

# Internet-Scale IP Alias Resolution Techniques \*

Ken Keys

Cooperative Association for Internet Data Analysis (CAIDA)  
University of California, San Diego  
kkeys@caida.org

## ABSTRACT

The well-known traceroute probing method discovers links between interfaces on Internet routers. IP alias resolution, the process of identifying IP addresses belonging to the same router, is a critical step in producing Internet topology maps. We compare the performance and accuracy of known alias resolution techniques, propose some enhancements, and suggest a practical combination of techniques that can produce the most accurate and complete IP-to-router mapping at macroscopic scale.

## Categories and Subject Descriptors

C.2.3 [Computer-Communication Networks]: Network Operations—*Network monitoring*; C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Network topology*

## General Terms

Measurement, Experimentation

## Keywords

Alias resolution, Mercator, iffinder, Ally, RadarGun, APAR, kapar, DisCarte

## 1. INTRODUCTION

The traceroute tool [13] and variants of it are widely used for discovery of network topology [14, 22, 21, 18]. Traceroute works by probing a destination address with a series of packets with increasing initial values of the IP TTL (Time To Live) field. When an IP router receives a packet to be forwarded, it first decrements the TTL field; if the new TTL is zero, the router does not forward the packet, but instead sends an ICMP Time Exceeded error back to the sender. The source address of the ICMP message identifies an interface on the router that sent the message. A series of probes with increasing initial TTL values will normally reveal an address at every hop along the path, with some exceptions. Repeating this process from multiple sources to multiple destinations can reveal many router addresses and links between them.

Each router by definition has at least two interfaces; Internet core routers commonly have 10 to 30. Additionally, each

interface may have multiple addresses. But each traceroute response reveals only a single address of a router. Converting raw IP address topology data discovered by traceroute into a more useful router topology requires identifying which addresses belong to the same routers. This process is called IP alias resolution.

In this report we compare the current state-of-the-art in alias resolution techniques. Section 2 describes how each technique works, and their strengths and weaknesses. Section 3 applies several techniques to comprehensive Internet topology measurements, validates their results against known ground truth, and evaluates their effectiveness. We conclude by identifying a practical combination of techniques that produces the most accurate and complete alias resolution information.

## 2. TECHNIQUES

We classify alias resolution techniques into two types: fingerprint techniques and analytical techniques.

Fingerprint techniques work by sending probe packets to different addresses, and identifying similarities in the responses that indicate the responses came from the same router. Fingerprint techniques are usually accurate, but incomplete because many routers do not respond to the probes.

Analytical techniques draw inferences about topology by analyzing the IP address graph. They are typically less accurate than fingerprinting because they make more assumptions about network engineering practice and deal with indirect, incomplete, and sometimes conflicting data. However, because they do not rely on responses to direct router probes, they can work on non-responding routers where fingerprinting techniques are useless.

### 2.1 Common source address

The earliest alias resolution technique was a fingerprint technique described by Pansiot and Grad [17], and implemented in Mercator [8] and iffinder [15]. It works by sending a UDP or TCP packet to an unused port and comparing the source IP address of the ICMP Port Unreachable error message sent in reply. The router could theoretically respond from any of its addresses. In practice, many routers respond from the address of the interface on the route back to the prober. On these routers, a probe sent to the address of any of the other interfaces will reveal that the probed address and response address are aliases. However, some routers always respond from the probed address, or do not respond at all to this type of probe, making their aliases undetectable

\*This project is sponsored by the U.S. Department of Homeland Security (DHS) Science and Technology (S&T) Directorate Cybersecurity Contract #N66001-08-C-2029.

with this method. If the source address of the error message is in private (RFC 1918) address space, it may not be unique, and additional techniques are required to disambiguate it.

