The Cost of the Path Not Taken

Max Schuchard
University of Tennessee
Email: mschucha@utk.edu

John Geddes
University of Minnesota
Email: geddes@cs.umn.edu

Mike Schliep
University of Minnesota
Email: schliep@cs.umn.edu

Nicholas Hopper
University of Minnesota
Email: hopper@cs.umn.edu

***Abstract*—We consider the problem of estimating the latency of a feasible but unused Autonomous System-level path on the Internet. This problem arises in evaluating the overhead incurred by censorship and surveillance circumvention schemes that alter the Internet routing infrastructure, and the cost of attacks against such schemes. Since these paths are not advertised by the current routing infrastructure, they cannot be directly measured by end hosts, leading researchers to estimate the costs indirectly. Using traceroute measurements of observed Internet paths, we measure the accuracy of the two methods used in the literature to date, finding that these methods have poor accuracy and correlation, explaining as low as 3% of the variation in observed AS path latencies, and at most 42%. We also describe an improved method that can balance accuracy and path coverage. At the high end our estimator can explain up to 83% of variation in observed AS path latencies, while still being able to achieve 56% when maximizing the number of paths able to be estimated.**

I. Introduction

While the "end to end principle" dictates that transactions between two Internet hosts should not depend on the underlying details of the path between them, in practice several properties of these paths can be highly relevant. For example, the list of Autonomous Systems (ASes) that appear on the path can help measure the exposure of the hosts to censorship and surveillance; and the latency of the path determines to a large extent the performance of the application in terms of responsiveness and throughput. Therefore measuring these properties can be important for the evaluation of systems, both proposed and deployed, which have an impact on the construction of paths on the Internet.

In the case of AS path inference, algorithms can use routing information and inferred AS relationships [7] to find the likely paths used between an arbitrary pair of networks with reasonably high accuracy, and can also be used to find alternate paths that, while not preferred under the current routing configurations of ASes, could be used without violating the "valley-free" routing principle.

Similarly, many techniques have been developed to estimate the latency between Internet hosts. These range from deploying specific measurement nodes [11], to using existing protocols [12], to inferring AS-level paths and using the underlying topology to build more accurate estimates [19]. In contrast to AS topology inferences, however, these methods only estimate the paths *currently* utilized by the Internet routing infrastructure.

Since ASes and ISPs represent a particularly relevant threat model in the context of censorship and surveillance, several recently proposed circumvention schemes [14], [15], [21], [22] involve changes to the routing infrastructure so that either the circumvention schemes or attacks against these schemes can lead to the use of AS paths that exist but are *not used by the current routing infrastructure*. In this case, current systems for estimating the latency of these paths do not function since the proposed paths are not actively in use.

In this paper, we examine techniques to estimate the latency cost of these unused AS paths. Previous work has used AS hop count [15], [21], or average inter-AS latency [14] to estimate these costs; we use observations of currently used Internet paths to assess the accuracy of these techniques, showing that they are only weakly correlated to actual performance. We also introduce a new technique to improve the accuracy of AS path latency estimation, based on conditional average inter-AS latencies. This technique can slightly reduce the *coverage* of AS paths it can estimate given a data set, while significantly improving the accuracy of the estimates it produces.

The rest of the paper is structured as follows. Section II covers previous work in latency estimation and some related research that has utilized such estimations; Section III goes into further details on existing methods on estimating latencies of new paths and introduces our new technique; Section IV details how the data was collected and processed, along with our experimental setup used in evaluating the various methods; Section V compares and contrasts the methods, examining the *accuracy* and *coverage* of the techniques discussed in Section III; Section VI examines a case study demonstrating the affects of method selection; and finally Section VII discusses conclusions of the difference of the methods.

II. Related Work

A. Estimating Latency

Previous work on latency estimation has had the goal of estimating the latency between pairs of hosts based on the current network conditions. The IDMaps algorithm [11] used a deployed infrastructure of tracer nodes that perform latency measurements. To estimate a latency between unknown end points, the system uses triangulation in order to calculate an upper bound on the latency as an estimate. Similarly, Meridian [24] relied on a set of Meridian nodes and measurements from the targets to estimate distance(latency) between endpoints. The King tool [12] relied on recursive DNS queries to measure latencies between servers. *Network coordinate schemes*, such as GNP [20] and Vivaldi [5] attempt to assign coordinates to hosts so that the distance between coordinates is a good estimate of the latency between the hosts. Topology-aware

schemes such as iPlane [18], [19] compute the underlying AS path used between a source and destination and attempt to stitch together paths from the source through an intermediate point, and from a separate "vantage point" through the intermediate point to the final destination. However, none of these systems is designed to estimate how inter-host latency would change based on the use of a different path.

