

# Automatic Large Scale Generation of Internet PoP Level Maps

Dima Feldman  
Tel-Aviv University

Yuval Shavitt  
Tel-Aviv University

**Abstract**—Point of presence (PoP) level Internet maps are promising for tasks such as reasoning about the Internet evolution in time or Internet delay estimation. We thus suggest an efficient algorithm for generating PoP level Internet maps directly from the traceroute measurement results. The algorithm avoids the noisy process of interface aggregation to routers. The PoP level maps we obtain are annotated with PoP level link delay. Both the map topography and the link delay estimation for tens of ASes were validated with a combination of DNS names and two geo-IP databases, and were found satisfactory.

## I. INTRODUCTION

Studying the Internet structure and evolution is a rather young research thread that was initiated by the pioneering work of Faloutsos *et al.* [1]. The first works looked mostly at the static characterization of the Internet graph at certain points in time [2], [3], but later researchers tried to understand how the network evolve, and suggested models to explain the network evolution [4], [5], [6]. These models were mainly evolving in the abstract Internet AS graph with no connection to the real world, or with some naive connection with the Internet underlying geography. Recently, there is a growing understanding that the evolution of an Internet region should be estimated by tightly correlating the Internet structure with its underlying geography, and the changes in the economic, social, and even political evolution of the region in question. For example, it is no surprise that as the economy of a certain region grows it will result in greater demand for Internet connectivity, which will manifested as growth in the Internet graph relate to this region. There are only a few works in this research direction [7], [8] due to the difficulty of obtaining good Internet maps.

Internet maps can be drawn in several levels of abstraction, each is suitable for studying various aspects of the network. The most commonly used abstraction is the autonomous systems (AS) level graph which has a relatively small amount of nodes (a few tens of thousands

of ASes) and thus it is easy to handle. While it is useful for many purposes, the AS representation of the Internet lacks a geographic grip. There are many large- cross country and cross Atlantic ASes that are represented by a single node on the AS level graph, virtually connecting ASes that have large physical distance between them. In addition, AS sizes spread over three orders of magnitude which makes any growth model on the AS graph unreliable. For example, the growth of a large AS by 25% is more significant than the addition of several tiny stub ASes, yet the former will not show any change in the number of nodes in the AS level while the latter will show a network growth. The other extreme is to use the router level graph, which contain too many details for most practical uses, and due to its large size it is hard to imagine a mapping system for obtaining a reliable graph at this level and track its evolution in time.

Large ASes are not spread uniformly in the area they span, but typically are build from a rather small number of points of presence (PoPs). A PoP is a concentration of router and other networking devices in a campus from which Internet connectivity is offered to the region around it, typically at least a large metropolitan area. For simplicity of the writing, small ISPs and customer ISPs can be treated as a single PoP ISP. Thus, by partitioning large and medium ISPs to their PoPs and looking at the PoP connectivity map we have a map with two advantages. First, each PoP is now distinctly related to some metropolitan geographic area allowing us to correlate its characteristics (e.g., birth time, connectivity) with the characteristics of the real world of its region. Second, all the nodes in the PoP level map are roughly the same size, whether these are PoPs or small ISPs, making growth model relevant for this abstraction.

In this work, we suggest a novel algorithm for the automatic construction of PoP level maps. Rocketfuel [9] is the only work we're aware of to suggest such an algorithm, but despite its reported success by the authors, there was criticism [10] about some of the

techniques they used, in particular their reliance on DNS. We have shown [11] that, indeed, DNS introduces noise to the PoP partition. Rocketfuel partition an ISP by first aggregating IP interfaces into routers and then aggregating routers into PoPs. The first stage is the one where the DNS resolution noise is injected. We suggest, instead, to skip the IP to router aggregation phase and use a novel partitioning algorithm on the IP interface graph. At the end of the process, we generate a list of interfaces per PoP without the internal division of the PoP to routers. For the Internet modeling we target, and for many other tasks this is sufficient. We argue that the partitioning of IP interfaces to routers is not functionally important: for most practical matters, a single huge router functions similarly to a small group of routers in the same room which are inter-connected by a high speed network. In particular, for the Internet evolution models and for Internet distance maps, which are the two prime motivations of this work, the partition of IP interfaces to routers is certainly of little value.

