Resolving IP Aliases in Building Traceroute-Based Internet Maps

Mehmet H. Gunes and Kamil Sarac
Department of Computer Science, University of Texas at Dallas
2601 N Floyd Rd. Richardson, TX 75083, USA
{mgunes, ksarac}@utdallas.edu

***Abstract*— Most network measurement studies utilize traceroute-based topology data collected from the Internet. This data is then processed to construct a sample Internet map. The map construction process includes an important step called IP alias resolution, the task of identifying IP addresses belonging to the same router in the collected data set. Inaccuracies in alias resolution affects the representativeness of the resulting map. This in turn impacts the observations or conclusions derived from the measurement study. In this paper, we present a new alias resolution algorithm called Analytical and Probe-based Alias Resolver (APAR). Given a set of path traces, APAR utilizes the common IP address assignment scheme to infer IP aliases within the collected path traces. APAR incurs minimum traffic overhead into the network. Our evaluation results show that APAR reveals a significant number of IP aliases which are not found by the existing approaches.**

I. Introduction

Internet measurement studies require the availability of representative Internet maps. However, no complete map depicting the Internet's router level topology exists. In order to facilitate the research activities in Internet measurement studies, several institutions continuously collect topology data from the Internet and make them available to the research community [1], [2], [3], [4], [5]. 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. Additionally, inaccuracies in this process may significantly affect the accuracy of the observations or results obtained in the measurement study [6], [7].

In this paper, we consider router-level topology measurement studies. Most router-level measurement studies utilize the well-known Internet debugging tool, *traceroute* [8], or its variants [1], [3], [4]. 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 error messages, traceroute returns the path information as a sequence of IP addresses each representing a router between the local system and the remote destination.

After collecting the path traces, the information needs to be processed to build the corresponding network topology. This step involves several tasks including (1) verifying the correctness of the path traces, (2) resolving unresponsive routers that are represented by '*'s in traceroute outputs, and (3) resolving IP addresses belonging to the same router. The first task has to do with the fact that certain types of load balancing implementations may cause traceroute to return path traces that do not correspond to a real path in the network. This may occur when a router forwards consecutive traceroute probes on different paths toward the destination. The second task has to do with the fact that not all routers respond to traceroute probes all the time. Finally, the last task is an artifact of the traceroute-based topology collection procedure and is the main focus of the work presented in this paper.

Topology construction requires identification and grouping of the IP addresses belonging to the same router, a task often referred to as *IP alias resolution*. Routers have multiple interfaces each one having a different IP address. Hence, a router may appear on multiple path traces with different interface IP addresses. The goal of IP alias resolution is to identify the IP addresses that belong to the same router and combine them into a single node in the resulting sample topology map. Without a proper alias resolution process, the resulting topology map may be significantly different from the underlying network.

Given the fact that traceroute-based topology maps are used in various research areas [9], [10], [11], accuracy of IP alias resolution process may significantly impact the accuracy of the observations made by using these maps. As an example, the authors of [7] identified limitations in IP alias resolution as the main reason for the limited success of the topology discovery method used in the *Rocketfuel* study [9]. In addition, in a related recent work [6], we demonstrated the impact of IP alias resolution on the utility of the recently proposed sampling bias tests [10] that are used to verify the representativeness of the collected Internet topologies. Finally, in order to quantify the impact of poor alias resolution, we conducted an experiment where we collected 5K path traces from a 10K Waxman graph and used them to build sample graphs. In building one of the sample graphs (S1), we had all the nodes to respond to *ally* based alias resolution probes. In the other one (S2), we had 40% of the nodes (40% is based on our experience presented in [6]) to ignore *ally* probes. After the alias resolution phase, the size of the sample graph S1 was 3093 and that of S2 was 4413. The difference (1320 nodes in S2) is an artificail increase due to imperfect alias resolution in building S2. Similarly, the average degree of the nodes in S1 was 2.07 and that of the nodes in S2 was 1.45. The results of this experiment shows the impact of imperfect alias resolution on the accuracy and characteristics of the resulting sample topologies.

Several mechanisms have been developed to resolve IP aliases. These include *mercator* [12], *iffinder* [13], and *ally* [9] tools (see Section II for more details). Given two IP addresses, these tools send probe messages to the IP addresses and study the responses to decide on aliases. These tools are easy-to-use and provide a very convenient way to verify if a given pair of IP addresses are alias or not. On the other hand, due to their active probing nature, they introduce additional traffic overhead into the network. Considering the increasing volume of measurement traffic in the Internet [14], many ISPs filter out or rate limit queries destined to themselves. This practice affects the utility of the existing alias resolution tools.