## 2.2 Common IP ID counter: Ally

The IP identification field of packets is designed to identify and allow reassembly of IP datagrams that have been fragmented. Many routers maintain a simple IP ID counter that is incremented with each use, and shared by all interfaces. Consecutive packets generated by such routers will have consecutive ID values, no matter which address is used as the source address of the packets. This shared counter serves as the basis of another fingerprint technique, which was implemented in the Ally component of Rocketfuel [21].

Ally tests a candidate pair of alias addresses by simultaneously sending a probe packet to each, and then sending a third probe to whichever address responds first. If the IP IDs of the three responses are in order and close in value, it suggests the two addresses belong to a single router using a simple shared counter. If not, it may be because the addresses are not aliases, or they could be aliases on a router that does not use a simple shared counter. This ID fingerprinting method is vulnerable to false positives due to two routers' IDs coincidentally synchronizing during the three-packet test, although this weakness can be mitigated by running Ally again at a later time on each pair identified in the first pass. Ally is also vulnerable to false negatives due to routers that do not respond to direct probes, or routers whose ID counters increment too quickly. Additionally, Ally cannot draw any conclusions about routers that appear to not use incrementing ID counters.

But the biggest drawback of the Ally technique is that, given  $n$  addresses, it would require  $O(n^2)$  probes to test all possible pairs. To make Ally more practical, some other heuristic is needed to reduce the size of the search space. One such heuristic, used by Rocketfuel, is to restrict the set of candidate pairs to those pairs in which both IP addresses have similar TTL values as measured from a number of different vantage points. Although this heuristic does reduce the amount of probing needed, it is still not enough for practical use on macroscopic-scale Internet graphs. Also, any pruning heuristic carries the risk of excluding some candidate pairs that would otherwise have been identified as aliases.

## 2.3 Common IP ID counter: RadarGun

RadarGun [7] avoids many of Ally's problems by not working with address pairs, but instead probing the entire list of  $n$  addresses, iterating over the list at least 30 times. Using all of the responses for an address, RadarGun can estimate the rate of change, or *velocity*, of the IP ID of the router with that address. Because two potential aliases may be probed tens of seconds apart, their ID values cannot be compared directly. But after calculating their velocities, RadarGun can interpolate what their values would be at any time during the probing process, and compare the interpolated values. Any two addresses can be inferred to be aliases if they have similar and constant ID velocities, and the ID value in every response from one address is similar to the interpolated ID value of the other address at the same time. Because each test uses many more responses than an Ally test, RadarGun is more tolerant of routers that are occasionally unresponsive due to rate limiting, dropped packets, or other intermittent

losses.

The number of RadarGun probes per address is typically configured to a value between 30 and 100, which much smaller than  $n$ , so the total number of probes is only  $O(n)$ . Although this is much better than Ally's  $O(n^2)$  probes, RadarGun still has some scaling difficulties when applied to large-scale Internet graphs generated by CAIDA, with values of  $n$  in the millions. Because IP ID counters are only 16 bits, they "wrap" back to 0 after reaching 65535. Occasional single wraps between probes are unavoidable, and can be taken into account when calculating velocities. However, if probes to the same address are spaced more than about 30 to 40 seconds apart, multiple wraps become so likely that it is impossible to confidently detect linear ID change or calculate ID velocities. So if the list is too large to probe in its entirety within this maximum packet spacing, while also staying under a desired maximum probing rate, some other heuristic must be used to break the list into smaller pieces.

## 2.4 DNS analysis

If an organization assigns DNS names to router interfaces with a systematic convention that identifies the router, this information can be extracted by decoding the names, as described by Spring *et al.* [20]. This approach requires a human to identify the naming convention of each ISP, and write rules for interpreting the names according to the convention. Additionally, naming conventions are subject to the whims of the ISP, and DNS databases are not always kept up to date. Because of the human intervention required and other problems, we do not currently consider DNS analysis a practical technique for automated macroscopic alias resolution, although future work and homogenization of conventions may improve this situation, and it can be useful for manual validation of other techniques.

