Resolving IP Aliases in Building Traceroute-Based Internet Maps

Mehmet H. Gunes, *Member*, IEEE and Kamil Sarac, *Member*, IEEE

Abstract— **Alias resolution, the task of identifying IP addresses belonging to the same router, is an important step in building traceroute-based Internet topology maps. Inaccuracies in alias resolution affect the representativeness of constructed topology maps. This in turn affects the conclusions derived from studies that use these maps. This paper presents two complementary studies on alias resolution. First, we present an experimental study to demonstrate the impact of alias resolution on topology measurement studies. Then, we introduce an alias resolution approach called *Analytic and Probe-based Alias Resolver (APAR)*. APAR consists of an analytical component and a probe-based component. Given a set of path traces, the analytical component utilizes the common IP address assignment scheme to infer IP aliases. The probe-based component introduces a minimal probing overhead to improve the accuracy of APAR. Compared to the existing state-of-the-art tool *ally*, APAR uses an orthogonal approach to resolve a large number of IP aliases that *ally* fails to identify. Our extensive verification study on sample data sets shows that our approach is effective in resolving many aliases with good accuracy. Our evaluations also indicate that the two approaches (*ally* and APAR) should be used together to maximize the success of the alias resolution process.**

I. Introduction

Internet topology measurement studies consist of three phases: (1) topology collection, (2) topology construction, and (3) topology analysis. Compared to the studies in topology collection and topology analysis, the amount of work in topology construction is fairly limited. As we briefly discuss below, topology construction is not a straightforward process. Inaccuracies in this process may significantly affect the accuracy of the observations or results obtained in the measurement study [1], [2], [3].

Most router-level measurement studies utilize the well-known Internet debugging tool, *traceroute* [4], or its variants [5], [6], [7], [8]. Traceroute returns a path from a local system to a given remote system by tracing the routers in between. It uses TTL-scoped probe packets to obtain ICMP error messages from the routers on the path. By collecting the source IP addresses from the incoming ICMP packets, traceroute returns the path as a sequence of IP addresses each representing a router between the local system and the remote destination. During topology collection, load balancing routers may affect the accuracy of path traces. Augustin et al. [8] discusses the problem and proposes a traceroute variant to ensure path accuracy in the presence of per-flow-based load balancing routers.

After the collection of the path traces, the information needs to be processed to build the corresponding network topology. This step involves a couple of tasks including (1) resolving anonymous routers that are represented by '*'s in traceroute outputs and (2) resolving IP addresses belonging to the same router. Topology measurement studies should pay the due attention to these tasks to obtain a representative topology map. The first task has to do with the fact that not all routers respond to traceroute probes all the time. Existing solutions to this problem include a graph theoretic approach [9] and a graph based induction approach [10]. The second task is an artifact of the traceroute-based topology collection procedure and is the main focus of this paper.

Routers have multiple interfaces each one having a different IP address. A router may appear on multiple path traces with different IP addresses. Therefore, there is a need to identify IP addresses belonging to the same router. This task is named as *IP alias resolution*. Without alias resolution the resulting topology map may be significantly different from the real topology.

Several mechanisms have been proposed to resolve IP aliases [11], [12] and few alias resolution tools have been developed including *iffinder* [13] and *ally* [14]. These tools use an active probing approach to resolve IP aliases. They are easy-to-use and provide a convenient way to verify if a given pair of IP addresses are alias or not. On the other hand, these tools depend on the participation of the routers in terms of responding to queries directed to themselves. This dependence introduces limitations to the success of alias resolution task as some network administrators configure their routers to ignore active probes directed to themselves. As an example, in our recent study, we observed that 40% of 7073 IP addresses we probed with *ally* did not return a response [1]. This observation motivates the following questions: (1) *what is the impact of alias resolution on topology measurement studies?* and (2) *how can we improve on alias resolution process to build more representative network topologies from a set of collected path traces?*

Given the fact that traceroute-based path traces are used in various research areas [1], [12], [15], [16], the alias resolution process may affect the observations made using this data. Even though several papers reported the impact of poor alias resolution on some specific measurement studies [1], [2], to the best of our knowledge, there is no systematic study that quantifies the impact of poor alias resolution on topology measurement studies. The first part of the paper presents our experimental study on the impact of alias resolution on various topological characteristics. The results indicate that the success rate of the alias resolution process significantly affects the observed characteristics. Section III presents a summary of these results.

In the second part of the paper, we present a new alias resolution approach called *Analytic and Probe-based Alias Resolver (APAR)*. APAR consists of an analytical component and a probe-based component. Given a set of path traces, the analytical component [17] utilizes the common IP address assignment scheme (see RFC 2050) to infer IP aliases. Contrary to probe-based approaches, it does not require router participation for alias resolution purposes. Note that router participation is required during the traceroute-based path collection process as otherwise

M. H. Gunes is with the Department of Computer Science & Engineering, University of Nevada, Reno, NV 89557 and K. Sarac is with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080, (emails: gunes@cse.unr.edu and ksarac@utdallas.edu).

we would not have a practical way to collect data to build topology maps. APAR operates in two phases. In the first phase, it uses the common IP address assignment practices to detect the subnets within the set of collected path traces. In the second phase, it uses identified subnets to align symmetric segments of different path traces and infers alias pairs among involved IP addresses. Path asymmetry is a commonly observed characteristic in the Internet. However, our approach does not require complete path symmetry. It uses symmetric path *segments* to resolve aliases. If a given pair of path traces are completely disjoint/asymmetric, then there are no alias pairs to resolve in the data set (see Section V.D). The probe-based component introduces a minimal probing overhead (one probe per IP address) to improve the accuracy of APAR. Note that *ally* and APAR use two orthogonal mechanisms to resolve IP aliases. Our comparisons between the two approaches show that the two tools need to be used together to maximize the success rate of the overall alias resolution process.

In summary, the contributions of this paper include an experimental study that demonstrates the impact of alias resolution on observed topological characteristics; a new alias resolution approach, APAR, that depends on an analytical approach to infer a large number of IP aliases; an experimental study that analyzes the performance and the accuracy of APAR. Accordingly, after presenting the related work in Section II, we divide the paper into two parts. In the first part, in Section III, we present a summary of the results of our experimental study on the impact of alias resolution on topology measurement studies. In the second part, we first highlight the main idea of the APAR algorithm in Section IV. In Section V, we present the details of the APAR algorithm and in Section VI we present our experimental evaluations of the algorithm in detail. Finally, Section VII concludes the paper.

II. Related Work

The initial work on alias resolution utilizes source IP addresses [11]. Given a set of IP addresses, the algorithm sends probe packets to IP addresses to solicit ICMP error messages. Probing an IP address IP1, if the returning ICMP port unreachable message has a source IP address of IP2 then IP1 and IP2 are set as aliases. *Mercator* [18] and *iffinder* [13] use this method. *Mercator* improves [11] by sending multiple probes to the given IP addresses from a number of different source-routing capable routers. *iffinder* discovers additional aliases by using the Route Record option of IP (RFC 791). The second approach uses potential similarity in IP identification field values in the returning ICMP packets [12]. Since some operating systems implement IP identification value as a monotonically increasing counter, successive packets originating from such a router would have consecutive IP identification values. *Ally* tool [14] combines the address based method with the IP identification based method to classify a pair of IP addresses as *alias*, *not-alias*, or *unknown*.

These methods are simple and powerful in resolving IP aliases. However, being active probing approaches, they introduce probing overhead ($O(n^2)$ probes in the case of *ally* where n is the number of IP addresses). They also depend on routers participation by replying to the probe messages. Considering the increasing volume of measurement traffic in the Internet [19], many ISPs configure their routers to respond to traceroute probes but ignore or rate limit direct probes destined to themselves. This practice affects the utility of the existing alias resolution tools [17].

Compared to the existing approaches, APAR introduces a more scalable approach to resolve IP aliases. The analytical component introduces no probing overhead and requires no router participation. The probe-based component incurs a significantly lower probing overhead ($O(n)$) to improve the overall accuracy of the APAR approach. Note that being two orthogonal approaches, APAR and *ally* do not compete but complement each other in maximizing the success rate of the overall alias resolution process.

III. Quantifying the Impact of Alias Resolution

In this section, we study the impact of alias resolution in building traceroute-based topology maps. We experimentally analyze the effect of imperfect alias resolution on a broad set of graph properties on a genuine topology. Due to size limitations, we present only a subset of the results to summarize our findings (see [20] for additional results on synthetic topologies).

Analysis Procedure: In our analysis, we use a topology map provided by iPlane [21] on July 17, 2007. The map has about 407K edges and 81K nodes. We annotate the graph such that each edge incident on a node introduces a unique node identifier called an *interface identifier*. Then, we take six different topology samples by collecting shortest paths (to emulate traceroute) among a number of (source,destination) pairs including S1=(10,1000), S2=(20,2000), S3=(30,3000), S4=(100,100), S5=(200,200), and S6=(300,300) pairs. We refer to the first three samples as `(k,m)`-traceroute data and the other samples as `(n,n)`-traceroute data for ease-of-presentation. Each path trace includes an interface identifier for each intermediate node on the path.

After collecting path traces, we apply alias resolution with different success rates including 0%, 25%, 50%, 75%, 85%, 95%, and 100% to generate different sample topologies from the same set of path traces. Here, 0% indicates that alias resolution fails for all nodes in the network and 100% indicates that alias resolution succeeds for all nodes in building a sample topology. In general, a failure in alias resolution results in *false negatives*. On the other hand, a *false positive* is introduced when two addresses are incorrectly considered as alias. We also consider the effect of false positives on the graph constructed with 100% alias resolution success rate. We consider false positive rates of 0%, 5%, 10% and 15% where percentages indicate the ratio of incorrectly formed pairs among identified alias pairs. Finally, we analyze various properties of the resulting topologies to quantify the impact of alias resolution on the observed topological characteristics. The considered graph characteristics can be grouped into size, node degree, clustering, path length, and betweenness related

