Internet-Scale IPv4 Alias Resolution with MIDAR

Ken Keys, Young Hyun, Matthew Luckie, and k claffy

***Abstract*—A critical step in creating accurate Internet topology maps from traceroute data is mapping IP addresses to routers, a process known as alias resolution. Recent work in alias resolution inferred aliases based on similarities in IP ID time series produced by different IP addresses. We design, implement, and experiment with a new tool that builds on these insights to scale to Internet-scale topologies, i.e., millions of addresses, with greater *precision* and *sensitivity*. MIDAR, our Monotonic ID-Based Alias Resolution tool, provides an extremely precise ID comparison test based on monotonicity rather than proximity. MIDAR integrates multiple probing methods, multiple vantage points, and a novel sliding-window probe scheduling algorithm to increase scalability to millions of IP addresses. Experiments show that MIDAR's approach is effective at minimizing the false positive rate sufficiently to achieve a high *positive predictive value* at Internet scale. We provide sample statistics from running MIDAR on over 2 million addresses. We also validate MIDAR and RadarGun against available ground truth and show that MIDAR's results are significantly better than RadarGun's. Tools such as MIDAR can enable longitudinal study of the Internet's topological evolution.**

***Index Terms*—alias resolution, Internet topology, network measurement, IP identifier**

I. Introduction

VARIANTS of the traceroute tool [1] are widely used to discover Internet topology [2]–[5]. Traceroute shows the sequence of router interfaces on the path from the source to the destination, and executing traceroute from multiple sources to multiple destinations reveals many router interfaces and allows us to infer links between them. A router by definition has at least two interfaces; Internet core routers often have dozens. Alias resolution is the process of identifying which interface IP addresses belong to the same routers and is required to convert the abstract IP-level topology discovered by traceroute to a more concrete router-level topology [6], [7] that better describes the physical infrastructure of routers and links and thus is more useful for studying the diversity and resiliency of the Internet infrastructure. The research community's current inability to draw a map that closely reflects physical connectivity is a fundamental gap in our knowledge of the Internet, and limits what else we can know or even study, including investigating the potential impact of loss of critical components of the infrastructure. In conjunction with techniques for annotating routers with AS ownership [8], router-level topologies also enable the study of Internet economics and policy questions, such as which paths are not only feasible but consistent with routing policies. Annotated with geolocation information [9], router-level topologies enable the study of ISP relationships at a finer granularity than allowed by an AS-level topology, such as peering at multiple geographical locations.

There are many alias resolution techniques and implementations available [10], but they are limited in accuracy and coverage, and none are practicable at Internet-wide scale, i.e., millions of IP addresses. The Mercator technique [6], [11], [12] identifies aliases by sending a probe packet to one address and getting a response from a different address. Ally [4] infers that a pair of addresses are aliases if probe packets sent to them produce responses with IP ID values in the correct order. Spring et al. [13] described techniques for drawing alias inferences from similarities in reverse DNS lookups, and from simple analysis of traceroute graphs. APAR [14], [15] and kapar [10] use more sophisticated graph analysis techniques to infer subnets linking routers, and from that, aliases. DisCarte [16] infers aliases from analysis of a graph created from combined traceroute and Record Route data. RadarGun [17] looks for similarities in IP ID time series collected from many addresses. Sherry et al. [18] describe the use of the IP prespecified timestamp option to infer aliases. MERLIN [19] and `mrinfo` [20] send an IGMP `ASK_NEIGHBORS` message to list the IPv4 multicast-enabled interfaces of a router.

In this paper, we introduce MIDAR, our Monotonic ID-Based Alias Resolution tool, an IP ID-based alias resolution technique inspired by Ally and RadarGun. An *IP ID value* is a 16-bit number stored in the IP ID field in the IP header, which the sender of a packet sets to some unique value so that the recipient can identify and reassemble fragmented packets. For alias resolution purposes, we are concerned with the IP ID values of packets *originated* by a router, rather than *forwarded* by a router. Routers themselves can send packets, for example, by responding to ping or traceroute; by running BGP or NTP; and by providing NetFlow, SNMP, or remote terminal access. There is no standard method for generating IP ID values, but many routers maintain a simple IP ID counter that is incremented for packets it generates and which *wraps* from 65 535 to 0. The key observation is that if a router uses a *shared IP ID counter* for generating IP ID values, then the router will use consecutive IP ID values when sending consecutive packets no matter which interface address it uses as the source address. Thus, if two addresses share a counter, then they are conclusively aliases, and their *IP ID time series*, a sequence of IP ID values collected over time, will have similar values in a given measurement period and will form a monotonically increasing IP ID sequence when merged together, modulo counter wraps. The latter expresses the *monotonicity requirement*, a necessary condition for two

Manuscript received June 3, 2011.

K. Keys, Y. Hyun, M. Luckie, and k claffy are with the Cooperative Association for Internet Data Analysis (CAIDA), University of California, San Diego, CA 92093 USA (e-mail: kkeys,youngh,mjl,kc @caida.org).

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

time series to be derived from a shared counter. IP ID-based alias resolution techniques infer aliases by analyzing the IP ID values in response packets and inferring which interface addresses use a shared counter. RadarGun infers a shared counter by looking for similar time series values, whereas Ally and MIDAR infer a shared counter by checking for the monotonicity requirement, though in different ways.

Most routers seem to use a single IP ID counter shared across all interfaces and protocols, but any IP ID based alias resolution technique must account for those that do not. Some routers set the IP ID to zero or some other constant value, a random value, or the value used in the probe packet [17]. Such non-counter IP ID values can be detected and excluded from IP ID-based alias resolution. Some other routers use separate counters for each interface or subset of interfaces. Use of such unshared IP ID counters is undetectable from an analysis of IP ID values alone. Because two addresses may be aliases but not share an IP ID counter, IP ID-based techniques cannot find all alias pairs, and cannot definitively conclude that two addresses that do not share a counter are not aliases. Thus, IP ID-based techniques can produce three results: 1) *positive* shared counter and *positive* aliases; 2) *negative* shared counter and *inconclusive* aliases; 3) *inconclusive* shared counter and *inconclusive* aliases.

Because of the limited precision of IP ID values and the limited variation in rates of change, or *velocities*, of IP ID counters (see Appendix A), it is inevitable that any large set of addresses will have many pairs of addresses with similar or aligned IP ID time series out of sheer coincidence, as predicted by the birthday paradox and the pigeonhole principle. Suppose a given alias resolution technique has a *false positive rate* of related to how much tolerance it allows when comparing two time series to infer a shared counter. Then for N addresses, there are $O(N^2)$ pairs of addresses, and we can thus expect $O(\quad N^2)$ false positives (FP), but only $O(N)$ true positives (TP) (see Appendix C). When $N > 10^6$, as in the case of Internet-scale alias resolution, the *prevalence* of aliases is extremely low, approximately 1 in a million. Hence, unless the false positive rate is extremely low, $\quad 1\ N = 10^{-6}$, false positives can overwhelm true positives, and the alias technique will not be reliable for identifying aliases. We can quantify the degree of reliability with the *positive predictive value* (PPV) metric, which measures the fraction of positive test results—"shared counter" and thus "alias"—that are correct; that is, TP $(\mathrm{TP}+\mathrm{FP})$. Another important metric for evaluating IP ID-based techniques is *sensitivity*, the fraction of shared counters that produce a positive test result; that is, TP $(\mathrm{TP}+\mathrm{FN})$, where FN stands for false negatives. For the purposes of comparing the effectiveness of IP ID-based techniques, the false negatives we are interested in are failures to detect counter sharing, not failures to detect aliases. (See Appendix B for further discussion of these and related terminology.)

There are two main challenges for an alias resolution technique as the number of addresses N increases: 1) probing and testing the $O(N^2)$ candidate alias pairs, and 2) minimizing the false positives relative to true positives; that is, ensuring a high PPV. The Ally technique requires $O(N^2)$ probes to test all possible pairs of addresses. RadarGun improves on Ally's scalability by probing an entire set of addresses as a whole with $O(N)$ probes, but scalability is still limited by a need to obtain overlapping time series from all addresses. Neither Ally nor RadarGun has a sufficiently low false positive rate to handle the millions of addresses that appear in macroscopic-scale Internet topology graphs such as that collected by Ark [21]. Repeating the alias tests of Ally and RadarGun can lower the overall false positive rate and thus increase the PPV, but because these tests suffer from false negatives, repetition can also decrease the sensitivity, causing aliases to be missed.

MIDAR is an attempt to overcome these limitations to scaling to millions of IPv4 addresses. In a nutshell, MIDAR collects IP ID time series data from many different vantage points, then mines the data using our Monotonic Bounds Test (Sec. III-B) to discover which IP addresses are likely aliases to the same router. The key features of MIDAR are the *Monotonic Bounds Test* (MBT), an ID comparison test with near perfect sensitivity based on monotonicity rather than proximity, which allows MIDAR to achieve an extremely low false positive rate and thus a high PPV; the use of multiple probing methods to increase the responsiveness of targets and thus extend the coverage of IP ID-based techniques; and the use of multiple vantage points and a novel sliding-window scheduling algorithm to achieve probing scalability.

This paper is organized as follows. Sec. II reviews the features and limitations of the two best known IP-ID based approaches: Ally and RadarGun. Sec. III presents the essential concepts and components of MIDAR. Sec. IV describes the four stages of our full MIDAR system: Estimation, Discovery, Elimination, and Corroboration. Sec. V reports results from an Internet-scale MIDAR experiment. Sec. VI compares the results against ground truth and compares MIDAR's MBT to RadarGun. Sec. VII summarizes our contributions and plans to integrate MIDAR into a larger system for alias resolution.

II. Existing IP ID Techniques and Limitations

A. Ally

The Ally component of Rocketfuel was the first tool to examine IP ID values for alias resolution. Several papers describe the Ally alias resolution tool [4], [13], [17]. We base our description on Bender et al. [17] and on the Ally source code included in Scriptroute v0.4.8 [22] (earlier standalone releases of Ally are now deprecated). The user can direct Ally to probe with one of UDP (default), TCP ACK, or ICMP.

A key step of Ally is checking whether the IP ID values of two candidate addresses are *in order*; that is, the values form an increasing sequence consistent with the use of a shared counter. Because IP ID counters wrap from 65 535 to 0, Ally must use sequence space arithmetic, similar to that defined in RFC 1982 [23]. We will use the notation $X \quad Y$ to denote the *less than* relationship within sequence space.

Ally uses the following procedure to test whether addresses A and B are aliases. First, Ally sends a probe to A, waits 1 ms, then sends a probe to B. Suppose the IP ID values in the responses are A_1 and B_1, respectively. Ally first checks whether A_1 and B_1 are in order and close enough to each other; namely, that $A_1 \quad 10 \quad B_1 \quad A_1 + 200$. If so, then

Ally waits 400 ms, sends a probe to B, waits 1 ms, and sends a probe to A. Ally then checks that the resulting IP ID values B_2 and A_2 meet the condition $B_2 \quad 10 \quad A_2 \quad B_2 + 200$ and that $A_1 \quad A_2$ and $B_1 \quad B_2$. If all these conditions are met, then Ally declares A and B to be aliases; otherwise, they are declared to be "non-aliases," but like all IP ID-based techniques, Ally can only infer that they do not share a counter.

Because Ally cannot know the exact generation time of each collected IP ID value, it uses an error margin when comparing the values. The $A_1 \quad 10$ margin accommodates reordering of probes on the forward path, which would cause B_1 to be generated before A_1, with $B_1 \quad A_1$. The $A_1 + 200$ margin accommodates the advancement of the IP ID counter between the arrival of the probe packets at A and B. To keep this inter-arrival advancement low, Ally sends probe packets just 1 ms apart, but the packets typically undergo some dispersion due to cross traffic, routing differences (for example, if A and B reside in different prefixes), load-balanced paths with different lengths, and other causes. More dispersion allows greater potential counter advancement between IP ID samples.