## 2.5 Simple graph analysis

In [20], Spring *et al.* described a simple pair of rules that could be used to infer aliases by analyzing the traceroute graph. Because most routers respond to a traceroute probe from the address on which the probe arrives, two addresses with a common successor in traceroute paths are aliases, assuming the IP links are point-to-point. That is, if we have observed path segments  $(a, c)$  and  $(b, c)$ , where letters represent interface addresses, and links are assumed to connect exactly two routers, then, as illustrated in Figure 1,  $a$  and  $b$  must be aliases. Second, addresses in the same (loop-free) path are not aliases. The utility of these rules is limited because they can infer aliases only where two paths from different sources converge, and because Layer 2 constructs allow an IP link to connect more than two routers.

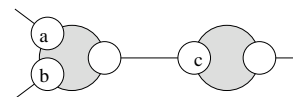
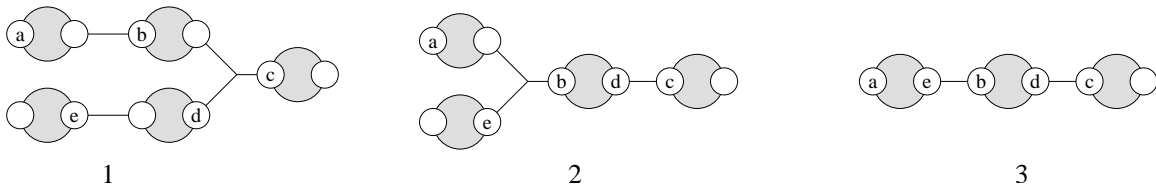


Figure 1: Possible topology for observed path segments  $(a, c)$  and  $(b, c)$ , assuming point-to-point links.

## 2.6 Graph analysis: APAR

The Analytic and Probe-based Alias Resolver (APAR) tool [10, 11] uses a richer set of inference rules, based on



**Figure 2: Three possible topologies for observed path segments  $(a, b, c)$  and  $(d, e)$  with  $c$  and  $d$  on the same subnet. Large circles represent routers; small circles are interfaces on the router.**

identifying the subnets linking routers and then aligning traceroute paths using those subnets.

APAR begins by inferring subnets among all the addresses collected from a large number of traceroutes. It starts by looking for addresses with the same /24 prefixes, splitting those prefixes into pairs of /25 prefixes, and so on down to the /31 prefixes. Any candidate prefix must meet several conditions in order to be considered a legitimate subnet.

Recall that the first and last addresses of an / $x$  subnet, where  $x < 31$ , are reserved for broadcast addresses, so if either of those addresses was observed in a path, the prefix can not be a subnet. This “no broadcast” condition rules out many /30 and larger subnets, but cannot rule out /31 subnets because they do not have broadcast addresses.

Second, the “accuracy” condition requires that no two addresses in the same subnet should appear as non-neighbors in the same trace, because a path should never traverse the same link twice. Appearing as neighbors is possible when the first router on a link responds from its outgoing interface instead of its incoming interface, contrary to standard practice. For example, in Figure 2.3, a traceroute probe sent from the left side would normally make the first router respond from interface  $a$ , and the second router from  $b$ ; however, if the first router responds from  $e$ , we see a path segment of  $(e, b)$ , where  $e$  and  $b$  are on the same subnet.

Third, the “completeness” heuristic requires that at least half of a subnet’s possible addresses were observed in the traceroute paths. For example, a /28 subnet is acceptable only if we observe at least 7 of its 14 possible addresses. Without this condition, APAR would incorrectly infer many sparsely populated large subnets whose interfaces really belong to a scattering of smaller subnets in the same address range. This condition may falsely reject a real subnet that was poorly covered by traceroutes, but smaller subnets (with longer prefixes) within it will still be allowed. This risk decreases with improved traceroute coverage. Point-to-point (/30 and /31) subnets will always be identified if both addresses were seen. APAR has greater confidence in subnets with higher completeness.

