Resource Deployment Based on Autonomous System Clustering

Jim Gast and Paul Barford
University of Wisconsin - Madison, Computer Science, {jgast,pb}@cs.wisc.edu

***Abstract*— Effective placement of resources used to support distributed services in the Internet depends on an accurate representation of Internet topology and routing. Representations of autonomous system (AS) level topology derived solely from BGP tables show only a subset of the connections that actually get used. However, in many cases, missing connections can be discovered by simple traceroutes. In addition, the differences between customer-to-provider links, peer-to-peer links, and sibling-to-sibling links are useful distinctions for the resource placement problem which is the focus of our work. Using two complementary mechanisms, we improve the accuracy of an AS forest as a predictor of packet paths. One mechanism uses recent insights that packets flow unidirectionally across customer-provider inter-AS links. Annotations are added to the AS forest to indicate links that appear to be peering versus those that appear to be customer-provider links. The other mechanism provides links between trees by remembering the most recently seen similar traceroute. The paper concludes by applying the annotated AS forest to a problem in resource placement to show that the representation is amenable to computationally inexpensive analysis.**

I. Introduction

Deploying services broadly across the Internet can provide additional capabilities, improve response times, reduce network congestion and reduce load over specific, crowded links. Effective placement of resources such as performance measurement nodes, proxy caches, application layer multicast distribution points, and anomaly detection engines would provide more accurate and more detailed services.

In this paper, we address the problem of effective resource deployment using client clustering at the AS level. We argue that deployment at the AS level is sufficient and appropriate for most services since routing within AS's is typically very efficient, and consideration of this problem at the router level makes it intractable for both the purpose of understanding topology and application of resource placement algorithms. We present a new method of clustering that generates a forest of trees of AS's. Branches of the tree form progressively smaller clusters, where each branch consists of AS's that are topologically near and over which packets are actually routed. Building and refining the AS forest consisted of three steps.

A. Step 1 - Building a representation of the Internet at the AS level

In step 1, we construct an initial AS forest based on BGP data. The floor of the forest is the centroid analogous to the strongly-connected portion of the Internet reported by Broido and Claffy [1]. From each of those nodes we grow a tree consisting of other AS's that most closely match neighbor sets. The presumed path of long distance packets would be from a leaf to the forest floor and then across the heavily connected floor (assumed to be a clique) and through the destination tree to the destination leaf. If the trees are accurate enough, the branches become *collection points* for traffic.

B. Step 2 - Calibrating the relative depths

To assign accurately a depth to each branch of each tree, we needed a mechanism for comparing trees. Using 200,000 traceroutes collected from public traceroute servers, we adjust the depth of each AS so that packet flows seen in actual traceroutes would adhere to a pattern noticed by Gao [2] in which packets flow only uphill, then laterally, then only downhill. Whenever a packet takes an unexpected hop between trees, we use the opportunity to compare those depths.

C. Step 3 - Adding missing links and an alternate parent

Our traceroute data shows many links that were not present in the BGP table data. This is consistent with the observations of Jamin et al. [3] and others since BGP contains only sparse information about non-local links. We again applied Gao's observations (but in this case to the traceroute results) and differentiated customer-to-provider links from sibling-to-sibling links. This allowed us to add an extra link to each node in each tree for an alternate parent.

D. Deploying resources using the AS forest

Finally, we apply the resulting AS forest to the problem of optimal resource placement. We employ a dynamic programming solution similar to the ones used by Raz [4] and Cidon [5] in which each sub-tree calculates the optimal use of 0 to ℓ servers in its sub-tree. The optimal placement solution is described in detail in [6].

II. Related Work

The need to separate the Internet into two very different regions, the highly interconnected core versus the periphery of hierarchical trees, was clearly stated by Broido and Claffy in [1]. They call the core portion the *giant component* and include inside it a large number of IP nodes.

An AS-level view of the Internet has much less detail, but would retain its value as an operational tool. And an AS-level view can be obtained from publicly available BGP tables like those at Oregon Route-Views [7]. Early researchers assumed that two AS's were linked if their AS numbers were adjacent in an AS path. Jamin et al. [3] showed that there are many actual links that do not show up in BGP. Then Gao and Rexford [8] made a substantial improvement in AS-graph accuracy when they noticed that customer-to-provider links create a hierarchy. Gao [2] went on to identify peer-to-peer and sibling-to-sibling relationships between Autonomous Systems and proposed a mechanism for inferring the relationships from the AS paths in BGP Tables.