We have been using the DIMES project [12], [13] measurement database for testing our PoP level map extraction. The main advantage of using DIMES is its enormous amounts of measurement and vantage points. DIMES is using a significantly higher amount of measurement agents from over two hundred vantage points, which measure at a higher rate than Rocketfuel (up to 4 *traceroute* a minute vs. one every 1.5 minutes in the Rocketfuel case).

There is one major difference between Rocketfuel goal and ours. While they attempted to discover as much as possible of the PoP level topology of a few ISPs, we attempt to *periodically* reveal the PoP level topology of the entire Internet. However, due to the yet limited capacity of our collection system, we do not have sufficient capacity to obtain the entire Internet PoP level graph. Instead, we concentrate on correctness and efficiency. In correctness we mean that we do not introduce wrong PoPs or PoP to PoP links. Due to the vast number of *traceroute* our system generates and periodic nature of the application, efficiency is an important feature of our algorithm: we can easily partition hundreds of ASes to PoPs directly from the DIMES *traceroute* database. Currently, we do not direct DIMES to measure in directions that will help the PoP topology construction, but this is certainly a future goal.

A typical PoP consists of two or more backbone/core routers and a number of client/access routers. The client/access routers are connected redundantly to more than one core router, while core routers are connected

to other PoPs of the ISP [14], [9]. The typical PoP has a clear bi-partite structure. However, we observe that when one uses *traceroute* probing from multiple sources to multiple destinations even simpler inter router connections create ‘bi-fans’<sup>1</sup>. We will use this type of structure to identify PoPs.

Beyond the topology, link delays can be calculated efficiently out of *traceroute* measurements [17]. Intra-PoP delays are in the order of a few milliseconds while inter-PoP delays grows linearly with distance (roughly 1ms RTT for 100km). We use the link delay in the bi-partite graphs we find to distinguish between intra and inter PoP interconnections.

We run our algorithm on the one hundred top degree ASes, and for most cases got reasonable partition. We closely examined the partition for a few US-wide and world wide providers, and compared the results with two geo-IP databases and the DNS names of the interfaces. The PoP partition results appear to be in agreement with geographic distances.

## II. THE POP EXTRACTION ALGORITHM

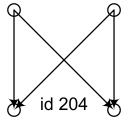
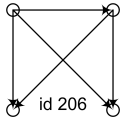
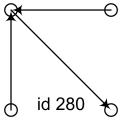
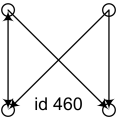
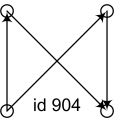
As described in the introduction, network graphs collected by DIMES agents has specific structures that are typical to the IP level internet graph. Alon *et al.* [15] showed that many complex networks has repetitive patterns of interconnections, called ‘network motifs’, which became a standard term in the networks analysis community. Their work showed that real-world networks outside the communication field are not purely random, but have a higher than (or lower than) expected number of specific motifs. In order to show the significance of a specific motif, their *mfinder* [16] software package uses the Z-score measure, which is calculated by

$$Z = \frac{X - \mu}{\sigma} \quad (1)$$

Where  $X$  is the number of occurrences of a motif in a specific network, and  $\mu$  and  $\sigma$  is mean and standard deviation of the motif occurrences within a random network with the same parameters. The Z-score reveals how many units of the standard deviation a specific case is above or below the mean. Unsurprisingly, we have found (using *mfinder* [16]) a number of motifs with a high Z-score across all the IP interface interconnection networks of various ASes. For example, Table I show the clear dominance of the ‘bi-fun’ motif (number 204)

<sup>1</sup>‘Bi-fan’ is a four node, specific network structure that was termed in [15]. In Section II the discussion about network motifs is extended and the bi-fan motif is defined by id 204 in a Table I.

TABLE I  
COMMON NETWORK MOTIFS IN IP INTERCONNECTIONS NETWORKS OF THREE ASes.

AS Number	Z-Score				
	 id 204	 id 206	 id 280	 id 460	 id 904
AS6395	377	-	9.51	43.84	148.39
AS5111	329.29	36.42	-	74.63	73.57
AS3549	154.8	5.38	37.87	19.51	-

in three large ISPs, Global Crossing, France Telecom, and Broadwind.

