Topology Inference from BGP Routing Dynamics

David G. Andersen, Nick Feamster, Steve Bauer, Hari Balakrishnan
MIT Laboratory for Computer Science
{dga, feamster, bauer, hari}@lcs.mit.edu

Abstract— **This paper describes a method of inferring logical relationships between network prefixes within an Autonomous System (AS) using only passive monitoring of BGP messages. By clustering these prefixes based upon similarities between their update times, we create a hierarchy linking the prefixes within the larger AS. We can frequently identify groups of prefixes routed to the same ISP Point of Presence (PoP), despite the lack of identifying information in the BGP messages. Similarly, we observe disparate prefixes under common organizational control, or with long shared network paths. In addition to discovering interesting network characteristics, our passive method facilitates topology discovery by potentially reducing the number of active probes required in traditional traceroute-based Internet mapping mechanisms.**

I. Introduction

Topological information about the Internet is useful for various reasons, including troubleshooting operational problems, and facilitating more realistic simulation and evaluation of network protocols and applications. Most approaches to topology discovery use `traceroute`–based measurements [1], [2], routing messages [3], or a combination of both [4]. We use clustering methods to derive a *logical* topology from routing messages. The results have more detail than topologies constructed only from routing data, and require no active probing. These topologies are structurally interesting on their own, and have application to existing topology measurements.

`traceroute`-based topologies observe the actual path taken by packets as they travel[1]. Unfortunately, this approach requires sending active data into remote networks—something that is not always well received, may be filtered by end networks, and which requires more network resources on the part of the mapper. [2]

Topology inference from routing messages is completely passive, but does not obtain as detailed or accurate a map. BGP data does not necessarily reflect the actual routing of packets [3]. To reduce their routing load, Internet Service Providers (ISPs) aggressively aggregate prefixes into supernets, hiding many internal details. Finally, many large ISPs announce hundreds of prefixes under the same Autonomous System (AS) path. For instance, Figure 1 shows the cumulative distribution of prefixes by origin AS, and lists the 5 most frequently occurring origins from MIT's routing table. Bu *et al.* speculate on a power law for routing tables [5], but regardless of the constants, it is clear that

[1]We consider the *path* to be the IP-layer path.

[2]This research was sponsored by Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare Systems Center San Diego under contract N66001-00-1-8933.

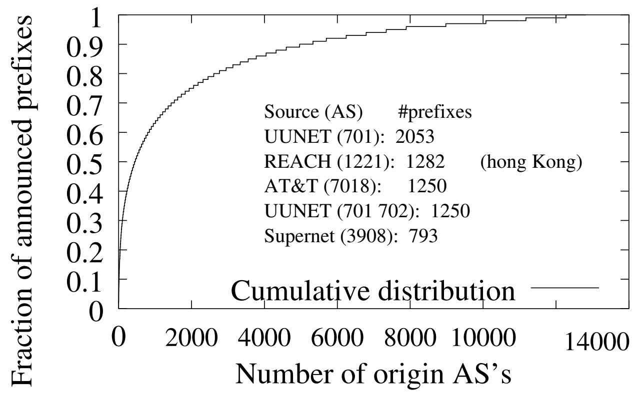


Figure 1. The 5 most frequent origin AS's account for 5.8% of the routes in an April 11, 2002 BGP table snapshot at MIT. The number of prefixes is dominated by a small number of prolific systems.

a small number of systems are responsible for a large fraction of the announced prefixes on the Internet. The 13 most common systems originate 10% of the total announced prefixes. Because of this, topologies that simply group all of the prefixes from a large, global ISP miss a great deal of information. Our work attempts to address this last failing of BGP-based topologies.

In this paper, we present a fully passive, BGP-based topology inference method. We group IP address prefixes based upon how frequently we observe BGP updates for both prefixes within the same time window. We then apply a standard clustering algorithm to join these prefixes into successively larger groups. Our temporal clustering produces higher-fidelity topologies than pure BGP table–based clustering. As an aid to other Internet mapping techniques, temporal clustering can guide the choice of paths on which to traceroute. Better destination selection can increase the information gained from fewer traceroute probes.

