IP Alias Resolution Techniques

version 1.1, 2009-01-19

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 known alias resolution techniques, and suggest a practical combination of techniques that can produce the most accurate and complete IP-to-router mapping.

1. INTRODUCTION

The traceroute tool[12] and variants of it are widely used for discovery of network topology[13][21][20][17]. 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 often have dozens. Additionally, each interface may have multiple addresses. But each traceroute response reveals only a single address of a router. We must identify which addresses belong to the same routers in order to convert the IP address topology discovered by traceroute to a more useful router topology. This process is called IP alias resolution.

In this report we compare the current state-of-the-art in alias resolution techniques. In section 2, we describe how each of the techniques works, and their strengths and weaknesses. Then in section 3, we apply several of the techniques to the real Internet, validate their results against known data, and evaluate their effectiveness. We conclude by identifying a practical combination of techniques that produces the most accurate and complete alias resolution information.

*This project is sponsored by the U.S. Department of Homeland Security (DHS) Science and Technology (S&T) Directorate.

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 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.

Table 1 contains a summary of the alias resolution techniques we will consider. The remainder of this section will describe each technique in more detail.

2.1 Common source address

The earliest alias resolution technique was a fingerprint technique described by Pansiot and Grad[16], and implemented in Mercator[7] and iffinder[14]. 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

Technique	Implementations	Advantages	Disadvantages
common source address	Mercator, iffinder	Effective on responding routers. Virtually no false positives.	Completely ineffective on unresponsive routers and routers that always respond from probed address.
common IP ID	Ally	Effective on responding routers. Low false positive rate.	Requires $O(n^2)$ probes. Completely ineffective on unresponsive routers.
common IP ID	RadarGun	Effective on responding routers. Low false positive rate. Requires less probing than Ally. More tolerant of noise than Ally.	Completely ineffective on unresponsive routers.
DNS analysis		No additional probing. Accurate on routers with a parsable naming convention.	Requires significant human input. Ineffective on routers without an identifiable naming convention.
simple graph analysis		No additional probing.	Accuracy and completeness limited by traceroute. Simplistic rules yield significant false positive rate.
graph analysis	APAR	Additional probing not required (but can be useful).	Accuracy and completeness limited by traceroute.
graph analysis	kapar	Additional probing not required (but can be useful). More accurate and efficient than original APAR.	Accuracy and completeness limited by traceroute.
graph analysis	DisCarte	Combines data from traceroute and Record Route; potentially highly accurate.	Prohibitively computationally expensive.

Table 1: Summary of alias resolution techniques.

to this type of probe, making their aliases undetectable with this method. Additionally, 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.

The Ally component of Rocketfuel[20] 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, routers that rate-limit their responses 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, a straightforward application of Ally 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 the large-scale Internet graphs generated by CAIDA. 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[6] 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 fixed to a relatively small number, typically 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 their 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.*[19]. 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 consider DNS analysis a viable technique for automated alias resolution, although it can be useful for manual validation of other techniques.

2.5 Simple graph analysis

Spring et al.[19] also described a simple set of rules that could be used to infer aliases based only on analyzing the traceroute graph. When responding to a traceroute probe, most routers will respond from the address on which the probe arrived. First, 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 lowercase letters represent interface addresses, and links are assumed to connect exactly two routers, then a and b must be aliases, as illustrated in Figure 1. Second, addresses in the same (loop-free) path are not aliases. These rules are of limited utility, since they can infer aliases only where two paths from different sources converge, and Layer 2 constructs may allow an IP link to connect more than two routers.

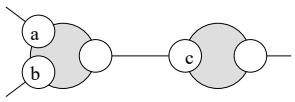

Figure 1: Possible topology for observed path segments (a, c) and (b, c), assuming point-to-point links. Large circles represent routers; small circles are interfaces on the router.

2.6 Graph analysis: APAR

The Analytic and Probe-based Alias Resolver (APAR) tool[9][10] uses a richer set of inference rules, based on identifying the subnets linking routers and then aligning traceroute paths using those subnets.

