The effect of IP prefixes on BGP visibility

THOMAS VAN DEN BERG, University of Twente, The Netherlands

Anycast is mostly used for hosting Domain Name System (DNS) servers, Content Delivery Networks (CDNs), and other purposes, enabling multiple devices to share a single IP address by leveraging the Border Gateway Protocol (BGP) to decide which device to route a packet to. Research exists on how these catchments of Anycast networks form. In this paper, we investigate the impact of IP prefix diversity on network visibility. We want to give insight into the size of discrepancies in the catchment between identical Anycast deployments with different IP addresses. We used Verfploeter [4], a tool to measure catchments of Anycast services, and the TANGLED testbed [3]. To reach our goal, we modified Verfploeter to be able to send requests from multiple source addresses simultaneously. Our results show that 0.5% of the IP addresses we measured responded differently based on the IP prefix we used to reach them.

Additional Key Words and Phrases: Anycast, IP prefix, Catchment, BGP, Verfploeter

1 INTRODUCTION

BGP [9] is used to exchange prefix reachability information across different Autonomous Systems (AS) [12]. BGP has many attributes for configuration options, and the treatment of these attributes is up to network operators [8]. The flexibility of BGP can be both positive and negative for network operators. For one, it allows for Anycast networks, where multiple devices (also called Points of Presence (PoPs)) use the same IP address or prefix. On the other hand, the flexibility and its subsequent complexity can cause difficulties in optimizing catchments (the mappings of PoPs to clients they serve) for better performance. Some routing policy aspects could center around the preferential treatment of specific IP prefixes over others. Therefore, prefixes are not guaranteed to be evaluated purely by their merits. For example, some prefixes could be ranked differently due to business agreements, which can be hard to implement correctly [2]. Because it can be hard to correctly implement these routing policies, some legacy routing policy remnants could remain in effect after the business agreements have expired. This uncertainty makes it harder to estimate the Anycast catchments of different IP prefixes, even after the routing policies should have returned to neutrally evaluating a prefix.

1.1 Methodology

We aim to explore the visibility difference between IP prefixes with the same AS configurations. We will first create a method for comparing visibility between prefixes. Next, we plan to minimize the effect of time between measurements on the resulting catchments by modifying the tools we use to measure catchments. Afterward, we will examine whether minimizing the effect of time also decreased the differences in catchments between different IP prefixes. Finally, we will measure the catchments of our different IP prefixes using the most stable configuration we have found and compare the measurements using our method for comparing visibility between prefixes.

1.2 Research questions

This paper focuses on the following research question: ***RQ:*** *To what degree is the visibility of Anycast deployments affected by the IP prefixes?* To answer this research question, we have used the following sub-research questions:

- **SRQ1:** How can the difference in the visibility of Anycast deployments be compared using catchment measurements?
- **SRQ2:** How can we change the usage of the testing tools to improve the stability of a single prefix?
- **SRQ3:** How can the catchment similarity between IP prefixes be affected by the changes in usage of our testing tools (**SRQ2**)?

1.3 Contributions

The contribution of this paper will be to provide a method to evaluate the differences in visibility between different IP prefixes. Research on the visibility differences of individual IP prefixes across time [7, 10, 14, 16] exists, but these studies focus mainly on how time, not the IP prefix, affects visibility. We aim to give insight into the scope of visibility differences between IP prefixes by analyzing catchments between Anycast deployments using the same testbed but differing IP prefixes.

The remaining part of the paper is organized as follows: First, we will examine the related works to this research and their relevance in the next section. Next, we will explore how we plan to answer our research questions in our methodologies. Then, we outline our results for each (sub-)research question in a dedicated section. Afterward, we will outline possible future work based on our findings. Finally, the last section contains the conclusions of this research.

2 RELATED WORK

This section will discuss some related works we plan to use when writing this paper.

De Vries et al. created a tool to be used to measure catchments called Verfploeter [4]. This tool uses what they called “passive vantage points (VPs)," network nodes not controlled by the researchers used for measurements. These VPs differ from VPs of other measurement tools, referred to by De Vries et al. as “active VPs." For the rest of this paper, what the Verfploeter paper called passive VPs will be referred to as “Verfploeter passive VPs (VpVPs)." Verfploeter sends requests to these VpVPs and analyzes their responses, including which PoP received the reply. It can send tens of thousands of these requests per second, but the authors recommend limiting the request rate to around 6,000 requests per second. They can then use this data to get an idea of the catchment of an Anycast network. Hendriks created a fork of Verfploeter called MAnycast-extended, which modernizes and extends Verfploeter [5]. This fork, among

TScIT 37, July 8, 2022, Enschede, The Netherlands

other things, allows the clients to send requests instead of a centralized server. Bertholdo et al. described an Anycast testbed for academic purposes [3]. It uses PoPs deployed all across the globe. An earlier approach to measuring catchments was using the RIPE Atlas tool [1] by RIPE NCC.

There also already exists research into how we can analyze catchment measurements. One such paper is about partitioning the internet using catchments [11] by Schomp and Al-Dalky. They describe a method to partition the internet into a limited number of partitions and find where the visibility of these partitions diverges instead of comparing individual vantage points. They found that some part of the routing is seemingly random to external parties. Another paper evaluating the differences between catchment visibility is about the differences between IPv4 and IPv6 Anycast deployments [15] by Wicaksana. This paper compared the catchments of Anycast networks using different versions of the Internet Protocol. They use a percentage of the measurements where routes diverge as the main result. This percentage is an acceptable measurement here since they explicitly state multiple times that they desire total convergence. They also reason that as IPv6 proliferates, the routes should converge. Additionally, Rexford et al. investigated the stability of popular BGP routes [10]. Finally, Sommese et al. developed a method to find Anycast deployments using the Verfploeter tool on the TANGLED testbed [13]. This method leverages multiple Verfploeter measurements of the same source addresses and compares their results. If a measured address replies to multiple TANGLED PoPs, its /24 prefix is assumed to be using Anycast. They note their accuracy in finding unicast addresses but also found that many of the Anycast addresses their approach finds might be false positives. These false positives could be due to multiple factors, including but not limited to preferred routes by AS operators or traffic engineering.