Ally has the following limitations. First, it is unclear whether these empirically-derived margins of error ($X \quad 10$ and $X+200$) are universally applicable to typical packet dispersion amounts and counter velocities. Second, using fixed margins of error is a fragile balancing act between minimizing false positives and false negatives. The wider these margins are, the more they allow false positives from chance alignments of IP ID values. However, if these margins are too narrow, then they can lead to false negatives if counters advance at a high rate or in bursts, or if probe packets undergo a significant amount of dispersion. Third, Ally relies on only four IP ID samples to infer aliases, which makes Ally susceptible to false positives caused by chance alignments, independent of the margins of error. Fourth, Ally cannot perform IP ID-based alias resolution on a router that rate limits its responses, because Ally needs the responses to be generated closely in time.

Finally, a significant drawback of the Ally technique is that, given N addresses, it requires $O(N^2)$ probes to test all possible pairs. To make Ally more practical, some heuristics are needed to reduce the size of the search space, such as requiring a pair of addresses to have similar return TTL values from a set of vantage points, as was done in Rocketfuel. Although this heuristic significantly reduces the amount of pairwise testing needed at moderate scales, its effectiveness with millions of addresses has never been demonstrated.

Even if it were possible to apply Ally to one million addresses, Ally's false positive rate ( ) would be too high to produce a useful positive predictive value. The margins of error in Ally's test allow samples to be 210 ID values apart, or $210 \quad 65536 = 0\,32\%$ of the ID space. The two halves of the test are closely correlated because they are taken only 400 ms apart and velocities are typically low (Fig. 7), suggesting is only slightly lower than 0.0032. But even if the two halves of the test were completely independent, would be at best about 0.00001. Extrapolating these rates to one million addresses suggests there would be at least 5 million false positives, and probably closer to 1.6 billion, which is orders of magnitude more than the expected 1 to 10 million true positives.

B. RadarGun

We base our discussion of RadarGun on the v0.3 source code [24] and the RadarGun paper [17]. The RadarGun implementation can probe with TCP ACK (default), UDP, or both (without special handling of cross-protocol comparisons).

RadarGun improves on Ally's scalability by probing an entire set of addresses in parallel, $O(N)$ probes, rather than a series of pairs, $O(N^2)$ probes. RadarGun makes 30 probing passes through the address list to obtain 30 IP ID samples from each address, with samples from different addresses interleaved with each other; for example, given addresses $A \ B \ C$, RadarGun takes the samples $A_1 \ B_1 \ C_1 \ A_2 \ B_2 \ C_2 \quad$. This probing scheme produces an IP ID time series for each address. An IP ID time series A (for address A) consists of a sequence of samples A_i , where each A_i specifies the sample time and the IP ID, $(t_i \ ID_i)$. RadarGun uses the measured receive time of a response packet as an approximation of the sample time, since it does not know exactly when a router generated a given IP ID sample. RadarGun discards a time series as unusable if 1) fewer than 25% of the 30 probes elicited responses (that is, RadarGun has fewer than 7 IP ID samples), 2) all collected samples have an IP ID of zero or all have the IP ID used by probes, or 3) the time series is *nonlinear*—that is, either the IP ID counter is advancing too quickly to measure, or IP ID values are randomly generated.

RadarGun classifies a time series as nonlinear in two cases. First, a time series with too many *negative deltas* is nonlinear. A *delta* is the difference of adjacent IP ID samples; that is, $ID_i = ID_{i+1} \quad ID_i$. Negative deltas occur naturally as an IP ID counter wraps from 65 535 to 0. For a given sampling interval, the faster an IP ID counter advances, the more frequently a negative delta will appear in a time series, since the counter can wrap more often within the sampled time period. Negative deltas can also occur when IP ID values are generated randomly, since the average probability of an *individual* delta being negative is 50% in a sequence of random values. RadarGun discards a time series as nonlinear if greater than 30% of the deltas are negative, since it cannot know if the cause is too fast a counter or random IP ID values.

Second, a time series with too high a velocity—that is, the apparent rate of advancement of an IP ID counter—is nonlinear. RadarGun computes the velocity from an *unwrapped* IP ID time series. A time series is typically unwrapped by adding 65 536 (the full IP ID space) whenever a negative delta occurs. RadarGun also tries to account for counter wraps that may have occurred in large gaps in time between samples, even when the delta is positive. RadarGun infers the number of possible wraps in each gap from an estimate of the time between wraps, t_{wrap}, derived from a simplistic calculation on the positive deltas in a time series. For a gap of duration t_{gap}, there are $t_{\text{gap}} \quad t_{\text{wrap}}$ inferred wraps, and RadarGun adds this many multiples of the IP ID space when unwrapping the time series. Once a time series has been unwrapped, RadarGun computes the velocity as the slope of the linear least squares line that best fits the unwrapped IP ID values. The more negative deltas there are in a time series, the higher the apparent velocity of the unwrapped samples. Therefore,

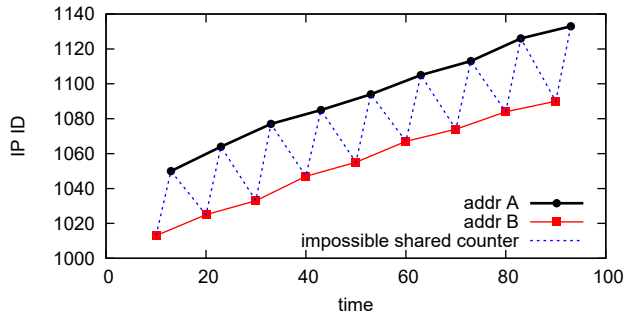

Fig. 1. There can always be a false positive under the RadarGun distance test regardless of the threshold used. For example, the average IP ID distance between these time series is only 40, which is below the 200 threshold for being shared counters, but these addresses cannot share a counter because the merged sample points do not form a monotonic sequence over time. A similar construction exists for any threshold.

both true high velocity counters and random IP ID values will lead to high apparent velocities. RadarGun discards a time series as nonlinear if the velocity is greater than 800 ID/s, since it cannot know the cause.

The key insight of RadarGun is that if two addresses share an IP ID counter, then their time series should have nearby IP ID values when overlapping in time. RadarGun checks for this condition by employing a *distance test* to measure how close two time series are in IP ID space. The key building block of the distance test is the calculation of the *sample distance*, the distance between an individual sample point in one time series and the expected value of the IP ID counter in the other time series at the same moment in time. There are two cases to computing the sample distance, with all calculations performed on the unwrapped time series A and B of the two addresses being tested. Let $B_j = (t_{B\,j}\ \ ID_{B\,j})$ be a sample of B. In the first (and more common) case, B_j lies between adjacent samples of A in time, that is, there is some i for which $t_{A\,i}\ \ t_{B\,j} < t_{A\,i+1}$. RadarGun then uses linear interpolation between A_i and A_{i+1} to estimate $ID_{A\,\mathrm{est}}$, the expected IP ID value of A at $t_{B\,j}$. In the second case, B_j does not lie between any two samples of A, and RadarGun extrapolates the best fit line through A (the same line used to calculate the velocity) to estimate $ID_{A\,\mathrm{est}}$. In either case, RadarGun then computes the sample distance $_{B\,j} = ID_{A\,\mathrm{est}}\ \ ID_{B\,j}$. After computing all sample distances $_{B\,j}$ between B and A, and similarly computing the sample distances $_{A\,i}$ between A and B, RadarGun calculates the average sample distance:

$$_{A\,B} = \frac{_{i}\ _{A\,i} + \ _{j}\ _{B\,j}}{A\ +\ B}$$

If $_{A\,B} < 200$, then RadarGun concludes A and B are aliases; if $_{A\,B} > 1000$, then they are not aliases. Otherwise, the distance test is inconclusive.

By employing the distance test on time series, RadarGun is more tolerant than Ally of ICMP rate limiting and less susceptible to false positives caused by chance alignments. However, RadarGun's distance test is still only a heuristic. There is no inherently right choice for the distance thresholds, since they must be low to minimize false positives with lower velocity addresses, and high to minimize false negatives with higher velocity addresses. Even if the thresholds were chosen adaptively to velocities, the thresholds must still have margins of error to allow for bursty IP ID counter advancement and other uncertainties, which prevents exact separation of shared counters from unshared. As a result, adjusting the distance thresholds never fully eliminates false positives, false negatives, or inconclusive results, but merely shifts the balance between them. Regardless of the threshold used, there can always be a false positive under the distance test that could be avoided by checking for monotonicity, as illustrated in Fig. 1.

As a consequence of the above weaknesses, the distance test produces too many false positives for RadarGun to scale to millions of addresses. For example, extrapolating the false positive rate (0.0005) implied in Bender et al. [17] to one million target addresses suggests there will be an order of magnitude more false positives (264 million pairs) than true positives, giving a very poor PPV. An obvious solution to this problem would be to repeat the distance test at a later time, but because the test also suffers from false negatives, some shared counter pairs would inevitably be lost with each repetition.

Furthermore, because RadarGun needs overlapping time series from all addresses for the distance test, there is a practical limit to the number of addresses RadarGun can handle before requiring network-unfriendly levels of probing bandwidth. For example, probing one million targets with 10 s spacing would require 100 000 packets per second, or 35 Mb/s.

III. MIDAR Design

To find aliases among a large number N of router addresses, MIDAR collects an IP ID time series from each of the addresses and tests for a shared IP ID counter in each of the $O(N^2)$ address pairs. We take a bottom-up approach to describing MIDAR. In this section, we describe the essential concepts and key features of MIDAR, and discuss our approach to mitigating false positives. In the following section we will describe how we integrate these components into the complete MIDAR system.

A. Time series in MIDAR

MIDAR takes a sampling of IP ID values to construct the time series used for alias resolution. MIDAR considers a time series *unusable* for alias resolution if 1) fewer than 75% of the probes elicited responses from the target; 2) 25% or more of the collected samples are *degenerate*, that is, have a constant IP ID value (such as zero) or echo the IP ID used by probes;[1] or 3) the time series cannot be modeled as a monotonically increasing sequence (that is, the observed frequency of negative deltas is so high that the IP ID values may have been generated randomly or by a counter that is wrapping too quickly to measure). Our quality thresholds are higher than RadarGun's, but in practice few interfaces produce a medium-quality time series that would be considered usable by RadarGun but unusable by MIDAR. However, the higher

[1] We do observe real time series with more than 25% but fewer than 100% degenerate samples. These samples appear to be the result of some mechanism other than chance occurrences in a time series produced by a counter.

thresholds do matter in MIDAR's procedure for choosing the best probing method for each target, as described in Sec. III-D.

To reliably detect all counter wraps, we must use a sampling interval shorter than the wrapping period, so we obtain exactly one negative delta whenever the counter wraps and positive deltas at all other times. Sampling even more frequently will yield more positive deltas while the counter is increasing monotonically but still only one negative delta at each counter wrap, decreasing the overall fraction of deltas that are negative. We adopt RadarGun's 30% threshold on the maximum allowed fraction of negative deltas before we consider a time series unusable. This 30% threshold is intentionally more conservative than the 50% threshold suggested by the Nyquist-Shannon sampling theorem when counter wrapping is thought of as a periodic signal. The 30% limit on negative deltas also has the advantage of excluding 98.8% of random time series, which cannot be used for alias resolution (see Appendix D).

We define the *maximum acceptable sampling interval* $I_{\max}$ to be the largest sampling interval that still ensures the fraction of negative deltas is no more than 30%. MIDAR collects an initial time series from each target address using a small fixed sampling interval and then calculates $I_{\max}$ individually for each target based on the target's observed velocity. MIDAR uses the computed $I_{\max}$ to customize the sampling interval individually for each target when collecting additional time series actually used for alias resolution (Sec. III-E).

Observe that limiting the fraction of negative deltas to 30% is equivalent to limiting the average counter advancement per sample to 30% of the ID space, because the counter advances through 100% of the ID space between each counter wrap. Hence, the maximum acceptable sampling interval for a time series with velocity v is

$$I_{\max} = (0\ 3 \quad 2^{16})\ v \tag{1}$$

We define the velocity v of a time series to be the average slope of the segments weighted by segment duration; that is,

$$v = \frac{ID_i}{t_i} \tag{2}$$

where ID_i and t_i are the change in ID and time, respectively, between samples i and $i+1$. If a ID_i would be negative, then we *unwrap* it by adding 2^{16}. To avoid distortions due to sampling gaps or atypical counter behavior, we exclude discontinuities (Appendix E) when calculating velocity.

