Modeling on Quicksand

Dealing with the Scarcity of Ground Truth in Interdomain Routing Data

Phillipa Gill
University of Toronto

Michael Schapira
Hebrew University of
Jerusalem and Google NYC

Sharon Goldberg
Boston University

Abstract

Researchers studying the interdomain routing system, its properties and new protocols, face many challenges in performing realistic evaluations and simulations. Modeling decisions with respect to AS-level topology, routing policies and traffic matrices are complicated by a scarcity of ground truth for each of these components. Moreover, scalability issues arise when attempting to simulate over large (although still incomplete) empirically-derived AS-level topologies. In this paper, we discuss our approach for analyzing the robustness of our results to incomplete empirical data. We do this by (1) developing fast simulation algorithms that enable us to (2) running multiple simulations with varied parameters that test the sensitivity of our research results.

Categories and Subject Descriptors: C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms: Algorithms, Experimentation.

1. INTRODUCTION

In measurement-based networking research, we often find that "what we can measure in an Internet-like environment is typically not the same as what we really want to measure" [30]. This is especially true in the context of interdomain routing. Indeed, many papers have exposed the myriad issues at play when studying the interdomain routing system *e.g.,* [18, 28]; while these works provide an important critique of existing data sets and techniques, they tend to highlight limitations rather than offer solutions. Here we take a more constructive approach; we use our work on the deployment dynamics of secure interdomain routing protocols (S*BGP protocols) [10] as a case study, and show how we demonstrated the robustness of the results we obtained via modeling and simulation over imperfect empirical data.

Data imperfections mean our approach is useful for identifying *trends*, rather than providing accurate *predictions*. Thus, demonstrating robustness meant showing that the trends we identified were insensitive to changes in parameters that are known to be incompletely captured by the data. Thus, this paper makes two contributions; first, we present our approach for performing robustness analysis, and second, we present the fast simulation algorithms that enabled the multiple simulations we needed for robustness analysis.

1.1 Simulation approaches.

A number of studies have relied on modeling and simulation of the interdomain routing system, ranging from work on network reliability [15, 32], to BGP convergence [26], to geopolitical control of network traffic [17], to secure routing [2, 3, 10]. Researchers commonly model the interdomain routing system as a graph $G = (V, E)$ where the vertices are autonomous systems (ASes), and edges indicate physical connections between ASes [13]. Simulations are used to determine how ASes select routes for traffic using BGP (the Border Gateway Protocol, the de facto routing protocol in the global Internet). Because the AS graph is quite large (currently, it is thought to contain 36K vertices, and 130K edges [5,6]), algorithms that run over the full AS graph quickly become computationally intensive. Thus, in designing simulations, researchers must contend with (a) the fact the interdomain routing system does *not* use shortest-path routing, and (b) scalability issues. Below we discuss a few approaches for overcoming these challenges.

Custom BGP simulators. Because ASes are controlled by autonomous, economically-motivated entities, their routing decisions often incorporate considerations (*e.g.,* economics, geography) beyond path length. Thus, standard shortest-path graph algorithms do not work 'out of the box'; instead, researchers have developed custom simulators that compute paths based on routing policy decisions of individual ASes.

Some BGP simulators are designed to allow network operators to answer "what-if" questions about routing configurations within a single AS [8, 27], or simulate BGP at the packet level [7, 26]. The faithful modeling provided by these simulators comes at a high computational cost, making them ill-suited to simulations over a large number of ASes. In contrast, BSIM [16] and BGPSIM [31] abstract away intradomain and packet-level considerations in favor of modeling BGP on an AS graph. While considerably more scalable than previous work, the computational overhead of these simulators remains high because they use discrete event simulation. As such, BGPSIM [31] must be run on a supercomputer, while BSIM [16] can only run a limited number of simulations over the entire AS graph.

Scaling down the AS-level topology. Alternatively, scalability issues may be mitigated by scaling down the AS graph itself. For example, one can use AS-graph generation tools like GT-ITM [34], BRITE [22], or Inet [14] to create a smaller synthetic AS graph, or use the techniques in [19] to scale down an empirically-derived AS graph (*e.g.,* as provided by CAIDA [6] or UCLA [5]) to a smaller set of ASes.