Initial work on clustering clients and proxy placement was done by Cuñha [9]. That work described a process of using *traceroute* to generate a tree graph of client accesses (using IP addresses collected from a Web server's logs). Proxies were then placed in the tree using three different algorithms and the effects on reduction of server load and network traffic were evaluated. More recent work by Krishnamurthy and Wang in [10] provide new mechanisms for clustering of clients at the router level.

A number of recent papers have addressed the issue of proxy placement based on assumptions about the underlying topological structure of the Internet [11], [12], [13]. Li et al. [11] describe an optimal dynamic programming algorithm for placing multiple proxies in a tree-based topology.

Jamin et al. [12] examine a number of proxy placement algorithms under the assumption that the underlying topological structure is not a tree. Their results show quickly diminishing benefits of placing additional mirrors (defined as proxies which service all client requests directed to them) even using sophisticated and computationally intensive techniques. Qiu et al. [13] also evaluate the effectiveness of a number of graph theoretic proxy placement techniques. They find that proxy placement that considers both distance and request load performs a factor of 2 to 5 better than a random proxy placement. They also find that a greedy algorithm for mirror placement (one which simply iteratively chooses the best node as the site for the next mirror) performs better than a tree based algorithm.

III. Topologically-guided Clustering

A. Finding the centroid

To determine the strongly connected component, we computed the portion of the AS list reachable by AS_n in h hops, varying h from 2 to 5. Almost all of the AS's (94%) can reach the bulk of the Internet (90% of the other AS's) in 5 hops. The 4 hop results were also not helpful. Many nodes high on the 4 hop list gained their height solely by virtue of having 2 well-connected neighbors. Finally, we found a useful gap in the 3 hop list in which 25 AS's can see 80% of the other AS's. Since we could visualize 25 trees on a single screenful of information, we declared those 25 AS's to be our (somewhat arbitrary) backbone.

B. Definitions

The clustering algorithm uses neighbor sets and a distance function that acts as the length of a link.

The following definitions are used throughout the rest of this section:

- $Depth_n$ is the shortest distance from AS_n to a centroid node measured in AS hops.
- AS_n is a *neighbor* of AS_m if it immediately follows or precedes AS_m in any AS path.
- The *set of neighbors* of AS_n is denoted by N_n.
- $outdegree(n)$ of AS_n is $|N_n|$.
- AS_m is a candidate parent of AS_n if $m \in N_n$ and $Depth_m = Depth_n - 1$. The set of candidate parents of AS_n is denoted C_n.
- The *Hamming distance* between AS_n and AS_m is the number of neighbors exclusive to only one of them.

$$dist(n, m) = |N_n \cup N_m| - |N_n \cap N_m|$$

C. Clustering AS's using BGP routing data

For each node, we first compute $Depth_n$ for each AS_n using Prim's [14] Algorithm. Over 51% of the nodes are directly connected to a backbone node ($Depth_n \leq 1$) and over 91% of the nodes have $Depth_n \leq 2$.

Each AS with $Depth_n > 0$ chooses the nearest parent:

$$nearest(n) = \min_{m \in C_n} \{dist(n, m)\}$$

D. Initial annotations

The remaining neighbors in N_n consist of nodes that either one hop farther from the backbone, the same $Depth_n$, or one hop closer. Here we explicitly assume that edges that take a packet closer to the backbone are uphill edges. We eliminate uphill edges that were not the nearest candidate parent. Links to nodes at the same $Depth_n$ are initially assumed to be siblings. Links that are one hop farther away from the backbone we (initially) assume to be customers.

IV. Annotating the AS Forest

To test our AS forest we ran traceroutes. From the list of 882 traceroute servers, route servers, and looking glass sites from www.traceroute.org, we chose 135 servers, each in a different AS, 2 or more hops from the centroid.

A. Choosing traceroute destinations

For the traceroute destinations, we chose a representative IP address for every AS. Even when the last hop fails to reach a working IP address, if a prior hop already shows the desired AS, it is a usable AS trace. In our case, 11% of our traceroutes did not end in the intended AS.

Over the week of March 11, 2002, we performed 200K traceroutes. Although this number is comparable to other studies [15], [16] and much smaller than one study [1], our study did not need repetitions of the same routes.