In this paper, we present a novel approach for IP alias resolution, called *Analytical and Probe-based Alias Resolver* (APAR*)*. Given a set of collected path traces, APAR uses the common IP address assignment practices (see RFC 2050) to infer IP aliases. APAR introduces minimum traffic overhead and it does not require active participation of the routers in the alias resolution process. 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 the symmetric segments of different path traces and infers alias pairs among the involved IP addresses. Path asymmetry is a commonly observed characteristic in the Internet. However, it does not negatively affect the performance of APAR. If a given pair of path traces are completely disjoint/asymmetric, then there is no alias pairs to resolve in the data set. In other words, alias resolution is relevant when the given set of path traces include some common segments in them. APAR leverages this commonality to infer IP aliases from within the given data set.

The work presented in this paper builds on our preliminary work in [15]. In that work, we considered a rather simple version of the problem called Two Path Alias Resolution Problem (TPARP). TPARP assumes point-to-point links to resolve IP aliases on a given pair of path traces between two vantage points and it completely ignores multi-access links. In the current work, we significantly extend the scope of the problem to multiple path traces and consider both point-to-point and multi-access links.

The rest of the paper is organized as follows. Section II summarizes the related work. Section III presents a graph theoretic formulation of the problem. Section IV discusses several observations that are used in the construction of the APAR algorithm. Section V presents the details of the APAR algorithm. Section VI includes our experimental evaluations of APAR. Finally, Section VII concludes the paper.

II. Related Work

The initial work on alias resolution is by Pansiot and Grad [16]. Given two IP addresses, they send probe messages to both IP addresses to solicit ICMP error messages. By comparing the source IP addresses of the returned ICMP error messages, they can set the two IP addresses as alias or not. *Mercator* [12] and *iffinder* [13] are two well-known tools to resolve IP aliases using this method. *Mercator* improves [16] 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 *ally* tool introduced by Spring et. al. [9] combines the address based method with an IP identification based method to classify a pair of IP addresses as *alias*, *not-alias*, or *unknown*. This approach uses potential similarity in IP identification field values in the returning ICMP error messages to infer IP aliases. 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* also uses similarities in the host names of the routers as well as the TTL values of the responses to narrow the search space and to increase the confidence in an identified alias pair.

These methods are simple and powerful in resolving IP aliases. However, being active probing approaches, they introduce additional traffic overhead. They also depend on routers' participation by replying to the probe messages. In addition, IP identification based method suffers from several limitations. Some operating systems assign random values to IP identification field and some (e.g., Linux OS) set this field to zero for each IP packet. Besides, at some high end routers each interface implements its own counter and ICMP packets generated by different interfaces may have unrelated IP identification values.

Compared to probing based approaches, APAR, in general, is a passive approach for alias resolution. It requires minimum active probing and does not depends on the active participation of the routers to resolve IP aliases. Our approach can identify a significant number of aliases by itself. It can also be combined with the probing based approaches to improve the success of alias resolution.

III. Alias Resolution Problem

Path traces collected by traceroute consist of interface IP addresses of devices. A problem during sample router-level map construction is to identify IP addresses belonging to the same device. This problem is defined as *IP alias resolution problem* and *alias resolution* is the process of identifying whether IP addresses belong to the same device or not.

We introduce a number of terms and symbols in formulating the problem as a graph theoretic problem. This notation will be used in 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 (Preferred Path): A *preferred path* $PP(v_i, v_j)$=$(V_{PP(v_i,v_j)}, E_{PP(v_i,v_j)})$ is a subgraph of G where $V_{PP(v_i,v_j)}$=$\{v_i$,v_{i+1},v_{i+2},$\ldots$,$v_j\}$ represents the sequence of vertices between v_i and v_j, and $E_{PP(v_i,v_j)}$=$\{e_{(v_i,v_{i+1})}$,$e_{(v_{i+1},v_{i+2})}$,$\ldots$,$e_{(v_{j-1},v_j)}\}$ represents the sequence of edges in E connecting the vertices in $V_{PP(v_i,v_j)}$. An edge $e_{(v_k,v_{k+1})} \in E_{PP(v_i,v_j)}$ connects v_p and v_{p+1} via interfaces $i_e^{v_p}$ and $i_f^{v_{p+1}}$. $\square$

A path is a preferred path based on some application specific criteria, e.g., shortest path, minimum cost path, etc. Note that $PP(v_j, v_i)$ may or may not be equal to $PP(v_i, v_j)$.