B. Monotonic Bounds Test

The Monotonic Bounds Test (MBT) checks whether the IP ID times series of two addresses meet the monotonicity requirement, a necessary condition for sharing an IP ID counter; that is, whether the two time series form a monotonically increasing IP ID sequence when considered as a single merged time series. The MBT is a rigorous test that does not employ ad hoc thresholds to accommodate uncertainties.

MBT checks that two time series A and B meet the monotonicity requirement by individually checking that each sample of B meets the monotonicity requirement with respect to the samples of A, and vice versa. If all sample tests pass, then A and B as a whole meet the monotonicity requirement. We first describe MBT in an idealized form and then describe the details needed to apply it to real data. Let $B_j = (t_{B\,j}\ \ ID_{B\,j})$ be a sample of B that we will check with respect to the samples of A. Let $A_i = (t_{A\,i}\ \ ID_{A\,i})$ and $A_{i+1} = (t_{A\,i+1}\ \ ID_{A\,i+1})$ be adjacent samples in A such that $t_{A\,i} < t_{B\,j} < t_{A\,i+1}$; that is, A_i and A_{i+1} are the nearest adjacent samples of A in time to B_j. Fig. 2 illustrates the two different MBT cases. In the first case, the counter has not wrapped between the samples A_i and A_{i+1} (that is, $ID_{A\,i} > 0$), and so we can simply check that $ID_{A\,i} < ID_{B\,j} < ID_{A\,i+1}$. We can visualize this constraint as requiring B_j to fall within the vertical bounds of the box with corners at A_i and A_{i+1}. For example, in Fig. 2, B_2 lies between A_1 and A_2 in time and falls within the bounding box of these samples, and thus B_2 meets the monotonicity requirement. In contrast, B_3 is between A_2 and A_3 in time but does not fall within the bounding box (because $ID_{B\,3} < ID_{A\,3}$) and thus violates the monotonicity requirement. In the second MBT case, the counter has wrapped between A_i and A_{i+1} (that is, $ID_{A\,i} < 0$). Therefore, the bounding box between A_i and A_{i+1} is split into two pieces, and we must have either $ID_{A\,i} < ID_{B\,j}$ or $ID_{B\,j} < ID_{A\,i+1}$. For example, B_5 lies between A_4 and A_5 and passes, since $ID_{A\,4} < ID_{B\,5}$. B_6 also lies between these samples but violates the monotonicity requirement by lying outside both pieces of the bounding box. If all samples of B pass, then MBT swaps the roles of A and B and repeats the procedure. If any sample-wise test fails, we can immediately conclude that that A and B do not share a counter without performing the remaining sample-wise tests.

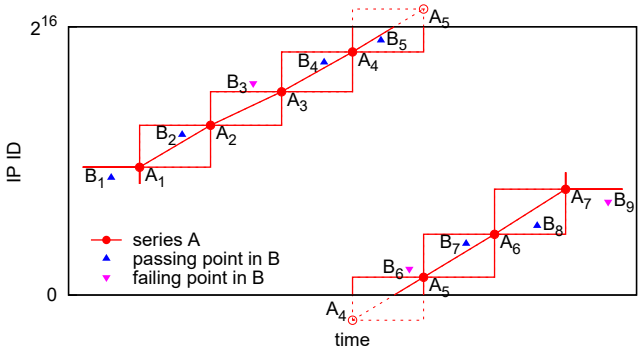

Fig. 2. Illustration of the sample-wise execution of the Monotonic Bounds Test (MBT). Each sample in one time series must lie within the monotonic bounds set by the closest surrounding samples (in time) of the other time series (e.g., B_2 must fall within the bounding box with corners at A_1 and A_2). When there is a counter wrap between the surrounding samples (e.g., between A_4 and A_5 when checking B_5), the monotonic bounds also wrap. (Velocity is exaggerated for clarity.)

So far, we have described a time series as being $(t_i\ \ ID_i)$ with t_i being the sample time. To maintain a virtually zero false negative rate (a crucial property relied on in MIDAR), MBT needs accurate sample times to determine which samples define the monotonic bounds for each sample-wise test. The *true sample time*, $_i$, is the exact moment in time that a router generated the value ID_i. We cannot determine $_i$ with active measurement, but we can calculate accurate bounds on $_i$. We

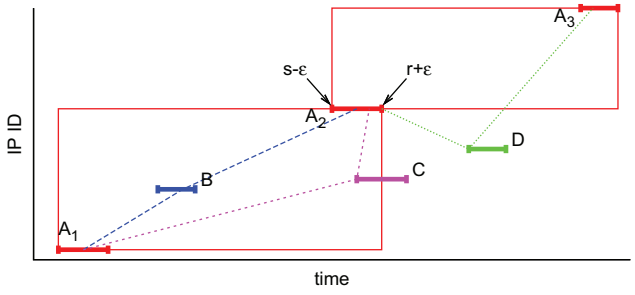

Fig. 3. Monotonic Bounds Test with imperfect time data. *Sample time ranges* are shown as horizontal bars (with exaggerated size for clarity). Samples B and C lie at least partially within the bounding boxes, making it possible to draw a monotonic curve (dotted lines) through them connecting neighboring samples from interface A, showing that samples B and C may come from a counter shared with A. Sample D lies completely outside the bounding boxes, so no monotonic curve can connect samples in A and still pass through D, so D cannot come from a counter shared with A.

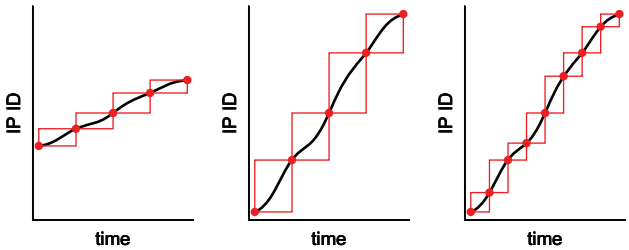

Fig. 4. With all other things being equal, monotonic bounds (that is, the range of IP ID values allowed by the MBT) become tighter when a time series has a lower velocity (subfigure (a) compared to (b)) or when IP ID values are sampled on shorter intervals ((c) vs. (b)).

know the measured times s_i when we sent our probe and r_i when we received the response, and that the true send and receive times are within $\pm\epsilon$ of the measured times, where ϵ is the maximum clock error of all monitors during a MIDAR run (see Sec. III-D). Since the response must have been generated between sending and receiving, we know that the true sample time must be within the *sample time range* $(s_i - \epsilon,\ r_i + \epsilon)$, which we will substitute for τ_i in MBT execution.

MIDAR obtains the samples of a single time series sequentially by sending a probe only after receiving the response to a prior probe or after a timeout, so there is never any uncertainty about the ordering of the samples within a single time series. However, since MIDAR probes multiple interfaces in parallel (Sec. III-E), two samples from separate time series can have overlapping time ranges, making the true ordering of these samples uncertain. When the time range of one of the bounding samples overlaps that of the test sample, MBT widens its bounds to the next closest sample whose time range does not overlap the test sample. This makes the monotonic bounds larger than they could have been if the true ordering of samples were known, but the sensitivity of the test is preserved despite these uncertainties—a test failure against the larger bounds conclusively means that there is no shared counter. We can thus accommodate uncertainties in both the response time and clock error without compromising the rigor of MBT.

Fig. 3 illustrates the execution of MBT using sample time ranges. We wish to individually test the samples B, C, and D against the surrounding samples of $\{A_i\}$. The time ranges of B and D (shown as horizontal bars) do not overlap with the time ranges of A_i, so we know the true ordering of these samples, and MBT execution is straightforward: sample B passes against A_1 and A_2, and sample D fails against A_2 and A_3. Sample C is the interesting new case, since the time ranges of C and its nearest surrounding sample A_2 overlap. Because of the overlap, we cannot know whether C precedes A_2 and therefore should be bounded by A_1 and A_2 (the left bounding box), or whether C follows A_2 and should be bounded by A_2 and A_3 (the right box). MIDAR simply avoids relying on the ambiguous A_2 and looks outward toward A_1 and A_3, the nearest samples that do not overlap with C, to find the suitable monotonic bounds. Sample C falls completely within this larger bounding box (that is, $ID_{A,1} < ID_C < ID_{A,3}$) and therefore passes. Equivalently, C falls at least partially within one of the two smaller bounding boxes. The dashed line passing through A_1, C, and A_2 illustrates a possible monotonic counter consistent with these sample values and time ranges.

Because MBT is based strictly on the definition of monotonicity, we must detect and account for discontinuities (Appendix E) and other anomalies (Appendix F) that occur occasionally in time series that otherwise adhere to the definition. Whenever a sample-wise MBT test involves an ID value that is questionable due to a discontinuity or anomaly, the test may generate a false negative. Rather than risk this rare error, we do not apply the MBT to that sample, and rely on the remaining sample-wise tests for the most accurate result.

In general, the more ID samples we can test with MBT, and the tighter the monotonic bounds, the more confident we can be that a positive test result means a shared counter. Monotonic bounds are typically small in practice for two reasons. First, monotonic bounds are defined separately by each pair of samples and are by construction as tight as possible. Thus, the lower velocity time series in Fig. 4a has tighter bounds than the higher velocity time series in Fig. 4b. Low velocity time series make up the majority of the cases observed in our data, and the monotonic bounds can be quite small; for example, adjacent samples with ID values 5 and 7 define monotonic bounds that can be satisfied by only a single ID value, 6. Second, we can use a shorter sampling interval to tighten the bounds independently of the target velocity, as illustrated by Fig. 4b and 4c, which have identical counters but different probe spacing. To the extent possible, MIDAR tries to keep monotonic bounds small by adapting probe spacing to the actual measured velocity of each target interface (Sec. III-E).

C. Multiple probing methods

Bender et al. [17] mentioned the possibility of combining TCP and UDP methods to increase the number of testable targets, but did not offer any procedure for doing so, nor investigate its effectiveness. The recent RadarGun implementation can probe each target with both TCP and UDP but does not handle cross-protocol comparisions in any special way. In

TABLE I
SUMMARY OF PROBING METHODS.

Method	Probe Packet	Expected Response
TCP	TCP ACK to port 80 on target	TCP RST or (rarely) ICMP *port unreachable* from target
UDP	UDP packet to port 33435 on target	ICMP *port unreachable* from target
ICMP	ICMP *echo request* to target	ICMP *echo reply* from target
Indirect	TTL-limited ICMP *echo request* to a host *past* the target	ICMP *time exceeded* from target

this section, we describe the procedure used by MIDAR to fully exploit four probing methods—TCP, UDP, ICMP, and a method we call *TTL-limited indirect probing*, or *Indirect* for short.

Table I summarizes the methods supported by MIDAR. The TCP, UDP, and ICMP methods are straightforward: send a probe packet to the target, and if the response is of the expected type, collect the IP ID value. Although UDP responses from a different address are often from a different interface on the same router, there is a risk that such responses are from a different router altogether, so we do not use them in MIDAR; interpreting these responses is more in the domain of the Mercator technique. The *Indirect* method imitates a traceroute measurement. Every intermediate address in a traceroute path responded with an ICMP *time exceeded* response, so in theory, we can elicit a *time exceeded* response again by reproducing the conditions of a traceroute measurement. For an address observed at hop h in a traceroute path, the *Indirect* method sends a probe with TTL $= h$ from the original vantage point to the original destination and obtains an IP ID sample from the *time exceeded* response from the target. To maximize the chances of the probe taking the same route as the original traceroute packet and expiring at the target address, we maintain the same Paris-traceroute flow label [25] as the original traceroute measurement. Nevertheless, the route can still change, and we may face a new route that either entirely bypasses the target address, or passes through the target address at a different hop. MIDAR does not currently handle the first case—this is the greatest weakness of *Indirect* probing. MIDAR handles the second case by hunting for the target at nearby TTLs. If an *Indirect* probe with TTL $= h$ does not elicit the expected response, but one of two additional probes with TTL $= h \quad 1$ does, we use that new TTL as the expected TTL for subsequent probes. MIDAR performs this *TTL expansion* process only in the Estimation stage (see Sec. IV-A). In our experiments, TTL expansion increased the fraction of sufficiently-responsive targets of *Indirect* probing from 76.5% to 80.8%. Expanding further to $h \quad 2$ provided only a negligible increase in the response rate while significantly increasing the probing cost.