Normal network engineering practice is to assign a unique address prefix or subnet to each IP link; each interface on the link gets an address within that prefix. The notation "/x" indicates a prefix length x, in which the first x bits identify the subnet and the remaining $32 - x$ bits identify an address on the subnet. In /31 subnets, there are exactly 2 interface addresses. In subnets with shorter prefixes, the first and last addresses in the range are broadcast addresses and are not used for interfaces, so there are $2^{32-x} - 2$ interface addresses available. In normal practice, the minimum prefix length used for a subnet is 24 bits, which can address up to 254 interfaces.

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 pre-

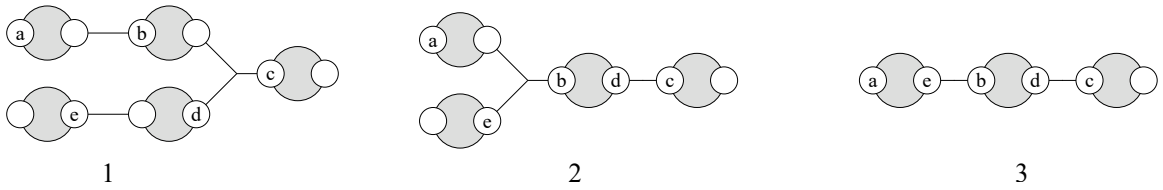

Figure 2: Three possible topologies for observed path segments (a, b, c) and (d, e) with c and d on the same subnet. Large circles represent routers; small circles are interfaces on the router.

fixes, 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.

First, by definition, if either of the broadcast addresses of the subnet were observed, the subnet cannot be real. 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 prefix should ever 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 interface b; however, if the first router responds from interface e, we see a path segment of (e, b), where e and b are on the same subnet.

Third, more heuristically, the "completeness" condition requires that at least half of a subnet's possible addresses were actually observed in the traceroute paths. For example, for a /28 subnet to be acceptable, we must have observed 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 much smaller subnets in the same address range. The completeness condition may falsely reject a real prefix that was poorly covered by traceroutes, but the smaller subnets (longer prefixes) within it will still be allowed. The risk of falsely rejecting real subnets 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 TTL Exceeded messages generated by traceroute are usually sent from the interface that received the probe. Therefore, given two path segments (a,b,c) and (d,e) from different paths, where c and d are on the same inferred subnet, we align the traces in the following manner:

$$\begin{array}{ccccc} a & \rightarrow & b & \rightarrow & c \\ & & & \swarrow & \\ e & \leftarrow & d & & \end{array}$$

There are three possible topologies that would explain this alignment, shown in Figure 2. Addresses b and d are potential aliases. The "common neighbor" condition looks for shared topology on the left side (in the alignment diagram) of the potential aliases. If b and e are also on the same subnet as each other[1], 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 (i.e., /30 and /31) subnets, this time without enforcing the common neighbor condition. The common neighbor condition is unnecessary because a point-to-point 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 size of the subnet containing c and d than we are of the existence of a subnet containing c and d. The "no loop" condition is particularly important in this second pass where the common neighbor condition is not required. 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 (...a,b,d,e...), the "no loop" condition will rule out (b,d) as an alias pair.

[1]The publicly available implementation of APAR[8] used a weaker definition for the neighboring subnet than that suggested by [10]: it required only that b and e shared a prefix that was not shorter than that of the c - d subnet. In particular, it did not require the b - e subnet to pass the accuracy and subnet distance conditions.

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 no more than 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 no more than 1.

2.7 Graph analysis: kapar

Although the APAR algorithm is promising, the original implementation[8] was not well optimized for production use on large-scale Internet topologies. We wrote our own highly optimized implementation of APAR, called "kapar", 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. the results of a fingerprint technique or even 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 subnet existence. Note that the existence of one of a subnet's broadcast addresses in a path (or any other source) rules out the existence of the subnet. Thus, any subnet with prefix shorter than 29 can be split into two subnets of half the size (with prefixes 1 bit longer) if its middle two addresses were not observed, because the middle two addresses correspond to broadcast addresses in the smaller subnets. APAR treats the large subnet and both small subnets as real 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 smaller subnet to which it would belong cannot exist, and the larger subnet is considered real. But if neither address elicits a response, kapar (optionally) assumes that the larger subnet is not real, but the two smaller subnets are still real. We call this conjecture the MM constraint. These additional probes can be conveniently folded into the probes used to collect TTL data, and/or the probes of a fingerprint technique whose results are fed into kapar.

