $j$  unstable.



(c) (ON,ON,ON)  $i$  unstable.



(d) (OFF,ON,ON)  $t_{ij}$  unstable.



(e) (OFF,ON,OFF) All nodes stable.

Figure 23: States of the Transition Gadget .

- **Override Traffic.** Traffic from  $Override(i,j)$  to  $d_j$  has provider route  $(Override(i,j), j, d_j)$  and  $(Override(i,j), t_{ij}, d_j)$ . Since  $j$  has lower AS number than  $t_{ij}$ , the latter route is chosen by  $Override(i,j)$  iff selector-transition node  $t_{ij}$  is ON and selector node  $j$  is OFF. When the former route is used, selector node  $j$  obtains  $10mk$  units of utility; note that this is approximately an order of magnitude more utility than  $j$  can ever hope to obtain from the SELECTOR GADGET (losing this utility will cause  $j$  to override the SELECTOR GADGET). Similarly,  $t_{ij}$  receives utility  $10mk$  when the latter route is chosen.
- **Hold Traffic.** Traffic from  $Hold(i,j)$  to  $t_{ij}$  has provider route  $(Hold(i,j), a_{ij}, t_{ij}, d_j)$  and  $(Hold(i,j), b_{ij}, t_{ij}, d_j)$ . Since  $a_{ij}$  has lower AS number than  $b_{ij}$ , the latter route is chosen by  $Hold(i,j)$  iff it is secure, i.e., the selector-transition node  $t_{ij}$  is ON. When the former route is used, selector-transition node  $t_{ij}$  obtains  $20mk$  units of utility, since it receives Hold traffic from its customer. When the latter route is used,  $t_{ij}$  receives this traffic from its peer, and thus obtains no utility.

Notice also that the utility of selector node  $i$  is not affected by *any* of the customer tress in the TRANSITION GADGET, so that its behavior will be completely dominated by the SELECTOR GADGET. Meanwhile, selector node  $j$  is only affected by the TRANSITION GADGET when  $j$  loses Override traffic (when  $j$  is OFF and  $t_{ij}$  is ON). Since the utility obtained from Override traffic dominates the utility obtained from the SELECTOR GADGET,  $j$  has an incentive to turn ON regardless of the state of the SELECTOR GADGET. Finally,  $t_{ij}$ 's behavior is completely determined by the state of the TRANSITION GADGET.

The idea of the TRANSITION GADGET, is that if the SELECTOR GADGET is set to a particular state (i.e., state  $i$ ), the selector-transition node turns ON, and causes a reset of the SELECTOR GADGET. We start with the following proposition, that argues that the TRANSITION GADGET will not turn on “by mistake”.

PROPOSITION K.7. *Node  $t_{ij}$  turns ON iff node  $i$  is ON.*

PROOF. This follows straight-forwardly from our discussion of routes and utilities above. If node  $i$  is ON,  $t_{ij}$  turns ON to attract the And traffic for  $30mk$ -units of utility, which exceed the utility  $t_{ij}$  could obtain by remaining OFF (namely,  $20mk$ -units of utility from the Hold traffic). For the other direction, suppose node  $i$  is OFF, and so  $t_{ij}$  cannot attract the And traffic. Then, if  $t_{ij}$  is OFF, it obtains  $20mk$ -units of utility from the Hold traffic, which exceeds the maximum utility  $t_{ij}$  can obtain by turning ON (namely,  $10mk$  of utility from the Override traffic).  $\square$

Next, we show how the TRANSITION GADGET progresses from a state where only  $i$  is ON, to a state where only  $j$  is ON. To do this, we present Figure 23, which depicts each of the five phases of the TRANSITION GADGET, and Table 6 presents the utility of each non-fixed node in each phase of the TRANSITION GADGET. Figure 23 and Table 6 follow from the discussion of routes and utility above, so we omit a detailed arguments of their correctness. Note that the node that wishes to change its action in each stage is shown in bold in Table 6.