Definition (Trace): A trace, $trace(v_i, v_j)$, is a function of a preferred path $PP(v_i, v_j)$ where the trace visits each vertex $v_k \in V_{PP(v_i,v_j)}$ starting from v_i all the way to v_j and returns a list of interfaces (one for each vertex) as its output. $\square$

A trace is said to arrive at a visited vertex from its incoming interface and the interface representing a visited vertex in the trace output is called a *shortest path interface*.

Definition (Successor/Predecessor): Given a trace output, $trace(v_i, v_j)$=$(\ldots, i_e^{v_p}, i_f^{v_r}, \ldots)$, 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 $subnet_s^x$ 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.

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

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

After defining the terminology used in the rest of the paper, we now give the definition of *Alias Resolution Problem.* Note that, in this paper, we mainly consider the problem in the context of traceroute based topology measurement studies. An alternative version of the alias resolution problem, which is not considered in this paper, may seek to identify if a given couple of IP addresses are alias or not.

Definition (Alias Resolution Problem): Given a number of path traces, $\bigcup trace(v_i, v_j)$, from a network graph $G = (V, E)$, alias resolution problem is to build a subgraph $\bar{G}$ of G such that

- $\bar{G}$=$(\bar{V}, \bar{E})$ where $\bar{V}$=$\bigcup V_{PP(v_i,v_j)}$ and $\bar{E}$=$\bigcup E_{PP(v_i,v_j)}$.
- If $i_e^{v_p} \in trace(v_k, v_l)$ and $i_f^{v_p} \in trace(v_m, v_n)$, then there should be one and only one $v_p \in \bar{V}$.
- For each $i_e^{v_p}$,$i_f^{v_r}$ such that $i_e^{v_p}.address \xleftrightarrow{x} i_f^{v_r}.address$ for a $/x$ subnet, there exists at most one link $e_{(v_p,v_r)} \in \bar{E}$.

IV. Observations

In this section, we present a summary of IP address assignment practices in the Internet and show a methodology to use it in identifying 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 $X = 32 - \lceil log_2(N + 2) \rceil$. 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.

For example, if a subnet has a network address of 192.168.0.0/28, then the last 4 bits are used to identify the individual IP addresses of the device interfaces in this subnet. These four bits can identify at most 14 device interfaces. The remaining two IP addresses, namely 192.168.0.0 and 192.168.0.15, have special meanings and are not typically used for assignment.

B. Identifying IP Aliases Using Subnets

The subnet relation between the 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. Larger subnets (/29 or larger) are not considered as they cause waste of IP addresses.

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 (IP$_A \xleftrightarrow{\mathbf{30}}$ IP$_B$) or (IP$_A \xleftrightarrow{\mathbf{31}}$ IP$_B$). 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. 1 where $h1$, $h2$, $h3$, and $h4$ are end-hosts and $r1$ and $r2$ are routers all connected

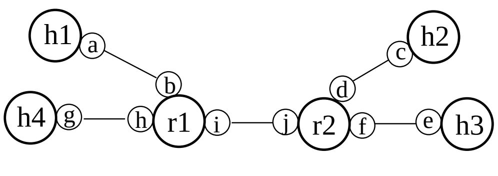

Fig. 1. A sample network between four end-hosts.

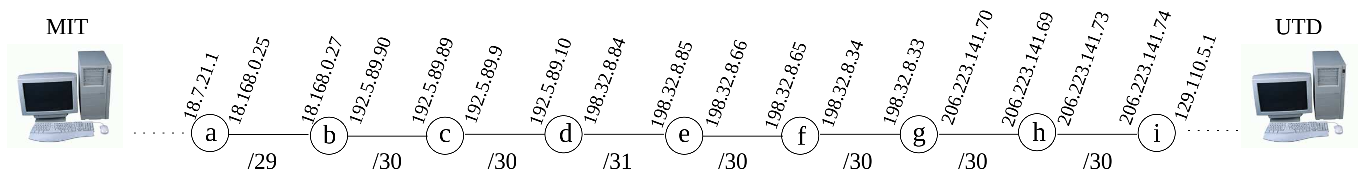

Fig. 2. The inferred subpath between MIT and UTD hosts.