Appendix H describes the extent to which employing multiple probing methods increases usable (i.e., sufficiently responsive, nondegenerate, and monotonic) time series for our dataset described in Sec. V. Using TCP alone resulted in only 34.6% of the addresses having usable time series, leaving nearly two-thirds completely untestable with IP ID based alias resolution. If we employ all four methods, 80.6% of addresses yield usable time series to at least one method.

The main concern with employing multiple probing methods is how consistently the interfaces on the same router behave. In the simplest case, either all or none of the interfaces of a single router respond with usable IP ID values to a given method. In this case, we can collect all samples with the same method, presenting no complications for alias inference. However, interfaces on a single router do not always behave consistently, perhaps due to different filtering on different routes to the various interfaces. In such cases, we can infer aliases only if a router uses the same IP ID counter to generate responses across probing methods as well as across interfaces. We expect a router to use a shared counter on all interfaces when responding to TCP and UDP probes, since we expect the responses to come from a shared CPU that executes (router-wide) services potentially reachable with these protocols. However, when we use ICMP or *Indirect* probing, the ICMP *echo reply* or *time exceeded* responses could be generated entirely on a line card (that is, on the fast path) [26], and a line card may have its own IP ID counter not shared with either the CPU or other line cards on the same router. Thus, there is a chance that responses to ICMP and *Indirect* probes may not share a counter with responses to TCP or UDP probes.

We can detect counter sharing across probing methods in the same way we identify shared counters across interfaces—we apply MBT to a pair of time series obtained from the same interface but with different probing methods. Note that these $O(N)$ cross-method comparisons do not suffer from the high false discovery rate of the $O(N^2)$ cross-interface comparisons described in Appendix C. We observe a relatively high incidence of counter sharing for our dataset (see Sec. V), ranging from 88.9% to 97.4% of addresses per pair of methods (see Appendix I for details). Because counter sharing across methods is common, we can usually detect a shared counter across addresses even if the addresses are probed with different methods. Thus, there would be little benefit to probing every address with every method when collecting data for MBT; one method per target is sufficient, and much more efficient.

To determine which methods are usable with each target, MIDAR probes all targets with all methods in the initial Estimation stage (see Sec. IV-A). This process is inherently scalable since time series do not need to overlap across targets. When multiple methods are usable for a given target, MIDAR selects one to use in subsequent stages based on the following preferences. We prefer TCP over UDP because routers are more likely to rate-limit their responses to UDP. We know this from our own observations, and the fact that modern Cisco routers by default generate at most one *unreachable* response every 500 ms [27]. If the UDP and ICMP methods do not appear to share a counter, we prefer UDP, because responses to UDP are more likely to be generated in the CPU using an ID counter that is shared across interfaces. But if UDP and ICMP do share a counter, the choice of method does not affect the chances of cross-interface counter sharing, and we prefer ICMP because its responses are less likely to be rate limited. We prefer *Indirect* the least because 1) its responses are less likely to be generated in the CPU with a shared counter, 2) its responses are more likely to be rate limited, 3) a routing change may prevent us from probing a target, and 4) if a vantage point is lost mid-run, we may not have the traceroute

information needed to probe from a different vantage point.

Use of multiple probing methods is the reason MIDAR uses higher quality thresholds for time series than RadarGun, as described in Sec. III-A. Accepting a medium-quality time series from a preferred method might mean MIDAR would ignore a high-quality time series for the same address from a less-preferred method.

D. Multiple vantage points

MIDAR employs multiple vantage points to increase the aggregate probing rate, an obvious approach to scalability suggested but not implemented in [17]. Because MIDAR needs to compare time series collected by the different vantage points, their clocks must be synchronized, for example with NTP or RADclock [28], [29]. MIDAR does not require extraordinarily precise clock synchronization, but it does require an estimate of the maximum clock error across all vantage points during execution.[2] The lower the , the tighter the monotonic bounds become in the Monotonic Bounds Test (Sec. III-B), so we recommend minimizing where possible by, for example, deploying RADclock instead of NTP.

The higher probing rate achievable by multiple vantage points is not enough by itself for true scalability as the number of targets increases to Internet-scale. We discuss another technique MIDAR employs for scalability in the next section.

E. Achieving probing scalability with sliding window

The simplest way to collect overlapping time series for a list of target addresses is to iterate over the list multiple times, probing the targets in order, like RadarGun. If we probe N addresses at p packets per second (pps), then each target is sampled every $I = N\ p$ seconds. The resulting time series for each target is usable only if the sampling interval I is less than or equal to the maximum acceptable sampling interval $I_{\max}$ for that target (see Sec. III-A). I must be short enough to accommodate the highest velocity of the N targets. Suppose the highest velocity is 2000 ID/s. Then, from (1), we must have I 9 83 s. If $N = 2$ 10^6, then to achieve $I = 9\ 83$ s, we must probe at 203 459 pps, which is at least 71.6 Mb/s of traffic with TCP probes. The brute force approach of probing from 1000 hosts in parallel would reduce the probing rate to 203 pps per host, but managing that many hosts is cumbersome. Here, we present a more scalable technique that can achieve even smaller intervals for high velocity targets, at half the per-host probing rate, using fewer than 40 hosts.

MIDAR achieves probing scalability with a *sliding window* scheduling algorithm that exploits two observations. The first observation is that if addresses have very dissimilar velocities, they cannot share a counter, so we do not need to apply the MBT to them and thus do not need their time series to overlap. That is, we can use velocity similarity as a high sensitivity (but low PPV) shared counter test, to filter out many unshared counter pairs at an early stage. The second observation is that target velocities vary widely from near zero to several thousand ID/s, but the vast majority of the targets we have observed have low velocities (see Appendix A), so we need a short sampling interval only for the minority of high velocity targets.

[2]To estimate , we used `ntpq`/`ntpdate` to determine the clock offset and delay of each vantage point during a MIDAR run.

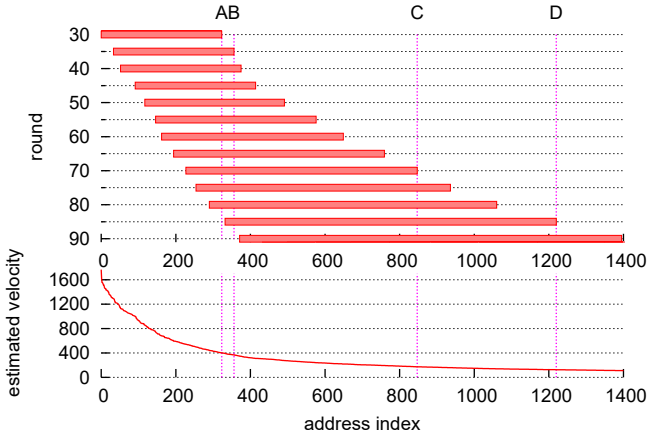

Fig. 5. Portion of sliding window schedule for a real large-scale run. The full schedule covers 272 rounds and 37 546 addresses per monitor, but for brevity, we show only every fifth round from 30 to 90 and only the first 1400 addresses. Addresses A and B have similar velocities, so are near each other in the velocity-sorted list, and thus share the window in rounds 35–83. Addresses A and C have less similar velocities, and share a window only in rounds 70–83. Addresses A and D have even less similar velocities, and so D does not enter the window until round 85, after A has exited.

MIDAR incrementally probes the target list over multiple *rounds*. In each round, MIDAR sends one probe to each target in a *window* in sequence. A *window* is a contiguous subset of the target list defined by starting and ending target indexes. The width and position of the window changes over time. The width of the window determines the time it takes to probe the window, and thus the sampling interval for targets within the window. The position or coverage of the window determines which targets will have overlapping time series. MIDAR ensures that the window covers likely shared-counter candidates by sorting the target list by descending velocity, which puts addresses with similar velocities near each other (MIDAR obtains the velocities in the Estimation stage, see Sec. IV-A). We can think of the simplest approach described at the beginning of this section as a degenerate case with a fixed window covering the entire target list, so that we collect overlapping time series for all targets.

Fig. 5 illustrates the execution of the sliding window. In the upper subfigure, each dashed horizontal line represents the target list at a particular round of execution (so each vertical line represents the same target address over all rounds), and each solid bar represents the window. For brevity, the figure only shows every fifth round. The lower subfigure shows the target velocities in ID/s, with the target indexes matching up vertically between the two subfigures. For discussion, we have labeled four target addresses (A, B, C, and D) and highlighted their target indexes with vertical lines.

Observe first that the width of the window increases over time, from around 300 targets at round 30 to 1000 targets at round 90. The window must be narrow near the beginning to ensure a sampling interval short enough for the highest velocity in the window. Because velocities vary widely in the beginning (from 1600 to 400 ID/s in the first 300 targets), the

narrow window includes all targets that could plausibly share a counter with the highest-velocity target, while excluding many targets that could not. The window is several times wider at round 90 than at round 30 because the target velocities at indexes 400–1400 are much lower with less variation, so the sampling interval can be longer, and more of the adjacent targets are shared-counter candidates based on their velocities.

Next, observe that the window gradually moves down the list over successive rounds. The more rounds two addresses co-occupy the window, the more overlap there will be between their time series. We wish to ensure sufficient overlap between targets with similar velocities, but avoid wasting resources on obtaining overlap between targets with sufficiently dissimilar velocities. Targets with nearly the same velocities, such as A and B in Fig. 5, are near each other in the target list and thus will co-occupy the window the longest and have the most overlap. Targets A and B have overlapping samples from round 35, when B first falls within the window, until round 83, when A last appears in the window. Targets with only somewhat similar velocities, such as A and C, are farther apart in the target list and thus will co-occupy the window for only a limited number of rounds (rounds 70–83 for A and C), but collecting even a few overlapping samples is still useful for ruling out these unlikely shared-counter pairs with MBT. Finally, targets with sufficiently dissimilar velocities, such as A and D, never co-occupy a window and have no overlap, but we presume they cannot share a counter, so lacking overlap is a feature that improves efficiency.

The sliding window must balance two competing requirements in each round—it must be narrow enough to ensure a sufficiently short sampling interval for the highest target velocity in the window, and it must be wide enough to include all nearby targets with velocities similar enough to share a counter. We can quantify this trade-off with metrics that depend only on target velocities and use the metrics to guide the choice of the optimal window size. Let v_{high} be the highest target velocity in a window, and v_{low} the lowest. These are the velocities of the first and last targets in the window, because the target list is sorted by descending velocity. We define a *spacing* metric for the quality of a window's sampling interval in terms of how much a counter with velocity v_{high} would advance between samples; specifically, we define a counter advancement of 16 384, or 1/4 of the ID space, to be one unit of *spacing*. If the counter advances 1/8 of the ID space, then the spacing will be 0.5. Lower spacing means more samples between counter wraps. We define a *similarity* metric for a window's inclusiveness of similar target velocities in terms of the ratio $v_{\text{low}}\ v_{\text{high}}$, with a value of 2/3 being one unit of *similarity*. A ratio of 1/3 would be a similarity of 0.5. The lower the similarity, the wider the range of velocities allowed as possible shared counter pairs. As a window becomes larger, the spacing metric increases (gets worse) and the similarity metric decreases (gets better). For each round, we choose the window's starting target index, and then choose the window size at which these two metric values are equal (or cross). If possible, we first advance the starting target index of the window past any targets in the beginning portion of the window that have already been probed at least 30 times while sharing a window with all targets of similar enough velocities to potentially share counters. In this way, the window eventually slides down the entirety of the target list over multiple rounds.