Khan et al. analyzed the accuracy of IP prefix information in Internet Routing Registries (IRRs) [6]. IRRs contain routing information about the different ASes and contain information updated by the people running the ASes. In this research, the authors look at how up-to-date and accurate the information that ASes put into IRRs is.

3 METHODOLOGIES

We have created our sub-research questions to aid us in answering our main research question. **RQ** requires us to compare the differences in catchments using different IP prefixes and account for other variables, such as the time between measurements of the prefixes. **SRQ1** has been added to help us create a method to compare these differences. **SRQ2** aims to minimize the effect of time on the results of the measurements by increasing catchment stability. Once we have answered **SRQ2**, we will have found a method to do measurements with more stable results. We expect this method to result in fewer differences when measuring between IP prefixes since any time-related instability would also occur when gathering these measurements. **SRQ3** thus aims to see if this assumption is correct by verifying whether the difference decreases when increasing time-related stability. We will already have worked toward answering our main research question by answering our sub-research questions. We can therefore focus more on the results we get rather than how we want to obtain results once we start answering the main research question.

3.1 On answering SRQ1

For our first research question, we aim to develop a method of evaluating the catchment similarity between IP prefixes to improve results. We plan to combine multiple strategies to achieve this goal. The first strategy is to look at related works and see what other approaches exist. Some papers we have found [10, 11] have already done some work comparing routes on the internet. We could use the paper about BGP route stability [10] to compare approaches to identifying route stability and set the expectations for our experiments. We will use the other paper [11] if we need another way to look at the catchment differences. The second strategy would be to look at which tools are available to us and their capabilities. The tools most relevant for this task are Verfploeter [4] and the RIPE Atlas platform [1]. The first tool will likely be used because of its flexibility, while the RIPE Atlas platform requires more coordination with external parties. Finally, we can also use measurement metrics to compare the measurements, like the percentage of VpVPs where the catchment differs. If we have time to run multiple tests for each source address we are testing, we could even look at the differences in the stability of the routes. Using these strategies, we hope to come up with a way to compare the catchment data from different Anycast measurements.

3.2 On answering SRQ2

Another aspect of stabilizing measurements is decreasing the time between these measurements. Verfploeter can ping the VpVPs in a pseudorandom order to avoid rate limits and abuse complaints [4]. Because measurements can take longer than 10 minutes, it takes quite a long time between the requests to a single VpVP. The worst-case scenario is around 20 minutes, but if we do not shuffle the hitlist every time, the time between requests to a VpVP would be about 10 minutes. We could adapt Verfploeter to measure multiple source addresses at the same time. By sending ICMP ECHO requests (also commonly known as ping-requests) for the different source addresses to the same VpVP directly after each other, we minimize the time between the requests to a VpVP. As stated before, the recommended rate of requests is about 6,000 per second. The suggested approach of consecutively sending ICMP ECHO requests for different source addresses to the same VpVPs would increase the traffic to this VpVP in the short term. It would also enable a single Verfploeter measurement to measure multiple Anycast deployments. To avoid rate limits and abuse complaints, we should divide the number of requests per second by the number of source addresses since we send requests to different source addresses without rate-limiting. With this decreased request rate and alternative approach, we could still see 1,000 requests per second from each address when measuring six different source addresses. This results in a significantly smaller time difference between sending the first and last request to a single VpVP. By decreasing the time between requests to a single VpVP with a factor up to more than 100,000 when testing six source addresses and even more when with fewer addresses, we hope to increase the stability of our measurements.

3.3 On answering SRQ3

With this sub-research question, we aim to see if the methods described in **SRQ2** affect the difference in catchment between IP prefixes. If the methods from **SRQ2** had little influence on the catchment stability, we expect these methods to have little effect on the catchment stability between source addresses on different IP prefixes. We reason that if the methods from **SRQ2** affect the catchment stability a lot, then the variety in the routes stems from factors other than the IP prefix. We expect that unstable catchments from evaluating a single IP prefix would result in more notable differences between catchments when comparing different IP prefixes. These differences, however, have multiple possible reasons, not just the differing IP prefixes. If we achieved perfect catchment stability for single IP prefixes, we would expect that most of the differences in catchment between IP prefixes stem from diverging routing policies based on IP prefixes.

3.4 On answering the main research question

To answer the research question, we can run experiments to see how stable catchments are across different IP prefixes. We will use the approach to measure catchment similarity from **SRQ1** to compare the results from the measurements. We will run the experiments with the configurations we have explored for **SRQ3**. **SRQ2** is not directly used to aid in answering the main research question. It just serves to answer **SRQ3**. In the end, we hope to see the effect of IP prefixes on the catchment of different source addresses, with as little interference from other variables as possible.

4 COMPARING VISIBILITY OF ANYCAST MEASUREMENTS

We will look into ways to compare the visibility of different IP prefixes in this section. We split this problem into two parts. First, we aim to isolate the measured differences caused by using different IP prefixes rather than other factors. Second, we need to find a way to interpret the resulting differences.

Conceptually, four factors can influence the differences found between measurements. Three of these factors are somewhat controllable by us. These factors are the packet content, packet origin, and time at which we send the packet. The final factor is what we call routing decisions. We use routing decisions to describe anything a network operator might use to diverge routing paths not based on the previous three factors.