.

However, these approaches come with the risk that scaling has modified or even destroyed important graph properties. The authors of [19] show that their scaling techniques preserve several important graph-theoretic properties (*e.g.,*

degree distribution, spectral properties). However, the AS graph contains structures which may not be captured by matching standard graph-theoretic properties. For example, in [10] our results relied heavily on the prevalence of a subgraph (the "Diamond" in Figure 5 of [10]) that is difficult to describe using standard properties. Moreover, the risk is compounded because *a priori* it not always clear which graph properties will be important for a particular study; this often becomes clear only *after* the study is complete.

Our simulator. To avoid potential inaccuracies that might result from scaling down the AS-level topology, we advocate running simulations on full empirical AS graphs (*e.g.,* [5,6]). To support running repeated simulations over the full AS graph, we developed lightweight simulation algorithms [10, 12] and parallelized them across a compute cluster running DryadLINQ [33]. Our approach drastically reduces computational overhead by *algorithmically* computing paths chosen by BGP. Moreover, to provide the speedup required to run simulations over the entire AS graph, our simulator (a) abstracts away intradomain details, (b) assumes that all ASes use the routing policies proposed in [9].

1.2 Scarcity of ground truth.

Studies of interdomain routing require models of the AS-level topology, routing policies, and sometimes also traffic matrices. Unfortunately, ground-truth data is scarce. Empirically-derived AS level topologies [5,6] are known to be incomplete [23]. Furthermore, inter-domain routing policies are often kept private and are known to differ across ISPs. (However, models of these routing policies have been developed [9] and are widely used by researchers). Finally, data on inter-domain traffic flows [4] remains elusive (*e.g.,* because traffic volumes differ between vantage points in the network) and ever-changing (*e.g.,* the emergence of "hyper giant" ASes that source large volumes of traffic [20]).

While models and inferred AS level topologies serve as a useful starting point for studying the inter-domain routing system, they can only take us so far. In the remainder of the paper, we discuss the robustness analysis we used in support of our results in [10]. We start with a model of BGP (Section 2) and then discuss the scaling challenges faced in our simulation studies (Section 3) and the algorithms and techniques we use to overcome them (Sections 4 - 5). We then discuss (Section 6) how these algorithms enabled us to affirmatively answer the question: "Do the available measurements and their analysis and modeling efforts support the claims that are being made [in the paper at hand]" [18].

2. MODELING BGP

We now overview aspects of the models that we used in [10, 12] (and that have appeared in many other papers, see references in [12]). Full details are in the original papers.

2.1 AS graph and entities.

The AS graph. The AS-level topology of the Internet is modeled as graph $G = (V, E)$ where vertices represent ASes and edges represent connections between them. Each edge is annotated with the standard model for business relationships in the Internet [9]: *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. We distinguish three types of ASes:

Content providers (CPs) are ASes that generate revenue by providing popular content to as many users as possible.

Stubs are ASes that have no customers of their own and are not CPs.

ISPs. The remaining ASes in the graph (that are not stubs or CPs) are ISPs. They earn revenue by transiting Internet traffic.

2.2 Routing policies.

ASes design routing policies subject to their own individual cost and performance constraints. Since routing policies are considered sensitive information, we use a standard model of routing policies based on [9,13], and used in [10,12] and many other works.

We follow [13] 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 paths it learns from its 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 paths with highest local preference, prefer shortest paths (*i.e.,* fewest AS hops).

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.[1]

This standard model of local preference [9] captures the idea that an AS has incentives to prefer routing through a customer (that pays it) over a peer (no money is exchanged) over a provider (that it must pay).