We chose the coefficients and balance point for the two metrics based on our experiments with the sliding window schedule. These choices reflect an informed judgment on the relative importance of various measurement parameters, and it may be worthwhile to further study the tradeoffs involved, but the exact chosen values are not critical. Because the window follows a smooth transition, a small change in these parameters or in velocity estimates would cause only a small change in the amount of overlap between targets. If we had taken a simpler approach of grouping the targets by velocity and probing each group separately, a small change in parameters or velocity estimates could change group assignments and cause some pairs of targets to go from full overlap to no overlap.

MIDAR partitions the full target list across multiple vantage points and simultaneously probes with a sliding window from all locations. To ensure that all targets with similar velocities have overlapping time series even across vantage points, MIDAR assigns, to the extent possible, both an equal number of targets and an equal distribution of target velocities to each vantage point, so that the windows of different vantage points cover the same range of velocities at the same time. Targets that can only be probed with the *Indirect* method can only be assigned to vantage points that saw that target in a traceroute path, but targets usable with other methods can be assigned to any vantage point, giving us the flexibility needed to achieve nearly identical velocity distributions.

The same sliding window schedule drives probing on each vantage point. We can pre-calculate the schedule because the windows depend only on target velocities, which are known in advance. The schedule includes a delay in each round for any vantage point that was assigned less than its share of targets for that round, allowing us to finely synchronize the probing of a given velocity range across all vantage points.

The sliding window scales gracefully without manual parameter adjustment to varying numbers of targets and vantage points and varying levels of overlap quality between time series. Using this approach, with 40 vantage points and a self-imposed limit of 100 pps per vantage point to minimize impact on the network, we were able to collect the required overlapping time series for 1.9 million addresses in 5.9 hours, with a worst case sampling interval of 15% of the wrap period. This aggregate probing rate of 4 000 pps is significantly lower than the 459 496 pps that would be needed by the brute force approach to achieve the same sampling interval.

Like TTL clustering in Rocketfuel, MIDAR's sliding window eliminates many pairs from consideration before fully probing them. In the context of MIDAR, we chose to use a sliding window based on IP ID, the same parameter already used by MBT, rather than introduce potential additional complications of a new parameter, TTL.

F. Further reducing false positives

For the millions of addresses typically discovered by Internet-scale mapping experiments, some of the trillions of

possible pairs of addresses will have similar IP ID time series over a given measurement period out of sheer coincidence (see Appendix C). Thus, all IP ID based alias tests will be susceptible to false positives at this scale. Fortunately, unrelated IP ID counters that were coincidentally similar during one period of time will eventually diverge under continued observation. We exploit this fact to substantially improve confidence in positive test results and to rule out false positives. We repeatedly test pairs that pass MBT, delaying hours or days between tests. The more times a pair passes MBT, the higher our confidence of a shared counter; but a single MBT failure conclusively rules out a shared counter. Because of the virtually zero false negative rate of MBT, assuming the clock error is not underestimated and counter anomalies are detected (see Appendix F), we can repeatedly apply MBT with negligible risk of losing aliases.

When starting with a large number N of addresses, our probing schedule must be tuned to handle the $O(N^2)$ possible pairs, at some expense to accuracy. But as we repeat the MBT, the number of alias candidates gets smaller, allowing us to tune our probing schedule to give more accurate MBT results. Exactly how we do this is described in the next section.

IV. MIDAR Implementation

A complete execution of MIDAR is divided into four stages. In the *Estimation* stage, we determine the velocity and best probe method for each address for use in subsequent stages. In the *Discovery* stage, we probe all target addresses with a sliding window schedule that allows us to efficiently probe and apply MBT to a large number of pairs to discover pairs that potentially share an IP ID counter. In the *Elimination* stage, we re-probe and repeat MBT on these potential alias pairs to rule out most false positives. Finally, in the *Corroboration* stage, we probe and apply MBT to each candidate alias set as a whole to confirm them and to rule out remaining false positives. After completion of all probing stages, we infer reliable alias sets.

A. Estimation stage

In the *Estimation* stage, we ascertain two fundamental properties of each target. We first identify the preferred probing method for each target, as discussed in Sec. III-C. All subsequent MIDAR stages probe each target with only the target's preferred method. We next estimate the velocity of each target by applying (2) to the time series collected by a target's preferred method. The subsequent Discovery stage uses these estimated target velocities to calculate its $I_{\max}$ according to (1) and create the sliding window schedule.

We partition the target list across vantage points and probe every target with every probing method. Because we care only about the properties of individual targets, we do not need to collect overlapping time series across targets, so the probing procedure is inherently scalable to any number of targets. To avoid potential bias in selecting a probing method, we randomize the probing order of the methods for each target. We probe each target 30 times, with an average interval of about 7.8 s between probes of a given method to the same target. This interval is short enough to reliably sample targets with velocities up to 2 520 ID/s, according to (1).

B. Discovery stage

The *Discovery* stage is our first pass at identifying address pairs that appear to share a counter. We start by generating a sliding window probing schedule using the velocities found in the Estimation stage, and, following this schedule, probe each target with its best probing method. We then analyze the results of these Discovery probes, applying our shared counter tests to every pair of targets with overlapping time series. Our most important test for shared counters is the Monotonic Bounds Test (Sec. III-B). But before applying the MBT, we can sometimes rule out a shared counter with two simpler checks on IP ID byte order and precision (see Appendix F). We explicitly do not use tests based on hop distance between monitor and target, or on the inferred initial TTL set by the target in the response, because these tests provide little additional benefit and risk causing false negatives.

C. Elimination stage

In the *Elimination* stage, we repeat the MBT on every apparent shared counter pair found in Discovery in order to eliminate the bulk of the false positives. Because we now have a more manageable set of candidate pairs, we no longer need a sliding window probing schedule. To achieve minimal probe spacing, and thus minimize the ID bounds in the MBT, we could probe each shared-counter pair separately. The main drawbacks of this approach are the high cost of probing a large number of candidate pairs (6.8 million pairs in our experiment, see Sec. V) and the undesirability of repeatedly probing addresses that are involved in many candidate pairs.

We can achieve far greater probing efficiency by exploiting the graph structure of shared-counter sets, with addresses as nodes and candidate shared-counter relationships as edges. Large shared-counter sets generated by Discovery tend to be very sparse graphs with many smaller cliques or near-cliques of real aliases linked together by relatively few false edges created by chance alignments. In Elimination, we decompose each large shared-counter set into overlapping smaller subgraphs, ensuring each edge occurs in at least one subgraph. We try to extract subgraphs that are as close to a clique as possible, since we can efficiently collect overlapping time series between all pairs in a clique with the minimal number of probes, but if the resulting subgraph would cause ID to exceed 5% of the ID space, we choose a smaller subgraph to guarantee tight probe spacing for more effective elimination of false positives. To reduce repeated probing of addresses, we try to minimize the number of subgraphs that include any given address. In the experiment described in Sec. V, this subgraph-based probing generated only 15% as many probes as would have been needed by pair-wise probing.

We probe each subgraph for 10 rounds, where a round consists of a single probe to each member of the subgraph consecutively, which guarantees maximum overlap between the time series of the addresses. We send probes to members of the same subgraph no faster than once every 600 ms and no slower than once every second. The purpose of the lower bound is to avoid the appearance of an attack and to avoid rate limiting at the target, since at least one popular brand of

router will by default rate-limit ICMP *unreachable* responses to one every 500 ms. For each subgraph $\mathcal{S}$, the round duration is typically a little over $\mathcal{S}$ 600 ms and at most $\mathcal{S}$ 1 s.

To reduce total run time, we probe multiple subgraphs in parallel. The artificial delay within each round allows us to interleave rounds of different subgraphs without significantly increasing the duration of each round. We can control our aggregate probing rate by adjusting the number of subgraphs we probe in parallel.

D. Corroboration stage

In the *Corroboration* stage, we take the transitive closure of all candidate shared-counter pairs that passed the Elimination stage to obtain candidate shared-counter sets. We then probe each of the sets as a whole and apply MBT to both the apparent shared-counter pairs we have already discovered and the pairs implied by transitive closure of those discovered pairs.

Probing in the Corroboration stage is the same as in the Elimination stage. The only difference is in the input—the sets are smaller, but we want coverage of every possible transitive closure pair in each set, not just the previously discovered pairs. Although most sets are small, some are still large enough or have high enough velocity that they need to be broken into subgraphs as in Elimination. Compared to Elimination, more subgraphs are required to cover an alias set of a given size in Corroboration because we must probe every pair in the transitive closure. Minimizing the size of these sets by eliminating as many false positives as possible in Elimination allows Corroboration to work with reasonable efficiency.

The Corroboration stage can also be used as a standalone tool to retest a previously collected set of aliases that have undergone months of potential address churn, or to test potential alias sets discovered by other means, such as with DNS name inference or other alias resolution techniques. Used this way, the Corroboration stage is more efficient and has better PPV and sensitivity than Ally or RadarGun.

E. Final alias inference

After all probing stages, we can finally infer reliable alias sets. First, we find all pairs that passed MBT in Discovery, were not ruled out by Elimination, and were reconfirmed by MBT in Corroboration. Each of these pairs has passed the MBT at least two times, so we have fairly high confidence that they actually share counters. The transitive closure of these pairs yields the alias sets corresponding to routers. For each new pair created through transitive closure, we perform the MBT and other alias tests using probe data already collected in the Corroboration stage. Because the Corroboration stage was specifically designed to obtain overlapping time series for every pair in every alias set, we will be able to perform at least one MBT on each of these previously untested transitive closure pairs, except when addresses are unresponsive. A *transitive closure conflict* occurs when addresses A and B appear to share a counter, B and C appear to share a counter, but A and C do not share a counter. Such a conflict cannot occur in an actual alias set, but can occur in experimental data due to a false positive or false negative, or a change in the network topology during data collection. We conservatively discard any alias sets with transitive closure conflicts; the sets that remain are MIDAR's final router alias sets.

TABLE II
CLASSIFICATION OF ADDRESS PAIRS IN OUR EXPERIMENT. BOLD LINES INDICATE PAIRS THAT WERE PASSED ON TO LATER STAGES.

stage/classification	address pairs	percent
Estimation		
usable	1 753 713 330 078	100.00
Discovery		
unusable	110 318 572 353	6.29
dissimilar Est. v (likely unshared)	802 244 369 262	45.75
failed MBT (definitely unshared)	841 143 560 073	47.96
passed MBT (possibly shared) $\mathcal{D}$	6 828 390	0.000389
...and belong to a large set $\mathcal{D}_L$	6 450 911	0.000368
...and belong to a small set $\mathcal{D}_S$	377 479	0.000022
Elimination		
in $\mathcal{D}_L$ and passed MBT $\mathcal{E}_L$	2 790 570	0.000159
survived Elim. $\mathcal{D}_S$ $\mathcal{E}_L$ $\mathcal{E}$	3 168 049	0.000181
Corroboration		
in $\mathcal{E}$ and passed MBT $\mathcal{C}$	2 783 801	0.000159
Final analysis		
in transitiveClosure($\mathcal{C}$) $\mathcal{C}_T$	2 935 558	0.000167
...in $\mathcal{C}_T$ and passed Cor.	2 930 698	0.000167
...in $\mathcal{C}_T$ and failed Cor.	2 500	0.000000
...in $\mathcal{C}_T$ and untested in Cor.	2 360	0.000000
$\mathcal{C}_T$ **minus pairs in conflicted sets**	2 800 727	0.000160

V. EXPERIMENTAL RESULTS

We now describe an Internet-scale experiment with MIDAR performed on CAIDA's Archipelago (Ark) [21] infrastructure on April 18–26, 2011. Results are summarized in Table II.

For input to MIDAR, we collected 2 323 682 addresses, primarily from intermediate (router) addresses in 189 million Paris-traceroute paths taken April 1–15, 2011 in the *IPv4 Routed /24 Topology Dataset* [30], which is an effort to systematically measure IP-level paths from Ark monitors to a dynamically generated list of IP addresses covering all /24 prefixes in routed IPv4 address space.[3] Of these addresses, MIDAR's Estimation stage found that 1 872 813 (80.6%) had usable time series (Sec. III-A). For more detailed classification of Estimation responses, see Appendix G.