using point-to-point links. The lower case letters a, b, ..., j represent interface IP addresses. Assume that we have two path traces $trace(h1, h3)$=(a, b, j, e) and $trace(h2, h4)$=(c, d, i, g) that are taken from this network. Comparing the two path traces, we observe that ($i \overset{\mathbf{30}}{\longleftrightarrow} j$). Based on this subnet relation, we can align the path traces as

$$\begin{tikzcd} a & b & j & e & \text{(trace from h1 to h3)} \\ g & i \arrow[ur] & d & c & \text{(reverse of trace from h2 to h4)} \end{tikzcd}$$

and identify IP alias pairs as (b, i) and (d, j). Please note that there is an alternative alignment of the path traces as

$$\begin{tikzcd} a & b & j \arrow[dr] & e & & & \text{(trace from h1 to h3)} \\ & & g & i & d & c & \text{(reverse of trace from h2 to h4)} \end{tikzcd}$$

However, since routers are assumed to send ICMP error messages from their shortest path interface to the vantage point, the second alignment is incorrect. 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.

Using Multi-Access Links

Multi-access links are used to connect several device interfaces to form a subnet. In general, these subnets include more than two interfaces connected to them. A number of technologies can be used to build multi-access subnets including Ethernet, FDDI, token ring, etc. When building a subnet, one chooses a subnet number that has enough IP addresses for unique address assignment for each interface on the subnet. As an example, if a subnet is to include five device interfaces, one defines a /29 subnet to assign unique IP addresses to each interface[1].

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. However, it may also introduce false positives if the inferred subnets do not correspond to real subnets in the network. We analyze this and several other related issues in more detail in the next section.

Finally, we end this section by providing a real life example to demonstrate the utility of the observations in inferring IP aliases. Table I presents traceroute outputs between two end hosts, one in MIT and the other in UTD computer network. The first column shows the MIT-to-UTD trace and the second column shows the *reverse* of 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.

[1]In a /29 network, we have $2^{(32-29)}$-2=6 IPv4 addresses for assignment.

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

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. 2. 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.

V. ANALYTICAL ALIAS RESOLUTION

In this section, we present analytical alias resolution approach to resolve IP aliases in a set of collected path traces. This approach is mainly built on the observations presented in the previous section. APAR includes two steps (1) analyzing IP addresses in the set of collected path traces to identify a set of candidate subnets and (2) using the identified subnets to resolve IP aliases. In the following, we present each of these steps and the details of the APAR algorithm.

A. *Subnet Formation Rules*

Our alias resolution method relies on identifying subnets from the set of collected path traces. If we were to know the underlying network topology at the subnet level, 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 identify subnets that the IP addresses are involved in.

We use an iterative approach to form all candidate subnets starting from /22 subnets to /31 subnets using the IP addresses at hand. Here we assume that the largest subnet in the Internet has at most $2^{10}-2$ nodes in it (i.e., a /22 subnet). First, we form all candidate /22 subnets from the data set by combining the IP addresses whose first 22 bits match. Next, we recursively form smaller subnets (e.g., /23, /24/, . . ., /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 in the Internet. Similarly, the candidate /29 subnet may be part of a bigger subnet in the Internet. Therefore, we need to use some criteria to eliminate non-existent subnets from our candidate subnet list. We identify two conditions to achieve this as below.

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 $subnet_s^x$

$$\begin{aligned}\nexists\, trace(v_i,\ v_j) \mid\ & (i^{v_p}, i^{v_r} \in\ subnet_s^x) \text{ and} \\ & (i^{v_p}, i^{v_r} \in\ trace(v_i,\ v_j)) \text{ and} \\ & (i^{v_{p+1}} \neq i^{v_r}) \text{ and } (i^{v_{p-1}} \neq i^{v_r})\end{aligned} \quad \square$$

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 is used to detect inaccurate subnets in our candidate subnet set.

Condition 2: COMPLETENESS
Ignore candidate subnets that have less than half of their IP addresses present in the collected data set. A subnet $subnet_s^x$ can include up to $2^{32-x} - 2$ IP addresses and we require that at least half of these addresses appear in our data set. □

This requirement helps 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.

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. Based on this, 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 and take precedence.

B. Accuracy of Identified Aliases

After obtaining the candidate subnets, we next use this information to infer IP aliases. The alias resolution procedure follows the observations that we present in Section IV. In this section, we present several rules that we use to avoid false positives in inferring IP aliases.

Condition 4: NO LOOP
Assuming that the 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

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

If a routing loop is to emerge in any of the path traces as a result of inferring the two IP addresses as alias, 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

a b c d
i b h g f e (reversed trace)

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 in the Internet.