Export Policies. This standard model of export policies captures the idea that an AS should only be willing to load its network with transit traffic if its customer pays it to do so [9]:

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

2.3 BGP, convergence, and routing trees.

In BGP, each AS uses its ranking function to select a single path from the set of paths it learns from its neighbors, and then announces this path to the set of neighbors dictated by its export policy. Because each AS may only select paths from the set of paths learned from neighbors, the set of selected routes induces a tree rooted at the destination d, which we refer to as the *routing tree* to destination AS d, *i.e.,* $T(d)$, and $T_n(d)$ as the subtree of this routing tree rooted at node n. Moreover, we do not focus on BGP convergence dynamics (as in *e.g.,* [13]); instead we consider only the routing trees created when each AS no longer wants to change its route. Indeed, in [11] we prove that BGP will converge to a single stable routing tree if every AS uses the routing policies in Section 2.2; moreover, the proof holds even if the AS graph has cycles of customer-provider edges.

---

[1]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".

3. ALGORITHMIC CHALLENGES

To establish the robustness of our results, our approach [10, 12] involves running repeated simulations over the entire $36K$-node empirical AS graph [5, 6]. Of course, running repeated simulations at this scale takes time. While it is quite feasible to run simulation algorithms that take time $O(|V|^2)$ (e.g., $(36K)^2$ operations that take $1\mu s$ each runs in ~ 20 minutes), once we start getting into the realm of $O(|V|^3)$, simulation very quickly approaches the border of infeasibility (e.g., $(36K)^3$ $1\mu s$-operations takes ~ 17 months!). Thus, when working with $O(|V|^3)$ simulations, optimizing the *constants* in the O-notation becomes crucial; a $1000x$ speedup can be the difference between reporting on the results of a single simulation with a fixed set of parameters, and establishing the robustness of these results by performing 1000 different simulations with varied parameters.

In the following two sections, we discuss the algorithms that formed the basis of our simulations in [10,12], and show how we used *amortization* and *parallelization* to obtain a significant constant speedup that made our simulations feasible even when they crossed the $O(|V|^3)$ barrier.

4. ROUTING TREE ALGORITHM

At the core of our simulation methodology is a *routing tree algorithm* that computes the best available paths for each source AS to a given destination AS d, i.e., $T(d)$ (once BGP has converged so that no AS wishes to change its route). The algorithm assumes that ASes use the routing policies of Section 2.2, and takes time $O(|V|^2)$ to compute paths between all source-destination pairs, which is significantly faster than the $O(|V|^3)$ algorithm proposed in [32].

The algorithm is a specialized three-stage breadth-first search (BFS) on the AS graph; each stage of the BFS computes shortest paths of a particular type (i.e., customer / peer/provider). We explain the algorithm using Figure 1:

1st *stage: Customer paths.* We construct a partial routing tree by performing a BFS 'upwards' from the 'root' node d, using only customer-to-provider edges. (In Figure 1, we add edges $(d,1)$, $(d,2)$, $(d,5)$, $(2,6)$, and $(5,6)$).

2nd *stage: Peer paths.* Next, we use only single peer-to-peer edges to connect *new* ASes to the ASes *already added* to the partial routing tree in the 1st stage of the algorithm. (In Figure 1, this amounts to adding edges $(1,4)$ and $(1,3)$ but not $(1,2)$ or $(4,9)$).

3rd *stage: Provider paths and Tiebreak.* Next, we traverse the existing partial routing tree with a BFS, and add *new* ASes to the tree using provider-to-customer edges. (In Figure 1, this amounts to adding edges $(1,9)$, $(4,8)$ and finally $(3,8)$). Each source AS that is traversed now has a set of equally-good paths (in terms of **LP** and **SP**) to the destination. Thus, for each source AS, we sort its set of paths according to **TB** and determine its chosen path to d.