After identifying subnets, APAR iterates over them starting with those in which it has the most confidence. It uses each subnet to align segments of path traces and infer aliases. Note that the responses generated by traceroute are usually sent from the interface that received the probe. Thus, given two path segments  $(a, b, c)$  and  $(d, e)$  from different paths, where  $c$  and  $d$  are on the same inferred subnet, we infer that  $c$  and  $d$  are connected, and that the two path segments crossed the  $c$ - $d$  link in opposite directions. There are three possible topologies that would explain this result, shown in Figure 2. Addresses  $b$  and  $d$  are potential aliases. The “common neighbor” condition looks for shared topology

on the left side of the potential aliases. If  $b$  and  $e$  are also on the same subnet as each other<sup>1</sup>, only cases 2 and 3 can be correct, and  $b$  and  $d$  must be aliases. Similarly, if  $a$  and  $e$  are the same address or are already known aliases, only case 3 can be correct, and again  $b$  and  $d$  must be aliases. The “no loop” condition must also hold: two addresses cannot be aliases if they both appear in the same trace, because that would imply a path passed through the same router twice.

After one pass over the inferred subnets, we make a second pass over just the point-to-point subnets, without enforcing the common neighbor condition. That condition is unnecessary because the subnet cannot contain a third address, and so case 1 is ruled out and  $b$  and  $d$  must be aliases. This relaxation of the common neighbor condition is done in a second pass because we are less confident of the fact that the  $c$  -  $d$  subnet is point-to-point than we are of its existence, and we want the earlier inferences in which we are more confident to take precedence. The “no loop” condition is particularly important in this pass where the common neighbor condition is omitted. For example, in case 1 of Figure 2, if we incorrectly infer that the  $c$  -  $d$  subnet is point-to-point, but there was another path observed that passed through  $(\dots a, b, d, e, \dots)$ , the “no loop” condition will rule out  $(b, d)$  as an alias pair.

Additionally, APAR can make use of TTL data to avoid some false positives. To collect this data, one monitor must directly probe each interface address observed in the traceroute paths. Because interfaces on the same subnet are topological neighbors, their distance from a given vantage point should differ by at most 1 hop. Thus, during the subnet inference phase, the “subnet distance” condition requires that the minimum and maximum values of TTLs of interfaces within a potential subnet must differ by at most 1. Avoiding false positive subnet inferences can prevent both false positive and false negative alias inferences later. Similarly, during the alias inference phase, the “alias distance” condition requires that for any two interfaces to be aliases, their TTL values must differ by at most 1. On the other hand, inconsistencies in TTL data may cause us to incorrectly reject some valid subnets and aliases. So, depending on the quality of the TTL data, the net effect could be good or bad.

## 2.7 Graph analysis: kapar

Although the APAR algorithm is promising, the original implementation [9] was not well optimized for production use on large-scale Internet topologies. We wrote our own highly optimized implementation of APAR, called “kapar”,

<sup>1</sup>The publicly available implementation of APAR [9] used a weaker definition for the neighboring subnet than that suggested by [11]: it required only that the  $b$  -  $e$  prefix was not shorter than the  $c$  -  $d$  prefix. In particular, it did not enforce the no broadcast, accuracy, or subnet distance conditions.

to overcome this problem and to fix a few bugs, as well as to experiment with our own improvements to the algorithm.

The most significant optimization was to avoid storing the complete set of paths in memory. Instead, kapar makes a single pass over the set of traces and extracts only the minimum information it needs. First, it finds all unique 3-hop segments to use for the alias resolution phase. Second, it identifies common prefixes of length 24 or greater among addresses in the same trace to generate a list of subnets that cannot exist according to the subnet accuracy condition. Finally, it assigns a unique ID number to each trace, and stores a list of all observed addresses and a compressed bitmap of the IDs of the traces in which each address appeared. These trace ID sets contain sufficient information for checking the no-loop condition.