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. □

We use an example to explain each case. Consider the sample topology in Fig. 3 where $h1$, $h2$, $h3$, and $h4$ represents 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 topol-

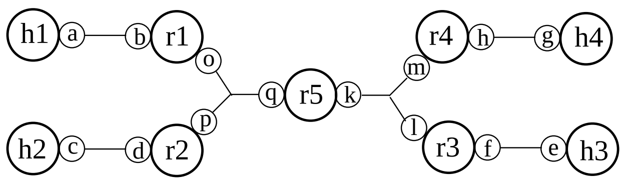

Fig. 3. A sample network using multi-access links.

ogy 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 the path traces $trace(h1, h4)$ and $trace(h2, h1)$. Observing ($q \xleftrightarrow{x} o$), we align the two 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 also observe that a is a neighbor for 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 and therefore ignore it.

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). As seen in these examples, the above rules help us increase our confidence in inferred IP aliases and help us avoid setting unrelated IP addresses as alias pairs with each other.

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 the 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.

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 improve the accuracy of formed alias pairs but the algorithm yields a reasonable accuracy level without the use of this condition as well.

C. APAR Algorithm

Analytical Alias Resolver algorithm 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 any two path traces. By symmetry, APAR locates the links connecting vertices in the preferred paths 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 $\bar{G}$.

The APAR algorithm in Fig. 4 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 identifies 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 in the data set. It then

```
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 $Subnets \leftarrow$ getSubnets($\bar{V}, compl$) /* subnet formation */
 6 findAliases(0)                                   /* alias resolution phase 1 */
 7 findAliases(1)                                   /* alias resolution phase 2 */

FUNCTION findAliases($mode$)
 8 for ( $\forall\ subnet_s^x \mid subnet_s^x \in Subnets$ and
             ($\neg$mode or $x \geq 30$) and
             $subnet_s^x.rank = $ maxRank($Subnets$) )
 9   for ( $\forall\ (v_p, v_r) \mid v_p, v_r \in subnet_s^x$ and $v_p \neq v_r$ )
10     for ( $\forall\ trace(v_k, v_l) \mid v_p \in trace(v_k, v_l)$ )
11       if ( $TTL(v_{p-1}) \simeq TTL(v_r)$ ) then
12         if ( noLoop($v_{p-1}, v_r$) ) then
13           for ( $\forall\ trace(v_m, v_n) \mid v_r \in trace(v_m, v_n)$)
14             if ( mode or ($v_{p-2} = v_{r+1}$) or
                   ($v_{p-2}, v_{r+1}) \in Alias$ or
                   ($v_{p-1}, v_{r+1}) \in subnet_{s'}^{x'}$ ) then
15               $v_r = v_{p-1}$ /* merge into one vertex */
16               $Alias \leftarrow Alias \cup (v_r, v_{p-1})$
```

Fig. 4. Analytical and Probe-based Alias Resolver algorithm.

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 6-7: 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 is used to limit *findAliases* function only to subnets of point-to-point links and ignore the Common Neighbor condition in the second phase.

Line 8-16: 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 8, 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 $subnet_s^x$, APAR looks for alias pairs analyzing all path traces passing through v_p and v_r. First, the Distance condition (Condition 6) in Line 11 drops candidate alias pairs that appear to be apart. *noLoop* function, in line 12, 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 14 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 9. Eventually, as a byproduct of the algorithm, alias pairs are recorded in a set called $Alias$.

D. Discussion

In this section, we discuss the limitations of our alias resolution approach. First of all, we should note that the existing alias resolution approaches depend on heuristics and may introduce false positives and false negatives. In addition, verification process requires the availability of the underlying Internet topology map which (1) is difficult to obtain and (2) obviates the need for topology collection along with alias resolution.

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

Another issue is that our path alignment scheme depends on routers' compliance with the RFC 1812 in sending ICMP messages. As mentioned before, RFC 1812 states that a router sends an ICMP message from the shortest path interface to the probe originator. If in reality a router uses some other interface to send the ICMP message, APAR may not be able to align the path segments properly. In this situation, with high probability, the No Loop, the Common Neighbor, and the Distance conditions will prevent inferring incorrect IP aliases. But the approach will likely fail to find the existing aliases involving the path segments.