II. Clustering of Routing Updates

The input to our clustering is a time-series of routing updates. An update is any BGP routing message that is specific to a prefix, such as an announcement, or withdrawal. Each update contains a timestamp indicating the second at which it was received, and the prefix ("18.0.0.0/8") that was affected. The updates are ordered by timestamp. Clustering proceeds by grouping the prefixes that are frequently updated in the same time window. The result of our clustering is a binary tree on the input prefixes, where prefixes that are more tightly correlated are linked more closely in the binary tree. We find that prefixes linked by our algorithm often share administrative or topological features.

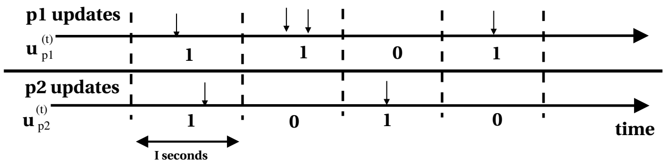


Figure 2. Generating the update vectors $u_{p1}^{(t)}$ and $u_{p2}^{(t)}$ from a stream of BGP updates for prefixes $(p1, p2)$. Time is discretized into I-second bins. $u_p^{(t)} = 1$ if and only if at least one BGP update for that prefix is received in the time window t.

A. Preprocessing

Before clustering, we *group* and *filter* the BGP updates. We first group BGP updates into time intervals of I seconds. We filter out massive updates that were typically caused by BGP session resets near our monitoring host. The information contained in massive updates is very limited—it reflects the measuring network's topology and immediate upstream connections, not the topology of any connections deeper in the network. Because clustering scales with the number of items that have connections to each other, this filtering can greatly increase the speed of the cluster computation.

The output of the preprocessing stage is a series of *update groups*, each of which contains the set of prefixes updated at that time.

B. Clustering

Clustering requires two components. The **distance metric** is a function that determines how closely two items are related. The **clustering method** groups items based upon their distance.

Our primary distance metric (actually a nearness metric) is based upon the correlation between the two update streams. To compute this, we take the discretized update groups, and determine the update vector $u_p^{(t)}$ for each prefix p:

$$u_p^{(t)} = \begin{cases} 1 & \text{if } p \text{ updated during interval } t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Figure 2 gives an example of update vector creation.

The correlation coefficient distance metric between two prefixes (p_1, p_2) is the correlation coefficient between their corresponding one/zero update vectors across n time intervals:

$$\mathcal{C}(p1, p2) = \frac{\frac{1}{n}\sum_{t=1}^{n}\left(u_{p1}^{(t)} - \overline{u_{p1}}\right)\left(u_{p2}^{(t)} - \overline{u_{p2}}\right)}{\sqrt{\sigma_{p1}^2}\sqrt{\sigma_{p2}^2}} \tag{2}$$

$\overline{u_{p1}}$ is the average of u_{p1}, that is, the probability that the prefix will be updated in any given interval. σ_p^2 is its variance.

We also examined a second metric, the weighted sum of the number of times that two prefixes (p_1, p_2) occur in the same time interval. Each interval is normalized by the total number of updates observed in order to reduce the significance of p_1 and p_2 updates coinciding in a period of increased update traffic.

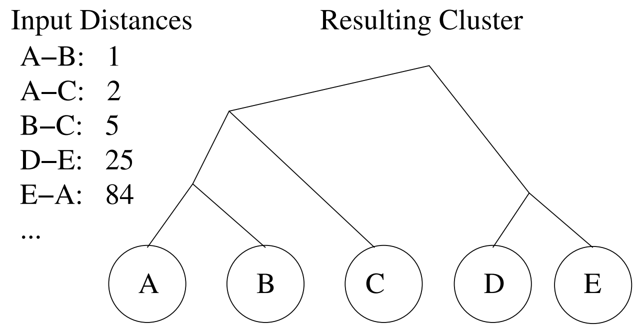


Figure 3. An example of the single-linkage (pairwise greedy) clustering method. The list on the left shows the 5 most closely related items, in order, and the graph on the right shows the clustering that results from this ordering.