TRANSITION GADGET is a three-(non-fixed)-node gadget, and so the proof follows by from arguing the node that moves after each phase has an incentive to change its action, and turn ON or OFF, while the node that does *not* move has no such incentive. (Note that the former holds also by inspection of Table 6). We do this now:

- 23(a) Initial (ON,OFF,OFF) stage.** Here  $t_{ij}$  wishes to change its action; by turning ON, it can attract both the And and the Override traffic, which is of higher utility than the Hold traffic it attracts by staying OFF. Also,  $j$  is does not wish to change its action; this follows from the fact that  $j$  attracts Override traffic regardless of its state; thus, the properties of the SELECTOR GADGET maintain  $j$  in the OFF state.
- 23(b) (ON,OFF,ON) stage.** Here  $j$  is wishes to change its action; in this state it has lost the Override traffic; thus  $j$  wants to turn ON to get it back (even though doing this will take the SELECTOR GADGET out of a selector stable state). Also,  $i$  does not wish to change its action; since  $i$ 's utility is not affected by the trafxfic flows in the TRANSITION GADGET, the properties of the SELECTOR GADGET maintain  $i$  in the ON state.
- 23(c) (ON,ON,ON) stage.** Here  $i$  wishes to change its action;  $i$ 's behavior is completely determined by the SELECTOR GADGET, which is currently *not* in a stable state. From the properties of the SELECTOR GADGET,  $i$  has an incentive to turn OFF. Also,  $t_{ij}$  does not wish to change its action; even though  $t_{ij}$  is only attracting And traffic (thus obtaining  $30mk$ -units of utility), this is still more than the utility it would obtain by turning OFF and attracting only Hold traffic (for  $20mk$ -units of utility).
- 23(c) (OFF,ON,ON) stage.** Here  $t_{ij}$  wishes to change its action;  $t_{ij}$  no longer attracts the And traffic (since  $i$  is off) and it is losing the Hold traffic as well (because  $t_{ij}$  is ON). Thus,  $t_{ij}$  has an incentive to turn OFF in order to attract the Hold traffic. Also,  $j$  is does not wis to change its action; it is in a selector stable state and is attracting Override traffic as well.

Figure	state			utility of $i$	utility of $j$		utility of $t_{ij}$
	$i$	$j$	$t_{ij}$	SELECTOR	SELECTOR	TRANSITION	TRANSITION
Fig. 23(a)	ON	OFF	OFF	$m(k+i-2) + \epsilon$	$m(k+j-2)$	$+10mk$	<b>20mk</b>
Fig. 23(b)	ON	OFF	ON	$m(k+i-2) + \epsilon$	$m(k+j-2)$	<b>+0</b>	$40mk$
Fig. 23(c)	ON	ON	ON	$\mathbf{m(k+i-3)} + \epsilon$	$m(k+j-3) + \epsilon$	$+10mk$	$30mk$
Fig. 23(d)	OFF	ON	ON	$m(k+i-2)$	$m(k+j-2) + \epsilon$	$+10mk$	<b>0</b>
Fig. 23(e)	OFF	ON	OFF	$m(k+i-2)$	$m(k+j-2) + \epsilon$	$+10mk$	$20mk$

Table 6: States of the Transition Gadget .

**23(e)** (*OFF, ON, ON*) stage. We have finally arrived at a stable state; the SELECTOR GADGET is in the selector stable state  $j$ , so  $i$  does not wish to change its action. Furthermore,  $j$  is attracting its Override traffic, so  $j$  does not wish to change its action as well. Finally,  $t_{ij}$  does not wish to change its action; by staying OFF,  $t_{ij}$  can attract only Hold traffic (thus obtaining  $20mk$ -units of utility), but this is still more than the utility it would obtain by turning ON and attracting only Override traffic (for only  $10mk$ -units of utility).

We can now derive the following proposition from the discussion above.

PROPOSITION K.8. *If selector-transition node  $t_{ij}$  turns ON, then the  $r$ -selector is set to state  $j$ .*

Combining Propositions K.7-K.8, it is clear that if  $i \neq j$ , the  $i \rightarrow j$   $r$ -TRANSITION GADGET can reset the  $r$ -SELECTOR GADGET from state  $i$  to state  $j$ .

## K.8 Combining Multiple TRANSITION Gadgets for a Given SELECTOR.

Thus far, we have argued only about a single  $i \rightarrow j$  TRANSITION GADGET (for given  $i, j$ ); we now consider arbitrarily resetting the SELECTOR GADGET from any state to any other state. We present the following proposition.