In our case, the only impactful difference in packet content should be the source address. Since we want to measure this impact, the difference in packet content from this will not be an issue. We also found two possible aspects regarding packet origin differences. The first is which network link we use, and the second is the source address we use to send the packet. To neutralize the impact of the network link, we will take all measurements of a VpVP from all Anycast PoPs, thus using the same network links every time. We aim to measure the difference in catchment when using different IP prefixes, so we want the difference in source addresses to exist. We aim to send measurement packets with minimal delay between them for the time aspect. The internet is not static and constantly changing, so less time between measurements will mean fewer changes. The final factor influencing differences between measurements is the effect of routing decisions. As stated before, in the paper about partitioning the internet using catchments [11], some routing happens according to rules that are, in essence, random. One such rule would be routing the packet according to the hash of its port number. These routing rules should route similar packets similarly, so we expect this to have a minimal impact.

We have found some papers that deal with interpreting catchment measurements [11, 15]. Just like the paper comparing IPv4 to IPv6 [15], we could use a percentage of the measurements where routes diverge as the main result. In our case, just as Wicaksana, we also want the routes to converge, but unlike them, we have not found a reason to suspect the improvement of the situation soon. The paper that partitioned the internet using catchment data [11] also presented an interesting approach to interpreting different results. We could determine what percentage of these partitions diverge based on the IP prefix if we partitioned the internet according to our results. This approach requires many different Anycast deployments, however. We only have access to a single customizable deployment, so if we wanted to do multiple measurements, we would not be able to do these simultaneously. The fact that we can not do the measurements simultaneously creates an issue. We suspect time to be a significant source of noise in our results, so if we run long experiments, we drastically increase the amount of noise in our data. Sadly, such a partition is thus likely unsuitable for us.

4.1 Isolating our measurements

As stated previously, we expect four reasons for routes to diverge. One of these is the difference in packet content, mainly stemming from the different IP prefixes. Two others are packet origin and time, which we can aim to minimize by changing Verfploeter [4] to send requests to measure different source addresses from all Anycast PoPs to the same VpVP directly after each other. The final reason is the effect of routing decisions, which we can measure by customizing Verfploeter to get multiple measurements for the same IP prefix from the same VpVPs.

We can minimize the time between measurements of a single VpVP by changing our testing tools, which is what we aim to do when modifying Verfploeter (**5**). It can never become zero, but it should be possible to send the requests to the same VpVP from the same PoP for different source addresses directly after each other without rate limiting.

We cannot affect the differences in routing that we attribute to routing decisions because we do not control the routing behavior of other Autonomous Systems. The opaqueness of routing decisions can cause us to believe that some IP prefixes with diverging routing behavior behave identically or cause us to conclude that IP prefixes with the same routing behavior behave differently. As previously discussed, we do not expect a significant impact of routing decisions based on factors other than the IP prefix on our measurements. If our results show that such an impact exists, we will try to mitigate the effects of these other routing decisions. One way to mitigate these effects is to run more measurements probing VpVPs multiple times using source addresses from a single IP prefix. Then we can look if they route to the same set of Anycast PoPs and if the ratios

of traffic received by Anycast PoPs are similar. This approach does require us to send more requests, which increases the time it takes to measure a single VpVP, increasing the risk of something along the route changing during the measurement. We also risk harassing VpVPs or other links if we send too many measurement requests without rate-limiting.

4.2 Interpreting the results

As stated before, partitioning the internet would require us to change our experiments to take longer. Experiments taking longer increases the risk of routing changes occurring in the meantime on the internet. Therefore, we will first focus on percentages of VpVPs occurring in the same catchment. We might find that the results when using a single measurement with multiple IP prefixes might not be satisfactory. In this case, we could try to run more measurements after changing the deployments or change our experiment to measure VpVPs multiple times using the same IP prefix. Both of these approaches would increase the time it takes to measure, thus adding some noise to the data from changes in routes across the internet.

4.3 Summary

Interpreting the results is not only a crucial part of our study, but our approach also depends on the results. We might find that our measurement noise is negligible depending on the results of our tests after modifying Verfploeter (**5**) or that we need to account for noise.

5 EXAMINING THE EFFECT OF TIME BETWEEN MEASUREMENTS ON ROUTE STABILITY

In this section, we will modify Verfploeter to minimize the time between requests to the same VpVP to see if this affects route stability.

We use a fork of Verfploeter called MAnycast-extended [5]. As stated before, this fork allows the Verfploeter clients to send the requests. We want to take it a step further and send requests to the same VpVP directly after each other without rate-limiting.

5.1 Testing the effect of time on catchments

After we made our changes to MAnycast-extended, which we describe in Appendix B, we want to compare the differences in catchment between multiple runs with different amounts of time between them. Our modified version cannot determine whether a response replies to a request of the first or second test, meaning we cannot apply MAnycast^2's approach with the same accuracy as we can when we can separate results. We could try to modify Verfploeter to include information to separate the tests in the results, but this fell outside of the scope of this study. Therefore, we decided to run two tests. The first would compare running our modified version of Verfploeter twice with a single source address to running it once with two identical source addresses. Here, we can not ascertain whether networks are using Anycast as well as before because we don't know which request resulted in which responses when using multiple identical source addresses. Next, we would test using two different IP addresses in the same /31 IP prefix in the same manner. We can remove Anycast addresses with greater accuracy using different IP addresses, but we risk ECMP load balancing distorting our results.