Running time. Consider a AS graph $G(V,E)$ where $E_{cp} \subseteq E$ is the subset of customer-provider edges. The 1st and 3rd stages visit customer-provider edges only (giving a worst-case running time of $|V| + |E_{cp}|$ each), while the 2nd stage visits peer-peer edges only ($|V| + |E \backslash E_{cp}|$). Thus, the full algorithm has worst-case running time of $3|V| + 2|E|$ to compute paths to a *single destination*. Since the AS graph is sparse, with $|E| \approx 4|V|$, we can compute paths for *all source-destination pairs* paths in time $11|V|^2$.
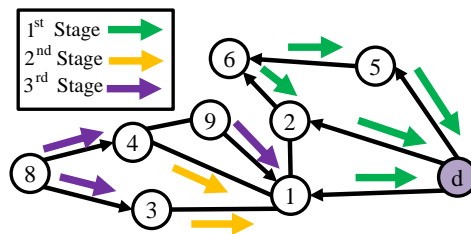

Figure 1: Routing tree algorithm.

5. ALGORITHMS: S*BGP DEPLOYMENT

In [12], we repeatedly computed paths from all sources to a single destination d, keeping us firmly in the realm of $O(|V|^2)$. However, in our study of S*BGP deployment dynamics [10], we hit the $O(|V|^3)$ barrier. We now show how amortization and parallelization made our study [10] feasible, by affording us a $\sim 1000x$ constant speedup.

5.1 The setting.

In [10], we assumed each AS could be 'secure' (i.e., deploy S*BGP) or 'insecure', where the *state S* is the set of secure ASes in the AS graph. Our goal was to study the evolution of S over time. We assumed that (a) an ISP n will become secure if doing so allows n to attract more traffic (i.e., increase the size of $T_n(d)$) and (b) secure ASes prefer to send their traffic via secure routes. As such, we modified the routing policies of Section 2.2 with the following step, that occurs *between* the **SP** and **TB** steps:

SecP Secure Paths. If there are multiple such paths, and node a is secure, then prefer *secure paths (i.e., paths on which every node is secure).*

Our simulation was initialized with a state S_o, which models a set of *early adopter ASes* that are the first to become secure. In each iteration with state S, (a) stubs deploy S*BGP iff one of their providers deploys S*BGP, and (b) each ISP n decides to deploy S*BGP iff n has 'utility', i.e.,

$$\sum_d |T_n(d)|$$

($T_n(d)$ defined in Section 2.3) that is higher, by a constant factor θ, in state $(\neg S_n, S_{-n})$ (the state S with ISP n reverses its deployment decision) relative to the original state S.

5.2 Overview of our algorithmic approach.

Scaling challenges. In [10], the combination of the dynamic state S with the addition of **SecP** to the routing policy means that paths can change in *each iteration*, and moreover we need to compute paths in state $(\neg S_n, S_{-n})$ for *each ISP* in each iteration. Since there are $15\% \cdot |V|$ ISPs in the AS graph, the bottom line is that *in each iteration* we need compute all-pairs paths $15\%|V|$ times. While the routing tree algorithm can be used for the all-pairs path computation, this gives a running time of $\sim 0.15|V| \cdot 11|V|^2 = 1.65|V|^3$, which quickly becomes intractable; if the routing tree algorithm for a single destination takes 10 ms, running it $15\%|V|^2 = 194M$ times takes 22 *days* for a *single iteration*!

Thus, we now show how we achieved the $1000x$ constant speedup that made our simulations feasible:

Amortization. Instead of repeating the routing tree algorithm $15\%|V|^2$ in each iteration, our approach was to run the

(relatively-slow) routing tree algorithm *once*, and save the intermediate results; we then use these intermediate results as input to a faster algorithm that we run repeatedly in each iteration, thus *amortizing* our computation. In Section 5.3 we show how our faster algorithm exploits the sparsity of the AS graph to run in average time $1.18|V|$ (*cf.,* the routing tree algorithm's $11|V|$ worst-case running time). More concretely, our C# routing tree algorithm implementation ran in $\approx 10ms$, while our faster algorithm ran in $\approx 2ms$.