Although *mfinder* is a very useful tool for identification of important motifs, it is not designed to be used for network clustering. In our work we do not look for a specific motif in the network, but for highly connected clusters. However, we do search for ‘bi-fun’s (id204) repetitions under a certain weight constraints as cores of the PoPs.

The algorithm works on the IP interface graph of a single AS. We use the following steps to reduce the IP level graph  $G(V,E)$  to a PoP level network.

**Initial Partition.** We remove all edges with delay higher than 5ms (namely 500km) and edges with a shorter delay but with a small number of measurements<sup>2</sup>. As a result, a non-connected graph  $G'$  is obtained. Then, for each connected component of  $G'$  we build an induced sub graph<sup>3</sup> of  $G$ . In cases when two or more sub graphs are connected by a few 5ms or less edges in  $G$ , they are also connected to a single induced sub graph. Now, each connected group is a candidate to become one or more PoPs.

There are two reasons for a connected group to include more than a single PoP. The first and most obvious reason is geographically adjacent PoPs, e.g., New York, NY and Newark, NJ. The other is caused by wrong delay estimation of a small amount of links. For instance a single incorrectly estimated link between Los Angeles, CA and Dallas, TX might unify the groups obtained by such a naive method.

#### Refined Partition.

<sup>2</sup>Since, at this stage, we are trying to avoid false indication of PoPs, we take into account only links with a reliable delay estimation. Thus, we selected to use links with at least 10 traceroute measurements [17].

<sup>3</sup>A subgraph  $S$  of a graph  $G$  is said to be induced if, for any pair of vertices  $u$  and  $v$  of  $S$ ,  $uv$  is an edge of  $S$  if and only if  $uv$  is an edge of  $G$ . In other words, induced subgraph is one that consists of some of the vertices of the original graph and all of the edges that connect them in the original graph.

(a) *Parent-Child classification.* Next, we check whether each connected group has more than one PoP. We note that each suspected partition looks like a collection of highly connected bipartite graphs with rich connectivity between them. Then we divide the whole partition to parents and children according to the measurement direction (each node or group of nodes simultaneously can be parents of one bipartite and children of another) in the bipartite graph, in this operation we ignore the weights of the edges. The minimal size of each group is two nodes, the process is described in Section II-A and the example of such a classification is shown in Fig. 1.

(b) *localization.* Using the high connectivity of the bipartite graph, for each group (parents or children) we divide the group to the physical collocations using the localization algorithm defined in Section II-B.

(c) *Unification.* Unifying *parent/child* group to the same PoP. If *parent pair* and *child pair* groups are connected<sup>4</sup>, then we calculate the weighted distance between the groups; if it is smaller than a certain threshold the pair of groups is declared as belonging to the same PoP.

#### Final Refinements.

(a) *Unification of loosely connected components.* In some cases, e.g., due to insufficient measurements, different parts of a PoP are only loosely connected in a way that does not form a bi-fan motif, in the extreme case only a single link connects two parts of a PoP. This will not allow the unification process, which is described above, to identify the parts as belonging to the same PoP. Thus, we look for connected components (PoP candidates) that are connected by links whose median distance is very short (we chose 1mSec). Note that at this point, due to the unification process, the graph shrank considerably, and thus the search for ‘close’ components is easy.

<sup>4</sup>If they are connected, by definition more than one edge connect the two groups

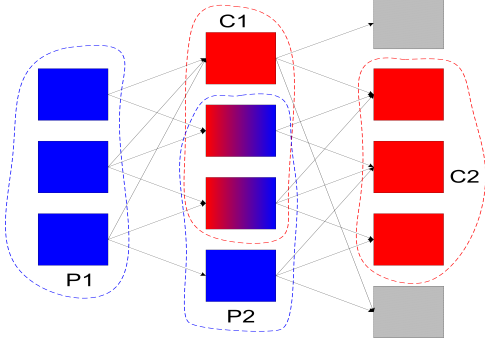


Fig. 1. Parent-Child classification: blue nodes - *parent pair*, red nodes - *child*, blue and red nodes - both parent and child, gray nodes - not classified

### (b) Singleton Treatment

At the end of the process, the ISP graph has evolved through the multiple node unifications described above into a graph that is comprised of several multi-nodes (the PoPs) and a larger number of nodes (IP interfaces) that were not assigned to any PoP. Typically, these nodes have only one or two links connecting them to the rest of the graph, and the path from a node to the closest PoP is in most cases one hop and sometimes two. We connect a singleton to the closest PoP, providing the distance (in miliSec) is shorter than a small threshold.