Formally, let $|t|$ be the number of prefixes updated in time bin t. The weighted sum is:

$$\mathcal{S}(p1, p2) = \sum_{t=1}^{n} \begin{cases} \frac{1}{|t|} & \text{if } u_{p1}^{(t)} = u_{p1}^{(t)} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We found the correlation metric superior to the weighted sum metric in almost all cases, and focus on it during our evaluation. The weighted sum normalizes against increased update traffic, but admits a small bias towards groups of prefixes that experience frequent updates. In contrast, correlation normalizes against update frequency, but does not incorporate the confidence of the correlation. In future work, we intend to address this by adding a small prior to the correlation computation—a bias towards believing that prefixes are unrelated, so that the confidence in the correlation plays a part as well.

Single-linkage clustering is a simple and efficient pairwise greedy clustering method [6]. This method first computes the $O(n^2)$ pairwise distances between objects and stores them in sorted order. It iterates through the prefix pairs from closest to farthest. When it encounters a new node in a pair, the single-linkage method joins the node to its neighbor, or its neighbor's cluster if the neighbor has already been clustered. If both prefixes are in the same cluster, it does nothing. If the prefixes are in different clusters, the two are merged. This process is illustrated in Figure 3.

This straightforward clustering requires no recomputation of the $O(n^2)$ pairwise distances between objects, so the algorithm runs in $O(n^2 \log n)$ time. While single-linkage can be prone to artifacts like long unbranched chains, we require a fast algorithm to handle the large number of objects that we cluster (1,000–110,000). Using this method, it is feasible, if time consuming, to cluster all 110,000 prefixes in the BGP routing tables. Clustering 2,000 nodes runs in about 3 minutes.

III. Data Collection

We collected a large set of BGP traffic on which to perform our analysis, and a snapshot of traceroutes with which to evaluate our clustering. In this section, we describe how and when we collected routing traffic and traceroute data.

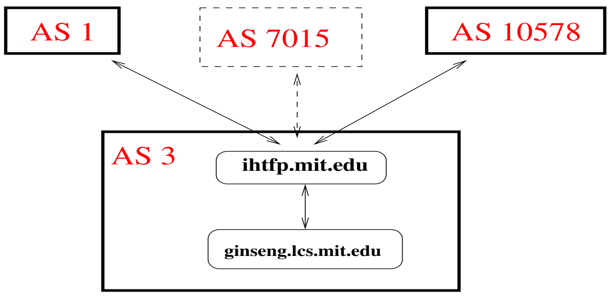


Figure 4. We collected BGP announcements at ginseng.lcs.mit.edu, which listened to routing updates from MIT's border router, ihtfp.mit.edu. MIT obtains upstream connectivity from Genuity (AS 1) and the Northeast Exchange (via AS 10578). On April 18, 2002 at 9:30am GMT, a private link between MIT and AT&T Broadband (AS 7015) was established.

A. BGP Collection

We collected BGP announcements using a Linux machine at the MIT Laboratory for Computer Science running Zebra 0.91a, an open source software router [7], which peered with MIT's border router. We configured Zebra to log all BGP updates, and archived these updates nightly. We have collected almost 70 million BGP announcements since June 28, 2001.

Figure 4 shows where our collection machine, ginseng, sits in relation to MIT's border router, ihtfp, and the rest of the Internet. During this collection, MIT received two upstream feeds: a commercial feed via Genuity (AS 1), and an Internet2 feed via the Northeast Exchange (AS 10578). Because of our configuration, we hear only MIT's *best* route to any given prefix. This means that our BGP monitor will not see all announcements seen by MIT; it will only see BGP announcements that caused a change in MIT's choice of best route to a prefix.

One noteworthy feature of MIT's peering session with Genuity is that we often see route updates that change only the Multi-Exit Discriminator (MED) of the route. These updates likely reflect a routing change internal to Genuity, or the selection of a different peer router out of which to send traffic. These announcements may propagate more topology information to us than other BGP routing feeds. We leave a quantification of these effects as future work.

As noted by Mahajan *et al.*, many announcements in the BGP table are accidental [8]. With one ISP, we observed a monthly factor of two increase in the number of announced prefixes, when an access list change permitted internal routes to leak to the rest of the network. Such leaks provide a possible source of data for obtaining higher-resolution topology maps. However, these accidental leaks may be reduced as a result of the Mahajan study, removing this potential source of information. We therefore remove these spurious announcements from our data set before clustering.