In the Discovery stage, we probed these usable addresses from 40 Ark monitors with a sliding window schedule. Of the $\binom{N}{2} = 1\,75 \quad 10^{12}$ address pairs, 6 828 390 (0.0004%) appeared to use a shared counter. The small fraction is not surprising because the number of shared pairs should be $O(N)$ whereas the total number of pairs is $O(N^2)$. The 45.75% of pairs with dissimilar Estimation velocities were very unlikely to share a counter, and the sliding window was intentionally engineered to not waste resources collecting the overlapping time series needed to apply the MBT to such pairs.

Analyzing all possible pairs in the Discovery stage is by far the most computationally expensive task in a large-scale MIDAR run; using a server with eight hyperthreaded 3.0 GHz CPUs, analysis of the 1.75 trillion pairs took 20 hours. Transitive closure of the 6 828 390 apparent shared-counter pairs resulted in 75 350 apparent shared-counter sets containing a total of 1 033 759 addresses.

[3]We used 21 cycles of traces (7 per team) collected by all 54 active Ark monitors rather than just the 40 monitors used for MIDAR measurements.

Of the 75 350 apparent shared-counter sets, 72 644 sets were already small enough to be efficiently tested in the Corroboration stage, but 2 706 sets were large enough that we wanted to use an Elimination stage to break them up before Corroboration. The largest of these contained 618 877 addresses, but only 6.3 million of its 191 billion possible pairs were actually classified as shared. This very sparse graph is consistent with our expectation of many smaller cliques or near-cliques of true shared-counter sets being linked together by relatively few false shared-counter pairs. These large sets were successfully broken up by eliminating pairs that the Elimination stage classified as unshared or untestable, leaving 174 075 sets containing 704 506 addresses, with the largest set containing 658 addresses. Of the 2 790 570 pairs that passed the MBT in Elimination, 2 705 601 (97.0%) would be reconfirmed as being shared in Corroboration, suggesting that Elimination had already removed the majority of Discovery's false positives among those pairs.

At this point, there are 3 168 049 pairs that survived Elimination: the 377 479 pairs from small Discovery sets that were not subjected to Elimination and the 2 790 570 pairs from large Discovery sets that passed Elimination. In Corroboration, we probe those pairs to reconfirm them, and also probe the 1 202 111 additional pairs implied by transitive closure of those pairs so we will be able to perform full-mesh conflict testing. Of the pairs that survived Elimination, 2 783 801 passed Corroboration, and have now passed MBT at least twice. (The pairs that failed mostly belonged to small Discovery sets and had been previously tested only once, and so were likely to still contain a significant number of false positives.) Transitive closure of these high-confidence pairs yielded 126 147 sets containing 427 199 addresses and 2 935 558 total pairs. The largest set was the same 658-address set found by the Elimination stage. Of these sets, only 23 contained transitive closure conflicts. Treating the conflicted sets as untrustworthy leaves us with 126 124 sets containing 426 152 addresses and 2 800 727 total alias pairs. Only 2325 (0.08%) of the pairs in 352 (0.28%) of the sets were untested by MBT, that is, inferred only via transitive closure. The high degree of internal consistency in the face of nearly complete full-mesh testing of every set is strong evidence that MIDAR's positive predictive value is extremely high (that is, it finds very few false positives).

VI. Validation

For validation, we used two sets of ground truth data: *R&E*, a collection of known topologies provided by research and educational networks (CAnet [31], CENIC [32], GÉANT [33], I-Light [34], Internet2 [35], and NLR [36]); and *Tier1*, a known topology provided by a Tier 1 ISP.

The most direct validation we can do is test whether MIDAR and a validation set agree on the classification of alias pairs. Table III shows the result of this comparison for the full-scale experiment described in Sec. V against the two validation sets. Note that disagreements may indicate not just errors in MIDAR, but also errors in the validation set or real changes in the network between collection of MIDAR data and validation

TABLE III
GROUND TRUTH VALIDATION OF FULL-SCALE MIDAR SHARED-COUNTER PAIRS. A PAIR IS "IN TARGET LIST" OR "USABLE" IF THAT CONDITION IS TRUE FOR BOTH OF ITS ADDRESSES. "UNDETERMINED" IS MOSTLY PAIRS THAT WERE NOT TESTED WITH MBT BECAUSE THEY HAD DISSIMILAR VELOCITIES. ACTUAL ALIASES CAN LEGITIMATELY BE CLASSIFIED AS "UNSHARED" OR "UNDETERMINED" IF THEY BELONG TO A ROUTER THAT DOES NOT SHARE A COUNTER ACROSS INTERFACES.

MIDAR	Tier1		R&E	
result	aliases	nonaliases	aliases	nonaliases
in target list	72 300	38 810 671	17 930	1 710 940
unusable in Estimation	33 909	19 867 021	9 869	900 441
shared (positive)	**26 522**	**0**	**5 856**	**0**
unshared (negative)	**372**	**8 758 079**	**8**	**335 297**
undetermined	11 497	10 185 571	2 197	475 202

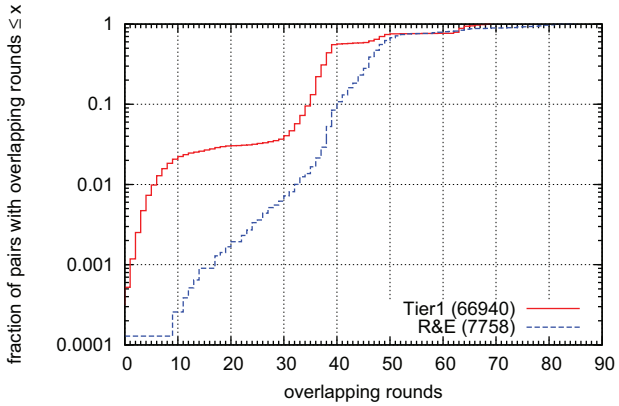

Fig. 6. Time series overlap between known shared-counter pairs during the sliding window for a newer MIDAR run on Oct 24-Nov 3, 2011. The sliding window failed to achieve 3 rounds of overlap for only 0.25% of known shared-counter pairs in the Tier1 dataset, and only 0.01% in the R&E dataset.

sets. For both sets, the number of false positives (shared nonaliases) is zero, showing that MIDAR has a high positive predictive value. There were few actual alias pairs that the MBT explicitly classified as unshared; closer analysis of their time series supports the conclusion that they are actually unshared, and not false negatives in MBT. The majority of aliases missed by MIDAR were due to routers that do not respond, do not use monotonic counters, or do not share a counter across interfaces, making them undetectable with any IP ID based technique.

To test the quality of the sliding window schedule, we examine how many rounds of overlap were achieved between the time series of pairs of addresses known to share counters.[4] We identify known shared counters by performing a standalone Corroboration run on known aliases. In Discovery, we require at least 5 sample points to pass the MBT, which can be obtained in 3 overlapping rounds, assuming no discontinuities or unresponsive probes. Fig. 6 shows that the sliding window achieved this 3 round minimum for 99.75% of known shared-counter pairs in the Tier1 dataset, and 99.99% in the R&E dataset. Thus, of all the shared counter pairs that could have been found by Discovery, only a tiny fraction were missed due to poor overlap in the sliding window.

[4]We did this overlap analysis on a new MIDAR run on Oct 24–Nov 3, 2011, which has similar overall results to the April 2011 MIDAR run described in the rest of this paper. We are unable to do this analysis on the April run.

TABLE IV
GROUND TRUTH VALIDATION OF RADARGUN AND MIDAR DISCOVERY ANALYSIS ON THE SAME PROBE DATA. A "TESTABLE ALIAS" IS AN ACTUAL ALIAS PAIR IN WHICH BOTH ADDRESSES WERE TESTABLE.

ground truth	tool and threshold	testable aliases	TP	FP	PPV
Tier1	RadarGun 1	62 817	1 500	0	1.0000
	RadarGun 10	62 817	48 146	5	0.9999
	RadarGun 100	62 817	61 773	7 358	0.8936
	RadarGun 200	62 817	62 392	23 110	0.7297
	RadarGun 300	62 817	62 392	56 885	0.5231
	MIDAR MBT	62 801	62 801	5	0.9999
R&E	RadarGun 1	2 307	228	2	0.9913
	RadarGun 10	2 307	1 010	4	0.9960
	RadarGun 100	2 307	1 426	110	0.9283
	RadarGun 200	2 307	1 608	225	0.8773
	RadarGun 300	2 307	1 820	252	0.8783
	MIDAR MBT	2 513	2 457	0	1.0000

TABLE V
LARGE-SCALE VALIDATION OF FALSE POSITIVES IN RADARGUN AND MIDAR DISCOVERY ANALYSIS ON THE SAME PROBE DATA TO 100 000 PRESUMABLY UNRELATED ADDRESSES.

tool and threshold	upper bound FP		lower bound FP	
RadarGun 1	326	0.000 000 6	266	0.000 000 5
RadarGun 10	25 235	0.000 048	21 994	0.000 041
RadarGun 100	631 995	0.001 9	559 189	0.001 1
RadarGun 200	1 470 877	0.002 8	1 298 696	0.002 4
RadarGun 300	2 397 778	0.004 5	2 114 970	0.004 0
MIDAR MBT	327	0.000 000 7	—	—

Although it is not feasible to run RadarGun at the scale of our MIDAR run in Sec. V, we did run it in April 2012 against our entire ground truth, which is larger and more reliable than the inference-based validation data used by Bender et al. [17]. We used TCP probes and an average probe spacing of 30.1 s and 13.1 s for the Tier1 and R&E datasets, respectively. We then analyzed the results with various settings of RadarGun's alias distance threshold. We also applied MIDAR's Discovery stage analysis to the data collected by RadarGun, allowing us to directly compare RadarGun's time series modeling and distance test to MIDAR's time series modeling and MBT. Note that we are not testing the full MIDAR system; in particular, we are not testing multiple probing methods, multiple stages, or adaptive probe spacing. Table IV summarizes the results.[5] Given identical probe data, there is no setting of RadarGun's distance threshold that allows it to achieve results as good as MIDAR's MBT in both TP and FP at the same time.

Because the available ground truth is too small to explore large-scale phenomena, we constructed a special set of target addresses for this purpose. From the 126 124 multi-address and 1.7 10^6 single-address shared-counter sets identified by the full MIDAR run of April 2011, we randomly picked one address in each of 100 000 random sets. Although MIDAR cannot always identify the full set of addresses belonging to every router, this construction minimizes the chance that some of our selected addresses legitimately share a counter.

In April 2012, we ran RadarGun on these addresses with TCP probes and an average probe spacing of 41.8 s. We also ran MIDAR Discovery stage analysis on the data collected by RadarGun. Of the 100 000 addresses, 32 604 were testable with RadarGun, and 29 678 with MBT. RadarGun with its default alias distance threshold of 200 found 1 470 877 positives out of 531 494 106 testable pairs, and MBT found 327 out of 440 377 003 (and a full MIDAR run with multiple stages and adaptive probe spacing would find even fewer). Assuming that all positives are false gives an upper bound on false positives. To obtain a lower bound, we must count only the positives that we can be sure are actual negatives. We rely on the fact that MBT tests a necessary condition for counter sharing, and is not specific to MIDAR. If two time series fail MBT, they cannot share a counter (assuming no undetected anomalies). Of the 1 470 887 probable false positives found by RadarGun, 1 298 696 failed the MBT and so are definitely false positives for RadarGun, giving a lower bound on of 0.0024. (We cannot use this approach to obtain a lower bound on MIDAR's false positive rate.) Table V shows the results for upper and lower bounds on FP and for various settings of the RadarGun distance threshold, showing that RadarGun is highly susceptible to false positives, regardless of the chosen threshold.