As can be seen from the results in Figure 1, using multiple source addresses seems to reduce test results where an IP address only responds to a single source address. This decrease is especially apparent when we use multiple source addresses of the same /32 prefix, where we classified no VpVPs as only responding once. It is unlikely that all VpVPs answered requests of both tests, especially when we compare the results to what happens when we use an identical /31 IP prefix instead of an identical /32 prefix. The lack of "One occurrence" results is due to a limitation of our approach. Only VpVPs that replied once can be classified as "One occurrence" when using multiple source addresses of the same /32 IP prefix. We can not determine whether these requests came from the first source address or the second if they replied to multiple requests. Therefore, we cannot classify a VpVP as replying to only one test if we use multiple identical source addresses. As long as the VpVP responds more than once, even if these replies were to the same source address, we must classify the VpVP as replying to both tests. We can also see that using the same /31 IP prefix instead of the same /32 prefix did not affect the share of measurements determined to have reached other Anycast networks. Additionally, as expected, when using two different IP addresses in the same /31 prefix, the share of measurements responding to different PoPs increases significantly.

6 EXAMINING THE EFFECT OF USING MULTIPLE SOURCE ADDRESSES (5) ON CATCHMENT SIMILARITY BETWEEN IP PREFIXES

In this section, we will find out whether the changes from **SRQ2** affected the results when we compare IP prefixes.

To determine the effect of using multiple source addresses, we ran a test where we looked for catchment similarity in two different ways. First, by running a test with a single source address twice and comparing the catchment, followed by a test with two source addresses and comparing the catchment between the source addresses.

As we expected, we can see in the results in Figure 2 that, just as with the comparison between using single and multiple addresses within the same /31 prefix (Figure 1), the most significant result is a decrease in the number of passive VPs which only replied once. It also seems that occasionally the number of different replies increases when using multiple measurements with a single source address. This increase is, however, not substantial enough to draw any meaningful conclusions.

In general, minimizing the amount of time seems not to do much more than decrease the number of passive VPs who reply once. The results might have been more significant if the tests using single source addresses took longer, resulting in more time between measurements. However, we remain convinced that the decrease in time between requests to the individual VPs was the right choice, even if the effect is not very apparent.

7 THE EFFECT OF IP PREFIXES ON DEPLOYMENT VISIBILITY

Now we will compare a /31 and /23 prefix to determine whether the /24 prefix matters for the visibility of an Anycast network. We repeated this test five times to ensure some stability in our results. We have put all our results in Figure 3. As is visible in this figure, the different /32 prefixes seem to vary more in catchment than the /24 prefixes. This unexpected result could be the case because of ECMP load balancing. It could be the case that ECMP load balancing rules state that adjacent IP addresses should take different routes but that it is more random when only the 24th bit differs. Because of these results, we ran another test where we modified the 32nd bit. We hope that this test will allow us to classify VpVPs using ECMP. To filter out ECMP results, we ran a test with four source addresses with different 24th and 32nd bits. First, we filtered out the “Anycast" and “One occurrence" results, and then we classified results that responded to different PoPs when varying the 32nd bit of the source address as ECMP. After this, we just compared PoPs between the source addresses with different 24th bits to find which results diverge in visibility due to the IP prefix. We put the results of this test in Figure 4. In these results, around 0.5% of the VpVPs that responded seem to discriminate based on the IP prefix. However, the scope of this test was quite limited, and it would be interesting to see how a /23 IP prefix compares to a smaller prefix.

Another disclaimer could be that we compare two different /24 IP prefixes in the same /23 prefix. This relatively small prefix difference means there could be discrimination when using /24 prefixes further apart, but this would require more resources to test.

8 FUTURE WORK

There are two types of future work that we have found which we believe would be beneficial. The first is more testing using different source addresses, BGP configurations, and more extensive tests. The second is by improving the Verfploeter application further. As has been discovered in subsection C.3, flushing results immediately to disk causes some clients to crash.

When it comes to doing more testing, there are multiple possible approaches. One of the most interesting ones would be to change the BGP configurations or IP addresses. These tests could aim to see if IP prefixes that lie further apart do show a difference. Furthermore, more research into ECMP routing could explore the role this load-balancing technique plays in catchment overlap. Finally, more extensive testing could prove valuable because of the limited scope of our tests. More prolonged testing with more IP addresses could increase our knowledge of the role of prefix discrimination on the internet.

As stated before, Verfploeter clients can crash when flushing results to disk immediately, and we expect this to be the case because result flushing causes a packet to not arrive in time. Flushing results to disk before the program finishes is beneficial since this would allow for experiments with more source or target addresses or more probes. There is probably a better approach using multithreading than writing every single result to disk independently, which could be added. Furthermore, the fact that clients can crash when something goes wrong with a packet should be resolved. It happens more frequently when flushing to disk immediately but logically is a risk when something goes wrong with a packet either way. The crash message states that the issue manifests when unwrapping the data field of a Task message. This location is strange, however, since according to the Protocol Buffers data structure, this is a ‘oneof’ field. It does not make sense that this field is optional in Rust if this field means one of the options and not one or none. I do not know enough about the Protocol Buffers data structure to say what ‘oneof’ should look like. Either way, maybe a more modern version of the Protocol Buffers libraries for Rust fix this issue, or a rewrite of the Verfploeter protocol is possible to avoid this issue. This issue with Protocol Buffers is only an issue of error clarity, and the fact that a packet arrives incomplete is the issue causing the program to crash. I don’t know enough about the sockets used to offer good insight here, but it should be possible to ask the server to repeat the packet if it arrives corrupted instead of just crashing the client.

Upon this suggested work’s completion, we could have a better view of the internet and better tools to measure catchments.

9 CONCLUSION

We think it is legitimate to say that nearly nothing went as expected with our experiments. From a relatively simple modification of the order of packets consistently crashing the Verfploeter CLI to the stability of results increasing from using different /32 prefixes to /24 prefixes. However, this lack of predictability makes the conclusions more interesting.

In creating a method to compare the visibility of catchment measurements, we learned that there are multiple ways to present our catchment measurements, some of which are more interesting than others. One limitation of the more elaborate approaches to classifying results is that it would take a large amount of data, which fell outside the scope of this study.