B. Traceroutes and Analytical Data

We took a "snapshot" of traceroutes to all BGP-announced prefixes in our routing table (109,783 destinations) by probing the first IP address within the prefix. After collection, we processed the traceroutes to assign DNS names and autonomous systems, and collected `whois` data for the prefixes.

Because we evaluate our clustering only in a tree context, not a full-mesh mapping context, we did not need to use the router interface identification techniques used by Mercator [2] or Rocketfuel [4]. We did, however, need to identify and merge cases of multi-path routing. To do so, we chose to compare paths based upon their most distant shared IP. This could cause problems when comparing networks that re-use interface IP addresses, but we did not encounter this problem in our study. We perform loop elimination by identifying and removing cycles from the traceroutes.

IV. Results

We performed clustering on 2338 prefixes announced by UUNET (AS 701) and 1310 prefixes announced by AT&T (AS 7018), the largest domestic chunks of indivisible prefixes in our routing table[3]. Some of these prefixes may also have been announced by other ISPs; we do not make use of this information. We examine those prefixes that were a) present in a static routing table snapshot at MIT on April 11, 2002, and b) announced with an AS path of 701 at some point in the preceding 90 days.

For these experiments, we grouped prefixes with a time window of 30 seconds, a common minimum advertisement time in BGP implementations. Experiments with a 60-second time interval showed little difference. We filtered massive updates at a 90% threshold—updates containing more than 90% of all of the advertised prefixes for a particular origin AS were discarded. This filtering removed about 20 half-minute intervals over a 90 day period, but removing these updates made the algorithm run much more quickly.

Clustering produces a rooted tree of "similarity" pairings where the intermediate nodes are meaningless (and implicit). In contrast, traceroute and the underlying network have actual routers at the intermediate nodes, and these routers are identified in network mapping. To evaluate our clustering, we must compare these two different types of graphs. In general, evaluating graph homomorphism is a hard problem. We first explore the structure of the clusters we obtain. With an understanding of that structure, we develop three general heuristics with which to evaluate the clusters: (1) Are the clustered IP addresses adjacent to each other? (2) Are the prefixes routed to the same destination? (3) How deep into the network do the prefixes share a path? To put these heuristics in context, we anecdotally examine some of the clustering decisions for which none of the heuristics apply.

Important cluster decisions occur *first*. In single-linkage clustering, the most important clustering decisions are made early, when joining the most strongly connected prefixes. Figure 5 shows the number of prefixes processed and the number of clusters formed, as a function of the number of pairwise prefix comparisons examined. UUNET ends up with 6 clusters after 2.3 million comparisons; AT&T with 5 clusters after $\sim 800K$ comparisons. We note that correlation-based clustering main-

[3]These numbers are larger than those from the static routing table snapshot because they're aggregated over 90 days of routing updates. AT&T appears to have fewer prefixes because they make many allocations from an internal class-A netblock, 12/8.

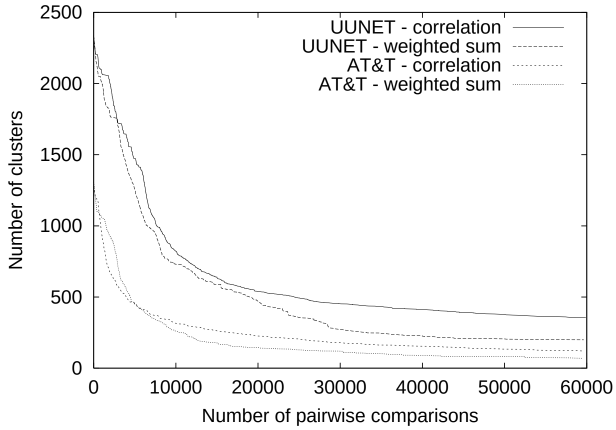


Figure 5. The evolution of the clustering over time. We begin with each of the 2338 nodes in an individual cluster. Cluster formation speed drops off rapidly — the most important low-level clustering decisions are made after considering the most closely paired prefixes.

tains a larger number of distinct clusters earlier in the clustering process. We interpret this as showing that the correlation score results in more intra-cluster linkages. This is a good thing—it indicates that correlation-based clustering is more internally consistent, though it says nothing about the relationship of these consistent subgraphs to other external metrics. In contrast, a random grouping of the AT&T nodes would be expected to converge to a single giant component in under $2n \log n = 30,000$ comparisons, whereas here it has 179 distinct sub-clusters [9]. For the remainder of the paper, we focus only on correlation-based clustering, which outperformed score-based clustering on nearly all metrics.