B. Applications for unused paths

Tor Relay Selection: Tor [8] is a system that provides anonymity by forwarding data through multiple proxies, called relays, to an end destination. The AS where each relay resides has implications both for anonymity [9], [10], [16] and performance [4], [23]. Work dealing with anonymity is generally only concerned with what AS-level paths are traversed from client to end destination through Tor. When evaluating performance, latency measurements or estimates are needed in order to determine how relay selection in Tor can affect performance. Akhoondi *et al.* [4] used only measured latencies associated with specific paths to perform their evaluation. Due to the limitations of scalability, Wacek *et al.* developed methods of performing larger scale simulations when using AS-aware path selection strategies in Tor. Using CAIDA traceroutes, they construct a network topology at the Point-of-Presence (PoP) granularity, allowing them to avoid some of the large variances seen in AS-to-AS latency measurements. However, due to the incompleteness of the traceroute data, they were only able to incorporate about 60% of relays into the model.

Decoy Routing and Censorship: To overcome some of the limitations seen in systems such as Tor, a new type of system called decoy routing [13], [17], [25] was proposed, seen in systems such as Telex [25], Cirripede [13] and TapDance [26]. These systems use ISPs to redirect traffic to a covert destination a censor might be attempting to block, while still appearing to travel to a benign overt destination.

Schuchard *et al.* [22] showed that a routing capable adversary could choose alternate AS paths to avoid ISPs deploying decoy routers. Houmansadr, Wong, and Shmatikov [14] looked at the cost in terms of latency that such a censor would face in attempting to circumvent ISPs using decoy routing. In evaluating these attacks simulations need to be run to produce an Internet-wide AS-topology to determine what actual paths are available to a censor. For evaluating the performance cost of the attack, Houmansadr *et al.* estimated the latency of the path used before and after the attack. This can be difficult because in order to induce a cost, the path used afterward must be one that is not used in the global routing table before, meaning measurements of the actual path will be limited. To overcome this, estimates of inter-AS latencies are computed, which are then simply summed to estimate the latency along an entire path.

Anonymity in Future Internet Designs: While decoy routing systems take advantage of the current state of the Internet, systems such as LAP [15], ANDaNA [6], and Dovetail [21] have proposed protocols for lower-latency anonymous communication under "future Internet designs" that retain autonomous systems but change the protocols used to propagate and select routes between ASes. Because these routing protocols inherently use different AS paths than BGP, estimating the latency of paths used with or without anonymity overlays involves AS paths that are not currently used in the global routing table. To circumvent this, both LAP and Dovetail estimate the cost of all AS paths by the number of ASes on the path, and evaluate the overhead in terms of this estimate.

III. Methods

To build their respective models, all methods take in training data of a list of traceroutes, where each traceroute is a sequence of triplets containing router IP, ASN, and round trip time (RTT) from the source to router.

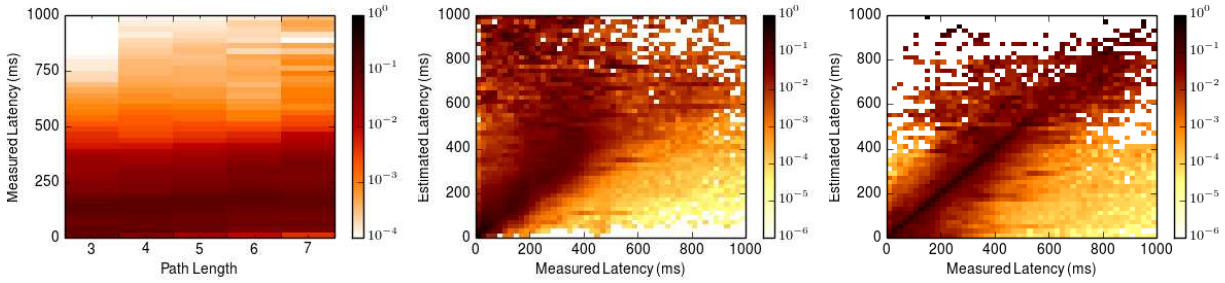
Hop Counting: Many times the goal of estimating new path latencies is to determine how much slower or faster a system would be when using newly generated paths. To do so, one of the simplest heuristics is to just use the raw number of AS hops in the path as a proxy for actual path latency. This naturally assumes a level of homogeneity in AS links that is not truly reflected in the real-world Internet, but still can be useful when the interest is to get a general sense of the impact of using new paths.