By plotting the number of false positives in the above tests against increasingly larger random subsets of the targets, we found that the number of false positives for both tools was, as expected, directly proportional to the total number of testable *pairs*, not addresses. Specifically, the number of false positives matched the predicted curve $N \quad (N \quad 1)$ 2, with the values of given in Table V.

VII. CONCLUSIONS

No one alias resolution technique is perfect. All have some amount of false positives, all have significant incompleteness, and most have scalability issues in the operational difficulty of working at large scale or in the way errors grow superlinearly as the scale increases. MIDAR extends recent work in IP ID-based alias resolution with new, highly scalable techniques that minimize false positives sufficiently to achieve a high positive predictive value at Internet scale (that is, millions of addresses). Although we believe false positives and scalability are now essentially solved problems for IP ID-based techniques, completeness will always be limited by routers that simply do not share IP ID counters across interfaces. Furthermore, rate-limiting may slightly reduce the completeness in MIDAR, and overcoming this limitation is an area for future study. Using multiple alias resolution techniques in parallel can improve completeness, but resolving disagreements between techniques is a challenge we hope to pursue in the future.

We are currently using a combination of MIDAR, iffinder, and kapar (working toward our larger Multi-Approach Alias Resolution System (MAARS)) to periodically capture router-level topology which we curate and share with researchers [37]. The MIDAR tool and all data of this paper will be released to the community by June 2012 [38].

[5]The MIDAR validation in Table III had fewer testable pairs than this validation because it included only addresses that also appeared in Ark traces.

REFERENCES

[1] V. Jacobson, "traceroute tool," ftp://ftp.ee.lbl.gov/traceroute.tar.gz.
[2] k. claffy, T. Monk, and D. McRobb, "Internet tomography," in *Nature*, Jan. 1999.

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

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

[5] Y. Shavitt and E. Shir, "DIMES: Let the Internet measure itself," in *ACM Computer Communications Review*, Oct. 2005.

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

[7] M. H. Gunes and K. Sarac, "Importance of IP alias resolution in sampling internet topologies," in *IEEE Global Internet 2007 (GI 2007)*, May 2007.

[8] B. Huffaker, A. Dhamdhere, M. Fomenkov, and k. claffy, "Toward Topology Dualism: Improving the Accuracy of AS Annotations for Routers," in *PAM*, Apr 2010.

[9] B. Huffaker, M. Fomenkov, and kc claffy, "Geocompare: a comparison of public and commercial geolocation databases," in *Network Mapping and Measurement Conference (NMMC)*, May 2011, http://www.caida.org/publications/papers/2011/geocompare-tr/.

[10] K. Keys, "IP alias resolution techniques," Tech. Rep., 2008, http://www.caida.org/publications/papers/2008/alias_resolution_techreport/.

[11] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *INFOCOM*, Mar. 2000.

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

[13] N. Spring, M. Dontcheva, M. Rodrig, and D. Wetherall, "How to resolve IP aliases," Tech. Rep., May 2004.

[14] M. H. Gunes and K. Sarac, "Analytical IP alias resolution," in *IEEE International Conference on Communications (ICC 2006)*, Jun. 2006.

[15] ——, "Resolving IP aliases in building traceroute-based internet maps," Tech. Rep., Dec. 2006.

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

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

[18] J. Sherry, E. Katz-Bassett, M. Pimenova, H. V. Madhyastha, A. Krishnamurthy, and T. Anderson, "Resolving IP aliases with prespecified timestamps," in *IMC*, 2010.

[19] P. Mérindol, B. Donnet, J. Pansiot, M. Luckie, and Y. Hyun, "MERLIN: MEasure the Router Level of the INternet," in *Conference on Next Generation Internet*, Jun 2011.

[20] J.-J. Pansiot, P. Mérindol, B. Donnet, and O. Bonaventure, "Extracting intra-domain topology from `mrinfo` probing," in *PAM*, April 2010.

[21] Y. Hyun, "Archipelago measurement infrastructure," http://www.caida.org/projects/ark/.

[22] "Scriptroute source code," http://www.scriptroute.org/source/scriptroute-0.4.8.tar.gz.

[23] R. Elz and R. Bush, "Serial number arithmetic," RFC 1982, Aug. 1996.

[24] "RadarGun source code," http://www.cs.umd.edu/~bender/radargun/radargun-0.3.tgz.

[25] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, and M. Latapy, "Avoiding traceroute anomalies with Paris traceroute," in *IMC*, Oct. 2006.

[26] J. Aweya, "IP router architectures: an overview," in *International Journal of Communication Systems*, 2001, pp. 447–475.

[27] "Cisco IOS release 12.0 network protocols command reference," http://www.cisco.com/en/US/docs/ios/12_0/np1/command/reference/1rip.html#wp1028904.

[28] D. Veitch, J. Ridoux, and S. B. Korada, "Robust Synchronization of Absolute and Difference Clocks over Networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 417–430, April 2009.

[29] J. Ridoux and D. Veitch, "Principles of Robust Timing Over the Internet," *ACM Queue, Communications of the ACM*, vol. 53, no. 5, pp. 54–61, May 2010.

[30] http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml.

[31] https://amidala.canet4.net/cgi-bin/reports.pl.

[32] D. Newcomb, CENIC, La Mirada, CA, private communication, May 2011.

[33] http://stats.geant2.net/lg/.

[34] http://routerproxy.grnoc.iu.edu/ilight/.

[35] http://routerproxy.grnoc.iu.edu/internet2/.

[36] http://routerproxy.grnoc.iu.edu/nlr/.

[37] http://www.caida.org/data/active/internet-topology-data-kit/.

[38] http://www.caida.org/tools/measurement/midar/.

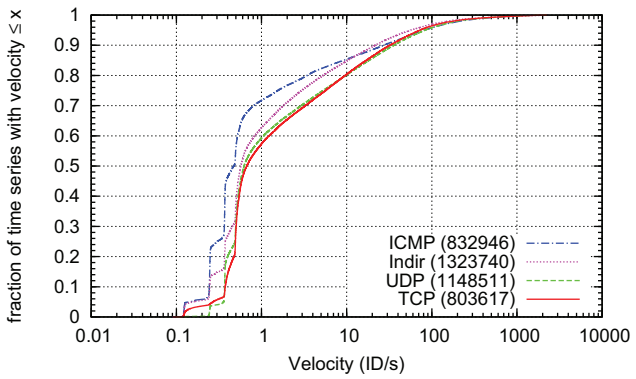

Fig. 7. Cumulative distribution of IP ID velocities for usable time series collected by the Estimation stage using four probing methods (the key includes the count of time series for each method). The distribution is heavily skewed toward low velocities and tapers off long before reaching the maximum discernible velocity of 2 520 ID/s for the given sampling interval.

TABLE VI
RELATIONSHIPS BETWEEN BINARY CLASSIFICATION TERMS.

		Actual value: Positive	Actual value: Negative	
Test result	Positive	TP	FP	→ PPV
	Negative	FN	TN	→ NPV
		↓ Sensitivity	↓ Specificity	↓ → Accuracy

APPENDIX A
VELOCITY DISTRIBUTION

Fig. 7 shows the distribution of velocities for usable time series (Sec. III-A) collected by the Estimation stage (Sec. IV-A). The figure shows separate distributions for each of the four supported MIDAR probing methods (Sec. III-C). The upper bound on the plot is approximately 2 520 ID/s, the maximum we could detect with our chosen sampling interval. The CDF tapers off long before reaching this upper bound, suggesting there are not many actual interfaces using monotonic IP ID counters with velocities higher than this bound; that is, any apparent velocities higher than this bound are likely due to randomly generated IP ID values.

APPENDIX B
BINARY CLASSIFICATION

To aid in discussion of alias resolution tests, it is useful to review some terminology commonly used in epidemiology and other fields. Some of these terms and their relationships are illustrated in table VI.

- *positive*: having the condition in question (e.g., a pair of addresses sharing an IP ID counter, or being aliases)
- *negative*: not having the condition in question
- *actual positives* (AP) and *actual negatives* (AN): number of cases that do or do not actually have the condition
- *prevalence*: fraction of cases that actually have the condition, $\mathrm{AP}/(\mathrm{AP}+\mathrm{AN})$
- *true positives* (TP): actual positives that test as positive
- *true negatives* (TN): actual negatives that test as negative
- *false positives* (FP): actual negatives that test as positive
- *false negatives* (FN): actual positives that test as negative

- *sensitivity* *true positive rate* (TPR) *recall*: fraction of actual positives that test as positive, TP AP
- *positive predictive value* (PPV) *precision*: fraction of positive tests that are correct, TP (TP + FP)
- *accuracy*: fraction of tests that are correct, (TP + TN) (TP + TN + FP + FN)
- *false positive rate* (FPR or): fraction of actual negatives that test positive, FP AN
- *false discovery rate* (FDR): fraction of positive tests that are incorrect, FP (TP + FP) = 1 PPV

Accuracy alone is not a good measure of the quality of a test. When prevalence is low, as in the case of large scale alias resolution, a test that mostly gives negative results will have a large number of true negatives and thus high accuracy, but might still have poor sensitivity and PPV.

APPENDIX C
FALSE POSITIVES

The false discovery rate is potentially very high when using IP ID time series for alias resolution at Internet scale.

According to the well-known "birthday problem," in a group of just 23 or more randomly chosen people, there is greater than 50% probability that at least one pair of people will have the same birthday. Similarly, given that the IP ID space has $2^{16} = 65\,536$ distinct values, it takes a group of just 302 IP addresses to have a 50% probability of some pair of addresses having the same IP ID value at any given time, and just 777 addresses for a 99% probability. When the number of targets N passes the number of possible values H, collisions are guaranteed by the pigeonhole principle. Even worse, if our IP ID test allows a range of nearby values instead of just equal values, the frequency of collisions increases with the size of the range. This would be the case if it were possible to probe all N targets instantaneously with Ally. In the context of the birthday problem, this requirement would be like requiring a pair of people in a group to have birthdays within 4 days of each other (which happens with 50% probability in a group of just 9 people).

Given N target addresses, and an average of d addresses per router, the number of shared-counter pairs (actual positives) is approximately the number of interface pairs per router times the number of routers:

$$\text{AP} \quad \frac{d}{2} \quad \frac{N}{d} = \frac{N(d \quad 1)}{2} \tag{3}$$

and the number of non-shared-counter pairs (actual negatives) is the total number of possible pairs minus the actual positives:

$$\text{AN} = \quad \frac{N}{2} \quad \text{AP} \quad \frac{N(N \quad d)}{2} \tag{4}$$

The prevalence is then $(d \quad 1) \ (N \quad 1)$. Some fraction of the tests on actual negatives will give false positive results when counters belonging to unrelated addresses are coincidentally synchronized to within the tolerance of the test. Then, the total number of false positives will be

$$\text{FP} = \quad \text{AN} \quad \frac{N(N \quad d)}{2} \tag{5}$$

For alias resolution results to have a useful *positive predictive value*, there must be significantly fewer false positives than actual positives. Comparing (5) and (3), and solving for , gives us an upper bound on useful values of :

$$\frac{d \quad 1}{N \quad d} \tag{6}$$

Thus, when N d, the maximum acceptable false positive rate of the test is inversely proportional to the number of target addresses.

To decrease , RadarGun and MIDAR compare tens of sample points in time series, as opposed to Ally's two. However, the decrease is not as much as one might expect, for two reasons. First, the samples in a single series are not independent, but are related by an underlying counter that increments with a somewhat regular rate. From this perspective, we can view the test as requiring that two counters have similar *initial* ID values and similar *velocity* (rate of ID change). Second, because the velocity distribution of real ID time series is heavily skewed towards low velocities as seen in Fig. 7, many pairs of counters will have a low velocity difference. Two unrelated counters with a similar initial ID value and a low velocity difference will take a long time to diverge.

Furthermore, note that the alias relationship is transitive. That is, if addresses A and B are aliases, and B and C are aliases, we must infer that A and C are also aliases; all three addresses belong to the same router. Even a small set of false positives, interpreted at face value, could lead us to incorrectly merge many distinct routers into one. The topology distortion caused by false positives is thus amplified by transitive closure.