PROPOSITION K.9. *If we attach a single  $k$ -TRANSITION GADGET from each node  $i \in \{1, \dots, k\}$  to each other node  $j \neq i$  in an  $k$ -SELECTOR GADGET, then each  $i \rightarrow j$  TRANSITION GADGET can move the SELECTOR GADGET from selector stable state  $i$  to the selector stable state  $j$ .*

PROOF. Figure 24 shows how the TRANSITION GADGET is attached to arbitrary nodes  $i$  and  $j$  of the SELECTOR GADGET. It is easy to see that adding TRANSITION GADGET to the SELECTOR GADGET does not introduce any customer-provider loops; individual TRANSITION GADGET have no customer-provider loops, and since each TRANSITION GADGET has its own independent selector-transition node, that may only be directly connected a single selector node (*i.e.*, node  $i$  in Figure 24), adding multiple TRANSITION GADGET to a SELECTOR GADGET cannot introduce customer-provider

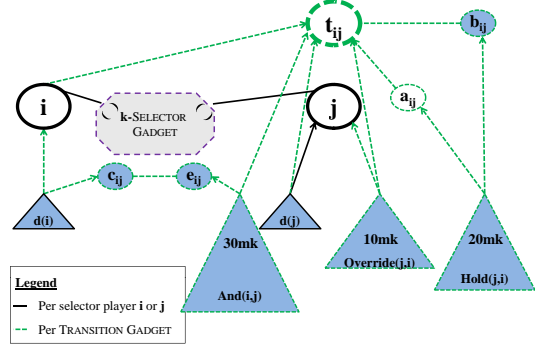


Figure 24: Transition Gadget between nodes  $i, j$  of Selector Gadget

loops. Suppose the selector is set to state  $i$ . Each selector node is attached to its own, independent selector-transition node, and thus by Proposition K.7, only the selector-transition node attached to node  $i$  will turn ON. Finally, by Proposition K.8, we have that the SELECTOR GADGET will be reset to state  $j_i$ .  $\square$

## K.9 TRIPLE TRANSITION GADGET .

The final element of our reduction requires us to be able to simulatively reset *three* independent SELECTOR GADGET from state  $\langle i_1, i_2, i_3 \rangle$  to states  $\langle j_1, j_2, j_3 \rangle$ .

The TRIPLE TRANSITION GADGET ‘AND’s together three TRANSITION GADGET, using a similar trick to that used in the AND GADGET. The idea is to connect three SELECTOR GADGET to a single selector-transition node, and to ensure that the selector-transition node turns ON (*i.e.*, moves from the first phase of the TRANSITION GADGET, (ON, OFF, OFF) in Figure 23(a)), to the second stage ((ON, OFF, ON) in Figure 23(c)) iff all three of the SELECTOR GADGET are set to the correct states  $\langle i_1, i_2, i_3 \rangle$ .

### K.9.1 Constructing the TRIPLE TRANSITION GADGET .

Figure 26 presents an overview of the TRIPLE TRANSITION GADGET. We have three independent SELECTOR GADGET that *are not* considered part of the TRIPLE TRANSITION GADGET; a  $k_1$ -SELECTOR GADGET, a  $k_2$ -SELECTOR GADGET, and a  $k_3$ -SELECTOR GADGET

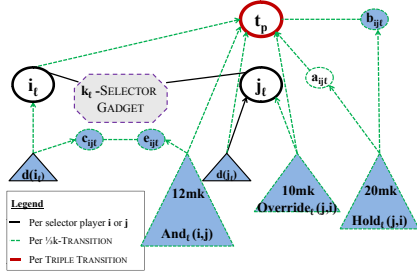


Figure 25:  $\frac{1}{3}k$ -Transition Gadget

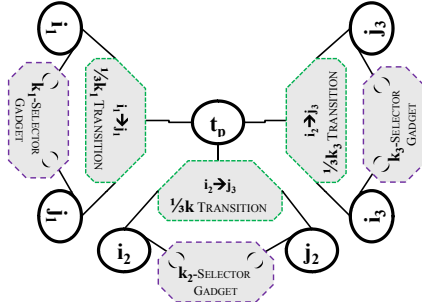


Figure 26: Schematic of Triple Transition Gadget

. Each SELECTOR GADGET has its respective selector nodes,  $i_1, j_1, i_2, j_2, i_3, j_3$ . The selector nodes associated with each SELECTOR GADGET are connected to the single selector-transition node,  $t_p$  via a modified TRANSITION GADGET, which we call a  $\frac{1}{3}k$ -TRANSITION GADGET and depict in Figure 25. As shown in Figure 25, the  $\frac{1}{3}k$ -TRANSITION GADGET is identical to the TRANSITION GADGET, with the modification that the And traffic from a single  $\frac{1}{3}k$ -TRANSITION GADGET is not sufficient to turn ON the selector-transition node  $t_p$ .