When we wanted to determine the effect time had on our measurements, we found many limitations of the Verfploeter-based measurement tools we planned to use. Some limitations had solutions (such as the issues with Verfploeter crashing), but others have no apparent answer, or we left them as future work.

We also looked at whether our modifications of Verfploeter affected the results for our main research question. While answering this sub-research question, we did not know we would be classifying ECMP yet. We saw some differences in measurements, for instance, the decrease in VpVPs that participated in only a single test. Because the test where we classify ECMP VpVPs used four source addresses, we expect the effect of our modifications to be even more significant here.

Our research question concludes that there seem to be VpVPs that differ in routing based on the /24 IP prefix. Initially, when comparing a /23 to an /31 prefix, we saw little difference due to ECMP, but after we changed the test to filter out these results, we found that 0.5% of VpVPs seemed to change routing behavior based on the /24 prefix. It is positive to see that this percentage is small because this means that, for the most part, the internet is working as intended and not discriminating on IP prefixes.

REFERENCES

[1] RIPE Network Coordination Centre 2010. *Home | RIPE Atlas.* RIPE Network Coordination Centre. https://atlas.ripe.net/

[2] Leandro M. Bertholdo. 2021. *Did you validate your routing policies?* https://blog.apnic.net/2021/09/22/did-you-validate-your-routing-policies/

[3] Leandro M. Bertholdo, João M. Ceron, Wouter B. de Vries, Ricardo de Oliveira Schmidt, Lisandro Zambenedetti Granville, Roland van Rijswijk-Deij, and Aiko Pras. 2021. TANGLED: A Cooperative Anycast Testbed. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (2021-05). 766–771. ISSN: 1573-0077.

[4] Wouter B. De Vries, Ricardo De O. Schmidt, Wes Hardaker, John Heidemann, Pieter-Tjerk De Boer, and Aiko Pras. 2017. Broad and load-aware anycast mapping with verfploeter. In *Proceedings of the 2017 Internet Measurement Conference* (London United Kingdom, 2017-11). ACM, 477–488. https://doi.org/10.1145/3131365.3131371

[5] Remi Hendriks. 2023. Hendriks, R. (Remi, student M-CS) / manycast-extended · GITLAB. https://gitlab.utwente.nl/s1978047/manycast-extended

[6] Akmal Khan, Hyun-chul Kim, Taekyoung Kwon, and Yanghee Choi. 2013. A Comparative Study on IP Prefixes and Their Origin Ases in BGP and the IRR. *SIGCOMM Comput. Commun. Rev.* 43, 3 (jul 2013), 16–24. https://doi.org/10.1145/2500098.2500101

[7] Qi Li, Mingwei Xu, Jianping Wu, Patrick P. C. Lee, and Dah Ming Chiu. 2011. Toward a practical approach for BGP stability with root cause check. 71, 8 (2011), 1098–1110. https://doi.org/10.1016/j.jpdc.2011.04.009

[8] Tony Li, Ravi Chandra, and Paul S. Traina. 1996. BGP Communities Attribute. RFC 1997. https://doi.org/10.17487/RFC1997

[9] Yakov Rekhter, Tony Li, and Susan Hares. 2006. *A border gateway protocol 4 (BGP-4).* Technical Report.

[10] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. 2002. BGP routing stability of popular destinations. In *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment - IMW '02* (Marseille, France, 2002). ACM Press, 197. https://doi.org/10.1145/637201.637232

[11] Kyle Schomp and Rami Al-Dalky. 2020. Partitioning the Internet Using Anycast Catchments. *SIGCOMM Comput. Commun. Rev.* 50, 4 (oct 2020), 3–9. https://doi.org/10.1145/3431832.3431834

[12] Pavlos Sermpezis and Vasileios Kotronis. 2019. Inferring Catchment in Internet Routing. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 2, Article 30 (jun 2019), 31 pages. https://doi.org/10.1145/3341617.3326145

[13] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, KC Claffy, and Anna Sperotto. 2020. MAnycast2. In *Proceedings of the ACM Internet Measurement Conference.* ACM. https://doi.org/10.1145/3419394.3423646

[14] Yi Wang, Michael Schapira, and Jennifer Rexford. 2009. Neighbor-specific BGP: more flexible routing policies while improving global stability. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems* (New York, NY, USA, 2009-06-15) *(SIGMETRICS '09).* Association for Computing Machinery, 217–228. https://doi.org/10.1145/1555349.1555375

[15] M.A. Wicaksana. 2016. IPv4 vs IPv6 Anycast Catchment: a Root DNS Study. http://essay.utwente.nl/70921/

[16] Yan Yang, Xingang Shi, Qiang Ma, Yahui Li, Xia Yin, and Zhiliang Wang. 2022. Path stability in partially deployed secure BGP routing. 206 (2022), 108762. https://doi.org/10.1016/j.comnet.2022.108762