APPENDIX D
NEGATIVE DELTA RATE OF RANDOM TIME SERIES

A *random time series* is produced from random IP ID values rather than from a monotonic counter. In random time series, the average probability of an *individual* delta being negative is 50%, regardless of the sampling rate. Therefore, the expected number of negative deltas appearing in a random time series of n values is given by the binomial distribution for $n \quad 1$ trials and $p = 0\,5$. This distribution is a bell-shaped curve with mode at $(n \quad 1) \ 2$.

For a time series of 30 samples (29 deltas), we would allow a maximum of $0\,3 \quad 29 \quad = 8$ negative deltas before classifying the time series as unusable based on our 30% threshold on negative deltas (Sec. III-A). The probability of getting 8 or fewer negative deltas out of 29 random deltas is just 0.012, so 98.8% of random time series will be correctly identified as unusable. Eliminating addresses with random time series at an early stage is more efficient than ruling out shared counters involving those addresses later with MBT.

APPENDIX E
DISCONTINUITIES IN TIME SERIES

An IP ID time series that appears mostly monotonic may have an occasional *discontinuity*, a local region of uncertainty where we cannot be confident that a counter remained monotonic between individual samples.

There are two types of discontinuity. First, there is a discontinuity if the time gap between samples is too large, or more precisely, if t_i is greater than 3.5 times the median t of the same time series. This means that we lost three or more consecutive samples (assuming a regular spacing of probes) due to rate limited responses or packet loss. Recall that our definition of a usable time series required at least three samples between counter wraps, so if we have lost three or more samples, the gap may hide one or more counter wraps.

The second type of discontinuity occurs when the counter advances too quickly between samples, which could be due to a burst of router traffic causing high velocity monotonic ID advancement, but could also be due to the router's counter being reset, causing a non-monotonic ID change. Let v be the median segment velocity for a given time series. If either the actual counter advancement ID_i or the expected counter advancement $v \quad t_i$ of a segment is greater than 30% of the ID space, we mark that segment as a discontinuity.

We take discontinuities into account in all our analyses, allowing us to use time series that would otherwise introduce errors or be unusable. For example, we exclude discontinuities when computing v in (2); that is, for a discontinuity between samples i and $i+1$, we exclude ID_i and t_i, thus improving the robustness of v to atypical or transient counter behavior. We observed a discontinuity in approximately 0.8% of the usable time series we collected.

APPENDIX F
ANOMALIES IN MONOTONIC COUNTERS

We observe several types of anomalies in IP ID values. MIDAR detects and accounts for these anomalies in order to maximize its sensitivity and positive predictive value.

Most routers transmit ID values in big-endian order (network byte order), but some use *little-endian* order. If ID values from a low-velocity counter are interpreted in the wrong byte order, then the counter will appear to have a velocity about 256 times greater than its true velocity. On the other hand, if a high-velocity counter is interpreted with the wrong byte order, it will be indistinguishable from random. We developed an inexpensive test to detect the correct byte order, and found that approximately 0.6% of usable time series were little endian. We can also use byte order as an additional criterion for ruling out aliases, assuming that every interface of a router would use the same byte order for a given probe method (Sec. III-C) (but we do not assume that every router uses the same byte order for different probe methods).

The second type of anomaly is caused by routers that do not use all 16 bits of the IP ID field. Such a *limited precision* counter will wrap around its smaller ID space more frequently than a full precision counter with the same velocity, as shown in Fig. 8. To identify a b-bit limited precision counter, we require not only that the $16 \quad b$ high bits are constant, but also that there is at least one wrap, and that every wrapped segment, after being unwrapped, has a velocity similar to that of the non-wrapped segments. Failing to identify limited precision counters would not directly lead to false results, but it would lead us to unnecessarily mark their wrapped segments as discontinuities. We can also use limited precision counters to rule out shared counters: two time series with different ranges cannot share a counter. We found about 0.39% of usable time series had limited precision, most commonly using 12 bits with values between 0x6000 and 0x6FFF.

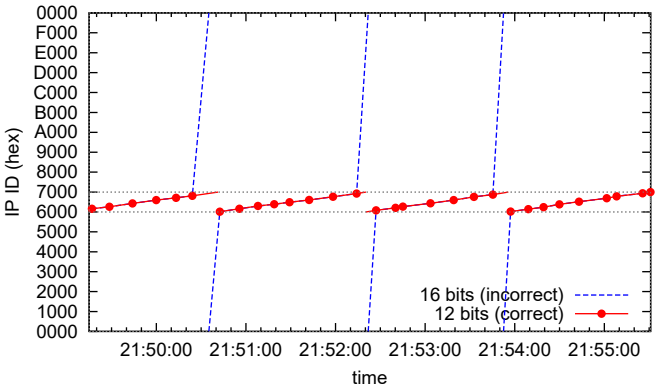

Fig. 8. An example of the limited-precision counter anomaly. The naïve interpretation shows a much larger unwrapped delta (dashed line) than the correct interpretation.

The final type of anomaly, *XX00 and XXFF outliers*, is most easily explained by an example. Given a sequence of (hexadecimal) ID values 11FC, 11FE, X, 1202, 1204 , one would expect to see X=1200, but might actually see X=1100. Taking this at face value would mean the counter advanced by FF02 (from 11FE to 1100, almost the full ID space) in the time it would normally advance by just 0002. We speculate that a more likely explanation is that the two bytes of the counter are updated asynchronously, and the value was generated *between* the time that the low byte wrapped from FF to 00 and the high byte incremented from 11 to 12. We also saw cases where the bytes seem to be updated in the opposite order, giving X=12FF in our example. Rather than allowing one possibly incorrect value to interfere with the MBT, we discard the one dubious point and keep the rest of the series. Whatever the cause of the anomaly, discarding one point will not significantly hurt MBT results, but if the value is indeed incorrect, keeping it would definitely cause false negatives. We found these anomalies in only 0.01% of 3.2 million observed monotonic time series.

APPENDIX G
RESPONSE RATE AND IP ID CHARACTERISTICS

To study the usefulness of the probing methods, we analyze our Estimation stage, in which we attempted to collect IP ID time series from 2 323 641 target addresses with all available probing methods. The Indirect method could not be used with addresses gathered from non-Ark sources, because we do not have the necessary traceroute information for them. Table VII shows the results.

We count a time series as having *insufficient responses* if fewer than 75% of the probes to the target elicit the expected response. The subcategories enumerate the most common reasons. *Unresponsive* means more than 75% of probes did not elicit *any* response. During Indirect probing, a sequence of TTL expansion that does not elicit any response from the target is counted as a single non-response. We count a time series as

TABLE VII
CLASSIFICATION OF IP ID BEHAVIOR OF ADDRESSES PROBED WITH VARIOUS METHODS IN THE ESTIMATION STAGE.

	TCP		UDP		ICMP		Indirect	
addresses probed	2 323 641	100.00%	2 323 641	100.00%	2 323 641	100.00%	1 832 771	100.00%
insufficient responses	905 267	38.96%	1 151 476	49.55%	482 399	20.76%	352 537	19.24%
mostly unresponsive	865 498	37.25%	1 014 227	43.65%	459 545	19.78%	322 991	17.62%
mostly unexpected	39 741	1.71%	136 863	5.89%	22 723	0.98%	28 454	1.55%
responsive, but degnerate ID values	137 363	5.91%	17 164	0.74%	999 677	43.02%	138 553	7.56%
mostly zero	130 744	5.63%	15 044	0.65%	1 293	0.06%	110 849	6.05%
mostly repeat	100	0.00%	568	0.02%	236	0.01%	985	0.05%
mostly reflect	6 516	0.28%	1 349	0.06%	998 077	42.95%	26 270	1.43%
responsive and nondegenerate, but nonmonotonic	477 394	20.55%	6 490	0.28%	8 619	0.37%	17 941	0.98%
responsive, nondegenerate, and monotonic (usable)	803 617	34.58%	1 148 511	49.43%	832 946	35.85%	1 323 740	72.23%

unexpected if more than 75% of the probes elicit a response of an unexpected type. For most time series, either all or none of the responses are unexpected. Most of the unexpected responses are ICMP *destination unreachable* messages from non-target addresses.

The main cause of unresponsiveness for the Indirect method appears to be network changes during the delay between the traceroutes and our experimental probes. When the delay is shorter, the response rate is higher. For example, in a different Indirect probing run to 3 000 targets from a single monitor, using addresses gathered from traceroutes taken only 3–4 hours earlier, only 1.2% of time series had insufficient responses. The traceroutes collected for Table VII were taken up to 18 days before the Estimation run, showing that Indirect probes can still be useful even after a moderate delay. However, we do see significant variability between monitors in the response rate to Indirect probing, suggesting different levels of route instability and per-packet load balancing near each location.

We classify a time series as having *degenerate ID values* if it had sufficient responses but 25% or more of the ID values were zero, some other constant value, or the value used in the probe packet. Such ID values are not useful to us because they do not reveal the state of an underlying shared counter. The subcategories enumerate time series for which more than 75% of IDs had the same type of degenerate value. Nearly half of the targets respond to ICMP *echo request* by echoing the ID, and a significant fraction of targets respond to TCP and Indirect with zero-valued IDs.

Any time series that passed all of the above tests is tested for monotonicity. If its ID values cannot be modeled as a monotonic counter, is is classified as *nonmonotonic*. TCP is the only method for which a significant fraction of targets passed the earlier criteria only to be classified as nonmonotonic.

Finally, any time series that is responsive, not degenerate, and monotonic, is classified as *usable* for testing with MBT.

APPENDIX H
UTILITY OF MULTIPLE PROBING METHODS

Table VIII shows the increase in target responsiveness and usable time series achievable by employing multiple probing methods for our dataset. Individually, ICMP has the highest responsiveness but the lowest amount of usable time series due to many addresses echoing the IP ID of the probe in the response. UDP and Indir have the highest amount of usable time series of any single method despite being more susceptible to rate limiting. If we employ all four methods, 89.2% of addresses respond to at least one method, and 80.6% yield usable time series to at least one method. This improved coverage will make alias resolution much more complete.

TABLE VIII
UTILITY OF COMBINING MULTIPLE PROBING METHODS. PERCENTAGES ARE RELATIVE TO 2 323 641 TOTAL ADDRESSES.

combination of methods				responsive	usable
TCP				62.75%	34.58%
	UDP			56.35%	49.43%
		ICMP		80.22%	35.85%
			Indir	64.97%	56.97%
	UDP	ICMP	Indir	88.27%	77.39%
TCP		ICMP	Indir	89.11%	76.02%
TCP	UDP		Indir	85.72%	77.28%
TCP	UDP	ICMP		82.66%	68.91%
TCP	UDP	ICMP	Indir	89.25%	80.60%

TABLE IX
CROSS-METHOD COUNTER SHARING FOR ADDRESSES THAT YIELD USABLE TIME SERIES TO MULTIPLE METHODS.

methods	addresses	shared	
TCP:UDP	595 465	562 582	94.48%
TCP:ICMP	383 712	341 247	88.93%
TCP:Indir	511 111	456 523	89.32%
UDP:ICMP	523 710	509 951	97.37%
UDP:Indir	774 993	745 913	96.25%
ICMP:Indir	545 585	525 224	96.27%

APPENDIX I
CROSS-METHOD IP ID COUNTER SHARING

Table IX shows the prevalence of cross-method counter sharing for our dataset. For each pair of methods, Table IX lists the number of addresses that responded to both methods with usable IP ID values and then the count and percentage of those addresses that had a shared counter. Overall, there is a high incidence of counter sharing, ranging from 88.9% to 97.4%. As expected, TCP and UDP share often at 94.5%. The sharing rates of the remaining pairs seem to be correlated with the response type; that is, counters seem more likely to be shared when two probing methods elicit a similar type of response. For instance, the sharing rate of TCP with either ICMP or *Indirect* is comparatively low perhaps because TCP rarely elicits an ICMP response. In contrast, UDP always elicits an ICMP response, and we thus see comparatively greater counter sharing between UDP, ICMP, and *Indirect*.