#### A. Parent-Child classification

A pair of nodes is marked as *parent pair* if **both** of them points to two or more nodes. We combine all the *parent pair* nodes into groups by pairwise unifying *parent pair* nodes. For example in Fig. 2, nodes {1,2}, {2,5} and {3,4} are defined as *parent pairs*, and are united to two *parent pair* groups {1,2,5} and {3,4}. In a similar way, two nodes are marked as *child pair* if **both** of them are pointed by at least two nodes, and *child pair* groups are defined like wise. Some nodes may belong to both categories and it is allowable for a node to belong to one *parent pair* group and to one *child pair* group. By definition, if a node belongs to two or more groups of the same kind, these groups are unified. Fig. 1 shows an example of *parent/child* classification.

#### B. Localization algorithm

The localization algorithm input is a highly connected bipartite graph  $G(V, E)$  with a weight function  $W : E \rightarrow \mathbb{R}$  representing the estimated physical link delay, as shown in Fig. 2. The other input to the algorithm is a partition of the graph to the *parent/child* groups as described in section II-A. The localization algorithm checks whether nodes of the same type (*parent/child*)

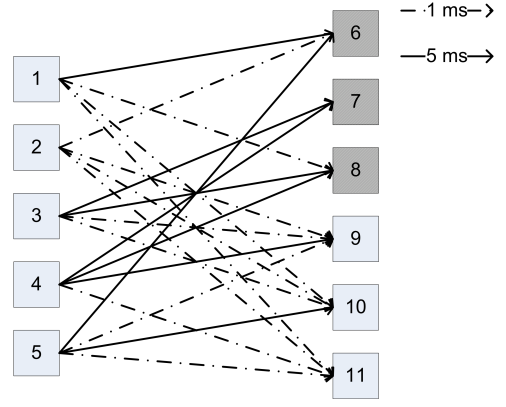


Fig. 2. Bipartite graph example, on the right side dark and bright nodes belongs to different collocation

belong to the same physical collocation. For this task the algorithm takes advantage of the topological structure of the group, for instance if we check the parent group  $P$  we note that each child node of the group is pointed by at least two parent nodes. Comparing the delays from the *child pair* nodes we can partition nodes of the *parent pair* group to one or more geographic collocations.

Formally, we represent each member of a group of two or more nodes (either *parent pair* or *child pair* group) in a coordinate space of the nodes that points to them using the weight of the edges. Next, we check the distance between each pair of nodes in that coordinate space. We assume that the link delay estimation errors [17] are caused mainly by an impulse noise, i.e., most of the measurements are fairly precise or have only small noise, while a small portion of the measurements may have large errors. Therefore, unlike the Gaussian noise case, where Euclidean distance is used as a representation of the distance between nodes, we compare the similarity over the coordinates. An example is shown in Fig. 2: nodes 6 and 10 are pointed by nodes 1,2,3, and 5. Taking into account the delays, one might assume that nodes 1,2, and 3 are at the same location while node 5 is located elsewhere. Nevertheless, nodes 9 and 11 imply that node 5 is collocated with nodes 1,2, and 3. In addition, using the weights we can conclude that the dark nodes (6,7,8) are located on a different location than the bright nodes (9,10,11).

We propose the following deterministic algorithm to classify the locations of nodes in the bipartite graph. For each pair of parent nodes  $(u, v) \in P, u \neq v$ , we define the ‘common children’ group,  $CC$  by

$$CC(u, v) = \left\{ w \in G \mid (u, w) \in E \cap (v, w) \in E \right\} \quad (2)$$

We denote the members of  $CC(u, v)$  as  $\{cc_1, cc_2, \dots, cc_m\}$ . Using the weights of the edges from the pair of parent nodes to the ‘common children’,  $W(u, cc_i)$  and  $W(v, cc_i)$ , we calculate the ‘Error Ratio’ vector,  $ER$ :

$$\overline{ER}(u, v) = \left[ \frac{W(u, cc_1)}{W(v, cc_1)}, \frac{W(u, cc_2)}{W(v, cc_2)}, \dots, \frac{W(u, cc_m)}{W(v, cc_m)} \right]$$

The important property of  $|\log(\overline{ER}(u, v))|$  is that for coordinates with a small relative error the values of the elements in  $ER(u, v)$  will be rather small, and will increase with a loss of the accuracy. Therefore comparing  $er(u, v) = \text{median}(|\log(\overline{ER}(u, v))|)$  to a certain threshold gives a proper indication for the accuracy in the majority of the measurements.