Hop-to-Hop Average: Houmansadr et al. describe a method in [14] using average latencies between ASes. Their method originally used the direct point to point latencies from the IPlane POP dataset, regardless of whether or not the path between the points of presence contains intervening ASes. We refine this method to only consider latencies between AS when we can observe no intervening ASes on a traceroute. Given two directly connected ASes in a traceroute, the latency between each router in the AS is stored in a global list for the ASes. For example, given a segment of a traceroute: $\langle (IP_1, A, t_1), (IP_2, A, t_2), (IP_3, B, t_3), (IP_4, B, t_4) \rangle$, which crosses AS A and AS B, we store latencies $(t_3 - t_1), (t_3 - t_2), (t_4 - t_1), (t_4 - t_2)$ in the list of latencies "from A to B". Then when producing an estimate for an AS-level path $[A, B, C, D]$, it simply return the sum of latencies between ASes, $lat(A, B) + lat(B, C) + lat(C, D)$, where $lat(AS_1, AS_2)$ is the mean of all latencies stored from AS_1 to AS_2.

Composite Construction: Our composite construction method has three main improvements. First, along with keeping track of latencies for each hop-to-hop seen, we also keep track of latencies for *all* partial AS paths contained in each traceroute. Second, instead of using all latencies between the first and last AS in the partial AS path, we only use the difference of the RTT for the first router in the first AS, and the first router in the last AS. Using the example traceroute from the Hop-to-Hop method, instead of storing all 4 of the times calculated, we would only store the time from IP_1, the first router in A, to IP_3, the first router in B, in this case $t_3 - t_1$. This gives a more accurate reflection of the time it takes to go all the way through AS A and over the link directly connecting A to B. Finally, since the ingress point to an AS can also effect what the expected latency would be, we

	iPlane		CAIDA		Combined	
	Coverage	R^2	Coverage	R^2	Coverage	R^2
Path Length	100%	0.022	99.9%	0.025	100%	0.013
Hop Average	99.4%	0.564	92.9%	0.312	99.2%	0.419
2-hop Global	99.4%	0.488	92.9%	0.468	99.2%	0.459
3-hop Global	90.5%	0.593	78.6%	0.548	89.8%	0.586
4-hop Global	55.3%	0.698	37.5%	0.632	54.1%	0.700
5-hop Global	17.5%	0.790	8.2%	0.686	17.1%	0.780
6-hop Global	2.5%	0.804	0.9%	0.655	2.5%	0.807
2-hop Incoming	77.7%	0.649	50.0%	0.599	76.7%	0.635
3-hop Incoming	77.7%	0.677	50.0%	0.611	76.7%	0.659
4-hop Incoming	50.6%	0.730	29.8%	0.652	49.5%	0.724
5-hop Incoming	16.0%	0.803	7.3%	0.712	15.6%	0.800
6-hop Incoming	2.5%	0.822	0.8%	0.678	2.5%	0.828
Full CC	99.4%	0.618	92.9%	0.514	99.2%	0.560

TABLE I: Coverage and correlation scores of estimates for the different methods and across the different data sets



(a) (b) (c)

Fig. 1: (a) heatmap of path length verses measured latency (b) heatmap of estimates from the hop average method compared to the measured latency (c) heatmap of estimates from the full composite construction method compared to measured latency

keep track of two databases of latencies. Along with a global database of latencies seen for each subpath, we also have a database where we index each latency seen for the subpath by the incoming AS. In the rest of the paper we refer to these as *global* and *incoming* databases respectively. When a database is queried for the latency of a subpath, it returns the median value of the list of latencies for the subpath.