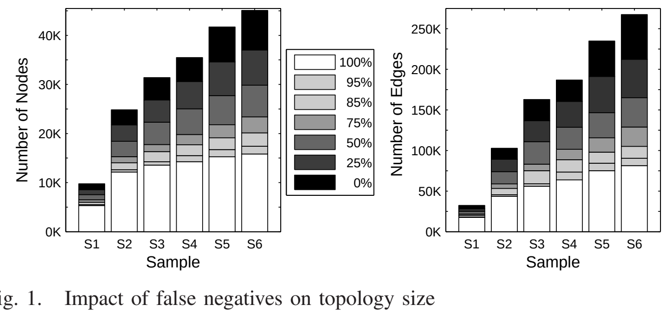


Fig. 1. Impact of false negatives on topology size

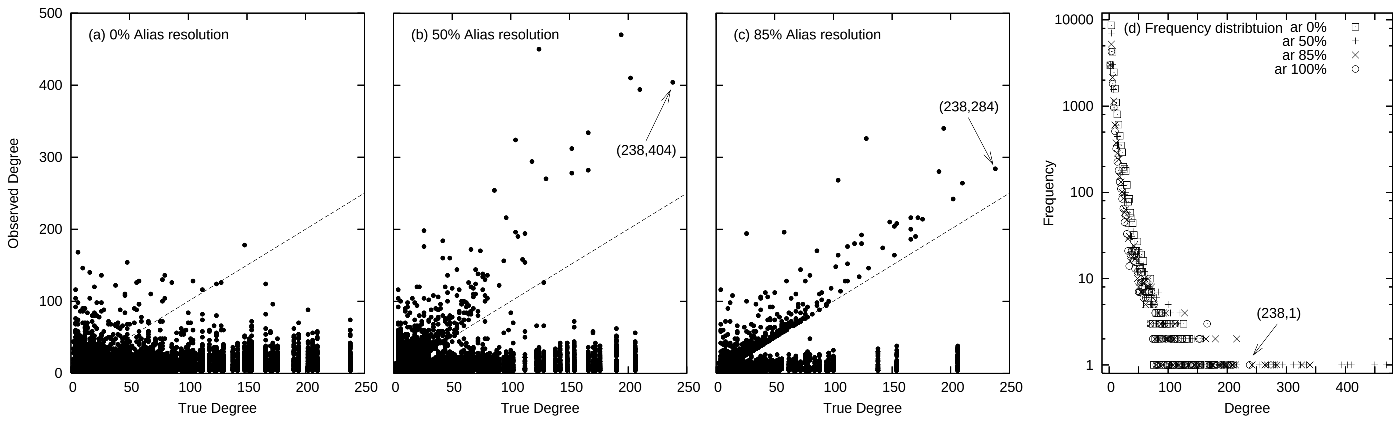


Fig. 2. Degree comparison for (20,2000)-sample topologies

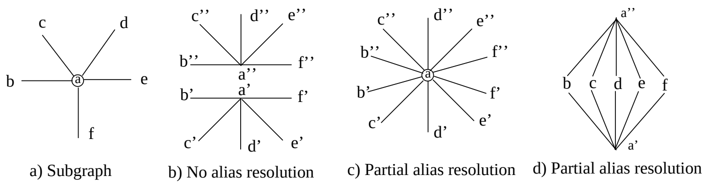


Fig. 3. Effect of partial alias resolution

characteristics. In the analysis, the sample topologies obtained with a 100% alias resolution success rate and 0% false positives correspond to *real* sample topologies.

Topology Size: Topology size, in terms of the number of nodes n and links m, is the basic information regarding a network. It also defines the average node degree k as $k = 2m/n$. According to the experiment results, the alias resolution success rate, i.e., false negatives, has a big impact on the topology size as seen in Fig. 1. In the figure, each color shows the additional artificial nodes/links added into the final topology map with diminishing success rate. On average, the number of nodes and edges is 2.38 and 2.74 times of the real topology, respectively, when alias resolution success rate is 0%. The number of artificial links due to imperfect alias resolution is more than that of artificial nodes in the sample topology. This is because, in the worst case, a node of degree d appears as d different nodes each with a degree d, introducing $d * d - d$ artificial links.

In contrast to false negatives, false positives reduce the topology size by incorrectly merging unique nodes (figures not shown). On average, the number of nodes is reduced by 5.3%, 10.5% and 15.5% when 5%, 10% and 15% false positives, respectively, exist in the set of detected alias pairs under 100% alias resolution success rate. Similarly, the number of edges is reduced by 0.01%, 0.03% and 0.05% with 5%, 10% and 15% false positives, respectively. The reduction in the number of edges seems to be insignificant when percentages are considered but it is linear with respect to the reduction in the number of nodes.

Node Degree: The accuracy of the alias resolution process has considerable impact on the node degree-related characteristics. Although one may intuitively expect an improvement in the accuracy of degree-related characteristics with an increasing success rate of the alias resolution process, we may not necessarily observe such a trend all the time as seen in Fig. 2. We use a small subgraph in Fig. 3 to analyze changes in node degree. In the figure, ‘no-alias resolution’ case (Fig. 3-b) results in a better approximation to (1) the degree of node a and (2) the average and the maximum degrees of the original subgraph (Fig. 3-a) compared to the ‘partial alias resolution’ case (Fig. 3-c) when we resolve aliases only for a. Similarly, maximum degree characteristics of the sample topologies show a trend where the maximum degree of a sample topology first increases and then decreases as the success rate of the alias resolution process increases from 0% to 100% for all sample topologies. The example scenario shown in Fig. 3 presents a potential explanation for this trend where the maximum degree increases from 5 (in Fig. 3-b) to 10 (in Fig. 3-c) and then goes back to its correct value 5 (in Fig. 3-a).

Next, we study several sample topologies to observe the changes in node degrees as the success rate of the alias resolution process increases. This helps us gain more insight into the impact of the alias resolution process on the node degree characteristics. Fig. 2-a,-b,-c show changes in node degrees for the (20,2000)-sample topology for 0%, 50%, and 85% alias resolution success rates. In these figures, ‘Observed Degree’ indicates the degrees of the nodes in the sample topology with imperfect alias resolution and ‘True Degree’ indicates the degrees in the sample topology with perfect alias resolution. Each point in these figures may correspond to one or more nodes in the sample topology. The number of nodes corresponding to each point is presented in the frequency distribution graph in Fig. 2-d. As an example, the ‘⊙’ tick at location (238,1) in Fig. 2-d indicates that there exists only one node with an ‘Observed Degree’ of 238 under 100% alias resolution success rate. This node corresponds to the node at (238,404) in Fig. 2-b and (238,284) in Fig. 2-c. The coordinate (238,404) indicates that the ‘Observed Degree’ of the node is 404 under 50% and 284 under 85% alias resolution success rates.

We now present several observations about the results presented in these figures. The points above the x=y line in Fig. 2-a,-b,-c correspond to overestimation of node degrees and the points below the x=y line correspond to underestimations of node degrees in sample topologies. In general, overestimation is caused by alias resolution problems at the neighboring nodes of a given node (see Fig. 3-c). Similarly, underestimation is caused by alias resolution problems at the node itself. In addition, the comparison of Fig. 2-a,-b,-c show that the *observed* maximum degree of the graph increases from 178 in Fig. 2-a to 470 in Fig. 2-b. It then goes down to 340 in Fig. 2-c (and down to 238 with 100% alias resolution success rate). Another observation

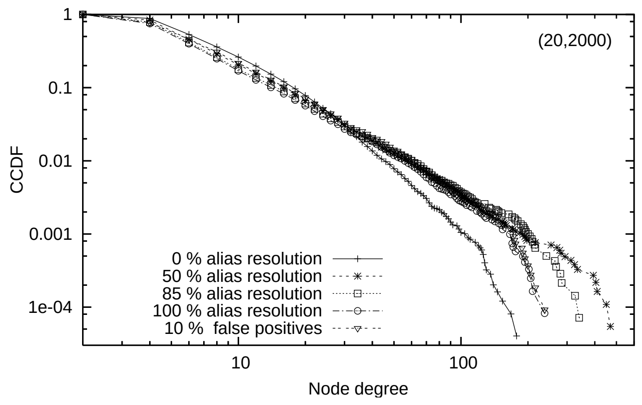


Fig. 4. Degree distribution for (20,2000)-sample

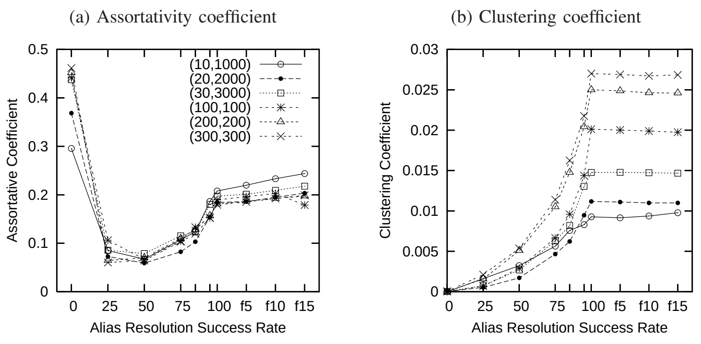


Fig. 5. Assortativity and clustering coefficients

from the figure is that alias resolution problems at a node may introduce a significantly large number of artificial nodes in the resulting sample topologies. As an example, according to Fig. 2-d, there is only one node with *true degree* of 238 in the *real* sample graph (i.e., refer to (238,1) in Fig. 2-d). On the other hand, Fig. 2-a shows a large number of nodes with *observed degrees* less than 238 that correspond to a node with a *true degree* of 238. Finally, we observe that as the alias resolution success rate increases some of the underestimation cases change to overestimation (compare Fig. 2-a vs. Fig. 2-b). This indicates that although the alias resolution problems of the corresponding nodes are fixed, there exists some neighbors of these nodes with alias resolution problems causing overestimation.

Furthermore, we analyze the effect of false positives on node degrees under 100% alias resolution success rate (figures not shown). In most cases, adding false positives increases the observed degree of falsely merged nodes. This is expected because merged nodes' degrees are combined in the new node. On the other hand, there are few nodes whose degrees reduce with increasing false positive rates. This occurs at nodes that are connected to both of the merged nodes.

Degree Distribution: Degree distribution represents the probability *P(k)* that a randomly chosen node has degree *k*. Degree distribution has been used to characterize network topologies [22] and several topology generators use this characteristic to generate synthetic topologies [23], [24]. In our experiments, we observe that degree distribution changes with the changing success rate of the alias resolution process, but different effects are observed with different samples. Fig. 4 presents sample results for the impact of alias resolution, i.e., false negatives and false positives, on degree distribution characteristics of (20,2000)-sample. In all samples, we first observe an overestimation at high degrees. As alias resolution success rate improves, they converge to the distribution of real topologies. In general, resolving aliases at half of nodes provides a degree distribution whose power-law slope matches the real topology while its tail is greatly distorted. The tail of distribution mends with improving alias resolution and becomes close to the original at 95% alias resolution. Moreover, false positives slightly diverge the distribution with the presence of no false negatives. This suggest that the impact of false positives is less than that of false negatives for the degree distribution.