There are many metrics for evaluating a cluster. We observe a large number of "obvious" good clusterings — prefixes that are adjacent to each other and routed to the same customer, or even to the same router. These are generally easy to identify via traceroute information. The less easily identified cases are, however, more informative about the results that failure-based clustering produces. Before presenting automated analyses of our clusters, we discuss some examples.

A. *Selected Prefixes in the Same Cluster*

The prefixes 200.50.192.0/19 and 196.3.153.0/24 appear to have little to do with each other. Their traceroutes only stay together for 10 hops, to New York. An examination of `whois` data, however, reveals that both are in the Caribbean — one in Jamaica, the other in Haiti.

199.230.128.0/23 and 204.154.48.0/21 are located about 45 miles away from each other in Illinois — but traceroute doesn't reveal this, because the default route to one now goes through a different provider, with a backup link to UUNET. This relationship was only exposed by using the historical data of BGP updates.

205.159.243.0/24 and 204.86.96.0/24 share only 10 traceroute hops, but they both end up in the same UUNET PoP in Chicago 18 hops later, following a parallel load-balanced path.

198.51.238.0/24 and 192.150.253.0/24 shouldn't be in our data set: they both have Internet2 routes. During an outage, however, their backup route was announced through UUNET. Both of these netblocks go through Sandia National Labs to get to their final destinations. In fact, we discovered a small cluster of 6 such prefixes.

66.220.192.0/20 and 208.23.232.0/21 route through different ISPs (UUNET and Sprint). They're both assigned to Carrot Technologies; their supernets go through Sprint, but Carrot advertises both prefixes specifically through UUNET, without an autonomous system number.

199.242.4.0/23 and 198.22.254.0/24 share the same PoP, so why are they interesting? One is allocated to a company in New York, and one to a company in Delaware—both of which are a component of the same institutional fund management company.

More intriguingly, 135.36.0.0/16 and 135.12.0.0/14 have almost nothing to do with each other—but have almost identical patterns of updates and withdrawals. More investigation reveals that the companies, Agere Systems and Lucent Technologies, spun off in July 2000—but their networks still behave surprisingly similarly.

Some of these relationships are clear from network data — traceroutes, or examining DNS names. Other relationships only show up when examining data such as IP address allocation databases, or even the companies' web pages. From these examples, we know that we sometimes cluster based upon non-network features; however, we also frequently cluster in ways that have a strong relationship to the underlying network.

B. *Network-based Heuristics*

Krishnamurthy and Wang use the last two traceroute hops to a client and the last 2 or 3 components of the client DNS name to validate their clustering [3]. Because we frequently cluster prefixes belonging to different organizations, we generalize these validation methods slightly. We consider three network metrics:

- IP address similarity
- Ratio of shared to unshared traceroute path length
- DNS-based Point of Presence (PoP) comparison

For the geographic comparisons, we extract router location from the ISP's naming convention. These techniques generalize reasonably to other networks, though with some manual effort required [10], [4]. For our dataset, we assigned 97% of internal UUNET hops to a PoP, and a slightly smaller fraction of AT&T hops.

We evaluate each metric's score of the cluster whenever two clusters are merged. Recall that clustering is achieved by traversing an ordering of pairwise comparisons, and merging these in order. At each merge, we examine the sum of the pairwise scores for that metric that have been placed into the cluster at that point in time. This gives us an idea of how accurate the clustering is with respect to that metric. As the number of clusters reaches its minimum, the average value of the metric should drop to its actual average value; a good clustering will initially keep the average value high. For some graphs, where noted, we show an exponential weighted moving average instead of the total average, to better show the variation.