As usual, we use standard tricks to ensure that interconnecting the gadgets this way does not introduce new traffic flows through the gadgets that affect nodes' actions. Notice also that interconnecting the gadgets this way cannot introduce any new customer-provider loops; this follows from the fact that the individual gadgets have no customer provider loops, and are interconnected *solely* through the selector-transition node  $t_p$  (see Figure 26), and the selector-transition node is always a provider or a peer of every node in the gadget (see Figure 25).

### K.9.2 Why it works.

The key to why the gadget works lies in the fact that the And traffic from each  $\frac{1}{3}k$ -TRANSITION GADGET now provides only  $12mk$ -units of utility, so that **all** three selector nodes  $i_1, i_2, i_3$  must be ON before the selector-transition node  $t(i_1, i_2, i_3)$  turns ON:

LEMMA K.10. *Node  $t(i_1, i_2, i_3)$  turns ON iff nodes  $i_1, i_2, i_3$  are all ON.*

PROOF. Recall that traffic  $And(i_1, j_1)$  provides utility to the transition node  $t(i_1, i_2, i_3)$  iff both  $i_1$  and  $t(i_1, i_2, i_3)$  are on. We start with the reverse direction. If only two of selector nodes ( $i_1, i_2, i_3$ ) are ON, the selector-transition node can receive at most  $(12mk + 10mk) + (12mk + 10mk) + 10mk = 54mk$ -units of utility by turning ON (attracting And traffic from two  $\frac{1}{3}k$ -TRANSITION GADGET, and Override traffic from each of the three  $\frac{1}{3}k$ -TRANSITION GADGET). Meanwhile, if the  $t_p$  is OFF, it obtains least  $20mk * 3 = 60mk$ -units of utility by attracting Hold traffic from each of the  $\frac{1}{3}k$ -TRANSITION GADGET, so in this case the selector-transition node has an incentive to remain OFF. For the forward direction, if all three selector nodes ( $i_1, i_2, i_3$ ) are ON, the selector-transition node earns utility  $(12mk + 10mk) * 3 = 66mk$  by turning ON, which exceeds the  $60mk$ -units of utility the selector-transition node could obtain by remaining OFF.  $\square$

An argument similar to that of Proposition K.8 gives us:

LEMMA K.11. *If  $i_1 \neq j_1, i_2 \neq j_2, i_3 \neq j_3$ , then if  $t(i_1, i_2, i_3)$  turns ON, then nodes  $j_1, j_2, j_3$  all turn ON.*

### K.9.3 What happens if $i_\ell = j_\ell$ ?

Our reduction may also require that we reset the three SELECTOR GADGET from states  $\langle i_1, i_2, i_3 \rangle$  to state  $\langle j_1, j_2, j_3 \rangle$ , where some  $i_\ell = j_\ell$ . Modifying the TRIPLE TRANSITION GADGET to do this is simple; we need only remove the Override traffic from the appropriate  $\frac{1}{3}k$ -TRANSITION GADGET, and reduce the Hold traffic to  $10mk$  (see Figure 25). It is not hard to see that this will ensure that the selector-transition node turns ON iff the three SELECTOR GADGET are set to  $\langle i_1, i_2, i_3 \rangle$  (i.e., analogous to Lemma K.10), without requiring a reset of the  $\ell^{th}$  SELECTOR GADGET gadget. Thus, by applying this modification to the TRIPLE TRANSITION GADGET whenever some  $i_\ell = j_\ell$ , we have the following, stronger version of Lemma K.11:

LEMMA K.12. *If  $t(i_1, i_2, i_3)$  turns ON, then nodes  $j_1, j_2, j_3$  all turn ON.*

### K.9.4 Multiple TRIPLE TRANSITION GADGET.

Finally, we argue that we can have multiple TRIPLE TRANSITION GADGET that can simultaneously reset three selectors from any state to any other state.