Joint Degree Distribution: Joint Degree Distribution (JDD) $P(k, k')$ characterizes the degree relation of nodes, i.e., it reports the probability that a node of degree k and a node of degree k' are connected [25]. Assortative coefficient r is a summary statistic of JDD and it measures the tendency of a network to connect nodes of the same or different degrees [26]. Positive values indicate assortativity (i.e., most of the links are between similar degree nodes) and negative values indicate disassortativity. As seen in Fig.5-a, assortativity of the topologies changes drastically in samples with an increase in alias resolution success rate (f5, f10, and f15 on x-axis show 5%, 10%, and 15% false positives). `(n,n)`-samples seem to be assortative with 0% alias resolution but non-assortative with 25% and 50% alias resolution. As alias resolution gets closer to 100%, they appear to be slightly assortative. Furthermore, except one case, adding false positives slightly increases the assortativity of topologies.

Clustering: Clustering $C(n)$ characterizes the density of the connections in the neighborhood of a node n [27]. We analyze clustering distribution with respect to node degree and observe an increase with increasing alias resolution success rate. In addition, we analyzed clustering coefficient C, which is a summary metric of clustering. It is the ratio of the number of triangles to the number of triplets. All samples yield a clustering coefficient of 0 with 0% alias resolution success rate (Fig. 5-b). Then, it always increases with the increasing alias resolution success rate. Finally, adding false positives slightly alter (increase or decrease) the clustering coefficient of the resulting topology. As expected, `(n,n)`-samples have higher clustering than `(k,m)`-samples.

Characteristic Path Length: Characteristic path length (CPL) $\bar{l}$ measures the average of the shortest path lengths between all node pairs in a network. On average, CPL values are 67.6%, 39.7%, 22.6%, 9.4%, 5.2%, and 1.3% higher than CPL of the original topology with 0%, 25%, 50%, 75%, 85%, and 95% alias resolution success rates, respectively, (figures not shown). Changes in the CPL also correlate with the hop distribution characteristic which shows the average percentage of the nodes reached at each hop. As an example, in the case of (10,1000)-sample topologies, while 21.4% of the nodes are reachable within 7 hops with 0% alias resolution, this rate increases to 92.8% with 100% alias resolution. Similarly, adding false positives reduce the CPL values. On average, CPL values are reduced by 2.1%, 3.8%, and 5.6% with respect to the real topology when 5%, 10%, and 15% false positives exist.

Betweenness: Betweenness $\sigma(n)$ is a mesure of centrality. It reports the total number of shortest paths that pass through node n [28]. Usually betweenness is normalized with the maximum

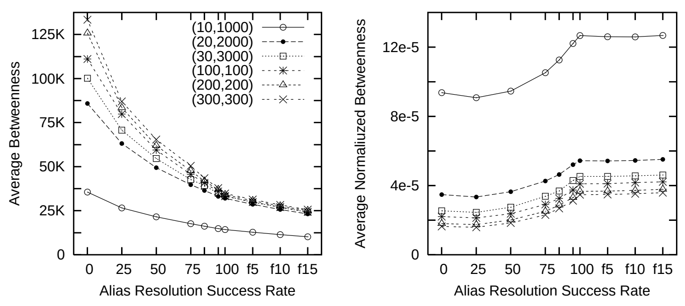


Fig. 6. Betweenness (average, normalized average)

possible value, i.e, $n(n-1)$. We analyze betweenness distribution and observe considerable changes with the increasing alias resolution success rate as seen in Fig.6. Average betweenness reduces with an improvement in the alias resolution success rate. This is due to the fact that as the alias resolution rate improves, artificial nodes are removed from the network causing a reduction in the number of path pairs that contribute to the betweenness (e.g., compare $\sigma(a)$ in Fig. 3-a and Fig. 3-c). On the other hand, normalized betweenness presents a reverse trend where as the alias resolution success rate increases, the normalized betweenness also increases. This is due to the fact that the normalized betweenness of the artificially replicated copies of a node n are less than the normalized betweenness of the node n when the rest of the network is the same (e.g., compare $\sigma(a)$ in Fig. 3-a and Fig. 3-d). Finally, the changes with false positives is less disruptive than false negatives.

From the above analysis on various characteristics, we conclude that the completeness and the accuracy of the alias resolution process has a significant impact on the results of studies that use traceroute-based topology data. In most of the characteristics, false negatives had more impact than false positives, e.g. 85% alias resolution success rate vs. 15% false positive (one important exception is the characteristic path length). This is mainly because the number of possible false negatives is larger than that of possible false positives and hence its impact. The number of false positives is proportional to the number of nodes with 100% alias resolution. However, the number of false negatives is proportional to the number of nodes in the initial topology which is more than double of the number of nodes with 100% alias resolution. Results also suggest that `(n,n)`-samples are affected more by imperfect alias resolution. `(k,m)`-samples, by nature, have fewer routers with alias resolution problem due to the tree-oriented nature of collected path traces. In addition, the impact of imperfect alias resolution increases as the size of the sample topology increases.

IV. Observations

In this section, we present a summary of IP address assignment practices in the Internet and show how it can be used to identify IP aliases from a given set of path traces.

A. IP Address Assignment Practices

IP address space is a scarce commodity and is used in a systematic way with a great care. The IP address assignment mechanism adheres to the guidelines presented in the Internet Registry IP Allocation Guidelines (RFC 2050). Basically, IP addresses belonging to a domain or an ISP network are divided into subnet ranges for each connection medium. Each subnet has a network address and each interface, belonging to an end host or a router within the subnet, gets an IP address from the range of the network address given to the subnet. In general, up to n device interfaces can be connected using a /x subnet where $n = 2^{32-x} - 2$. The first x bits of assigned IP addresses denote the subnet address and the last $32 - x$ bits identify the device interfaces within the subnet.

B. Identifying IP Aliases Using Subnets

The subnet relation between IP addresses of the devices can be used to identify IP alias pairs. In this subsection, we demonstrate how this can be done in subnets with point-to-point links and multi-access links separately.

Using Point-to-Point Links: The smallest subnet in the Internet is built by using a point-to-point link to connect two device interfaces. A /30 subnet or a /31 subnet (the latter is introduced in RFC 3021) is defined and used to assign IP addresses to the interfaces in this type of networks.

IP address assignment on point-to-point links can be used to identify symmetric path segments in the collected path traces. Given a set of path traces, one can compare segments of different path traces to find IP addresses, say IP$_A$ and IP$_B$, such that the two addresses belong to the same /30 or /31 subnet. Once such a match is observed, IP aliases can be inferred from the proper alignment of the path traces. We explain this with an example.

Consider the sample topology in Fig. 7 where $h1$, $h2$, $h3$, and $h4$ are end-hosts and $r1$ and $r2$ are routers all connected using point-to-point links. The lower case letters a, b, ..., j represent interface IP addresses. Assume that we have two path traces one from $h1$ to $h3$ as (a, b, j, e) and the other from $h2$ to $h4$ as (c, d, i, g) taken from this network. Comparing the two path traces, we observe that i and j belong to a /30 subnet. Based on the observed subnet relation, we can align the path traces as

```
a  b ↙ j  e   (trace from h1 to h3)
g  i   d  c   (reverse of trace from h2 to h4)
```

and identify IP alias pairs as (b, i) and (d, j). Note that RFC 792 states that an ICMP error message should be sent back with the IP address of the incoming interface (see Section V-D for more details). IP address j is observed from the vantage point $h1$, so its subnet pair i should be closer to $h1$ given that i and j are connected by a point-to-point link. This suggests that the alternative alignment

```
a  b  j ↘ e          (trace from h1 to h3)
      g   i  d  c    (reverse of trace from h2 to h4)
```

cannot be true.

Using Multi-Access Links: Multi-access links are used to connect several device interfaces to form a subnet. In general, these

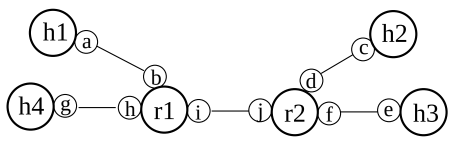


Fig. 7. A sample network between four end-hosts.

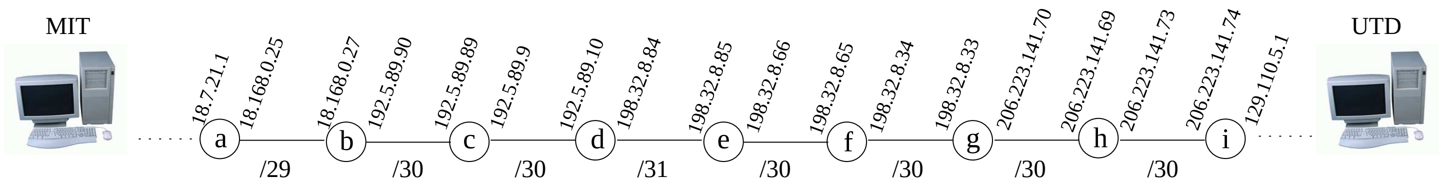


Fig. 8. The inferred subpath between MIT and UTD hosts.

TABLE I
TRACEROUTE RESULTS BETWEEN MIT AND UTD

	MIT-to-UTD (Direct path)	UTD-to-MIT (Reverse path)
1	18.7.21.1	18.7.21.84
2	**18.168.0.27**	**18.168.0.25**
3	**192.5.89.89**	**192.5.89.90**
4	**192.5.89.10**	**192.5.89.9**
5	**198.32.8.85**	**198.32.8.84**
6	**198.32.8.65**	**198.32.8.66**
7	**198.32.8.33**	**198.32.8.34**
8	**206.223.141.69**	**206.223.141.70**
9	**206.223.141.74**	**206.223.141.73**
10	*	129.110.5.1
11	*	129.110.95.1

subnets include more than two interfaces connected to them. When building a subnet, one chooses a subnet that has enough IP addresses for unique address assignment to each interface on the subnet. Similar to the case with the point-to-point links, we can identify IP addresses belonging to larger subnets (subnets with a mask of $/x$ where x<30) and use this information to infer IP aliases. This procedure helps us detect additional IP aliases.

Consider the example in Table I where we present traceroute outputs between two end hosts, one in MIT and the other in UTD networks. The first column shows the MIT-to-UTD trace and the second column shows the *reverse* of the UTD-to-MIT trace. Analyzing the IP addresses of these two traces, we can observe correlations between the IP addresses in the 2^{nd} row until the 9^{th} row. Assuming point-to-point links with /31 or /30 subnets or multi-access link with /29 subnet, we can construct the path segment corresponding to the traces as in Fig. 8. This arrangement can be used to detect IP aliases, e.g., 18.7.21.1 and 18.168.0.25 are IP aliases representing router a, 18.168.0.27 and 192.5.89.90 are IP aliases representing router b, etc.

In summary, we can infer IP aliases from collected path traces utilizing the subnet relation between IP addresses. This analytical approach does not require probing to elicit information from routers but benefits from IP address assignment practices. Additionally, it may be utilized on historical data sets where probing bases alias resolution can not be applied.