IP address space distance. It seems likely that adjacent netblocks may often be allocated "near" each other, for some

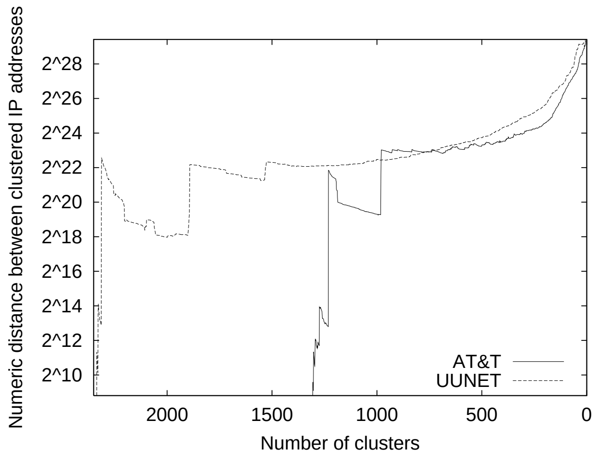


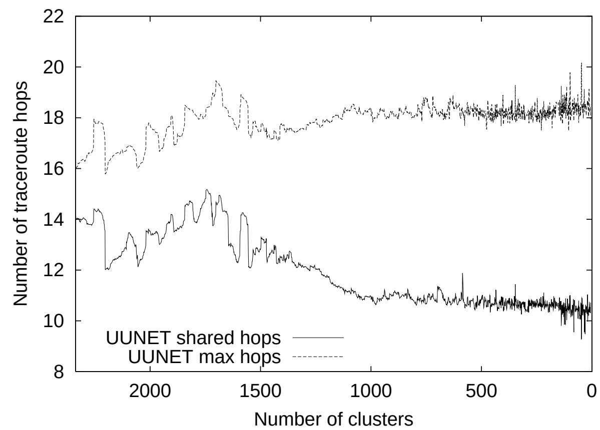
Figure 6. A moving average (q=0.995) of the numeric distance between the clustered IP address ranges. Two adjacent netblocks have a distance of zero. Two netblocks separated by a 'class C' netblock would have a distance of 2^8, and so on.

metric. We examine the distance between netblocks to see if the clustering groups netblocks that are close, numerically. We count the number of IP addresses that fall between adjacent netblocks, without accounting for the size of the blocks involved. Two adjacent netblocks (of any size) have a distance of zero. Two blocks separated by a "/24" (or an old-style "class C" netblock) have a distance of 2^8. Figure 6 shows these distances for AT&T and UUNET. The first few clustering decisions cluster prefixes that are numerically close to each other; the next 1/4 or so cluster prefixes that are closer than average. This is intuitively compatible with the idea that providers allocate IP addresses in a logical, hierarchical fashion.

Shared traceroute distance. The number of hops two paths have in common gives us one approximation of "network locality." Figure 7 shows the number of shared traceroute distances between pairs as clustering progresses. From this figure, we conclude that temporal clustering produces a grouping of prefixes that is accurate to several more network hops than a random grouping would be, at least, for the first few hundred clusters. By comparing with the length of the longest traceroute, we confirm that this effect is not simply due to clustering prefixes that are farther away.

Assignment to the same PoP. Because our clustering may often aggregate network blocks that do not follow the same path inside a customer, we next examine the Point-of-Presence (PoP)-level accuracy of our clustering. Figure 8 shows that correlation-based clustering groups the 2338 UUNET prefixes into about 1200 distinct clusters while retaining over 95% PoP-level accuracy. The 1310 AT&T prefixes group into about 900 clusters with about 97% accuracy. These numbers are quite informative; they suggest that many BGP updates occur for multiple prefixes at the PoP level.