Kapar also improves upon the APAR algorithm in several ways. First, it can load a set of aliases obtained from another source, e.g. results of a fingerprint technique or published topologies. These aliases are considered more reliable than traceroute paths, so when a combination of an alias pair and a path would violate the no-loop condition, the path is considered incorrect. Rejecting incorrect paths leads to fewer false inferences.

Second, during the subnet formation phase, kapar optionally uses a stricter test for point-to-point subnet existence. Recall that the existence of one of a subnet’s broadcast addresses rules out the existence of the subnet. Thus, any  $/29$  subnet can be split into two  $/30$  subnets if its middle two addresses were not observed, because the middle two addresses correspond to broadcast addresses in the  $/30$  subnets. APAR treats the  $/29$  subnet and both  $/30$  subnets as plausible in this case. However, kapar can use probes to these two “missing middle” (MM) addresses to help disambiguate. If either address elicits a response, the  $/30$  subnet to which it would belong cannot exist, and only the  $/29$  subnet is considered real. Ruling out false  $/30$  subnets prevents kapar from incorrectly relaxing the common neighbor condition and potentially generating false positive aliases.

Third, during the alias inference phase, kapar uses stricter tests for the common neighbor condition. When testing a  $b - e$  subnet, kapar requires that the subnet was inferred during the subnet inference stage (i.e., that it passed the no broadcast, accuracy, subnet distance, and completeness conditions), and that it have a higher rank than the  $c - d$  subnet. However, if there are no inferred subnets that contain  $b$  and  $e$ , kapar must rely on an  $(a, e)$  alias pair. Finding such a pair suggests not only that  $b$  and  $d$  are aliases, but also that  $b$  and  $e$  are on the same subnet, one that was not inferred during the subnet inference phase because of low completeness. Kapar (optionally) does not consider the common neighbor condition satisfied unless this implied  $b - e$  subnet meets the no broadcast, accuracy, and distance conditions. We call this optional extra test in the case of an  $(a, e)$  pair “Subnet Neighbor Verification” (SNV).

Finally, the kapar implementation can make use of TTL data obtained from multiple vantage points. This additional data imposes more constraints on both the subnet formation phase and alias resolution phase, further reducing the rate of false positives in each.

## 2.8 Graph analysis: DisCarte

The Record Route IP option, although disabled on many routers, provides another source of topology data that could

be used to complement traceroute data analysis. However, it is difficult to use effectively because of inconsistent implementations by routers and conflicts when attempting to align it with traceroute data. DisCarte [19] addresses these problems by using disjunctive logic programming (DLP) to apply a set of practical network engineering constraints to Record Route and traceroute data to make logical inferences about topology and alias resolution.

Unfortunately, DisCarte’s DLP approach as currently implemented is extremely computationally expensive. Working with traces between 379 sources and 376,408 destinations, the authors found the complete solution intractable. Even after dividing the data into subsets that would result in an incomplete solution, their runtime on a 341-processor Condor cluster was measured in CPU-years. For this reason, we do not currently consider DisCarte a practical technique for routine alias resolution at macroscopic scale.

## 3. EVALUATION

We used the scamper [16] tool on CAIDA’s Archipelago (“Ark”) [12] measurement infrastructure to collect 190 million ICMP Paris [6] traceroute-style traces over 28 days from 26 geographically distributed monitors. The destinations of these traces were selected from every  $/24$  sub-prefix of every routed prefix on the Internet. These traces served as the base input to all of our tests. We extracted two sets of addresses from this dataset: all 2,409,959 intermediate path addresses, i.e. router interfaces, and 65,626 “missing middle” (MM) addresses as described in section 2.7. Additionally, TTL datasets were collected 4 days after the end of the trace collection period by sending a single ICMP Echo Request (“ping”) probe from each Ark monitor to every observed and MM address.

To test the common source address method, we ran our own implementation, iffinder [15], on all 26 Ark monitors, 5 days after the end of trace collection. As input, we used the router interface addresses, both with and without the MM addresses. We also tested the use of TTL data to reduce false positives.