V. ANALYTICAL AND PROBE-BASED ALIAS RESOLUTION

In this section, we present a new approach, APAR, to improve the success of the alias resolution process in traceroute-based topology collection studies. The discussion in Section IV demonstrates the potential in leveraging the IP address assignment methodology to infer IP aliases. However, a straightforward application of this approach may introduce a significant amount of inaccurate aliases. In general, any two IP addresses can be considered to be members of the same subnet under some subnet prefix length. Since the subnet information of the underlying network is not available to us, we may end up incorrectly assuming the existence of subnets among a number of IP addresses but in reality these IP addresses may not belong to the same subnet. Therefore, an effective use of this approach requires a carefully designed procedure to accurately form subnets and infer IP aliases.

The main idea in APAR is to use an analytical approach to resolve IP aliases while a probe-based component is included to improve the accuracy of the overall process. The APAR algorithm includes two steps: (1) analyzing IP addresses to identify a set of candidate subnets within the collected path traces and (2) using the identified subnets to resolve IP aliases. In the following, we present each of these steps and the details of the algorithm after defining several terms/symbols that will be employed in the development of the algorithm.

Definition (Router-Level Graph): Let $G = (V, E)$ be a router level network graph where V represents the set of vertices (i.e., routers and end-hosts) and E represents the set of edges (i.e., communication links) connecting the vertices in V. Each vertex $v \in V$ has one or more interfaces $(i_1^v, i_2^v, ..., i_{degree(v)}^v)$ where $degree(v)$ represents the number of interfaces of v. Each interface i_e^v of a vertex v has an *address*, $i_e^v.address$, that is unique in G. An edge $e \in E$ connects two adjacent vertices v_p and v_{p+1} by connecting interfaces $i_e^{v_p}$ of v_p and $i_f^{v_{p+1}}$ of v_{p+1}. $\square$

Definition (Trace): A trace, $trace(v_i, v_j) = (i_a^{v_i}, \ldots, i_e^{v_p}, i_f^{v_r}, \ldots, i_z^{v_j})$, is a subgraph of G where an edge $e_{(v_p,v_r)}$ connects v_p and v_r via interfaces $i_e^{v_p}$ and $i_f^{v_r}$. $\square$

Trace returns a path from v_i to v_j in G reporting an interface $i_e^{v_p}$ for each visited vertex v_p and is determined based on some application specific criteria, e.g., shortest path, minimum cost path, etc. Note that $trace(v_j, v_i)$ may not be equal to $trace(v_i, v_j)$.

Definition (Successor/Predecessor): Given a trace output, $trace(v_i, v_j)$=$(i_a^{v_i}, \ldots, i_e^{v_p}, i_f^{v_r}, \ldots, i_z^{v_j})$, v_r is said to be the *successor* of v_p (shown as v_{p+1}) in $trace(v_i, v_j)$. Similarly, v_p is said to be the *predecessor* of v_r (shown as v_{r-1}). $\square$

Definition (Subnet): A subnet sn_s^x denotes a network whose subnet address is s and subnet mask is of length x. $\square$

From a practical point of view, due to the limited size of the IP address space (less than 2^{32} addresses), the size of the set V is limited. This indicates that given two node interface addresses, $i_e^{v_p}.address$ and $i_f^{v_r}.address$, the two addresses can be assumed to be (1) within the same subnet or (2) in two different subnets based on a subnet mask of length x where $0 \leq x \leq 32$.

Definition (a $\overset{\mathbf{x}}{\longleftrightarrow}$ b): Given two interface addresses $a = i_e^{v_p}.address$, $b = i_f^{v_r}.address$, and a subnet mask length x, ($a \overset{x}{\longleftrightarrow} b$) is a logic operation that returns TRUE if a and b belongs to the same subnet sn_s^x. Else, it returns FALSE. $\square$

In practice, two interfaces with addresses a and b are within the same $/x$ subnet if the leftmost x bits of the addresses match.

A. Subnet Formation

In this subsection, we present our approach to form candidate subnets that will be utilized by APAR. Note that our alias resolution method relies on inferring subnets in the collected data set. If we were to know the underlying subnets, we could group the IP addresses in our data set into these subnets and then use the above mentioned alignment procedure to infer IP aliases. However, the only information available is the set of path traces that provides a number of IP addresses and neighbor relation among the IP addresses within each path trace. Therefore, in this step, we need to analyze the existing data to infer subnets that the IP addresses are involved in.

We use an iterative approach to form all candidate subnets starting from /x subnets ($x < 31$) to /31 subnets using the IP addresses at hand. First, we form all candidate /x subnets from the data set by combining the IP addresses whose first x bits match. Next, we recursively form smaller subnets (e.g., /x, /x+1, …, /31 subnets). At this point, we need to decide if the candidate subnets correspond to real subnets in the Internet or not. That is, even though a given set of IP addresses can map to a, say, candidate /29 subnet, there may not be a real /29 subnet in the underlying network among these IP addresses. Instead, the addresses may belong to two separate /30 subnets. Similarly, the candidate /29 subnet may be part of a bigger subnet. Therefore, we need to use some criteria to eliminate non-existent subnets from our candidate subnet list. We identify several conditions to achieve this. Given that we do not have any information about the real subnets, our conditions are mostly heuristic in nature.

Condition 1: ACCURACY
Given a loop-free path trace, two or more IP addresses from the same subnet cannot appear in the trace without having a successor/predecessor relationship with each other. More specifically, given a subnet sn_s^x

$\nexists\, trace(v_i,\ v_j) \mid (i^{v_p}, i^{v_r} \in\ sn_s^x)$ and
$(i^{v_p}, i^{v_r} \in\ trace(v_i,\ v_j))$ and
$(i^{v_{p+1}} \neq i^{v_r})$ and $(i^{v_{p-1}} \neq i^{v_r})$ □

IP addresses in a subnet should appear next to each other whenever they appear in the same trace. This condition arises from the fact that nodes within the same subnet are directly connected and should appear one hop away from each other in a path trace. This condition detects inaccurate candidate subnets.

Condition 2: COMPLETENESS
A subnet sn_s^x can include up to $2^{32-x} - 2$ IP addresses for assignment and we require that some fraction of these addresses (e.g., half of them) appear in our data set. □

This is a heuristic to help us increase our confidence in the accuracy of the candidate subnets. Without this requirement, it would be easy to form a candidate subnet (likely a large one) using a few IP addresses falling into the same subnet range. However, the existence of a small number of IP addresses within the candidate subnet makes it difficult to verify the accuracy of this subnet. Depending on the completeness ratio, this condition may cause us discard a real subnet of size, say /28, and instead consider one or more smaller subnets of size /29, /30, or /31 that satisfy the completeness criteria. The likely impact of this situation is the omission of possible aliases that could have been identified under /28 subnet but not under the smaller subnets. But the algorithm would still identify a number of aliases using the smaller subnets.

Note that the completeness of subnets can be improved by probing non-observed IP addresses from the subnet range [29]. Additional probing will increase our confidence in identified candidate subnets and help in eliminating incorrect ones. In general, the recently presented approach of [29] will generate more accurate subnets but will require additional probing which we try to minimize in APAR.

Condition 3: PROCESSING ORDER
The output of the subnet formation step is a number of subnets with different subnet mask lengths. During alias resolution, we start our processing by considering the IP aliases introduced by subnets with higher completeness ratio. If there are multiple subnets with the same completeness ratio, then priority is given to the subnets involving more path traces. □

Note that we have more confidence on the accuracy of the subnets with high completeness ratio. Consequently, we consider IP alias pairs inferred from these subnets as more reliable. Hence, we process subnets with higher completeness first. Later on, when two separate alias pairs introduce a conflict with each other, we prefer the ones that are inferred earlier (i.e., inferred using a more complete subnet) and ignore the ones that are inferred later. Note that by definition, all /31 and /30 subnets are 100% complete.

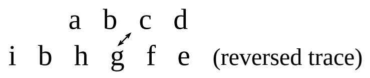
B. Identification of IP Aliases

In this subsection, we present our approach to infer alias IP addresses after obtaining candidate subnets. The alias resolution procedure follows the observations that we present in Section IV. In the following, we present rules to avoid false positives in inferring IP aliases.

Condition 4: NO LOOP
Assuming that path traces are loop-free to start with, the inferred alias pairs should not introduce/suggest any routing loops in any of the path traces. That is, given two candidate alias IP addresses $i_e^{v_p}.address$ and $i_f^{v_p}.address$ where $i_e^{v_p} \in\ trace(v_k, v_l)$ and $i_f^{v_p} \in\ trace(v_m, v_n)$, then

$\nexists\, trace(v_i, v_j) \mid i_e^{v_p}, i_f^{v_p} \in trace(v_i, v_j)$ □

If a routing loop is to emerge in any of the path traces as a result of inferring two IP addresses as aliases, the aliasing is considered to be inaccurate. This situation also indicates that either the alignment of the path segments or the inferred subnet(s) involved in the resolution of the alias pairs are inaccurate. To illustrate this condition, consider two path segments $(\ldots, a, b, c, d, \ldots)$ and $(\ldots, e, f, g, h, b, i, \ldots)$ belonging to two different path traces where $(g \overset{x}{\longleftrightarrow} c)$. Based on the subnet relation, we can align the path segments as



and infer two alias pairs as $(b,\ g)$ and $(c,\ f)$. However, the alias pair $(b,\ g)$ suggests the existence of a routing loop in the second path segment (i.e., both b and g appear in the same path). This suggests that the inferred alias pair $(b,\ g)$ cannot be accurate.

In addition to direct loops as shown above, the inferred alias pairs should not introduce any indirect loops in any of the traces.

An indirect loop occurs when we consider two IP addresses p and r as potential aliases while a previously detected alias of r, say, q appears on the same path with p in a path trace. Finally, IP addresses from the same subnet should not be set as alias. This last rule may not be valid for cases where a device has multiple interfaces connected to a subnet, a rarely used practice.

Condition 5: COMMON NEIGHBOR
Given two IP addresses s and t that are candidate aliases belonging to a router R, we require that one of the following rules hold for setting them as alias:

1) s and t have a common neighbor in some path trace, or
2) there exists a previously inferred alias pair (b, o) such that b is a successor (or predecessor) of s and o is a predecessor (or successor) of t, or
3) the involved path traces are aligned such that they form two subnets, one at each side of the router R. □

These rules help us increase our confidence in inferred IP aliases and help us avoid setting unrelated IP addresses as alias pairs with each other.

We use an example to explain each case. Consider the sample topology in Fig. 9 where $h1$, $h2$, $h3$, and $h4$ represent trace vantage points (e.g., end hosts) and $r1$, $r2$, $r3$, $r4$, and $r5$ represent routers in between. Assume that we have three path traces from this topology as $trace(h1, h4) = (a, b, q, m, g)$, $trace(h2, h1) = (c, d, o, a)$, and $trace(h3, h1) = (e, f, k, o, a)$.