We use the  $er$  values for the parents, to partition parents group to smaller parent groups which are geographically collocated. For this end, we produce a weighted clique of all the parent nodes in a group, where the weight of the edge  $(u, v)$  is  $er(u, v)$ . We remove all the links with a weight below a certain small threshold. Each connected component in the remaining graph becomes a parent group for the next step. To summarize, we partitioned the parent group to several parent groups that are geographically co-located.

The same process is repeated for child groups, where the error vectors are calculated by the distances to the common parents.

This kind of localization helps us to overcome a relatively large number of errors. However, if more than half of the measurements to a certain node are incorrect, the algorithm may fail to determine its location. Otherwise, there is no impact on the overall performance. Those ‘badly’ measured nodes might not become a part of the correct PoP, but the PoP map will be formed correctly in spite of them, i.e., no new PoPs will be created.

### III. RESULTS

We have run the algorithm described above for the hundred AS networks with the highest degree (Fig. 3). In 17 of them, the algorithm failed to identify even a single PoP. An examination of their identity revealed that 3 were exchange points (LIE, DE-CIX, and AMS-IX) and one was an experimental peering point, all four should not have PoPs; 5 were ASes in East Europe. However, a few of these ASes were networks where DIMES failed to supply a reasonable number of IP level measurements. For example, DIMES measured only 25 IP level links in AS 7911 that belongs to Level 3 Communications and only 10 of them had more than 10 measurements in the

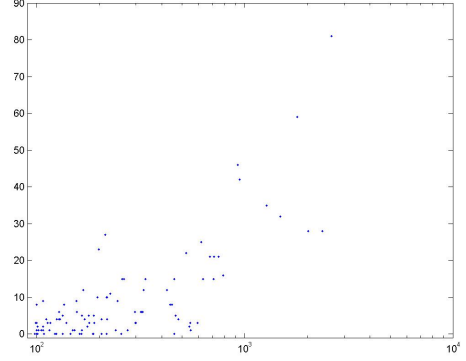


Fig. 3. The number of PoPs found (Y-axis) for the 100 top degree ASes (X-axis shows the AS degree).

two day sample, hardly enough data for the algorithm to run. The algorithm found more PoPs for high degree ASes, which is a positive sign.

We selected a few networks for a detailed evaluation of the algorithm performance. We report results for Broadwing, a US wide ISP that publishes its network structure on the web [18]. Visually, there is a great affinity of the obtained map (Fig. 4(a)) and the published networks (Fig. 4(b)), note that the published network has layer 2 nodes that cannot be detected with *traceroute*. Out of the 8 IP core sites we identified 6 (Hayward, Salt Lake City, Ft. Worth, Atlanta, Washington, and New York City) and missed Indianapolis and Seattle, for the latter we simply did not have measurements passing through. From the additional IP edge aggregate sites (Broadwing terminology) we have detected 4. We missed Miami (for which we had just a few measurements), Dallas (which we failed to distinguish from Ft. Worth), failed to distinguish Silicon Valley PoPs from Hayward, and failed to distinguish the smaller New England PoPs from NYC and Washington. It is not clear to us if closely located and highly interconnected PoPs can be separated using an automated algorithm.

All the edges we found (see Fig. 4(a)) were correct, besides two that we believe are layer 2 tunneled connections, thus true from the IP level aspects. To understand this, we note that in the network shown in Fig. 4(a) there are two triangles in which the sum of the delays along two edges approximately equals the delay on the third edge ( $nywk \rightarrow chcg + chcg \rightarrow cncn \approx nywk \rightarrow cncn$  and  $dlls \rightarrow lax + lax \rightarrow hywr \approx dlls \rightarrow hywr$ ). The ‘triangle equality’ implies that the long edges do not exist at the physical level, but that the  $nywk \rightarrow cncn$  link passes through Chicago in the example above, and