We did not attempt to evaluate any of the common IP ID counter techniques. According to the authors’ analysis and our own preliminary evaluation, Ally has been superseded by RadarGun in both accuracy and efficiency. However, RadarGun was released only recently, so we have not had time to fully evaluate it in our Ark environment nor implement any of our planned accuracy and scalability enhancements.

We also did not evaluate any DNS analysis technique, because they currently require too much human input for routine macroscopic automated use.

To test analytic techniques, the original implementation of the APAR algorithm proved too inefficient to run with our large dataset. Instead, we used our own kapar implementation, which is efficient enough to run many tests in a reasonable amount of time, and has the flexibility to experiment with various test and input configurations. We tested it with no TTL data, with TTL data from observed addresses, and with TTL data from observed and MM addresses.

Finally, we combined the results of iffinder and kapar to see how they could complement each other. Again, we used various combinations of TTL and MM input.

For validation of our alias resolution results, we obtained topology data for CANet [1], GÉANT[2], Internet2 [3] NLR

			CAnet			GÉANT			Internet2			NLR			WIDE			total	
			R	TP	FP	R	TP	FP	R	TP	FP	R	TP	FP	R	TP	FP	R	P
reality			5	117		19	478		9	713		7	231		6	88			
techniques																			
iffinder	kapar	TTLs																	
yes	-	-	0	0	0	0	0	0	0	0	0	6	100	0	3	22	0	52779	140407
yes M	-	-	0	0	0	0	0	0	0	0	0	6	100	0	3	22	0	52986	142450
yes M	-	1 H	0	0	0	0	0	0	0	0	0	6	95	0	3	22	0	52493	140355
yes M	-	all H	0	0	0	0	0	0	0	0	0	6	95	0	3	22	0	40361	94680
-	yes	-	4	52	0	22	104	28	17	190	45	9	69	11	5	38	0	131363	844059
-	yes N	-	4	52	0	22	118	16	17	191	44	9	70	11	5	38	0	131449	844293
-	yes	1	4	56	0	21	112	27	16	205	44	9	69	6	5	36	0	135238	838571
-	yes N	1	4	56	0	20	123	16	16	203	44	9	70	6	5	36	0	135291	838628
-	yes	all	4	45	0	16	99	12	16	165	4	8	76	6	5	36	0	131562	771812
-	yes N	all	4	45	0	16	99	12	15	171	4	8	76	6	5	36	0	131561	771776
yes M	yes	-	4	52	0	22	102	24	14	235	21	7	135	0	5	41	0	156471	902007
yes M	yes N	-	4	52	0	22	118	16	13	250	11	7	135	0	5	41	0	156573	902279
yes M	yes	1 L	4	56	0	20	111	23	14	244	13	7	135	0	4	38	0	160655	895859
yes M	yes N	1 L	4	56	0	20	123	16	13	238	37	7	135	0	4	38	0	160784	895948
yes M	yes	all L	4	45	0	16	103	12	14	159	6	6	134	0	5	39	0	159654	845083
yes M	yes N	all L	4	45	0	16	99	12	14	171	2	6	134	0	5	39	0	159662	845037

**Table 1: Comparison of alias resolution techniques on known networks. Result columns list the number of routers with multiple interfaces (R), and alias pairs: true (TP), false (FP), and total (P). In technique columns, “-” indicates the corresponding technique was not used. In the “iffinder” column, “M” indicates MM probes were used. The “kapar” column shows whether Subnet Neighbor Verification (N) was used. The “TTLs” column shows whether TTLs from one (1) or all (all) monitors were used, and whether the TTL precedence was high (H) or low (L) relative to iffinder.**

[4], and WIDE [5] networks. Data for the first four of these networks was available publicly.