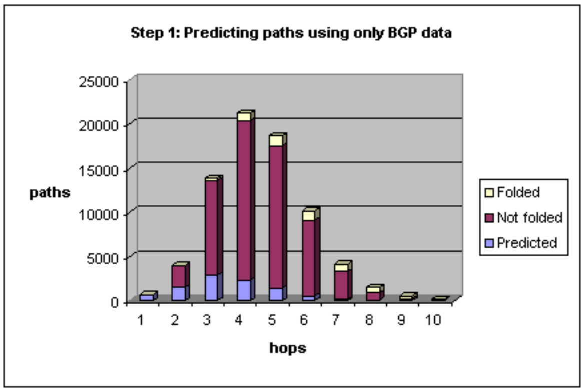

Fig. 1. Our early forest predicted only a tiny portion of the non-folded routes seen by traceroute.

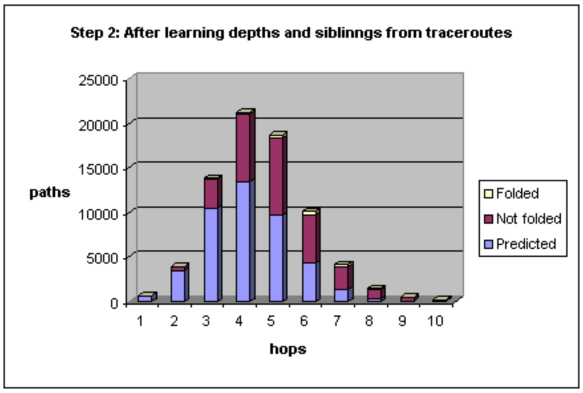

Fig. 2. Adjusting the annotations in the graph reduced the number of folded (implausible) paths and improved prediction.

B. Noting the relationship between AS's

The pattern we were hoping to see was the one identified by Gao [2]: each packet should flow uphill customer to provider, $c \rightarrow p$, (or laterally, sibling to sibling, $s \leftrightarrow s$) until it reaches the highest point needed to reach an AS (or a sibling or peer of an AS) upstream of the destination. Then the packet should flow only downhill provider to customer, $p \rightarrow c$, until it reaches the destination.

For each line of each traceroute, we translated the router link IP address to an AS number using the centralized BGP table. Our results sometimes skip over an AS because packets were lost, we got no response from the router, or because the router's interface had an IP address that belongs to the AS at the other end of the link. Because of route aggregation and other practical limitations of BGP, our translation from IP address to AS could be wrong as well. Finally, ISP's need not use globally-routable IP addresses for links inside their own domain. If we miss seeing the ingress into the AS, we might completely miss seeing the AS. Thus, our translated AS path might understate the length of the true AS path.

As other researchers previously noted [3], [2], a significant number of AS connections are hidden from most BGP tables. Figure 1 shows the results of the 74,963 unique complete traceroutes when applied to the AS forest derived solely from BGP information. The majority of the paths were from 3 to 6 AS hops long. A small number of paths were as long as 12 AS hops and a small number of IP addresses found routing loops at the inter-AS level.

The *folded traces* are the AS paths that appeared to flow uphill after having taken a downhill hop. At that point, our AS forest had only provisional labels to categorize each link as a customer-provider link or a sibling link. The dark area in the middle of each bar is the paths that did not violate the uphill-to-downhill laws but contained links not in our AS forest. For a hop from AS_m to AS_n we compare $Depth_m$ to $Depth_n$ in cases where the AS forest did not have a link at (m, n). Finally, the light area at the bottom is paths that only contained AS hops in the AS forest.

Figure 2 shows the same paths after the $Depth_n$ values have been refined. In this case, we pause for learning each time a traceroute shows an uphill hop after the packet had already reached a pinnacle. We used a *Current Best Hypothesis* algorithm [17] to test each hop of the traceroute. Imagine a trace $(k, l, ..., m, n)$ in which l was thought to be downhill from k, but n was thought to be uphill from m. This folded trace violates one or more of the annotations we have made. At least one of the links between k and m was annotated $A(k, l)$ as a $p \rightarrow c$ link. Choose k and l to be the closest instance of a $p \rightarrow c$ link. On the evidence of this traceroute, that could be a false positive. Alternatively, $A(m, n)$ was $c \rightarrow p$, preventing us from using it on the downhill side (a false negative). A special case where $l = m$ is easily handled.

To choose the appropriate generalization or specialization, we select the link most refuted by the evidence. That is, we track the failure count $F(m, n)$ and success count $S(m, n)$ of each annotation. If the total evidence $E = F(k, l) + S(k, l) + F(m, n) + S(m, n)$ exceeds a a learning rate threshold, α, we assume that we have seen enough cases to render a judgment. Each link, (k, l), has an error proportion $Err(k, l) = F(k, l)/(F(k, l) + S(k, l))$. If $Err(k, l) > Err(m, n)$ we change (m, n) to $s \leftrightarrow s$ by setting $Depth_m = Depth_n$. Alternatively, if the downhill link was more probably incorrect, we set $Depth_n = Depth_m$. Since we have changed the depth of an AS, we correct all of the annotations of the links to that AS.