The examples we presented show that the relationships found by failure-inspired logical clusters go beyond visible direct network connections. The network-based heuristics show us that even though our clusters arise from non-topological data, the resulting groups are informed by their underlying networks. These



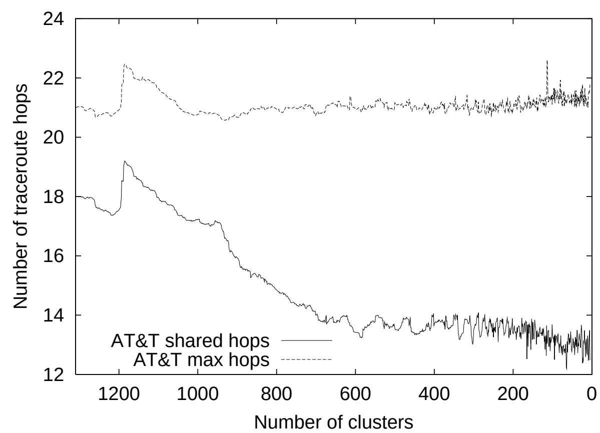


Figure 7. The pairwise shared/total traceroute hops. Computed with a moving average (q=0.995). The top line shows the average length of the longest traceroute in a pair, and the bottom line the number of hops that the two traceroutes had in common. The first few hundred cluster have longer-than-average shared paths to their prefixes.

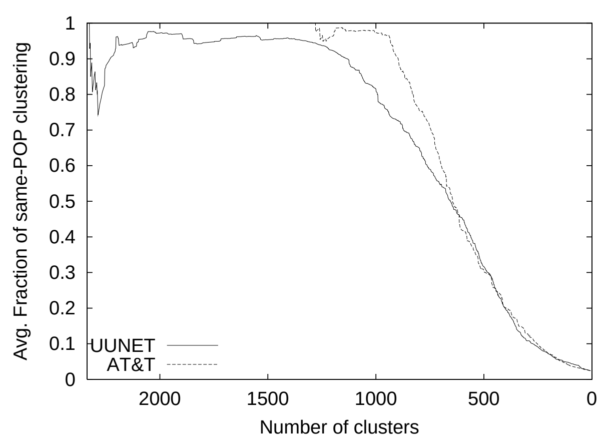


Figure 8. Number of clusters vs. the PoP-level accuracy of the clustering. UUNET reduces well from 2337 to about 1200 clusters; AT&T from 1310 to 900.

results imply that the information we extract from the temporal BGP signal has a strong basis in reality.

We speculate that BGP announcements for a particular prefix tend to be correlated with failure events in cases where a prefix has high geographic or topological specificity, as opposed to faults in highly aggregated prefixes. This agrees with our observation that clusters that are announced and withdrawn together tend to be located at the same POP.

V. Related Work

CAIDA's Skitter project is an ongoing, long-term collection of traceroutes and probes to large parts of the Internet, together with software to visualize and manipulate the data [1]. Like Skitter, Cheswick *et al.*'s early Internet mapping work [11] has a large visualization component to it. This early work traceroutes to destinations found in Internet Routing Registry databases, similar to using BGP-derived prefixes.

Mercator [2] uses "informed random-address probing" with `traceroute`-like probes to construct an Internet topology map. To choose its next target, Mercator considers both IP address allocation policies and the results of earlier probes.

Barford et al. examine the benefits from adding extra measurement hosts and destinations while performing Internet mapping [12]. They find that the marginal gain of adding additional sources drops off rapidly, but that the utility of adding additional random *destinations* remains nearly constant—discovering approximately 3 new nodes and 4 new links per destination, up to 1300 destinations. With this small number of destinations, it is unclear how the marginal gain applies with the larger set of target prefixes our work examines.

Chang *et al.* use traceroute data to infer relationships between autonomous systems [13]. Compared to passively using BGP tables, this active method can discover multiple links between ASs, and see through aggregation at ISP borders. We have not examined the ability of clustering to identify groups of prefixes from remote ASs that are likely shared at the same peering link, but it seems a promising avenue of exploration.

In an examination of routing table growth, Bu *et al.* claim that load balancing is responsible for a large number of identically-announced fragmented prefixes [5]. These prefixes present one obvious target for automatic aggregation by our methods.

Krishnamurthy and Wang use BGP routing information to cluster web clients [3]. They suggest that periodic traceroutes can be used to determine if clusters are too large or too small. In an analogous fashion, we propose a method for using BGP routing information to cluster *clusters*. We draw from this work's cluster validation methods to evaluate our own results in Section IV.