Table 1 shows the results of running various combinations of iffinder and kapar, compared to known topology data. The row labeled “reality” reflects only those alias pairs in the real published topologies for which both addresses appeared in Ark paths. A perfect alias resolution technique would discover all of those pairs, but should not be expected to discover pairs containing addresses that were not in its input. Note that a router with  $i$  interfaces has  $i * (i - 1) / 2$  pairs of interfaces; for example, a router with 10 interfaces has 45 pairs, and incorrectly merging a 3-interface router with a 4-interface router would introduce 12 false alias pairs. Also note that the number of routers can be too low if many aliases were missed, or too high if individual routers were split into multiple routers.

### 3.1 iffinder

Iffinder works well when probed routers respond from the interface on the route back to the prober as this tool expects. The NLR and WIDE columns of the first four experiments in the table illustrate that in this case iffinder correctly identifies many true aliases and yields no false positives. However, when routers do not respond to direct probing, or always respond from the probed address, iffinder is completely ineffective, as is the case in the other three networks used for verification. A single organization typically uses similar configurations on all of its routers, so iffinder tends to work either quite well or not at all on an entire network. Of the observed router interfaces we probed, 64% responded with a Port Unreachable message to at least one of the moni-

tors; of those responders, 5.6% responded to at least one monitor from an address other than the one probed, revealing an alias pair. This low resolution rate suggests that the common source address technique, while useful on some networks, is insufficient by itself on the Internet in general.

Adding the MM addresses to iffinder’s list of probe targets did not make any difference on the networks where we were able to verify the results of iffinder. However, probing MM addresses did produce a small increase in the number of routers and alias pairs found on the Internet in general. Although we can not conclusively prove that these additional inferences are correct, we believe that they are, since both theory and (limited) experimental evidence give no reason to expect a significant number of false positives. Because MM probes appear to have at least a small beneficial effect with almost no risk of detrimental effect, we continue to use MM probes in our later tests. The number of MM addresses was less than 3% of the number of observed router addresses, so probing them did not incur substantial additional overhead.

Since iffinder found no false positives, adding TTL constraints to it is unnecessary. In fact, adding TTL constraints actually hurt iffinder’s results slightly by incorrectly excluding some pairs that were correct. We expect that this relative reliability is because TTL comparisons rely on two separate probes, which are vulnerable to routing differences, whereas an iffinder inference relies on just one probe.

### 3.2 kapar

Kapar does not rely on direct probes and, therefore, works more consistently than iffinder across all of the networks with known topologies. However, kapar does not necessarily

find as many correct alias pairs on networks where both techniques work, and does find a small number of incorrect alias pairs. Kapar finds about 7 times more alias pairs than iffnder on the Internet in general, but we have no practical way to find out how many of those are false positives.

In most cases, Subnet Neighbor Verification (SNV) improved one or both of true positives and false positives, or had no effect. In only one case did SNV have a small detrimental effect, by incorrectly ruling out two true positives. Thus, SNV seems to be a useful addition to the APAR algorithm.

When we add in TTL data from one monitor, we see some decrease in false positives as expected, and, surprisingly, we also see some *increase* in true positives. This increase could be explained by the TTL data ruling out some false subnets which otherwise would have in turn ruled out some true aliases. More testing is necessary to decide if this effect is consistent or was just luck. With TTL data from all 26 monitors, the results are closer to our expectations: the decrease in false positives is even greater than that for TTLs from one monitor, but there is also an unfortunate decrease in true positives. Future work may help isolate the helpful effects of TTLs from the harmful effects and make TTLs more useful. For example, sending multiple pings to each address may reveal some addresses that reply with varying TTL values, making them too unreliable to be used in distance conditions.

### 3.3 iffnder and kapar

Combining iffnder and kapar lets us take advantage of the strengths of both methods. Because of iffnder's negligible false positive rate, we always configure kapar to treat iffnder's alias inferences as correct if they would conflict with its own inferences. Together, the two methods discover more true alias pairs than either method alone. Somewhat surprisingly, even on networks where routers do not respond to iffnder probes, adding iffnder results to kapar's analysis was sometimes helpful. This effect is likely due to the APAR algorithm propagating information from accurate iffnder results along path segments into neighboring networks.