LEMMA K.13. *Suppose we have at least three SELECTOR GADGET, and consider triplets of selector nodes  $(i_1, i_2, i_3)$ , where each selector node belongs to a different  $k_\ell$ -SELECTOR GADGET, and  $k_\ell \leq k$ . Then, for every such triplet of selector node  $(i_1, i_2, i_3)$ , we can*

attach a single  $k$ -TRIPLE TRANSITION GADGET that moves each of the three  $k_\ell$ -SELECTOR GADGET from state  $i_\ell$  to state  $j_\ell$ .

PROOF. Each triplet of selector nodes is attached to a unique selector-transition node  $t(i_1, i_2, i_3)$ , and we use standard tricks to ensure that non-designated does not flow into the gadgets and affect nodes' decisions. Thus, we can apply Lemma K.10 and Lemma K.12 in argument that is analogous to the one used to prove Proposition K.9.  $\square$

## K.10 Concluding the proof.

We are now ready to present our reduction. Given an input to STATIC-MODE, we construct a network in S\*BGP ADOPTION. Recall that the set of nodes in S\*BGP ADOPTION contains of  $r$  head nodes,  $q$  machine-state nodes, and  $k$  cell clusters, each containing  $\gamma$  symbol nodes. We construct a  $k$ -SELECTOR GADGET for the head nodes, a  $q$ -SELECTOR GADGET for the machine-nodes, a  $\gamma$ -SELECTOR GADGET for the symbol nodes in each of the cell clusters. Intuitively, this guarantees that, throughout the evolution of the network, in each of these subsets of nodes, only a single node is ON.

We construct, for every cell cluster  $C_i$ , a TRIPLE TRANSITION GADGET that connects the SELECTOR GADGET for the head nodes, the SELECTOR GADGET for the machine-state nodes and the selector for the specific cell cluster  $C_i$ . The TRIPLE TRANSITION GADGET guarantees the transition between triplets of stable states for the selectors as follows. When the head node  $h_i$  (that is, the head node that corresponds to the cell cluster  $C_i$ ) is ON, a machine-state node  $s_\beta$  is ON, and a symbol node (in  $C_i$ )  $g_{\beta,\sigma}$  is ON, the TRIPLE TRANSITION GADGET makes the three SELECTOR GADGET transition to a state where the head node ON, the machine-state node ON, and the symbol node in  $C_i$  ON are the nodes corresponding to the outcome of  $\delta(i, \beta, \sigma)$  in the Turing machine  $M$ . Thus, the different TRIPLE TRANSITION GADGET capture the transitions prescribed by the Turing machine's transition function.

Recall that a state of the network in S\*BGP ADOPTION is "clean" if only a single head node  $h$  is using S-BGP (is ON), only a single machine-state node  $s$  is ON, and only a single node  $g_i$  in every cell-cluster  $C_i$  is ON. Observe that every clean state in S\*BGP ADOPTION captures a configuration of the Turing machine  $M$  is STATIC-MODE (in which the index of the single head node using S-BGP is the location of  $M$ 's head, and so on). We make the following observation that immediately follows from our construction:

OBSERVATION K.14. *A configuration  $c$  in STATIC-MODE is static iff the clean state of the network in S\*BGP ADOPT that corresponds to  $c$  is stable.*

We set the initial state of the network in S\*BGP ADOPTION to be the clean state in which the single head node ON is  $h_1$ ; the single state node ON is the node that represents  $q_0$ ; and in each cell cluster  $C_i$ , the single symbol node ON is  $g_{i,x_i}$ . Observe that this clean state in S\*BGP ADOPTION captures the initial configuration of  $M$  (the head is pointing to the first cell, that contains  $x_1$ , and the machine-state is the initial machine-state  $q_0$ ).

The correctness of the reduction follows from the following observation about our construction.

OBSERVATION K.15. *From every clean state  $\alpha$  in S\*BGP ADOPTION that represents a configuration  $c = (h, s, g)$  in STATIC-MODE, the network evolves (within a constant number of time steps) to another clean state that represents the configuration that immediately follows  $c$  in STATIC-MODE (after applying  $\delta$ ).*

Thus, we show that when starting at an initial state of the network as described above, the network evolution essentially mimics the execution of  $M$  for the input string  $x$ . Theorem K.1 follows.