VI. Evaluations

In this section, we present our experimental evaluations of the analytical alias resolution approach on a data set collected by the AMP project [2]. The data set contains 144 sets of path traces among 130 vantage points taken on Aug 31, 2005. First, we pre-process the data to prepare it for our analysis. Then, we use the 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 compare the performance of our algorithm, APAR, with the current state of the art tool, *ally*, on our data set. Finally, we show the effectiveness of APAR on a sample network.

During the pre-processing phase, we filter out traces that appear less than 10 times (out of 144 runs). This filtering removes potential inaccuracies due to transient conditions in the network. We then combine the occurrences of '*'s in traceroute outputs. As we mentioned before, traceroute tool prints a '*' to represent a router that does not respond to traceroute probes. We use the technique that we presented in [6] to resolve unresponsive routers. The main idea in this process is to combine multiple appearances of '*'s in different path traces into a single router if (1) the previous router, (2) the next router, and (3) the trace destination are all same in the traces. In addition, we use a similar procedure to resolve '*'s that correspond to the routers that employ ICMP rate limiting and selectively respond to traceroute queries. At the end of this pre-processing step, we reduce 435,943 occurrences of '*'s to 616 unique nodes. Table II shows the properties of the data set before and after the pre-processing step. As seen from the table, we have 3,905 unique IP addresses and a total of 4,521 nodes in the graph after pre-processing.

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

A. Analyzing APAR

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

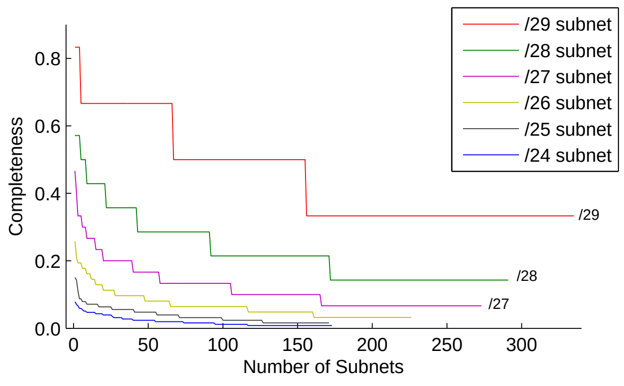

Fig. 5. Completeness distribution of subnets.

During subnet formation, we require that each candidate subnet satisfies our accuracy condition (Condition 1). Assuming that the largest subnet that may exist in the Internet is a /22 subnet (a subnet with 2^{10}-2 connected interfaces), we first form all the candidate subnets of /22, /23, . . ., /31 length and then apply the accuracy condition to filter out the candidate subnets which are known to be inaccurate. At the end of this process, we have 2,337 candidate subnets among 2,692 possible subnets to continue our processing.

In the next step, we analyze the completeness of each of the formed subnets in the data set. Fig. 5 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 /27 will be used to infer IP aliases and only 8 of /28 and 154 of /29 subnets can be used in alias resolution. Note that the results of the application of the completeness requirement indicate that the initial selection of x=24 is appropriate for this case.

For alias pair identification, we study the impact of the Common Neighbor and the Distance conditions with varying completeness ratios. The necessity of the No Loop condition is obvious and therefore its analysis is omitted. Table III presents the analysis results. The table consists of four parts which represent the number of alias pairs; the number of resolved unresponsive routers (routers returning a ’*’ to traceroute queries); the final topology size; and the (dis)agreement ratio with *ally*, respectively. 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. This step corresponds to the final APAR algorithm.

A comparison of the 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 the final algorithm, APAR, 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 the subnets) and the APAR step 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

(a) Number of Alias Pairs						
Completeness	0%	25%	33%	50%	66%	100%
Step 1	3234	2462	2438	2155	2087	2073
Step 2	1923	1919	1882	1780	1730	1702
Step 3	1771	1767	1733	1706	1665	1646
APAR	2185	2141	2121	2034	2009	2003
(b) Number of Resolved *’s						
Step 1	344	296	296	278	274	270
Step 2	289	259	255	244	235	229
Step 3	288	259	256	245	236	230
APAR	339	299	297	281	276	271
(c) Final Topology Size						
Step 1	1731	2420	2441	2706	2758	2773
Step 2	2796	2822	2858	2948	3004	3033
Step 3	2940	2958	2989	3021	3070	3090
APAR	2628	2704	2726	2815	2846	2854
(d) 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
APAR	47/3.2	47/3.9	47/3.0	48/2.2	49/2.1	49/2.0