When using both iffnder and kapar methods together, adding the Subnet Neighbor Verification technique appears to be a net benefit, although not as much as when we ran kapar without iffnder (section 3.2).

When testing TTL data with the iffnder/kapar combination, we have the choice of trusting the TTL data more or less than iffnder data. If the two data sources conflict, we can discard the iffnder inference and keep the TTL data for use in kapar, or keep the iffnder inference and discard the TTL data so it does not affect kapar. The experimental results (not shown) are somewhat mixed, but do slightly favor treating TTLs as less reliable than iffnder data, consistent with our results in section 3.1. As in section 3.2, future work may make TTLs more useful.

## 4. CONCLUSIONS

Every alias resolution technique tested has strengths and weaknesses. Fingerprint techniques are accurate when routers respond to their probes, but many routers do not, leaving large gaps in their coverage. Analytic techniques rely on indirect data and assumptions about network design, making them somewhat less accurate, but they do not depend on direct probing and thus work more evenly across the entire

Internet. Adding TTL constraints and intelligently chosen additional probing can further increase the accuracy of analytic approaches. By combining the strengths of these techniques, we can obtain a better set of results than we could with any one technique alone. Specifically, we found iffnder, kapar, TTL constraints, and "missing middle" probes to be an effective and scalable combination.

## 5. REFERENCES

- [1] <http://dooka.canet4.net/>.
- [2] <http://stats.geant2.net/lg/>.
- [3] <http://vn.gnroc.iu.edu/Internet2>.
- [4] <http://routerproxy.gnroc.iu.edu/nlr2/>.
- [5] Private communication with Kenjiro Cho and Yuji Sekiya at WIDE.
- [6] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, and M. Latapy. Avoiding traceroute anomalies with Paris traceroute. In *IMC*, Oct. 2006.
- [7] A. Bender, R. Sherwood, and N. Spring. Fixing Ally's growing pains with velocity modelling. In *IMC*, 2008.
- [8] R. Govindam and H. Tangmunarunkit. Heuristics for Internet map discovery. In *INFOCOM*, March 2000.
- [9] M. H. Gunes. APAR tool. <http://itom.utdallas.edu/data/APAR.tar.gz> (accessed 2008-07-02).
- [10] M. H. Gunes and K. Sarac. Analytical IP alias resolution. In *IEEE International Conference on Communications (ICC 2006)*, June 2006.
- [11] M. H. Gunes and K. Sarac. Resolving IP aliases in building traceroute-based internet maps. Technical report, December 2006.
- [12] Y. Hyun. Archipelago measurement infrastructure. <http://www.caida.org/projects/ark/>.
- [13] V. Jacobson. traceroute tool. <ftp://ftp.ee.lbl.gov/traceroute.tar.gz>.
- [14] k. claffy, T. Monk, and D. McRobb. Internet tomography. In *Nature*, January 1999.
- [15] K. Keys. iffnder tool, 2000. <http://www.caida.org/tools/measurement/iffnder/>.
- [16] M. Luckie. scamper tool. <http://www.wand.net.nz/scamper/>.
- [17] J.-J. Pansiot and D. Grad. On routes and multicast trees in the Internet. In *ACM SIGCOMM*, 1998.
- [18] Y. Shavitt and E. Shir. DIMES: Let the Internet measure itself. In *ACM Computer Communications Review*, October 2005.
- [19] R. Sherwood, A. Bender, and N. Spring. DisCarte: A disjunctive Internet cartographer. In *ACM SIGCOMM*, 2008.
- [20] N. Spring, M. Dontcheva, M. Rodrig, and D. Wetherall. How to resolve IP aliases. Technical report, May 2004.
- [21] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*, 2002.
- [22] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public Internet measurement facility. In *4th USENIX Symposium on Internet Technologies and Systems*, 2002.