Parallelization. Even with the amortized algorithm, we still needed parallelization to run our simulations in a reasonable amount of time; notice that with a 2 ms amortized algorithm, a single iteration would take $0.15|V|^2 \cdot 2ms = 4$ *days*! Instead, we leverage the fact that all our algorithms are independent across destinations, making them particularly amenable to Map-Reduce style parallelization. Thus, we parallelized (*i.e.,* mapped) our computation of the subtrees $T(d, S)$ for each state $(\neg S_n, S_{-n})$ across destinations d. We then aggregated (*i.e.,* reduced) them across the states $(\neg S_n, S_{-n})$ to obtain utility, *i.e.,* $\sum_d T_n(d)$ in state $(\neg S_n, S_{-n})$. We ran our code on a shared cluster of about 200 machines running DryadLINQ [33]; by parallelizing across destinations, each machine was roughly assigned computation for about $|V|/200 = 150$ destinations (but see [33] for details on how to implement this in DryadLINQ).

Total running time. The combination of the 5x speedup from amortization and the $\sim 200x$ speedup from parallelization gives running time of about $0.15 \cdot 1.18/200 \cdot |V|^3 = \frac{1}{1100}|V|^3$, giving us a $\sim 1000x$ constant speedup. Indeed, after a few more optimizations (discussed in our report [11]) a single iteration could run in 10-20 *minutes*.

5.3 Speedups via amortization.

The following observation allowed us to use amortization:

OBSERVATION 5.1. *If ASes use the routing policies of Section 2.2, 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.*

(Details in our report [11].) For each destination AS d, we do the following. First, we run the full routing tree algorithm *only once* to precompute the set of equally-good (in terms of type and length) paths from each source AS n to destination AS d. Then, for each of the $15\%|V|$ states $(\neg S_n, S_{-n})$ in each iteration, we repeat the following: given the tiebreak sets, we compute n's chosen path to d according the state, **SecP**, and **TB** (Section 2.2, 5.1).

The following values are precomputed from the routing tree algorithm for each destination d:

BucketTable: Table 1 is a BUCKETTABLE for the AS graph in Figure 1 with d as the destination. For each AS n, a BUCKETTABLE stores the (1) the path type, *i.e.,* the relationship between n and the next hop on n's best path to d, (*i.e., the* decision in the **LP** step) (2) the length of n's best path to d (*i.e.,* the decision in the **SP** step). Each AS n is placed in a cell of the BUCKETTABLE according to its best path length (row) and best path type (column), *e.g.,* AS 8 with a three hop path to d through a provider is in cell $(3, provider)$.

TieBreakSet: We have a TIEBREAKSET for each source AS n that stores the set of neighbors that offer n equally good paths to d according to **LP** and **SP**. The ASes in TIEBREAKSET are sorted by their ranking according to **TB**.

Table 1: BucketTable for routing tree in Figure 1.

Row	Cust.	Peer	Prov.
0	d	-	-
1	1, 2, 5	-	-
2	6	3, 4	9
3	-	-	8

For example, TIEBREAKSET$(6, d) = \{2, 5\}$ because according to **LP** and **SP**, both AS 2 and AS 5 offer AS 6 equally good paths to d (both are 2-hop paths where the next hop is a customer); AS 2 is stored first because it wins the **TB** step.

Amortized routing tree algorithm. This algorithm computes the routing tree for a given destination d and state S using the precomputed BUCKETTABLE and TIEBREAKSETS associated with d. Observe that every node in AS n's tiebreak set must have a path to d that is exactly one-hop shorter than the path that AS n has to d. Thus, we process each node n's routing decision in ascending order of path length, starting with the destination d. To do this, we start at the 0th row of the BUCKETTABLE (which can contain only the destination d) and walk down the rows BUCKETTABLE, processing each node n in the row as follows:

- If n is secure in state S and there are nodes in i's tiebreak set with a secure path to d, then n chooses a path through through first node in the TIEBREAKSET that offers n a secure path. AS n is marked as using a secure path to d.

- Otherwise, n chooses a path through the first node in its the TIEBREAKSET and n is marked as using an insecure path to d.

The average running time of the amortized routing tree algorithm is $t|V|$ for a single destination, where t is the average size of the TIEBREAKSET in the AS graph. Because the AS graph is quite sparse, it turns out that $t = 1.18$, which gives us up to a $6x$ speedup over the $11|V|$ routing tree algorithm.