A FIGURES

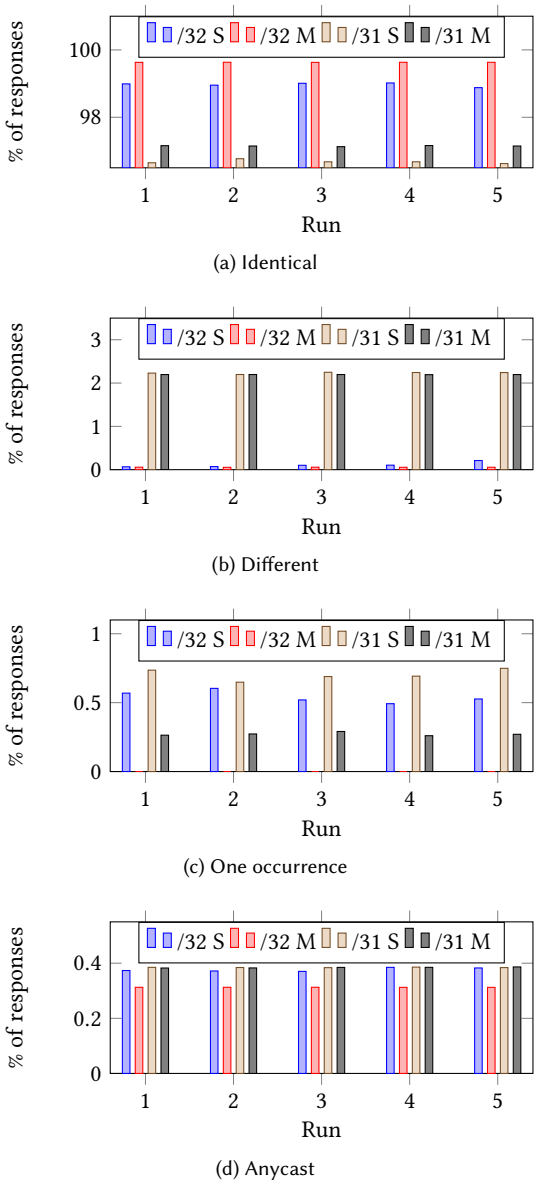
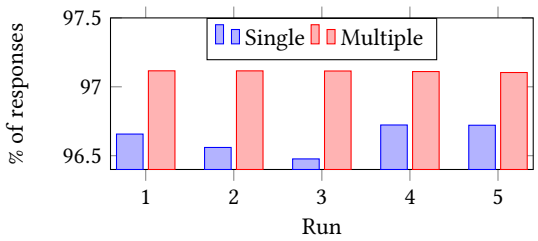
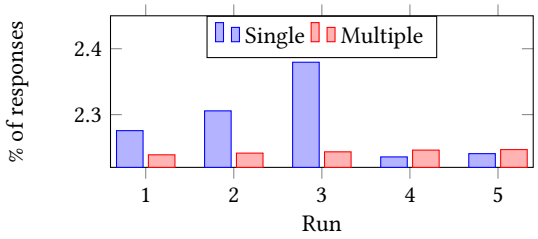


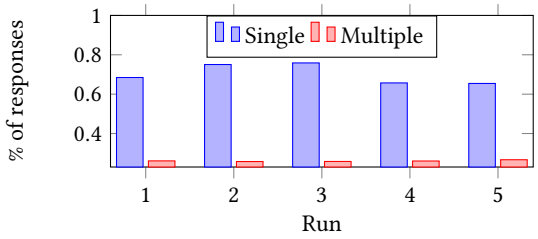
Fig. 1. Comparing visibility between source addresses in the same /31 prefix. Measurements marked with /x use the the same /x IP prefix, and S means that the measurement used only a single source address (so the comparison was done by running the measurement twice and comparing the results) while measurements marked with M used multiple source addresses. The run number is noted down because the test was repeated five times, and it shows which test run the result was a part of.



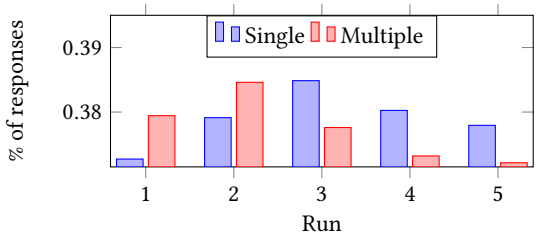
(a) Identical



(b) Different

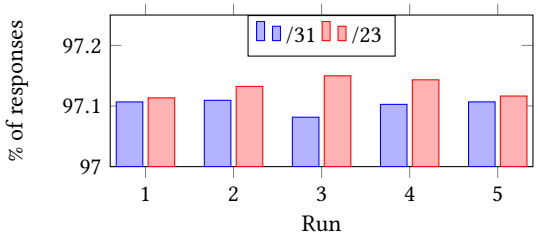


(c) One occurrence

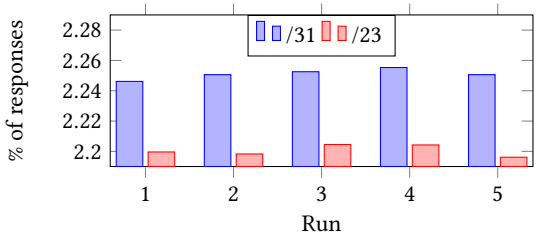


(d) Anycast

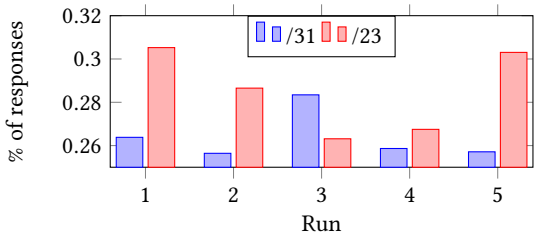
Fig. 2. Comparing the effect of time on catchment overlap on different source addresses in the same /23 IP prefix. S means that the measurement used only a single source address (so the comparison was done by running the measurement twice and comparing the results) while measurements marked with M used multiple source addresses. The run number is noted down because the test was repeated five times, and it shows which test run the result was a part of.



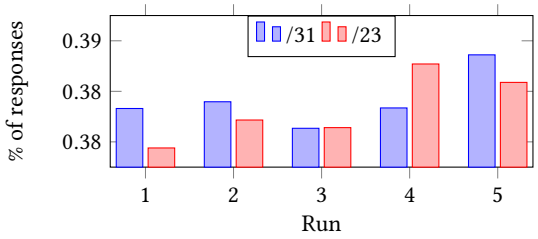
(a) Identical



(b) Different



(c) One occurrence



(d) Anycast

Fig. 3. Comparing visibility between source addresses in different /32 and /24 prefixes. Measurements marked with /x use the the same /x IP prefix. The run number is noted down because the test was repeated five times, and it shows which test run the result was a part of.