Rocketfuel improves on prior techniques to greatly reduce the number of traceroutes required for Internet mapping [4]. It uses multiple BGP tables from the RouteViews database [14], and attempts to map to both BGP-announced prefixes plus smaller blocks of 256 addresses within those prefixes. Rocketfuel stresses the import of doing the *right* traceroutes, instead of doing *all* of the traceroutes. Like Rocketfuel, the Cluster Graphs approach stresses the use of external information to guide Internet topology formation [15]. In this case, the authors use IP address activity information from other sources, such as WWW logs, to group IP addresses into clusters. Our work similarly presents a new source of information that can integrate with topology discovery: the temporal structure of BGP traffic itself.

As a caveat, we note that Spring et al. find that it is necessary to do probing at the sub-BGP prefix level to obtain complete coverage of an ISPs internal links [4], something that our BGP-only approach does not address.

VI. Conclusion

In this work, we have shown that the temporal structure of BGP messages can reveal interesting and important relationships between IP prefixes. Logical topologies derived from routing dynamics are also quite related to the underlying network structure. A clustering of prefixes inside UUNET can reduce the number of prefixes by about 50% while grouping 97% of the prefixes into groups that represent the same ISP Point-of-Presence. This data may prove useful for improving current Internet mapping topologies.

Our investigation is preliminary; in the future, we intend to analyze different distance metrics, examine other autonomous systems, and evaluate our techniques on a clustering of the entire Internet routing table. There is a great deal of information that can be mined from these sources; this paper provides only a glimpse at some of the relationships that can be inferred from completely passive analysis of BGP traffic patterns.

Acknowledgments

We would like to thank David Karger and David Gifford for much information about clustering—but any errors are exclusively of our own making. Jeff Schiller and Dorothy Curtis kindly helped arrange the BGP feed we analyzed. Magdalena Balazinska and Jaeyeon Jung provided valuable feedback on drafts of this paper. Finally, we thank our IMW'02 reviewers for their many good suggestions.

References

[1] "Skitter," `http://www.caida.org/tools/measurement/skitter/`, 2002.
[2] Ramesh Govindan and Hongsuda Tangmunarunkit, "Heuristics for internet map discovery," in *IEEE INFOCOM 2000*. IEEE, Mar. 2000, pp. 1371–1380.
[3] Balachander Krishnamurthy and Jia Wang, "On network-aware clustering of Web clients," in *Proc. ACM SIGCOMM*, 2000.
[4] Neil Spring, Ratul Mahajan, and David Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, Aug. 2002.
[5] Tian Bu, Lixin Gao, and Don Towsley, "On routing table growth," (Submitted for review) `http://www-net.cs.umass.edu/~tbu/`, 2002.
[6] Jure Zupan, *Clustering of Large Data Sets*, John Wiley and Sons, Ltd., 1982.
[7] "Gnu Zebra," `http://www.zebra.org/`.
[8] Ratul Mahajan, David Wetherall, and Tom Anderson, "Understanding BGP misconfiguration," in *Proc. ACM SIGCOMM*, Aug. 2002, (to appear) `http://www.cs.washington.edu/homes/ratul/bgp/bgp-misconfigs.ps`.
[9] Bela Bollobas, *Random Graphs*, Academic Press, London, UK, 1985.
[10] Venkata N.Padmanabhan and Lakshminarayanan Subramanian, "An investigation of geographic mapping techniques for internet hosts," in *Proc. ACM SIGCOMM*, 2001.
[11] Bill Cheswick, Hal Burch, and Steve Branigan, "Mapping and visualizing the Internet," in *Proc. USENIX Technical Conference*, 2000.
[12] Paul Barford, Azer Bestavros, John Byers, , and Mark Crovella, "On the marginal utility of network topology measurements," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.
[13] Hyunseok Chang, Sugih Jamin, and Walter Willinger, "Inferring AS-level Internet topology from router-level path traces," in *Proc. of SPIE ITCom*, Aug. 2001, pp. 19–24.
[14] University of Oregon, "RouteViews," `http://www.routeviews.org/`.
[15] Balachander Krishnamurthy and Jia Wang, "Topology modeling via cluster graphs," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.