6. ROBUSTNESS: S*BGP DEPLOYMENT

Our fast simulation algorithms allowed us to analyze the robustness and sensitivity of our results in [10], by performing repeated simulations with varied parameters. Guided by works like [18, 28], we had a good sense for where the gaps in our empirical data lie. Thus, we parameterized the gaps in our data, and then ran multiple simulations with varied parameters to ensure that our results hold within a "certain radius" of the ground truth.

We studied sensitivity to incompleteness in (1) empirical AS graphs [1, 5], and (2) interdomain traffic volumes. We also contended with (3) scarcity of information about our model-specific parameter, the *deployment threshold* θ (see Section 5.1). Finally, while our algorithms are "hard-coded" with the choice of routing policies (Section 2.2), we were able to reason about robustness to choice of routing policy without running additional simulations; see [11] for discussion.

6.1 AS graph.

We ran our simulations over the UCLA AS graph from Dec. 9, 2010 with additional IXP peering edges from [1], as summarized as the UCLA+IXP graph in Table 2.

Incompleteness of AS-level topologies. A ground truth for AS-level connectivity remains elusive [24], and is

especially problematic for large content providers that peer with a large number of ISPs at lower levels of the topology [1, 24, 28]. In [10], we singled out five of the largest CPs according to recent research [21, 29]: Google (AS 15169), Microsoft (AS 8075), Facebook (AS 32934), Akamai (AS 20940) and Limelight (AS 22822). Indeed, we observed that average path lengths for these CPs in the UCLA+IXP graph (according to the routing policies of Section 2.2) were around 2.7-3.5 hops, whereas the popular Knodes index [25] reports them to be much lower, around 2.2-2.4 hops.

Creating the augmented graph. We developed an augmented topology that focused on more accurate connectivity for the five CPs. We leveraged recent research on IXPs that indicates that many CPs are joining IXPs and peering with a large fraction of the IXP members [1]. Starting with the UCLA+IXP graph, we randomly connected the five CPs to ASes for which they are colocated at IXPs (using [1]'s data on ASes' presence at Internet Exchange Points (IXPs)), until their average path length decreased to 2.1-2.2 hops. Table 2 summarizes the resulting topology.

6.2 Traffic volumes.

Empirical data about Internet traffic volumes remains notoriously elusive. In our model, the utility of an ISP n is a function of traffic it attracts (see Section 5.1), so ideally we would weigh the contribution of an AS p that routes through ISP n by the volume of traffic that it sends to each destination d reached through ISP n. However, while some models of traffic volume exist (*e.g.,* the gravity model [4], which requires empirical data about every AS in the AS graph (which is known for some ASes, but not all)), modeling the interdomain traffic matrix is still a challenging problem. We separate the problem into two parts:

1. Where traffic originates. It is well known that a disproportionately large volume of Internet traffic originates at a few CPs. As such, we singled out the same five large CPs as before and assigned to each a 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. Since we didn't know x, we swept through different values $x = \{10\%, 20\%, 33\%, 50\%\}$ for the fraction of traffic originated by the five CPs; recent work suggests a reasonable range is $x = $10-20% [20, 21].

2. Where traffic is sent. We use two different models here. Our first model assumed that an AS spreads traffic evenly across all possible destination ASes. Our second model experiments with the idea of traffic *locality, i.e.,* that ASes are more likely to send more traffic to ASes closer to them. To model this, each ASes sends traffic proportional to $1/k$ to destination that are k hops away.

6.3 The deployment threshold.

Generally, we try to make our models as simple as possible, and avoid the introduction of multiple parameters that are difficult to quantify using empirical data. In [10], we needed to introduce a "deployment threshold" parameter θ (Section 5.1), which captures the cost of upgrading a network to S*BGP, relative the profits obtained from increases in utility. Because data on θ remains elusive, we assumed that each ISP uses the same threshold θ and swept through different values of θ, to get a sense of how deployment will progress with different deployment costs.