For the first rule above, consider path traces $trace(h1, h4)$ and $trace(h2, h1)$. Observing ($q \xleftrightarrow{x} o$), we align the traces as

```
a b q m g   (trace from h1 to h4)
   ↗
a o d c     (reverse of trace from h2 to h1)
```

From this alignment, we detect two candidate alias pairs as (b, o) and (q, d). We observe that a is a neighbor of both b and o. From the first rule above, we infer the alias pair (b, o). However, at this point we do not have enough evidence for inferring (q, d) as an alias pair. For the second rule, consider path traces $trace(h1, h4)$ and $trace(h3, h1)$. Observing ($k \xleftrightarrow{y} m$) as

```
a b q m g   (trace from h1 to h4)
     ↗
a o k f e   (reverse of trace from h3 to h1)
```

and considering the known alias pair (b, o), we infer the alias pair (q, k). Finally, for the third rule, we can again consider the path traces $trace(h1, h4)$ and $trace(h3, h1)$ and observe ($q \xleftrightarrow{x} o$) and ($k \xleftrightarrow{y} m$) as two subnets at both sides of a router as

```
a b q m g   (trace from h1 to h4)
   ↗ ↗
a o k f e   (reverse of trace from h3 to h1)
```

and infer the alias pair (q, k).

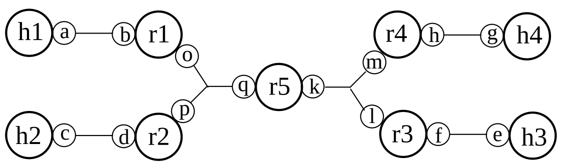


Fig. 9. A sample network using multi-access links.

Condition 6: DISTANCE
Given two IP addresses s and t that are candidate aliases belonging to a router R, s and t should be at similar distances to a vantage point. □

Given a set of IP addresses in the data set, a *ping* query is sent to each IP address. Dissimilarities in the TTL values of the returned *ping* responses are used to identify possibly inaccurate alias pairs. The Distance condition helps improve the accuracy of APAR. It also introduces an active probing component with a probing overhead of $O(n)$ where n is the number of IP addresses in the data set. Note that the same probes may also be used to incorporate the source IP address based alias resolution approach of [11] into APAR. Additionally, one may improve the accuracy, although not considerably, by observing distances from multiple topologically diverse vantage points.

If active probing is not possible, the APAR algorithm can also be used without this component. As we show in the evaluations section, the Distance condition improves the accuracy of formed alias pairs but the algorithm yields a reasonable accuracy level without the use of this condition as well (compare Step 2 and Step 3 rows in Table III-(d)).

C. APAR Algorithm

In this subsection, we present APAR, which produces IP aliases using the aforementioned observations and conditions. Given a set of path traces (i.e., $\bigcup trace(v_i, v_j)$), APAR uses the IP address assignment of subnets to identify symmetric path segments between path traces. By symmetry, APAR locates the links connecting vertices in path traces and checks for the existence of alias pairs. Each alias pair may help remove a potential artificial vertex and an artificial link from the final network graph. The APAR algorithm in Fig. 10 proceeds as follows:

Line 1-4: APAR populates $\bar{E}$ and $\bar{V}$. First, for each $trace(v_i, v_j)$, it populates $\bar{V}$ by including a new vertex v_p^e for each unique interface $i_e^{v_p}$ and populates $\bar{E}$ by including an edge between two consecutive interfaces in the trace output $trace(v_i, v_j)$. After this step, the graph $\bar{G}$ includes all connections between the vertices, but potentially has redundant vertices and edges. Note that each unique address is represented by a separate vertex in $\bar{V}$.

Line 5: APAR utilizes ping-like probes to get distances to each observed IP in $\bar{V}$. The same probes also help in finding IP aliases when source IP address of reply is different from the probed IP address. Note that *getDistances* function also updates $\bar{V}$ and *Alias* when aliases are identified.

Line 6: APAR infers all subnets that satisfy the Accuracy and Completeness conditions in the collected data set. *getSubnets* function first generates all possible /24 subnets and then recursively finds all smaller subnets. It then filters the subnets that fail the Accuracy and Completeness conditions. The resulting subnets are expected to correspond to the real subnets in the Internet.

Line 7-9: Two phases of alias resolution are executed in turn. The first phase operates on all subnets considering all conditions. Then, the second phase of the algorithm is run without the Common Neighbor condition (Condition 5) for only point-to-point links, i.e., /30 and /31 subnets. The *mode* parameter of *findAliases* function is used to ignore the Common Neighbor condition in the second phase.

```
INPUT: $\bigcup\ trace(v_i, v_j)$ taken from $G = (V, E)$
OUTPUT: $\bar{G} = \{\bar{V}, \bar{E}\}$ ; $Alias = \bigcup\ (i_a^{v_k}, i_b^{v_k}, i_c^{v_k}, ...)$
INITIALIZE: $\bar{V} \leftarrow \emptyset$ ; $\bar{E} \leftarrow \emptyset$ ; $Alias \leftarrow \emptyset$
1  for ( $\forall\ trace(v_i, v_j)$ )                    /* populate $\bar{V}$ and $\bar{E}$ */
2    for ( $\forall\ i_e^{v_p} \mid i_e^{v_p} \in\ trace(v_i, v_j)$ )
3      $\bar{V} \leftarrow \bar{V}\ \cup\ v_p^e$  for each $i_e^{v_p}$
4      if ( $\exists\ i_f^{v_{p-1}}$ ) then $\bar{E} \leftarrow \bar{E}\ \cup\ e(v_{p-1}^f, v_p^e)$
5  getDistances($\bar{V}$)                           /* probing component */
6  $Subnets \leftarrow$ getSubnets($\bar{V}, compl$) /* subnet formation */
7  findAliases(0)                                  /* alias resolution phase 1 */
8  $Subnets \leftarrow \bigcup\ (sn_s^x \mid sn_s^x \in Subnets$ and $x \geq 30)$
9  findAliases(1)                                  /* alias resolution phase 2 */
FUNCTION findAliases($mode$)
10   for ( $\forall\ sn_s^x \mid sn_s^x \in\ Subnets$ and
             $sn_s^x.rank = $ maxRank($Subnets$) )
11     for ( $\forall\ (v_p, v_r) \mid v_p, v_r \in\ sn_s^x$ and $v_p \neq v_r$ )
12       for ( $\forall\ trace(v_k, v_l) \mid v_p \in\ trace(v_k, v_l)$ )
13         if ( $TTL(v_{p-1}) \simeq TTL(v_r)$ ) then
14           if ( noLoop($v_{p-1}, v_r$) ) then
15             for ( $\forall\ trace(v_m, v_n) \mid v_r \in\ trace(v_m, v_n)$)
16               if ( mode or $(v_{p-2} = v_{r+1})$ or
                      $(v_{p-2}, v_{r+1}) \in Alias$ or
                      $(v_{p-1}, v_{r+1}) \in\ sn_{s'}^{x'}$ ) then
17                 $v_r = v_{p-1}$ /* merge into one vertex */
18                 $Alias \leftarrow Alias\ \cup\ (v_r, v_{p-1})$
```

Fig. 10. Analytical and Probe-based Alias Resolver algorithm.

Line 10-18: Alias resolution function minimizes the graph by finding alias pairs using the identified subnets and removing redundant vertices and edges. Alias resolution is performed for each candidate subnet starting from the highest ranking one. *maxRank* function, in line 10, returns the un-processed subnet with the highest rank as determined by the Processing Order condition (Condition 3). For each pair of vertices (v_p, v_r) (v_p and v_r represent unique interface addresses) in the subnet sn_s^x, APAR looks for alias pairs analyzing all path traces passing through v_p and v_r. First, the Distance condition (Condition 6) in Line 13 drops candidate alias pairs that appear to be apart. *noLoop* function, in line 14, analyzes all traces to see whether a loop is created by setting v_{p-1} and v_r as alias based on the No Loop condition (Condition 4). Line 16 applies the Common Neighbor condition (Condition 5). If (1) nodes appear to be close; (2) aliasing will not cause a loop; and (3) the Common Neighbor condition is satisfied, then the vertices in $\bar{V}$ representing the matched interfaces are unified by setting $v_r = v_{p-1}$. This also merges the corresponding edges in $\bar{E}$. Note that the other side of the alignment, i.e., $v_p = v_{r-1}$, will be handled by the symmetry in line 11. Eventually, as a byproduct of the algorithm, alias pairs are recorded in a set called $Alias$.

D. Discussion

In this subsection, we discuss the limitations of APAR. Note that both *ally* and APAR depend on heuristics and may introduce false positives and false negatives. In addition, verification process requires the availability of the underlying Internet map which is difficult to obtain.

APAR looks for clues in the data set to accurately identify IP aliases that satisfy the predefined conditions. APAR first clusters IP addresses into candidate subnets and filters the subnets that fail the Accuracy and Completeness conditions. In some cases, non-existing subnets may pass both conditions and appear at the alias resolution phase. However, the No Loop and Common Neighbor conditions will, most of the time, prevent such non-existing subnets from deceiving the algorithm in the alias resolution phase. In addition, the probing component, i.e., the Distance condition, further eliminates the possible false positives. The probes are also used to utilize the source IP based alias resolution at no extra cost. On the other hand, if APAR incorrectly separates a subnet into multiple smaller ones, it may fail to identify some of the IP aliases within the subnet.

One possible argument is that APAR depends on the path symmetry to be effective. As an example, inferring a /30 subnet requires a link to be traced in both directions. Assume that the link is not traced in both directions. This may not necessarily mean that APAR is ineffective (i.e., introduces false negatives) in this setting. The situation may be that there is no alias resolution problem related to this part of the path. We explore on this by considering four cases which cover most of the practical scenarios using the setup in Fig. 11. However, the relative occurrence ratios of these cases in practice are not easy to measure.