The process for computing an estimate for an unknown path is we first compute a set of all *composite constructions* of the path. A *composite construction* is a partitioning of the path edge list

$$[(A_1, A_2), (A_2, A_3), \ldots, (A_{n-1}, A_n)]$$

into sublists. For example, a composite construction of the path $[(A_1, A_2), (A_2, A_3), \ldots, (A_5, A_6)]$ might look like

$$\langle [(A_1, A_2)(A_2, A_3)(A_3, A_4)], [(A_4, A_5), (A_5, A_6)] \rangle$$

Given the full set of composite constructions, to produce an estimate we:

1) Filter out any constructions where the first subpath is not found in the *global* database.
2) Filter out any constructions where the remaining subpaths are not found indexed by the previous AS in the *incoming* database.
3) Compute the maximal subpath length across all constructions.
4) Filter out all constructions without a maximal subpath.
5) For each construction, compute the latency as the sum of latencies of each subpath.
6) Return the estimate for the unknown path as the median of the latencies of all constructions.

If after step 2 there are no remaining constructions, rerun the filter using the *global* database instead and continue with the remaining steps if possible.

IV. Experimental Setup

In this section we discuss the details on how the raw traceroute data was processed to give us a reliable set of AS level traceroutes, then cover the experimental setup.

A. Data Processing

The data set consisted of traceroutes from March 10, 2015 gathered from both iPlane [1] and CAIDA [3]. The iPlane data set consisted of 23,805,830 traceroutes and CAIDA had 5,600,176 traceroutes. We converted the IP addresses to their appropriate AS to give us AS-level traceroutes. Initially we used the origin to AS mapping provided by iPlane on March 10, 2015, but this resulted in many AS-level traceroutes containing *cycles*, where an AS would be shown to handle the traffic after already handing it off to a different AS.

To fix this issue, we built IP address to AS mappings based on observations of BGP routing tables from the same

date as the traceroutes were collected. We gathered dumps, taken in two hour intervals, of routing tables from RouteViews hosts [2]. Each dump consists of multiple global routing tables advertised by a diverse collection of ASes who volunteer to peer with RouteViews. For each observed path, we extracted both the advertised IP block along with the ASN of the AS that originated the path. Mappings of IP block to owning AS were built using these observations. Since we combined multiple observations which occurred at disparate times, in some cases multiple ASes were observed simultaneously asserting ownership over the same IP block. We resolved such situations by rejecting ownership mappings for IP blocks that did not have a single AS which appeared in a super majority, in our case 90%, of the observed routing tables. We also filtered out paths from several ASes which attempted to originate IP blocks which fell inside either reserved or internal portions of the IP space. We then resolved each IP address observed in our traceroute data set to the most specific IP block which both contained the IP address and had a valid ownership mapping.

Using our IP to AS mapping, when converting the iPlane traceroutes to their respective AS-level traceroutes, we saw a reduction in cycles from 1.5 million down to around 180,000, along with a reduction in raw IPs that were able to be resolved, from 407,948 down to 397,247. The CAIDA traceroutes data actually saw an *increase* in the number of IPs that were able to be resolved, from 5,911,880 up to 6,138,008.

In the case that we had an IP address that either could not be resolved, or belonged to a reserved block, if the directly preceding and succeeding AS were the same, we assumed it belonged to the same AS and mark it as so; otherwise the traceroute would be dropped from the dataset since we could not tell which AS it actually belonged too. After these traceroutes were filtered out we were left with 7,833,818 iPlane and 3,454,513 CAIDA AS-level traceroutes.

B. Evaluation Metrics

To analyze the accuracy of an estimator, we used a modified leave-one out cross validation technique. All the traceroutes were processed and added to training data set for each method. Then we iterated through the traceroutes using each one as test path. However, instead of just removing the single test path from the training data, we removed *all* traceroutes in the training data that contained the test path traceroute as a subpath. This was done to mimic the conditions we would normally be under, where we need to estimate the latency of a path that does not exist in the current Internet, meaning no latency information would be directly available for the entire path.