Table 2: Summary of AS graphs

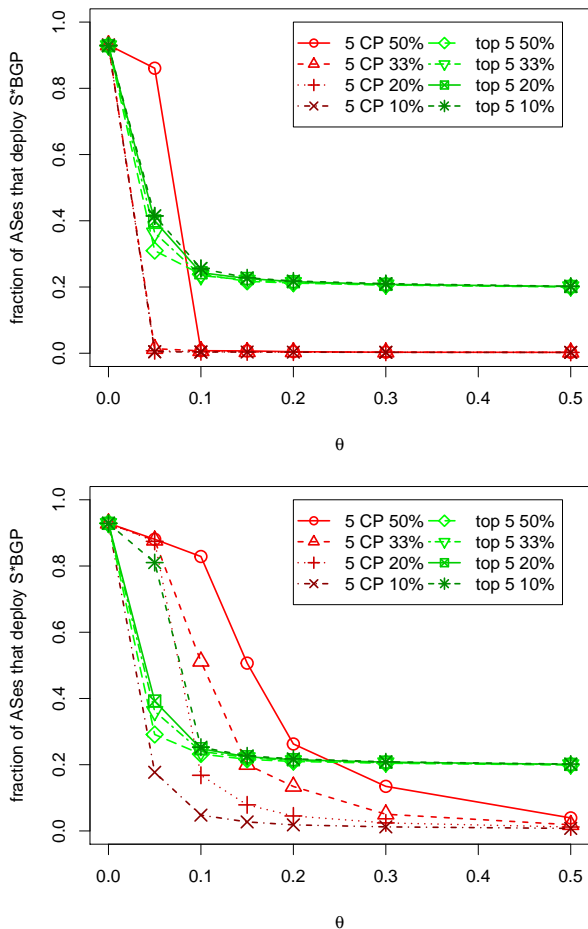| Graph	ASes	peering	customer-provider
UCLA+IXP [1, 5]	36,964	58,829	72,848
Augmented graph	36,966	77,380	72,848

Figure 2: Results of having the 5 CPs and top 5 Tier 1s as early adopters for varying traffic volume and deployment threshold θ in the UCLA+IXP (top) and augmented (bottom) AS graphs.

6.4 Results.

We show a sample of how we determined which sets of *early adopter ASes* (*i.e.,* different initial states), maximize the number of secure ASes.

Figure 2: We compare two early adopter sets: the five CPs vs. top five ASes in terms of degree (*i.e.,* the Tier 1s). We do this across all parameters discussed above: (a) in the original UCLA+IXP graph vs. the augmented AS graph (top figure vs. bottom), (b) different traffic volumes (different curves), and (c) by sweeping through values of θ (the x-axis). The y-axis shows the number of secure ASes when the simulation terminates (higher values are better).

Impact of the deployment threshold. Unsurprisingly, we find that our results are highly sensitive to the deployment threshold θ; the more expensive it is to deploy a secure routing protocol, the fewer ASes are likely to deploy it.

Results are generally robust. Fixing a particular deployment threshold, our results were virtually unaffected by

the spread of traffic across destinations (Figure 13 in [11]). Moreover, our results suggest that the Tier 1s perform best as early adopters for all parameters we considered, except for in one particular case: simulations on the augmented graph when the five content providers collectively source more than 20% of Internet traffic.

The scarcity of ground truth prevents us from concluding that the Tier 1s are better as early adopters than the CPs; instead, [10] suggests having an early adopter set consisting of *both* the five CPs and the five Tier 1s.

7. CONCLUSIONS

Studying interdomain properties of the Internet requires simulations where ground truth data is scarce. Understanding the impact of imperfect data and modeling assumptions on simulation results requires running large numbers of simulations. We make two contributions, presenting (1) scalable algorithms for simulating BGP routing on full-scale empirical AS graphs, and (2) techniques for assessing the robustness of results to gaps in empirical data, by running multiple simulations with different parameters. Our approach provides a significant speedup by hard-coding our simulator with the standard set of routing policies used in many BGP studies [9]; future work involves developing algorithms that support a wider range of routing policies.