B. *Verification of APAR*

In this subsection, we present our verification efforts on the accuracy of APAR. 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 four parts: (1) a comparison study between APAR and *ally*, (2) a similar study between APAR and DNS names of alias pairs, (3) use of APAR in resolving IP aliases for the Abilene Internet2 backbone whose IP level topology map is available at `www.internet2.edu`. In this study, we use APAR algorithm with 50% subnet completeness 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 [9] and identify 2,189,950 candidate alias pairs to probe with *ally*. For APAR, we used the above presented approach. Fig. 6 presents a set comparison of the number of alias pairs returned by two approaches. APAR fails to detect 898 alias pairs that *ally* resolves. On the other hand, *ally* fails for 1,048 pairs that APAR detects. Both approaches agree on 986 alias pairs and disagree on 45 pairs. The 34 alias pairs among the 898 pairs that *ally* resolves suggest routing loops on the path traces and are therefore marked as false positives. These results suggest a disagreement ratio of 4.4% (45/(986+45)) between *ally* and APAR. Assuming that, as the worst case for APAR, this ratio represents the error rate of APAR, APAR returns a total of 1,945 accurate alias pairs (1945 = 986+959 where 959 is 95.6% of 1,003). From this comparison, we can argue that the verifiable accuracy of APAR (w.r.t. *ally*) is over 95%.

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 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. 6 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 2084-657=1427 nodes whereas *ally* reduces the topology size by 1526-521=1005 nodes only. The difference emerges from the fact that *ally* resolve 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).

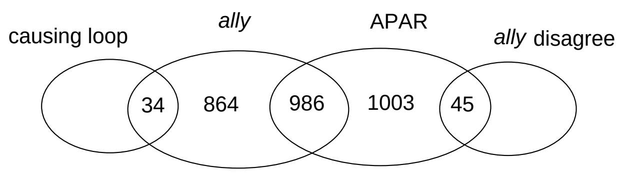

Fig. 6. Set comparison of the number of alias pairs found by APAR and *ally*.

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

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 (the third row) 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 3905 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. This means that APAR has a smaller false negative rate as compared to *ally* for our data set.

Finally, the last row in the table corresponds to an alternative approach where we combine *ally* and APAR in a more careful way In combined approach, we first run *ally* and collect 1884 alias pairs. Next, we apply the No Loop condition on these alias pairs and identify 34 alias pairs to cause loops. We remove these pairs and use the remaining ones during APAR based alias resolution process. At the end of the process, APAR identifies 1173 new alias pairs which yields a total of 3023 pairs. Among the approaches presented in Table IV, the combination procedure results in larger number of alias pairs (see the last row in the table). This results in further improvement than *union* in terms of number of aliased IPs and final topology size.

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 the router interfaces help infer topological information. As an example, the 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 the two of these IP addresses suggest a point-to-point link. In this case, by 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.

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	1021	43	223

Table V shows the results of this verification effort. According to the above verification procedure, APAR may introduce false positives by incorrectly forming non-existent subnets (43 incorrect subnets out of 1021 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 alias identification process. Similar to the subnet formation case, we use similarities in DNS names to identify alias pairs. As an example, the two DNS names `hous-denv-82.layer3.nlr.net` and `hos-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 identify DNS-based alias pairs for verification purposes. Table VI presents the DNS verification results on APAR identified alias pairs. According to the results, DNS agrees/disagrees with APAR on 1247/39 (out of 2034) alias pairs, respectively. For 748 pairs, we do not have meaningful information to make a decision. This results in a 3.0% disagreement (39/(1247+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 (21/1433) 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 verifiable accuracy of APAR (w.r.t. DNS information) is around 97%.

TABLE VI
VERIFICATION OF ALIAS PAIRS USING DNS INFO

DNS	agree($\surd$)	disagree ($\times$)	unknown (?)
/31	119	1	27
/30	832	21	580
/29	264	17	120
/28	32	0	21
Total	1247	39	748

Verifying APAR on Abilene Backbone

In this subsection, we want to verify the accuracy of APAR. This requires the availability of the underlying topology map of the sampled network. Abilene's Internet2 backbone map is available at `http://abilene.internet2.edu` site. Therefore, in this part of our evaluations, we run traceroutes to collect path traces crossing over the Abilene backbone and then use these traces to infer IP aliases using APAR. We then compare our sample topology with the Abilene backbone map.