Case 1: Let $trace(B, G) = (C_1, D_1, G_1)$ and $trace(G, B) = (F_2, E_2, B_3)$. In this case, the two paths are completely asymmetric and there is no instance of alias resolution problem. That is, in this case, path asymmetry does not affect the success of APAR as there are no aliases to resolve. □

Case 2: Let $trace(A, H) = (B_1, C_1, D_1, G_1, H)$ and $trace(H, A) = (G_2, F_2, E_2, B_3, A)$. In this case, we have two problem instances: one at B and the other at G. Note that the asymmetric path segments ($(\ldots, C_1, D_1, \ldots)$ and $(\ldots, F_2, E_2, \ldots)$ respectively) introduce no alias resolution problem. In this case, APAR can detect that B_1 and B_3 are aliases (using the subnet relation between A and B_1) and that G_1 and G_2 are aliases (using the subnet relation between G_2 and H). □

Case 3: Let $trace(A, H) = (B_1, C_1, D_1, G_1, H)$ and $trace(A, J) = (B_1, E_1, F_1, G_4, J)$. In this case, we have an alias resolution problem instance at G between G_1 and G_4. This is an example case where APAR remains ineffective as it does not have any information to identify that G_1 and G_4 are aliases. Note that if D, F and G were connected with a multi-access link, we would not have an alias resolution problem instance. □

Case 4: Let $trace(A, H) = (B_1, C_1, D_1, G_1, H)$ and $trace(J, A) = (G_3, F_2, E_2, B_3, A)$. In this case we have two problem instances: one at B and the other at G. As in Case 2

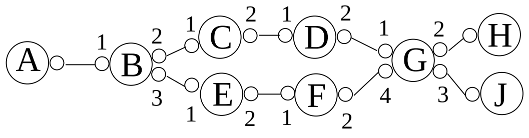


Fig. 11. A sample network with asymmetric paths.

above, APAR can identify B_1 and B_3 as aliases. However, it will fail to identify alias pair (G_1, G_3) due to lack of information. □

Another argument is that the success of APAR depends on collecting path traces from many vantage points. Having many vantage points increases the coverage of the underlying topology as well as the number of alias resolution problem instances at hand. Note that the success of APAR is not measured with the number of problem instances. However, increasing the number of vantage points positively affects APAR by improving subnet completeness and eliminating incorrect alias pairs. According to our evaluation results in Section VI-C, even though the approach returns IP aliases with a few vantage points, the increase in the number of vantage points helps improve the accuracy of APAR.

Another issue is that our path alignment scheme depends on routers compliance with the RFC 792 in sending ICMP messages. As mentioned before, RFC 792 states that a router sends an ICMP message with the IP address of the incoming interface, i.e., the shortest path interface, to the probe originator. If in reality a router uses IP address of some other interface in the ICMP message, APAR may not be able to align the path segments properly. In this situation, APAR may infer incorrect IP aliases. Note that it is difficult to evaluate how often our assumption holds and detect when it does not hold. However, our evaluations presented in the next section show that the false positive rate is less than 10%. This suggests that the probability of APAR introducing false positives due to routers non-compliance with RFC 792 is less than 10%. Results in Section III indicate that a false positive rate up to 10% does not seem to have a significant impact on topological characteristics of the resulting maps. But the approach will likely fail to find the existing aliases involving the path segments. As mentioned previously, our evaluations in Section VI-B shows less than 15% overall false negative rate on the utilized data set.

Finally, APAR does not consider MPLS clouds where LSR routers do not decrement IP TTL values. However, we expect that the IP addresses of the routers at end points of an MPLS tunnel will belong to different subnet ranges. Even if these IP addresses form a subnet, it is unlikely that the formed subnet will pass both Accuracy and Completeness conditions since the subnet range will likely be a large one.

VI. Evaluations

In this section, we present our experimental evaluations of APAR on data sets collected by AMP [30] and iPlane [21] measurement infrastructures. We use the AMP data set to study the impact of various conditions that we have defined and used in forming the subnets and in inferring the IP aliases. Next, we use several mechanisms to verify the accuracy of our algorithm on AMP data set. Finally, we use iPlane data set to show the utility of APAR on larger data sets collected from the public Internet. Note that one might utilize public traceroute servers to collect path traces from the Internet. However, during the course of this study we observed that most of these servers did not consider load balancing routers. Recall from Section I that due to load balancing practices, a path returned by traceroute may not correspond to a real path (see [8] for details).

A. Analyzing APAR

In this subsection, we analyze the effect of the conditions that we use in forming subnets and identifying IP aliases on AMP

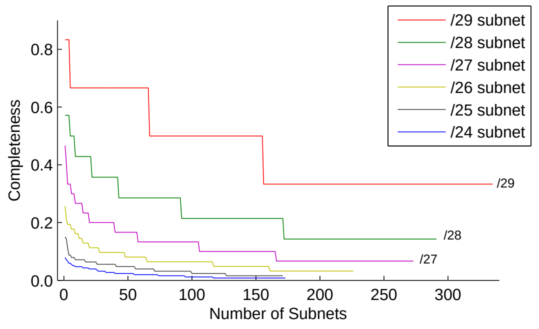


Fig. 12. Completeness distribution of subnets.

data set. The data set is collected among 130 AMP vantage points (up to 130x129 traces per set) on August 31, 2006. Most vantage points are located at US university networks that are connected over Internet2. In the AMP data set, we have 144 sets of path traces (6 sets per hour for 24 hours) among the vantage points. In our work, we filter out path traces that appear less than 10 times (out of 144 runs) to remove potentially inaccurate paths and consider the remaining ones as accurate. In addition, the AMP data set was small enough to use *ally* based verification by running *ally* in brute force manner, i.e., issuing $O(n^2)$ ally probes. After choosing the path traces to use, we resolve the occurrences of '*'s in the traceroute outputs by combining multiple appearances of '*'s in different path traces into a single router if the previous router and the next router are same in the traces. In addition, we use a similar procedure to resolve '*'s caused by routers that employ ICMP rate limiting or selectively respond to traceroute queries. At the end of this pre-processing step, we reduce 435,943 occurrences of '*'s to 616 unique nodes. At this point, our data set is ready to resolve IP aliases using APAR. Table II shows the properties of the data set before and after the above pre-processing step. As seen from the table, we have 3,905 unique IP addresses and a total of 4,521 nodes in the resulting graph.

We then study the impact of the Accuracy and Completeness conditions of subnet formation phase. During subnet formation, we require that each candidate subnet satisfies our Accuracy condition (Condition 1). We start subnet formation by initially setting x=24 to form all the candidate subnets of /24, /25, . . ., /31 prefix length and then apply the Accuracy condition to filter out the candidate subnets which appear to be inaccurate. At the end of this process, we have 2,337 candidate subnets among 2,692 possible subnets to continue our processing. Note that 2,337 subnets are not necessarily disjoint as some of the small subnets may be subsets of some larger subnets.

In the next step, we analyze the completeness of each of the formed subnets in the data set. Fig. 12 presents the completeness distribution for /24 to /29 subnets (after Condition 1) in our data set. Note that the completeness of /30 and /31 subnets is always 100%. According to the figure, if we use 50% completeness requirement (Condition 2), none of the subnets of size /24 to

TABLE II
Properties of Collected Topologies

Pre-processing	# Traces	# IPs	# *s	# Nodes
Before	2,306,395	3,952	435,943	439,895
After	19,358	3,905	616	4,521

/27 will be used to infer IP aliases and only 8 of /28 and 154 of /29 subnets can be used in alias resolution.

For IP alias pair identification, we study the impact of the Common Neighbor and Distance conditions with varying completeness ratios in Table III. The necessity of the No Loop condition is obvious and therefore its analysis is omitted. The table consists of three parts which represent the number of alias pairs; the final topology size; and the (dis)agreement ratio with *ally*, respectively. Note that APAR resolves anonymous routers (routers returning a '*' to traceroute queries) when a '*' is involved in aligned path traces. Final topology size considers such reductions which is on average 273 nodes out of 616 anonymous nodes. Since anonymous router resolution is not the focus of APAR, we do not present the details of resolved '*'s.

In the analysis, we divide APAR operation into four steps. In Step 1, we use the No Loop condition only. In Step 2, we add the Common Neighbor condition to avoid inaccuracies in subnet formation, especially for non-point-to-point subnets (i.e., subnets with a prefix length of /29 or smaller). In Step 3, we add the Distance condition to further eliminate potential false positives. Note that the Common Neighbor condition is a restrictive condition and in certain cases it may introduce false negatives. Since we have more confidence on the accuracy of the inferred point-to-point subnets, in the final step, we process these subnets again without applying the Common Neighbor condition.

A comparison of different columns in the table shows the impact of the Completeness condition. As the completeness rate increases, the number of alias pairs decreases (see Table III-(a)) but, as expected, the agreement/disagreement ratio with *ally* increases/decreases, respectively (see Table III-(d)). A comparison between Step 1 and Step 2 of the algorithm shows the impact of the Common Neighbor condition. As seen in Table III-(a), this condition reduces the number of alias pairs significantly, especially for small completeness rates. However, the increase in the agreement rate with *ally*, shown in Table III-(d), indicates that most of the filtered alias pairs are likely incorrect alias pairs and their elimination increases the relative accuracy (w.r.t. *ally*) of the process. A comparison between Step 2 and Step 3 shows the impact of the Distance condition. Similar to the previous case, this condition helps eliminate a number of likely incorrect alias pairs and improves agreement/disagreement rates with *ally*. Finally, a comparison between Step 3 and Step 4 shows the impact of relaxing the Common Neighbor condition for point-to-point subnets. This step helps increase the number of alias pairs (i.e., reduce false negatives) without reducing the agreement rate with *ally* except for one single case. In addition, the disagreement rates with *ally* stays very close to the ones in Step 3. The comparison of the agreement/disagreement rates with *ally* between Step 1 (where we do not apply the Common Neighbor condition for all subnets) and the Step 4 also indicates that the last step helps increase the number of alias pairs without introducing likely inaccuracies in the results.

TABLE III
EFFECT OF CONDITIONS IMPLEMENTED IN APAR

Completeness	0%	25%	33%	50%	66%	100%
(a) Number of Alias Pairs						
Step 1	3,234	2,462	2,438	2,155	2,087	2,073
Step 2	1,923	1,919	1,882	1,780	1,730	1,702
Step 3	1,771	1,767	1,733	1,706	1,665	1,646
Step 4	2,185	2,141	2,121	2,034	2,009	2,003
(b) Final Topology Size						
Step 1	1,731	2,420	2,441	2,706	2,758	2,773
Step 2	2,796	2,822	2,858	2,948	3,004	3,033
Step 3	2,940	2,958	2,989	3,021	3,070	3,090
Step 4	2,628	2,704	2,726	2,813	2,846	2,854
(c) Agreement/Disagreement Percentage with *ally*						
Step 1	33/16	41/9.1	42/8.4	46/4.0	47/3.4	47/3.2
Step 2	43/7.6	43/7.2	44/6.1	47/2.9	48/2.5	48/2.0
Step 3	46/3.0	46/4.0	47/2.8	49/2.0	49/1.9	49/1.6
Step 4	47/3.2	47/3.9	47/3.0	48/2.2	49/2.1	49/2.0

In summary, in this subsection we have analyze the effect of defined conditions on the accuracy of identified alias pairs. We have observed that requiring 50% subnet completeness provides a good balance between false positives and false negatives. In addition, the Common Neighbor and Distance conditions are effective in improving the accuracy of identified alias pairs. Especially, the Common Neighbor condition helps in reducing false positives introduced by multi-access-links.