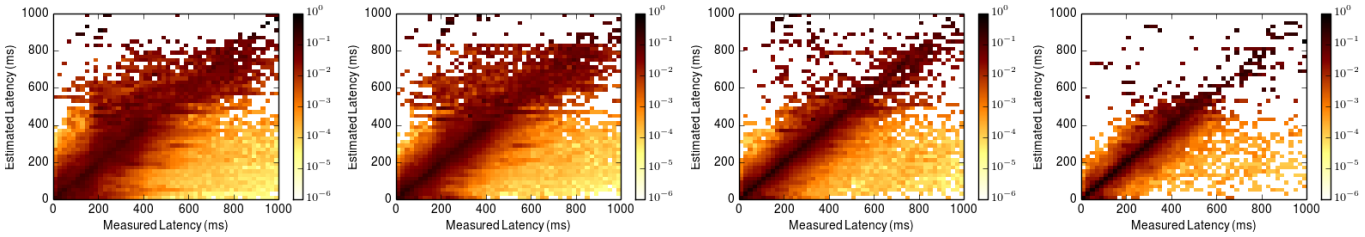
The main metrics we were interested in are coverage and accuracy. Coverage refers to the percent of traceroutes we were able to produce an estimate for after removing all the necessary training data. For accuracy we looked at two separate metrics, how well the estimates fit the data, and the error rates on the estimates. For determining fit we simply calculated the R^2 value when running a linear regression along the estimated verses measured latencies. Since the R^2 metric simply measures how well we fit the data and not necessarily how close we were to estimating the actual latency, we also examine how accurate the estimates the various methods produce, which is defined of the difference between measured and estimated latency. In order to prevent any potential selection bias, when directly comparing accuracy in terms of error between different methods we only included test path that all methods being compared could produce an estimate for.

V. Results

The high level results of the experiments can be seen in Table I. Here we broke down the results of using the iPlane and CAIDA data separately, and when combined into one data source. Also, instead of only reporting the results using the full composition construction method, we report results under more constrained parameters. We produced an estimate using *only* composite constructions with a maximal subpath length of $n \in \{2, 3, 4, 5, 6\}$, and only using either the incoming database or the global database. This allows us to see the trade off in coverage and accuracy as we tighten or loosen the restrictions on these parameters. Note that the full composite construction method has the same coverage as the hop average method since we will always have an estimate with maximal subpath of length 2 in the global database, just different latency estimates of the subpaths. When tightening the parameters in the composite construction method, we see that we tend to give up coverage for more accurate results. Not too surprisingly the path length method produces correlation scores around 2%, as it is unable to accommodate the wide range of latencies that can be produced within paths of the same length. Our composite construction method produces a fit almost twice as precise than the hop average estimator, but at the expense of a large reduction in coverage. However, when maintaining the same coverage we are still able to increase the fit by 14% compared to hop average.

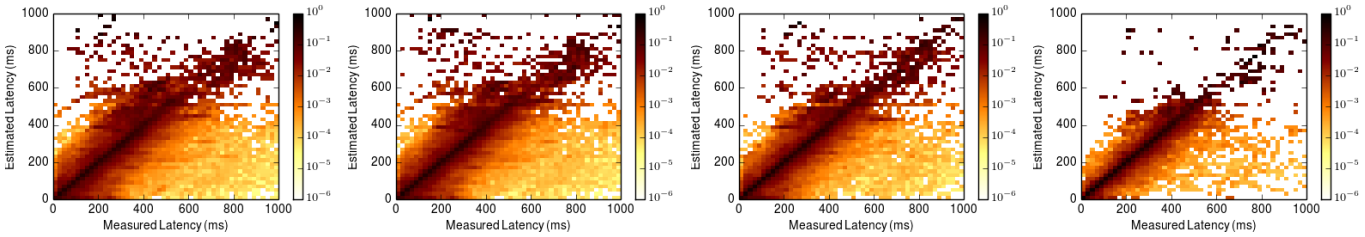
Figure 1a shows a heatmap of path length verses measured latency for all the paths. When path lengths rise, we see a slight correlation in the rise in density of latencies, but it does so slowly with a large tail end of the distribution. The averages of the latencies range from 150 to 250 ms, but we can see plenty of latencies reaching as high as 1000 ms. Results for the hop average method can be seen in Figure 1b. Here we see a much cleaner correlation of the measured and estimated latencies, with darker areas above the $y = x$ line indicating that the method is more likely to produce overestimates.