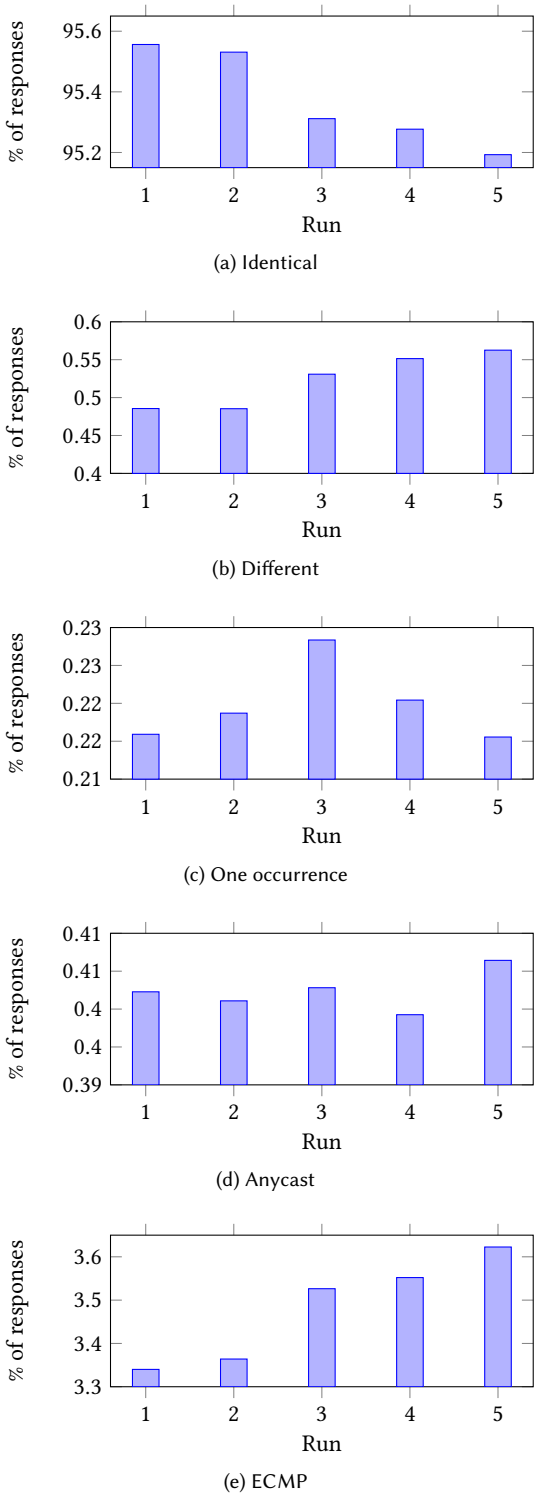


Fig. 4. Comparing visibility between source addresses in different /24 prefixes with ECMP classification. The run number is noted down because the test was repeated five times, and it shows which test run the result was a part of.

B MODIFICATION OF MANYCAST-EXTENDED

We modified MAnycast-extended to support multiple source addresses, allowing measurements to measure all source addresses in a single test. We removed the option to set a client source address. Custom source addresses for clients could be added back in but were not needed for our tests. We tried to minimize the amount of code we changed because it is easy to get overly ambitious while making simple modifications. Therefore we tried to limit ourselves to changing code only for good reasons: (i) to add needed functionality, (ii) to make code more readable and safer, and (iii) to save time. Most of our changes were to make MAnycast-extended accept multiple source addresses and to ensure we send requests to the same VpVPs with differing source addresses without delay.

After these changes, we started running experiments and ran into an issue where the CLI process would get killed as the test progressed. We later discovered that the program crashed because the system the command line interface (CLI) ran on ran out of memory. After reading more of the code for the CLI, we discovered that all test results are kept in memory until the test completes, and then the results are flushed to disk. So if we have all PoPs sending requests and multiple requests for different source addresses, we fill up the memory and the program crashes. We have three possible solutions for this problem that fit the scope of this paper. We can decrease the number of results we receive, flush everything to disk earlier, or increase the memory available to the CLI. We discuss these options and our findings when trying them out in the appendix C. From these findings, we concluded that it would be best for this study to use the CLI on a machine with more resources available, which is what we did for the rest of the experiments.

C COMPARING APPROACHES TO PREVENTING THE CLI FROM CRASHING

In this part of the appendix, we try to find the best way to avoid our CLI running out of memory.

C.1 Decreasing amount of results

One thing we could do to decrease the number of results is to send requests from a single client. This approach's most substantial side effect would be that it would become harder to filter out Anycast networks from our results using the method described by MAnycast^2 [13]. When a VpVP replies to two different PoPs, we cannot determine whether this was due to the VpVP being part of an Anycast network or because the VpVP responded differently to the individual measurements. Therefore, we have to rely more on an external dataset.

To test whether these datasets are accurate, we ran a test using the base version of MAnycast-extended and looked at the results. The test sent requests from all clients, so multiple responses are expected per IP address. Depending on the results, we suspect some IP addresses that replied to be hosting an Anycast network. If an IP address we pinged replied to multiple of our PoPs, we presume this IP address hosts an Anycast network. We expect this because we suspect our clients' requests reached various PoPs of the VpVP, which replied to different PoPs on the TANGLED testbed. We can then see the overlap between a dataset of Anycast networks [13]

Replies	In dataset	Suspected Anycast	Suspected not in dataset	In dataset not suspected
2629865	0.2967	0.3412	25.0209	8.7072
2629277	0.2968	0.3415	25.0244	8.6896
2629008	0.2968	0.3405	24.6623	8.6757
2628924	0.2969	0.3422	25.2750	8.6896
2628549	0.2970	0.3418	25.1079	8.7035

Table 1. A comparison of the MAnycast^2 dataset to our found Anycast networks using the MAnycast^2 method.