B. *Verification of APAR*

In this subsection, we present our verification efforts on the accuracy of APAR (without including source IP based technique in the probing step) on the AMP data set. Note that a complete verification is not possible as it requires the availability of the underlying network topology map. As mentioned before, the availability of this information obviates the need for topology collection along with alias resolution. Our verification efforts include two parts: a comparison study between APAR and *ally*, and a similar study between APAR and DNS names of alias pairs. In this study, we use APAR algorithm with 50% subnet completeness requirement as it gives a good performance as shown in the previous subsection.

***Ally*-based Verification**

In this part, we compare *ally* and APAR by looking at the level of agreement with each other. For *ally*, we use the methodology presented in [12] and identify 2,189,950 IP address pairs to probe with *ally*. Then, we run *ally* once again for identified alias pairs to eliminate possible false positives. Fig. 13 presents a set comparison of the number of alias pairs returned by both approaches. APAR fails to detect 898 alias pairs that *ally* resolves. Similarly, *ally* fails for 1,048 pairs that APAR detects. Both approaches agree on 986 pairs and disagree on 45 pairs yielding a disagreement ratio of 4.4% (45/(986+45)) between *ally* and APAR. *Ally* resolves 34 alias pairs that suggest loops; we mark these as false positives. From this comparison, we can argue that the false positive rate of APAR (w.r.t. *ally*) is less than 5%.

Table IV compares the two approaches on a few additional metrics. According to the first row, APAR combines 2,084 unique IP addresses into 657 unique routers (3.17 IPs per router) to result in a topology of size 2,813. Note that APAR maps 281

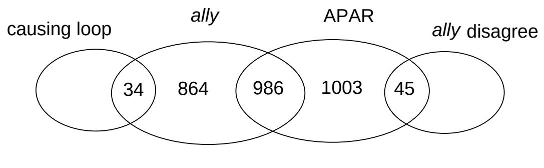


Fig. 13. Set comparison of the number of alias pairs found by APAR and *ally*.

occurrences of '*'s to their aliased IP addresses and this helps reduce the size of the resulting topology map. On the other hand, *ally* combines 1,526 unique IP addresses into 521 unique routers (2.93 IPs per router) resulting in a topology of size 3,516.

A comparison between Fig. 13 and Table IV indicates that even though the two approaches identify a similar number of alias pairs, the net results are somehow different. That is, APAR resolves IP aliases for more number of routers reducing the topology size by 2,084-657=1,427 nodes whereas *ally* reduces the topology size by 1,526-521=1,005 nodes only. The difference emerges from the fact that *ally* resolves more IP aliases per router (3.62 aliases per router) as compared to APAR (3.09 aliases per router). This is somehow expected as the number-of-alias-pairs-per-router rate depends on the availability of the path traces for APAR. On the other hand, if a router is responding to *ally* probes, then the approach can resolve most of the alias pairs for the router (provided that the router does not use ICMP rate limiting).

The third row in Table IV includes the results for the case where we take the union of the results by both approaches excluding the 34 alias pairs of *ally* that cause loops and 45 alias pairs of APAR that *ally* disagrees with. Assuming that the union results have acceptably few errors, the Aliased IPs column can be used to comment on the false negative rates of *ally* and APAR. That is, the third row suggests that there are 2,506 aliased IP addresses (out of the 3,905 total IP addresses that we have) in the data set. APAR identifies 2,084 aliased IPs and *ally* identifies 1,526 aliased IPs. This comparison suggests that *ally* misses larger number of aliased IPs as compared to APAR indicating that APAR has a smaller false negative rate for the data set.

The last row in the table corresponds to an alternative approach where we combine *ally* and APAR in a more careful way. First, we run *ally* and collect 1,884 alias pairs. Next, we apply the No Loop condition on these alias pairs and remove 34 alias pairs that cause loops. Then, we set the remaining alias pairs and run APAR on the data set. At the end, APAR identifies 1,173 new alias pairs which yields a total of 3,023 pairs. Among the approaches presented in Table IV, the combination procedure results in larger number of alias pairs. This results in further improvement than the *union* in terms of number of aliased IPs and final topology size. Hence, the combined approach should be used to resolve aliases in topology map construction studies.

Finally, assuming that the combined case gives the actual topology the differences in the topology sizes can be used to measure the false negative rates as follows. The difference between the original topology and the final topology in this case is 4,521-2,530=1,991. We need at least one alias pair to eliminate an artificial node in the topology. This suggests that we need at least 1,991 alias pairs to build the final topology correctly. The topology APAR builds has 2,813-2,530=283 artificial nodes. On the other hand, that of *ally* has 3,516-2,530=986 artificial nodes. From this, we can compute the false negative rates of each approach as (283/1,991)*100=14% for APAR and as (986/1,991)*100=50% for *ally*. Note that the number of alias pairs can not directly be used to quantify the false negative rate.

TABLE IV
IDENTIFIED IP ALIASES

Method	Alias pairs	Aliased IPs	Alias sets	Res.*s	Topo. size
APAR	2,034	2,084	657	281	2,813
ally	1,884	1,526	521	NA	3,516
Union	2,853	2,506	810	281	2,544
Combined	3,023	2,516	806	281	2,530

DNS-based Verification

In this part, we use the DNS names of the IP addresses to verify our subnet formation and alias identification steps. Some ISPs use naming practices where the DNS names of router interfaces help infer topological information. As an example, two IP addresses 216.24.186.8 and 216.24.186.9 have host names as `hous-atla-70.layer3.nlr.net` and `atla-hous-70.layer3.nlr.net`, respectively. The DNS names along with the IP addresses suggest that these interfaces form a /31 subnet. Therefore, this type of naming pattern can be used to detect point-to-point subnets.

The above heuristic may not apply to larger subnets. Instead, for these subnets, we use the heuristic to detect incorrectly formed subnets as follows. Assume that APAR forms a /29 subnet among three IP addresses and the host names of two of these IP addresses suggest a point-to-point link. In this case, using the DNS information, we decide that there should be a point-to-point link between the first two interfaces and the third interface belongs to another subnet, i.e., the initially formed /29 subnet among the three interfaces is incorrect.

Based on these observations, we visually analyze the relations among DNS names and use our findings to verify APAR. We first verify DNS names of inferred subnets, in Table V, and observe that APAR may introduce false positives by incorrectly forming non-existent subnets (43 incorrect subnets out of 1,021 verified subnets). The /29 subnets introduce the largest number of false subnets in the form of incorrectly combining two separate /30 (or /31) subnets into a /29 subnet. The Accuracy condition could not detect these errors due to the lack of traces that cross over both of these /30 (or /31) subnets. For the two cases of inferred /31 subnets, the DNS information suggests that the IP addresses actually belong to the same router. In addition, there are a large number of cases (a total of 223 cases) where no DNS information is available to make a decision.

Next, we use DNS information to verify the identified aliases. Similar to the subnet formation case, we use similarities in DNS names to verify alias pairs. As an example, the two DNS names `hous-denv-82.layer3.nlr.net` and `hous-atla-70.layer3.nlr.net` suggest that the corresponding IP addresses belong to a router located in Houston with a neighbor in Denver and another one in Atlanta. We use this and similar types of naming patterns to verify aliases. Table VI presents the DNS verification results on APAR identified alias pairs with a sub-classification of *ally* responses. According to the results, DNS agrees/disagrees with APAR on 1,247/39 (out of 2,034) alias pairs, respectively. For 748 pairs, we do not have meaningful information to make a decision. This results in a

TABLE V
VERIFICATION OF APAR SUBNETS USING DNS INFO

	total	incorrect	no-info
/31	131	2	10
/30	728	0	183
/29	154	38	27
/28	8	3	3
total	1,021	43	223

TABLE VI
VERIFICATION OF ALIAS PAIRS USING DNS INFO

DNS	agree($\surd$)				disagree ($\times$)				unknown (?)			
ally	$\surd$	$\times$	?	sum	$\surd$	$\times$	?	sum	$\surd$	$\times$	?	sum
/31	85	0	34	119	0	0	1	1	10	1	16	27
/30	430	4	398	832	1	8	12	21	238	10	332	580
/29	145	3	116	264	0	8	9	17	50	9	61	120
/28	20	0	12	32	0	0	0	0	7	2	12	21
sum	680	7	560	1247	1	16	22	39	305	22	421	748

3.0% disagreement (39/(1,247+39)) between the two approaches. According to the table, the majority of the disagreements (21 out of 39) are due to alias pairs inferred from /30 subnets. However, this corresponds to a smaller error rate (21/1,433) than that of /29 subnets (17/401). The results also suggest that the impact of most of the mis-formed /29 subnets (see Table V) are eliminated. Similar to *ally*-based verification, we can argue that the false positive rate of APAR (w.r.t. DNS information) is around 3%.

In summary, in this subsection we have used different approaches to verify the presented APAR algorithm. In each case, we have seen that APAR returned results have a small (less than 5%) rate of disagreement (i.e., false positives) with the other approaches. Eventually, there are 106 alias pairs that at least one of the verification approaches disagree with yielding an overall false positive rate of 5% for APAR. In addition, we have observed that APAR has a smaller false negatives rate (i.e., 14%) compared to false negative rate of *ally* (i.e., 50%) based on the combined approach in Table IV.

C. *Effectiveness of APAR*

In this subsection, we assess the effectiveness of APAR on iPlane data set [21]. The underlying network topology of the AMP is known to provide symmetric routes among vantage points. Nonetheless, path asymmetry is more prevalent in the public Internet. Using iPlane data allows us to measure the effectiveness of APAR on data sets collected from the public Internet. In addition, iPlane data includes IP aliases corresponding to the data set and we use them to quantify the effectiveness of APAR.

Comparison with Probing-based Approaches
In this part, we compare the effectiveness of APAR with probing based approaches. iPlane data set is collected from 184 vantage points on September 28, 2007. It contains 274,885 IP addresses in 13M path traces. One caveat of using iPlane data, on the other hand, is that we do not have much insight on the accuracy of path traces within the data set. In the data set, there are 687K path traces (5% of path traces) involving routing loops. Existence of routing loops suggests trace inaccuracies. Recall that trace inaccuracy negatively effects the map construction process. We filtered path traces with loops by removing the sub-trace corresponding to loop and using the remaining parts of the trace. We also resolved 7.1M '*'s to 712K anonymous routers.

In this experiment, we use *ally* and aliases provided by iPlane to quantify the effectiveness of APAR. iPlane uses source IP address approach of [18] and IP identifier approach of *ally* [12] for alias resolution. In order to reduce probing overhead, iPlane uses a heuristic to decide on which candidate alias pairs to query with *ally*. For instance, for 275K IP addresses in the data set there can be up to 37 billion pairs to probe which is prohibitive. iPlane probe reduction is that for each consecutive IP addresses, say IP_x and IP_y, in a path trace, iPlane probes IP_x (or IP_y) with /30 and /31 neighbor of IP_y (or IP_x). Using this approach iPlane resolves 28,513 alias pairs. Note that due to this heuristic, iPlane may miss alias pairs that could be identified by *ally*.