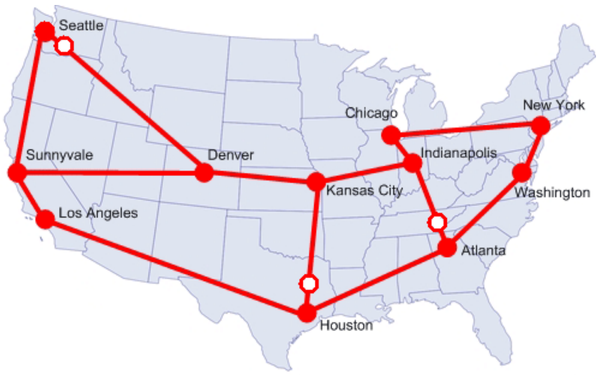

Fig. 7. Abilene Backbone topology by APAR (white nodes are artificial)

In order to sample the Abilene backbone network, we carefully choose 11 vantage points such that each router in the Abilene network appears as an entry or an exit point in at least one of the paths among these vantage points. After collecting path traces among vantage points, we obtain path segments where routers belongs to the Abilene network (i.e., IP addresses are in 198.32.8.0/22). Hence, our data set includes partial path traces belonging to the Abilene backbone.

After preparing the data set, we run APAR and *ally* on the data set to identify alias pairs corresponding to Abilene backbone routers. In this experiment, *Ally* remains completely ineffective since Abilene routers do not respond to *ally* probes. APAR successfully identifies 18 alias pairs out of 23 aliases that exist in the network without any false positives and produces a network map close to the original one as shown in Fig. 7. APAR fails to detect the IP address aliases at the Seattle router because this router does not appear on any path as an internal router. In other words, the Seattle router does not provide transit between Sunnyvale and Denver routers. Similarly, APAR fails to detect alias pairs at Atlanta and Houston routers since Atlanta-Indianapolis and Houston-Kansas City links do not appear in any of the collected paths. Our data set does not include necessary information to detect related alias pairs on these routers due to unicast routing preferences. As a result, this experiment shows that APAR is an effective approach in inferring IP aliases in a given set of path traces.

In summary, in this section we have used all the approaches that are available to us in verifying the presented APAR algorithm. In each case, we have seen that APAR returned results have a small (less than 5%) rate of disagreement with the other approaches. Eventually, there were 106 alias pairs that at least one of the verification approaches disagree with yielding an overall error rate of 5% for APAR. In addition, we have observed that APAR introduces a large number of new alias pairs that *ally* fails to detect.

VII. Conclusions

In this paper we have focused on the IP alias resolution problem. We have discussed the importance of alias resolution in the Internet topology measurement studies and motivated the need for an effective and scalable mechanism for resolving IP aliases. We have then presented an analytical alias resolution method, APAR, to resolve IP aliases without introducing any traffic overhead or requiring participation from the routers.

Our experimental evaluations on a set of collected path traces have shown that, compared to *ally*, the state-of-the-art probing based approach, APAR can resolve more IP aliases. Besides, the two approaches can complement each other to improve the alias resolution success rate significantly. The disagreement ratio between the two approaches is reasonably low. In addition, in our accuracy verification efforts using a small network topology, APAR returned most of the existing aliases without causing any false positives. In our verification efforts, we have seen that the verifiable accuracy of APAR was over 95%.

Based on these observations, the additional set of IP aliases identified by APAR would result in a great improvement on the accuracy of the resulting topology maps. As a result, given the importance of the IP alias resolution task in building realistic router-level Internet maps, the proposed approach makes an important contribution to the area of Internet measurements.

References

[1] D. McRobb, K. Claffy, and T. Monk, *Skitter: CAIDA's macroscopic Internet topology discovery and tracking tool*, 1999, available from http://www.caida.org/tools/skitter/.

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

[3] 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.

[4] Y. Shavitt and E. Shir, "DIMES: Let the Internet measure itself," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 71–74, 2005.

[5] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale Internet measurement," *IEEE Communications*, August 1998.

[6] S. Bilir, K. Sarac, and T. Korkmaz, "Intersection characteristics of end-to-end Internet paths and trees," in *IEEE International Conference on Network Protocols (ICNP)*, Boston, MA, USA, November 2005.

[7] 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.

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

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

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

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

[12] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *IEEE INFOCOM*, Tel Aviv, ISRAEL, March 2000.

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

[14] A. Nakao, L. Peterson, and A. Bavier, "Routing underlay for overlay networks," in *Proceedings of SIGCOMM*, Karlsruhe, GERMANY, August 2003.

[15] M. Gunes and K. Sarac, "Analytical IP alias resolution," in *IEEE International Conference on Communications (ICC)*, Istanbul, TURKEY, June 2006.

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