Acknowledgments

8. REFERENCES

[1] B. Augustin, B. Krishnamurthy, and W. Willinger. IXPs: Mapped? In *IMC*, 2009.

[2] I. Avramopoulos, M. Suchara, and J. Rexford. How small groups can secure interdomain routing. Technical report, Princeton University Comp. Sci., 2007.

[3] H. Chang, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure BGP protocol. In *Sigcomm*, 2006.

[4] H. Chang, S. Jamin, Z. Mao, and W. Willinger. An empirical approach to modeling inter-AS traffic matrices. In *IMC*, 2005.

[5] Y.-J. Chi, R. Oliveira, and L. Zhang. Cyclops: The Internet AS-level observatory. *ACM SIGCOMM CCR*, 2008.

[6] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, and kc claffy. AS relationships: Inference and validation. *ACM SIGCOMM Computer Communication Review*, JAN 2007.

[7] X. Dimitropoulos and G. Riley. Efficient large-scale BGP simulations. *Elsevier Computer Networks, Special Issue on Network Modeling and Simulation*, 50(12), 2006.

[8] N. Feamster, J. Winick, and J. Rexford. A model of BGP routing for network engineering. In *SIGMETRICS*, 2004.

[9] L. Gao and J. Rexford. Stable Internet routing without global coordination. *Trans. on Networking*, 2001.

[10] P. Gill, M. Schapira, and S. Goldberg. Let the market drive deployment: A strategy for transitioning to BGP security. In *SIGCOMM*, 2011.

[11] P. Gill, M. Schapira, and S. Goldberg. Let the market drive deployment: A strategy for transitioning to BGP security. Full version. Technical report, 2011.

[12] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How secure are secure interdomain routing protocols. In *Sigcomm*, 2010.

[13] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *Trans. on Networking*, 2002.

[14] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. Technical report, UM, 2000.

[15] J. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson, and A. Venkataramani. Consensus routing: The Internet as a distributed system. In *NSDI*, 2008.

[16] J. Karlin, S. Forrest, and J. Rexford. Autonomous security for autonomous systems. *Computer Networks*, oct 2008.

[17] J. Karlin, S. Forrest, and J. Rexford. Nation-state routing: Censorship, wiretapping, and BGP. *CoRR*, abs/0903.3218, 2009.

[18] B. Krishnamurthy, W. Willinger, P. Gill, and M. Arlitt. A socratic method for validation of measurement-based networking research. *Computer Communications*, 2011.

[19] 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.

[20] C. Labovitz. Arbor blog: Battle of the hyper giants. http://asert.arbornetworks.com/2010/04/the-battle-of-the-hyper-giants-part-i-2/.

[21] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Sigcomm*, 2010.

[22] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal topology generation. In *MASCOTS*, 2001.

[23] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. In search of the elusive ground truth: The Internet's AS-level connectivity structure. In *SIGMETRICS*, 2008.

[24] 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 - Techical Report TR-080026-2008*, Sept 2008.

[25] F. Orbit. http://www.fixedorbit.com/metrics.htm.

[26] B. J. Premore. *An analysis of convergence properties of the border gateway protocol usingdiscrete event simulation.* PhD thesis, Dartmouth College, 2003.

[27] B. Quoitin and S. Uhlig. Modeling the routing of an autonomous system with CBGP. *IEEE Network Magazine*, 2005.

[28] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 lessons from 10 years of measuring and modeling the Internet's autonomous systems. *Journal on Select Areas in Communications*, 2011.

[29] Sandvine. Fall 2010 global Internet phenomena, 2010.

[30] W. Willinger, D. Alderson, and J. Doyle. Mathematics and the Internet: A source of enourmous confusion and great potential. *Notices of the American Mathematical Society*, 2009.

[31] M. Wojciechowski. Border gateway protocol modeling and simulation. Master's thesis, University of Warsaw, 2008.

[32] J. Wu, Y. Zhang, Z. M. Mao, and K. Shin. Internet routing resilience to failures: Analysis and implications. In *CoNEXT*, 2007.

[33] 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 high-level language. In *Usenix OSDI*, 2008.

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