We do not expect MM probes to generate any additional complaints from network operators. If the MM addresses are normal addresses in the middle of a real subnet, they are qualitatively no different than the other addresses we already probe on the same subnet. And if the MM addresses are broadcast addresses belonging to the smaller subnets, they will normally be filtered at or before the router connected to the subnet to which they belong. Also, we expect the number of MM addresses that need to be probed to be only a small fraction of the total addresses probed.

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 provides another source of topology data that could be used for analytic alias resolution, but has historically been difficult to use effectively because of inconsistent implementations by routers and conflicts when attempting to align it with traceroute data. DisCarte[18] addresses these problems by using disjunctive logic programming (DLP) to apply a set of constraints based on Internet engineering practice to Record Route and traceroute data, and 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 consider DisCarte a practical technique for routine alias resolution.

3. EVALUATION

We used the scamper[15] tool running on CAIDA's

	network											
	CAnet			GÉANT			Internet2			NLR		
technique	R	TP	FP	R	TP	FP	R	TP	FP	R	TP	FP
reality	5	172		18	540		9	713		7	231	
iffinder	0	0	0	0	0	0	0	0	0	6	100	0
iffinder + all TTLs	0	0	0	0	0	0	0	0	0	6	95	0
kapar	5	30	0	14	75	9	14	193	25	9	61	7
kapar + MM	5	21	0	14	75	6	15	193	26	9	61	7
kapar + 1 TTL + MM	5	11	0	12	51	5	13	183	26	8	62	6
kapar + all TTLs	3	41	0	11	80	6	12	163	6	8	67	6
kapar + all TTLs + MM	3	41	0	11	80	6	12	163	6	8	67	6
iffinder + kapar + MM	4	29	0	16	63	6	15	209	13	6	132	0
iffinder + kapar + 1 TTL + MM	4	17	0	13	52	5	15	206	7	6	126	0
iffinder + kapar + all TTLs	3	41	0	11	84	6	14	167	4	7	127	0
iffinder + kapar + all TTLs + MM	3	41	0	11	84	6	14	167	4	7	127	0

Table 2: Comparison of alias resolution techniques on four known networks. Columns under each network list the number of routers with multiple interfaces (R), true positive alias pairs (TP), and false positive alias pairs (FP). In the technique labels, "1 TTL" means TTLs measured from a single monitor, "all TTLs" means TTLs measured from all monitors, and "MM" means the Missing Middle constraint was used.

Archipelago ("Ark")[11] measurement infrastructure to collect 372,623,376 ICMP Paris[5] traceroute-style traces over 58 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 by sending a single ICMP Echo Request ("ping") probe from each Ark monitor to every address in both sets.

To test the common source address method, we ran our own implementation, iffinder, on all 26 Ark monitors. 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 enhancements.

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

To test analytic techniques, the original implementation of the APAR algorithm proved too inefficient and could not run to completion with our large dataset on our available computing resources. 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 and without the MM constraint, as well as 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 be used to complement each other. Again, we used various combinations of TTL and MM input.

For validation of our alias resolution results, we obtained publicly available topology data for CAnet[1], GÉANT[2], Internet2[3], and NLR[4] networks.

Table 2 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.

Iffinder works well when probed routers respond from the interface on the route back to the prober as this tool expects. The iffinder/NLR cell in the table illustrates that in this case iffinder correctly identifies many true aliases and yields few or 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 monitors; 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.

Since iffinder found no false positives, the addition of TTL constraints is unnecessary. In fact, adding TTL constraints actually hurt iffinder's results slightly by incorrectly excluding some pairs that were correct. This finding supports our intuition that TTL data is more error-prone than direct probes as an indicator of topology.

Kapar does not rely on direct probes and, therefore, works more consistently than iffinder on all four networks. However, it does not find as many correct alias pairs on networks where both techniques work, and introduces a small number of incorrect alias pairs.

The addition of MM constraints to kapar appears to hurt slightly more than it helps. This effect is due to the fact that many real routers do not respond to the direct MM probes, making it a mistake to interpret a non-response as evidence of non-existence of an address. Unless a better test of non-existence is found, we should not use the MM constraint. However, it is still perfectly valid to use a response to an MM probe to prove that an address *does* exist and thus rule out any subnet that would have it as a broadcast address.