The algorithm found exchange points like the Russian Universities Federal Network (AS3267) quickly. $Depth_{3267}$ went from 9 hops from the backbone to 1. Others like the Milan Interconnection Point (AS16004) rose 4 times. Whenever a $Depth_n$ changes, other links become $c \rightarrow p$ or $p \rightarrow c$.

Figure 2 shows the results of learning depths. Bars show the average of 10 runs over the same traceroutes using 10-fold cross-validation with $\alpha = 6$. Higher values of α would require a larger data set.

Since this fixed many of our mistakenly labeled customer-provider paths, previously folded paths were now non-folded. Our algorithm had reversed some customer-provider pairs. Also, there were improvements when unidirectional customer-provider links were upgraded to bidirectional sibling links.

C. *Adding learned relatives*

In many cases, the traced routes showed links that were not present in our BGP-based AS forest or even the BGP-based AS graph. We decided to add the most recent *alternate parent* to each AS whenever a trace showed an unexpected uphill hop from that AS. We limited the learning to identifying a single alternate parent for each AS. If we saved all of the alternate parents, the program would eventually have learned all of the routes seen, but the number of "correct" paths from one AS to another would grow too fast. This would have made our subsequent service placement algorithm ineffective. We placed no limit on the number of learned siblings at the same $Depth_n$.

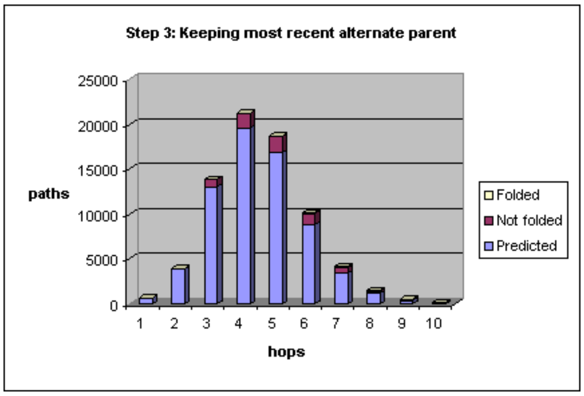

Fig. 3. Results with final AS forest

Figure 3 shows the results of allowing each node in the AS forest a list of siblings and a single, alternate uphill link. We considered more sophisticated techniques for discovering the best of the discovered links, but were satisfied that the simplest technique (saving the most recent) was effective and reacted well dynamically. Again, the results are the average of 10-fold cross validation with training sets of 67,467 traces and test sets of 7,496 traces. Over 91% of the test set traces correctly followed the uphill-then-downhill pattern and were composed only of links contained in our AS graph. Links with 5 or more AS hops had noticeably higher error rates.

Now that the AS forest can credibly predict the path of traceroutes, we turn our attention to the service placement problem.

V. Placement of Services

The goal in creating the augmented AS forest was to create a more accurate representation of the structure of the Internet so that our placement algorithm could more accurately place services.

To measure the effectiveness of our annotated AS forest, we consider the problem of placing web caches for a content provider. We accumulate bytes demanded by each AS using traffic seen by a set of commercial web server logs. Figure 4 shows the total traffic normalized to the traffic that would result if every request had to be satisfied at the centralized service. In our test data, 3.41 Gigabytes of replies came from 793 of the 12779 clusters. Using the refined forest produced by the clustering algorithm, on average traffic touched 2.49 AS's including the AS at the backbone and the originating AS. Throughout the rest of this section, we will use the term ASHop to avoid confusion with router hops. The total cost of traffic in this test data was 8.48 Gigabyte-ASHops.