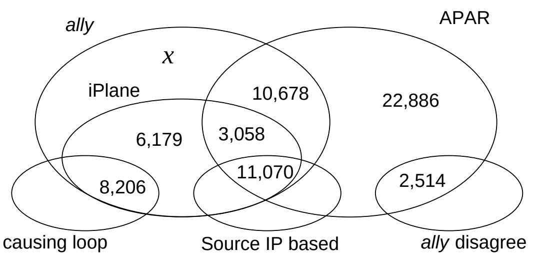


Fig. 14. Set comparison of the number of alias pairs found by APAR and *ally*.

We run APAR and resolve 50,206 IP alias pairs where 11,070 of them are identified using source IP address approach in the probing phase. Next, we use *ally* probes to verify the accuracy of alias pairs returned by APAR. Fig. 14 presents a set comparison of the number of alias pairs returned by both approaches. *Ally* agrees with 24,806 pairs and disagrees with 2,514 pairs yielding a false positive rate of 9.2% for APAR (w.r.t. *ally*). Overall, *ally* resolves a total of 39,191 alias pairs. Analyzing *ally* identified alias pairs, we observe that 8,206 pairs cause a routing loop in path traces yielding an error rate of 17.3% for *ally*. Note that due to practical limitations, we could not probe all 37 billion pairs with *ally* and therefore do not know the number of additional pairs (i.e., x in Fig. 14) that *ally* could find.

Ally based verification indicates that our false positive rate is less than 10%. In Section III, we observed that a false positive rate up to 10% does not seem to considerably alter the topological characteristics of a constructed network. Due to large number of possible pairs, we were not able to obtain the complete set of *ally*-identifiable alias pairs. Additionally, the large size of the iPlane data set prevented us from conducting a DNS based verification study as we have done for the AMP data set.

Effect of Number of Vantage Points
In this part, we analyze the effect of topology data size on the accuracy of APAR. We utilize an iPlane data set with 312,693 IP addresses and 25M path traces collected from 190 sources on May 30, 2008. Initially, we consider path traces collected by a single source as our topology data and use APAR and *ally* (as used by iPlane) to resolve IP aliases. Next, we increase the data size by adding path traces collected by a second source and then a third source and so on until we include path traces collected by 190 sources. At each step, we use APAR and *ally* to resolve IP aliases within that data set.

Fig. 15-a presents the number of alias pairs identified by APAR and *ally* as the number of sources increases. Figure also presents the number of alias pairs identified by both APAR and *ally* (labeled as 'common') and the number of alias pairs identified by the source IP address based approach. Fig. 15-b presents the accuracy of APAR in terms of false positives and false negatives w.r.t. the union of alias pairs found by APAR and *ally*. As expected, false negative rate of APAR is high for few sources but improves as the number of sources increases. According to the figure, with 10 or more sources, both false positive and false

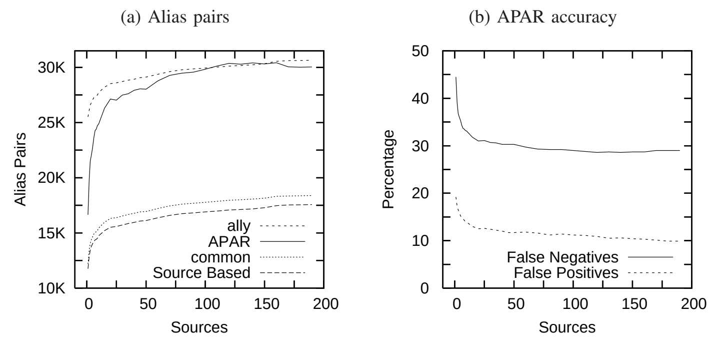


Fig. 15. Effect of number of sources

negative rates stabilize at a rate which is similar to the previous results.

In this subsection, we have evaluated APAR on two data sets collected from the public Internet. In data sets with large number of sources, the false positive rate of APAR is less than 10%. However, in data sets with few sources, both the effectiveness and accuracy of APAR is limited. In such cases, the accuracy can be improved by using multiple vantage points for the Distance condition.

VII. Conclusions

In this paper, we have focused on the IP alias resolution problem. The experimental study presented in the first part of the paper has demonstrated that the IP alias resolution task has a considerable effect on many topological characteristics of network maps that are built from a set of traceroute collected path traces. In the second part of the paper, we presented an analytical approach to resolve aliases among IP addresses in a set of path traces. The main contribution of this part has been the introduction of an analytical approach that introduces no or minimal active probing overhead for alias resolution. Our experimental evaluations on several genuine topology data have shown that the proposed approach complements the existing probe based alias resolution approaches and significantly improves the success of overall IP alias resolution process in topology measurement studies. Finally, our project site at `http://itom.utdallas.edu` includes both source code and data sets for APAR.

References

[1] M.H. Gunes, S. Bilir, K. Sarac, and T. Korkmaz, “A measurement study on overhead distribution of value-added Internet services,” in *Computer Networks*, vol. 51, no 14, pp. 4153–4173, October 2007.

[2] R. Teixeira, K. Marzullo, S. Savage, and G. Voelker, “In search of path diversity in ISP networks,” in *Proceedings of the USENIX/ACM Internet Measurement Conference*, Miami, FL, USA, October 2003.

[3] L. Amini, A. Shaikh, and H. Schulzrinne, “Issues with inferring Internet topological attributes,” in *Proceedings of SPIE ITCom*, Boston, MA, USA, July/August 2002.

[4] V. Jacobson, *Traceroute*, Lawrence Berkeley Laboratory (LBL), February 1989, available from ftp://ee.lbl.gov/traceroute.tar.Z.

[5] D. McRobb, K. Claffy, and T. Monk, *Skitter*, CAIDA, 1999, available from http://www.caida.org/tools/skitter/.

[6] N. Spring, D. Wetherall, and T. Anderson, “Scriptroute: A public Internet measurement facility,” in *USENIX Symposium on Internet Technologies and Systems (USITS)*, Seattle, WA, USA, March 2003.

[7] Y. Shavitt and E. Shir, “DIMES: Let the Internet measure itself,” in *SIGCOMM Computer Communication Review*, vol. 35, no 5, pp. 71–74, 2005.

[8] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, “Avoiding traceroute anomalies with Paris traceroute,” in *Proceedings of IMC*, Rio de Janeiro, Brazil, October 2006.

[9] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, “Topology inference in the presence of anonymous routers,” in *IEEE INFOCOM*, San Francisco, CA, USA, March 2003.

[10] M.H. Gunes and K. Sarac, “Resolving anonymous routers in Internet topology measurement studies,” in *IEEE INFOCOM*, Phoenix, AZ, USA, April 2008.

[11] J. Pansiot and D. Grad, “On routes and multicast trees in the Internet,” in *ACM Computer Communication Review*, vol. 28, no 1, January 1998.

[12] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, “Measuring ISP topologies using Rocketfuel,” in *IEEE/ACM Transactions on Networking*, vol. 12, no 1, pp. 2–16, February 2004.

[13] *iffinder tool*, http://www.caida.org/tools/measurement/iffinder/.

[14] *ally tool*, http://www.cs.washington.edu/research/networking/rocketfuel/.

[15] A. Lakhina, J. Byers, M. Crovella, and P. Xie, “Sampling biases in IP topology measurements,” in *IEEE INFOCOM*, San Francisco, CA, USA, March 2003.

[16] J. Han, D. Watson, and F. Jahanian, “Topology aware overlay networks,” in *IEEE INFOCOM*, Miami, FL, USA, March 2005.

[17] M.H. Gunes and K. Sarac, “Analytical IP alias resolution,” in *IEEE ICC*, Istanbul, Turkey, June 2006.

[18] R. Govindan and H. Tangmunarunkit, “Heuristics for Internet map discovery,” in *IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[19] A. Nakao, L. Peterson, and A. Bavier, “Routing underlay for overlay networks,” in *Proceedings of SIGCOMM*, Karlsruhe, Germany, August 2003.

[20] M.H. Gunes and K. Sarac, “Importance of IP alias resolution in sampling Internet topologies,” in *IEEE Global Internet*, Anchorage, AK, USA, May 2007.

[21] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, “iPlane: An information plane for distributed services,” in *OSDI 2006*, Seattle, WA, November 2006.

[22] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos, “Power-laws and the AS-level Internet topology,” in *IEEE/ACM Transactions on Networking*, vol. 11, no 4, pp. 514–524, August 2003.

[23] J.Winick and S. Jamin, “Inet-3.0: Internet topology generator,” University of Michigan, Tech. Rep., 2002.

[24] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: An approach to universal topology generation,” in *MASCOTS*, Cincinnati, OH, USA, August 2001.

[25] P. L. Krapivsky, S. Redner, and F. Leyvraz, “Connectivity of growing random networks,” in *Physical Review Letters*, vol. 85, p. 4629, 2000.

[26] M. E. J. Newman, “Assortative mixing in networks,” in *Physical Review Letters*, vol. 89, p. 208701, 2002.

[27] D. J. Watts and S. H. Strogatz, “Collective dynamics of ’small-world’ networks.” in *Nature*, vol. 393, no 6684, pp. 440–442, June 1998.

[28] L. Freeman, “A set of measures of centrality based on betweenness,” in *Sociometry*, vol. 40, pp. 35–41, 1977.

[29] M.H. Gunes and K. Sarac, “Inferring subnets in router-level topology collection studies,” in *ACM SIGCOMM IMC* , San Diego, CA, Oct 24-26 2007.

[30] A. McGregor, H.-W. Braun, and J. Brown, “The NLANR network analysis infrastructure,” in *IEEE Communications Magazine*, vol. 38, no 5, pp. 122–128, May 2000.

[31] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, “On routing asymmetry in the Internet,” in *IEEE GLOBECOM*, St Louis, MO, USA, November 2005.

PLACE PHOTO HERE

Mehmet H. Gunes received M.S. degree in Computer Engineering at Southern Methodist University in 2004 and Ph.D. degree in Computer Science at the University of Texas at Dallas in 2008. He is currently an Assistant Professor in the Department of Computer Science & Engineering at the University of Nevada, Reno. His research interests include computer networks, network security, network protocols, Internet measurements, graph data mining, and complex networks.

PLACE PHOTO HERE

Kamil Sarac received the M.S. and Ph.D. degrees in computer science from the University of California at Santa Barbara, in 1997 and 2002, respectively. He is currently an Assistant Professor in the Department of Computer Science at the University of Texas, Dallas. His research interests include computer networks and protocols; network and service monitoring and Internet measurements; overlay networks and their use in network security and denial-of-service defense; and multicast communication.