Figure 1c looks at the estimates for the full composite construction method. We see a much stronger density of estimates around the line $y = x$ indicating more accurate results, with much less overestimating as was seen when using the hop average method. We also broke down the results for composite construction when using the global database in Figure 2 and in Figure 3 when using the incoming database. We generally see more accurate results when using the incoming database, and as the maximal subpath length of the composite constructions increases, we see less noise and increasingly accurate estimates. One potential issue with



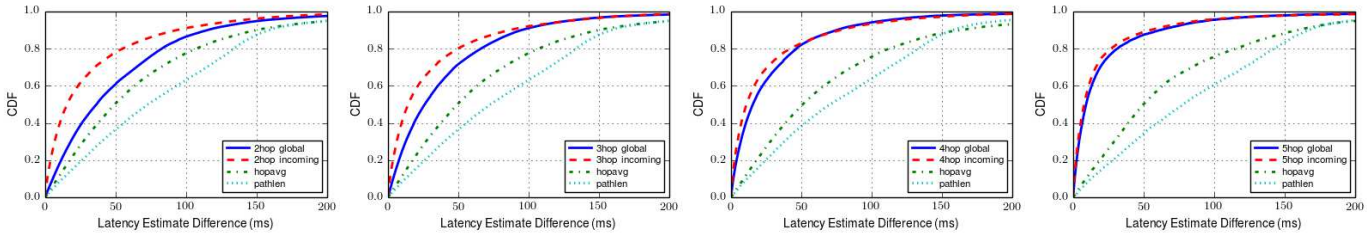
(a) Max Subpath Len 2 (b) Max Subpath Len 3 (c) Max Subpath Len 4 (d) Max Subpath Len 5

Fig. 2: Results from combined composite construction method, using global latencies



(a) Max Subpath Len 2 (b) Max Subpath Len 3 (c) Max Subpath Len 4 (d) Max Subpath Len 5

Fig. 3: Results from composite construction method, using latencies indexed by incoming hop



(a) Max Subpath Len 2 (b) Max Subpath Len 3 (c) Max Subpath Len 4 (d) Max Subpath Len 5

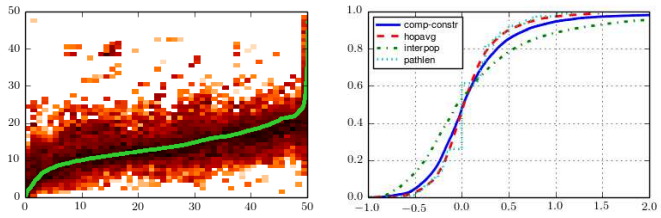
Fig. 4: Errors of composite construction compared to hop average and path length methods, across different maximal subpath lengths

just looking at the heatmaps is that they only show the potential accuracy of what can be estimated by each individual method. Perhaps the measurements that can be estimated by the composite construction method just happen to have less noise, leading to a selection bias. To check if this might be an issue, we compared the results across different methods including only measurements that each method could produce a result for, eliminating any potential selection bias. The results are shown in Figure 4, with each experiment split up into different maximal subpath lengths, including either only the global or incoming database. We see that after using maximum subpath length of 4 or longer, we gain little to no accuracy by considering the incoming hop. In all cases we are however seeing a substantial increase in accuracy compared to using the hop average and path length methods.

VI. Case Study

In this section we revisit the results of Houmansadr, Wong and Shmatikov [14], examining how latencies will be changed if a censor chooses routers to avoid a particular decoy routing deployment. The strategies used in our experiments differ slightly from [14]; as they are more expansive our before and after paths should contain a subset of the ones produced in [14]. In addition to path length, hop average, and composite construction methods, we also compared a hop average method that used the inter-PoP links data and the original IP to AS mapping provided by iPlane for parsing the training data, closely mimicking the methods *directly* used in [14]. Out of 58,953,866 total paths in the simulation, 9,378,666 paths changed. Of the paths that changed, the path length estimator was able to estimate 99.8% of paths, the hop average using the inter-PoP links data set could estimate 6.8% of paths, and the hop average and composite construction methods using the traceroute data produced estimates for 4.9% of paths. The main reason the inter-PoP estimator had a slightly larger coverage is it was less conservative in resolving IPs to ASes, resulting in a larger training data set to pull from. While coverage might be higher, this also has the downside of producing less accurate results.