Not filtered	Dataset filtered	Suspect filtered	Both filtered
99.3317	99.6393	99.6479	99.6480
99.3035	99.6442	99.6538	99.6538
99.3036	99.6542	99.6658	99.6659
99.3226	99.6390	99.6479	99.6479

Table 2. Catchment overlap using different filtering methods when using probes from multiple clients.

Not filtered	Dataset filtered	Suspect filtered	Both filtered
97.6462	98.7727	98.7763	98.7729
97.6171	98.7437	98.7473	98.7439
97.7655	98.9248	98.9277	98.9250
98.0539	99.0469	99.0500	99.0475

Table 3. Catchment overlap using different filtering methods when using probes from a single client.

and our found data. We have put the results of 5 tests into a table (See Table 1).

As can be seen, around a quarter of all the IP addresses we suspect of being part of an Anycast network are missing from the dataset. This high inaccuracy makes sense because the dataset is six months old and partially outdated. It does present a problem for us, however, as we can show by reanalyzing our results. We want to show the effect of only using a single probe compared to using multiple, so Tables 2 and 3 show the difference in catchment overlap. In this data, our accuracy drops from more than 99.6% when using filters on the dataset from multiple probes to often less than 99% when using filters and using only a single PoP to send requests. We could use a single probe to collect data to prevent the CLI from crashing, but it would add substantial noise to the data.

C.2 Flushing everything to disk sooner

Another approach could be to flush the results to disk before the program runs out of memory. This approach does require us to modify the program further, but it would allow us to run tests without considering the number of results we collect. We will, however, need to check that we do not miss packets while the program is writing to disk. To do so, we will compare our modified version without directly writing to disk with one that does immediately write to disk. We need to utilize a smaller IP list or fewer probing clients for this test. Otherwise, the version of our program that

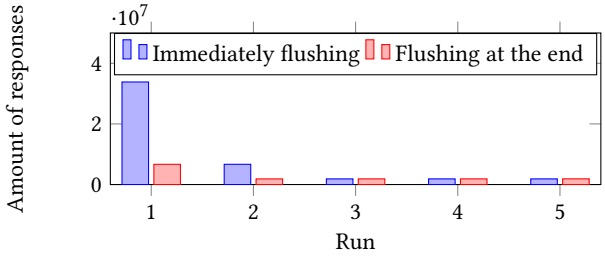


Fig. 5. Comparing amount of results when immediately flushing to disk or waiting until the end of the experiment. The run number is noted down because the test was repeated five times, and it shows which test run the result was a part of.

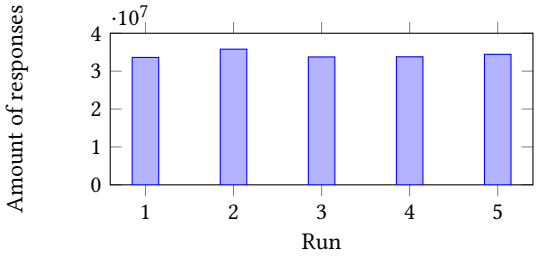


Fig. 6. Responses when running locally outside of the testbed. The run number is noted down because the test was repeated five times, and it shows which test run the result was a part of.

does not write to disk will crash. We ran a test where we ran the modified version that flushed to disk immediately, followed by one that kept all results in memory. Then, we repeated this five times and plotted the results in Figure 5. We can see that, for some reason, the number of results seems to decrease over time. Initially, we got 33 million responses. After the first run, it dropped to 6.6 million with the second and third runs and to 1.8 million responses from the fourth run onward. Looking at the number of responses after they have stabilized, we can see no difference between flushing to disk immediately or waiting.

C.3 Expanding the resources available to the CLI

The easiest way to expand the resources available to the CLI is to run the CLI on another machine. Running the CLI on another machine will always be less stable because it introduces another point of failure. We will also need to check if the extra latency to the CLI causes it to skip responses, as we did when testing the CLI that immediately flushes to disk. We cannot easily script a test where we switch between running the CLI on TANGLED or locally, so we will run it multiple times locally and compare the number of results to previous tests (which happened less than 8 hours before, but the clients and server were restarted in between tests).

As can be seen from the results plotted in Figure 6, this time, we don't see a decrease in the number of responses as we did before, and the amount of replies remains high. To verify these results, we ran more tests. We started by repeating the test on the machine with more resources. After this, we tried to run the same test on

the testbed itself. When we ran it on the testbed, however, it kept killing the CLI, which indicates that the number of results became too great to keep in memory. To resolve the crashes, we resorted to flushing everything immediately to disk, which revealed a similar pattern as before of the number of results decreasing over time, this time even more quickly. Afterward, we ran the CLI on the testbed without immediately flushing and still received few results. Finally, we ran the CLI locally with more resources available without immediately flushing and got similar results to the testbed. Baffled by these results, we checked the Verfploeter clients. Then we saw many of them had crashed, explaining the fewer results we received over time. The error messages indicated that the program crashed because some Tasks were missing data. The data was seemingly only missing when running the CLI and immediately flushing to disk. A hypothesis could be that the CLI or server might receive results from some Clients before it finished sending the start packets to the Clients and that the extra delay of immediately flushing to disk corrupts such packets. This theory would require further code digging or testing to confirm. Due to limited time constraints, this will be left as future work if there is interest. Therefore, for this study, we will use the CLI without immediately flushing to disk on a machine with more resources than the TANGLED control node.