A. *Placement By Customer*

For comparison, we evaluated costs for a placement algorithm that more closely matches the way services might

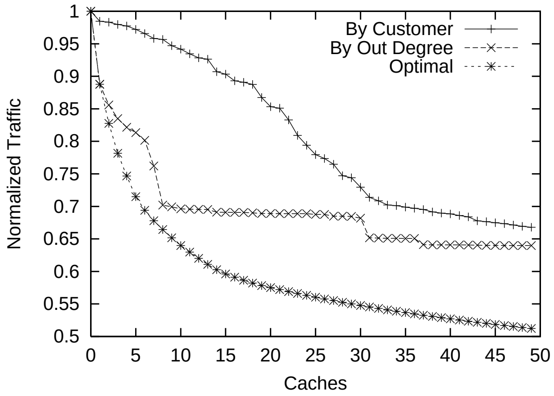

Fig. 4. Performance versus BGP-only placement algorithms

be placed opportunistically in a practical case. We randomly chose 50 locations out of the top 200 demand sites and averaged the results of 10 runs. The top line in Figure 4 shows the savings. On average, placing proxy caches in 15 AS's would have decreased the total number of byte-ASHops by 10%. The by-customer algorithm required 50 service locations to reduce the byte-ASHop traffic by 30%.

B. Placement by Out Degree

Figure 4 also shows the results of a placement algorithm that incrementally places each service at the AS with the highest remaining out degree (computed from the BGP table alone). This simple algorithm works well initially but quickly diminishes in utility after it uses up the innermost AS's in the centroid. The out-degree-based algorithm reduced the total traffic by 30% simply by using the 8 AS's with the highest out degree. The noticeable drops at 30 and 37 caches were caused by caches that had a large downstream demand.

These incremental placement algorithms (demand-based and out-degree-based) are appropriate for any service placement in which it is not economic to move a service from one location to another.

C. Simultaneous Placement

Running the dynamic programming algorithm [6] discovered ways to cut the total traffic byte-ASHops by half using 51 locations.

Perhaps the most useful value to the simultaneous placement algorithm is the shape of the graph of diminishing returns. By running the algorithm once, one can see the marginal benefit for the entire range of 0 to n caches.

VI. Conclusions

In this paper we have described methods for creating an AS forest based on BGP routing data and then refined by paths learned from traceroutes. The mechanism starts with an algorithm for finding the centroid of highly-connected nodes at the backbone of the Internet. It then grows a tree from each of those nodes based on BGP information. Rather than depend on subtle clues inside the BGP table to learn the relationships along the links, our mechanism learns those links from traceroute results.

The entire mechanism is suitable for use operationally, since it does not depend on proprietary information or large collections of route data. The resulting AS forest is computationally tractable for use in service placement.

We found that the mechanism learned a more accurate AS forest than what could be gleaned from the BGP tables alone, improving the prediction accuracy to 91% when tested against traced routes.

References

[1] A. Broido and kc claffy, "Internet topology: connectivity of IP graphs," Tech. Rep., CAIDA, http://www.caida.org/outreach/papers/topologylocal, 2001.

[2] L. Gao, "On inferring autonomous system relationships in the internet," in *IEEE Global Internet Symposium*, November 2000.

[3] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Towards capturing representative AS-level Internet topologies," in *To appear in ACM SIGMETRICS*, 2002.

[4] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 980–986, October 2000.

[5] I. Cidon, S. Kutten, and R. Soffer, "Optimal allocation of electronic content," *Proceedings of Infocomm*, April 2001.

[6] P. Barford, J. Y. Cai, and J. Gast, "Cache placement methods based on client demand clustering," Tech. Rep. TR1437, University of Wisconsin / Madison, March 2002.

[7] Route Views, "University of oregon," http://www.antc.uoregon.edu/routeviews.

[8] L. Gao and J. Rexford, "A stable internet routing without global coordination," in *Proceedings of ACM SIGMETRICS '00*, June 2000.

[9] C. Cuñha, *Trace Analysis and its Applications to Performance Enhancements of Distributed Information Systems*, Ph.D. thesis, Boston University, 1997.

[10] B. Krishnamurthy and J. Wang, "On network aware clustering of Web clients," in *Proceedings of ACM SIGCOMM '00*, Stockholm, Sweden, September 2000.

[11] B. Li, M. Golin, G. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of Web proxies in the Internet," in *Proceedings of IEEE INFOCOM '99*, New York, New York, March 1999.

[12] S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the Internet," in *Proceedings of IEEE INFOCOM '01*, Anchorage, Alaska, April 2001.

[13] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of Web server replicas," in *Proceedings of IEEE INFOCOM '01*, Anchorage, Alaska, April 2001.

[14] R. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[15] V. Paxson, "End-to-end Internet packet dynamics," in *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997.

[16] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proceedings of IEEE INFOCOM '00*, Tel Aviv, Isreal, March 2000, IEEE, pp. 1371–1380.

[17] J. S. Mill, "A system of logic, ratiocinative and inductive: Being a connected view of the principles of evidence, and methods of scientific investigation.," J.W. Parker, London, 1843.