Figure 5a shows the estimates using the composite construction method over top a heatmap of the latencies produced by the hop average method for the same path. While the



(a) (b)

Fig. 5: (a) Estimates from composite construction method verses hop average (b) Percent change in path latencies across the different estimate methods

trajectories for both methods overlap, there is a large range where the estimates fall, within 20% of the estimates produced by the composite construction method. However, even with the large variation in estimates produced, when we look at the percent change in latencies in the new path compared to the old path, the results look very similar in aggregate. Figure 5b shows the CDF in percent change in latencies, and even using the path length method produces almost identical results as the hop average and composite construction methods. The reason for the similar results is that if the methods are consistently biasing the estimated latencies, looking at the difference will still produce similar results. Interestingly, we still get results an order of magnitude less than those seen in [14], with our changes being $\pm 200\%$ compared to changes being in the 200-2000% range.

VII. Conclusion

In this paper we analyze some of the difficulties of estimating AS-level latencies across paths that are not used in the current internet topology. We discuss two previous methods, using raw path lengths and single hop averages, and introduce a new technique named composite construction. Our new technique is able to balance both *coverage* and *accuracy*, improving on the existing methods while maintaining high coverage. We finally look at a case study of newly generated paths, examine the range of estimates produced compared to our more accurate estimator, and analyze the methods in terms of change in latency in the newly generated path compared to the old existing path.

Acknowledgments: This research was funded by NSF grants 1314637, 1223421, and the University of Minnesota Doctoral Dissertation Fellowship.

References

[1] iPlane: Datasets. http://iplane.cs.washington.edu/data/iplane_logs/2015/03/10/.

[2] Route Views RIBS, March 2015. http://archive.routeviews.org/bgpdata/2015.03/RIBS/.

[3] The CAIDA UCSD IPv4 Routed /24 Topology Dataset - March 10, 2015. http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml.

[4] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A low-latency AS-aware Tor client. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 476–490. IEEE, 2012.

[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 15–26. ACM, 2004.

[6] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. Andana: Anonymous named data networking application. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[7] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, k. claffy, and G. Riley. AS relationships: Inference and validation. *SIGCOMM Comput. Commun. Rev.*, 37(1):29–40, Jan. 2007.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, 2004.

[9] M. Edman and P. Syverson. AS-awareness in Tor path selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 380–389. ACM, 2009.

[10] N. Feamster and R. Dingledine. Location diversity in anonymity networks. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, pages 66–76. ACM, 2004.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. Idmaps: A global internet host distance estimation service. *Networking, IEEE/ACM Transactions on*, 9(5):525–540, 2001.

[12] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 5–18. ACM, 2002.

[13] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov. Cirripede: circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 187–200. ACM, 2011.

[14] A. Houmansadr, E. L. Wong, and V. Shmatikov. No Direction Home: The True Cost of Routing Around Decoys. In *Network and Distributed System Security Symposium*, 2014.

[15] H.-C. Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. Nelson, M. Gruteser, and W. Ming. LAP: Lightweight anonymity and privacy. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, May 2012.

[16] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 337–348. ACM, 2013.

[17] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer. Decoy routing: Toward unblockable internet communication. In *USENIX Workshop on Free and Open Communications on the Internet*, 2011.

[18] H. V. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, and A. Venkataramani. A structural approach to latency prediction. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 99–104. ACM, 2006.

[19] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 367–380. USENIX Association, 2006.

[20] T. S. E. Ng and H. Zhang. Global network positioning: a new approach to network distance prediction. *Computer Communication Review*, 32(1):73, 2002.

[21] J. Sankey and M. Wright. Dovetail: Stronger Anonymity in Next-Generation Internet Routing. In *Privacy Enhancing Technologies Symposium*, 2014.

[22] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper. Routing around decoys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 85–96. ACM, 2012.

[23] C. Wacek, H. Tan, K. S. Bauer, and M. Sherr. An Empirical Evaluation of Relay Selection in Tor. In *Network and Distributed System Security Symposium*, 2013.

[24] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. *SIGCOMM Comput. Commun. Rev.*, 35(4):85–96, Aug. 2005.

[25] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the Network Infrastructure. In *USENIX Security Symposium*, 2011.

[26] Wustrow, Eric and Swanson, Colleen M and Halderman, J Alex. TapDance: End-to-Middle Anticensorship without Flow Blocking. In *USENIX Security Symposium*, 2014.