The addition of TTL data from a single monitor seems to do more harm than good. However, using TTL data from all 26 monitors improves the true positive rate on 3 of the 4 networks, and on the remaining network improves the false positive rate more than it worsens the true positive rate. This change appears to be a net improvement, although it is difficult to be sure with verification on only 4 networks. Future work may help isolate the helpful effects of TTLs from the harmful effects and make TTLs more useful.

Finally, combining iffinder and kapar lets us take advantage of the strengths of both methods. Together, the two methods discover more alias pairs than either method does alone. Somewhat surprisingly, even on networks where routers do not respond to iffinder probes, adding iffinder results to kapar's analysis was helpful. This effect is likely due to the APAR algorithm propagating information from accurate iffinder results along path segments into neighboring networks.

Unfortunately, the version of kapar we used for testing treated TTL data as more reliable than both iffinder data and path data. So, it is not surprising that the addition of TTL data gave mixed results. As demonstrated in previous tests, TTL data should be considered *less* reliable than iffinder, but *more* reliable than path data. That is, when TTL data conflicts with iffinder data, the iffinder data should be trusted, and the conflicting TTL data should be discarded; then if the remaining TTL data conflicts with path data, the conflicting path should be discarded. In the near future we will produce a version of kapar that ranks reliability in this manner, which we expect to generate results of equal or greater accuracy than those reported here.

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 iffinder, kapar, TTL constraints, and "missing middle" probes to be an effective combination.

5. REFERENCES

[1] `http://dooka.canet4.net/`.

[2] `http://stats.geant2.net/lg/`.

[3] `http://vn.grnoc.iu.edu/Internet2`.

[4] `http://routerproxy.grnoc.iu.edu/nlr2/`.

[5] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, and M. Latapy. Avoiding traceroute anonmalies with Paris traceroute. In *IMC*, October 2006.

[6] A. Bender, R. Sherwood, and N. Spring. Fixing Ally's growing pains with velocity modelling. In *IMC*, 2008.

[7] R. Govindam and H. Tangmunarunkit. Heuristics for Internet map discovery. In *INFOCOM*, March 2000.

[8] M. H. Gunes. APAR tool. `http://itom.utdallas.edu/data/APAR.tar.gz` (accessed

2008-07-02).

[9] M. H. Gunes and K. Sarac. Analytical IP alias resolution. In *IEEE International Conference on Communications (ICC 2006)*, June 2006.

[10] M. H. Gunes and K. Sarac. Resolving IP aliases in building traceroute-based internet maps. Technical report, December 2006.

[11] Y. Hyun. Archipelago measurment infrastructure. `http://www.caida.org/projects/ark/`.

[12] V. Jacobson. traceroute tool. `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`.

[13] k. claffy, T. Monk, and D. McRobb. Internet tomography. In *Nature*, January 1999.

[14] K. Keys. iffinder tool, 2000. `http://www.caida.org/tools/measurement/iffinder/`.

[15] M. Luckie. scamper tool. `http://www.wand.net.nz/scamper/`.

[16] J.-J. Pansiot and D. Grad. On routes and multicast trees in the Internet. In *ACM SIGCOMM*, 1998.

[17] Y. Shavitt and E. Shir. DIMES: Let the Internet measure itself. In *ACM Computer Communications Review*, October 2005.

[18] R. Sherwood, A. Bender, and N. Spring. Discarte: A disjunctive Internet cartographer. In *ACM SIGCOMM*, 2008.

[19] N. Spring, M. Dontcheva, M. Rodrig, and D. Wetherall. How to resolve IP aliases. Technical report, May 2004.

[20] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*, 2002.

[21] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public Internet measurement facility. In *4th USENIX Symposium on Internet Technologies and Systems*, 2002.

APPENDIX

Changes

- v1.0, 2008-12-12
  - initial revision
- v1.1, 2009-01-19
  - clarified relaxation of common neighbor condition for point-to-point subnets in APAR
  - added notes on differences between APAR description and implementation
  - added more detail on response rate to iffinder
  - other minor clarifications