Characterizing anycast convergence time for AS path prepends

ELIMALKO SAADO, University of Twente, The Netherlands

On the internet, IP addresses are used for identification of machines like servers and routers. These IP addresses are typically announced at a single physical Point of Presence (PoP). However, these announcements can be made at multiple PoPs, causing traffic to route to the closest PoP. This practice is called anycasting. Anycast is often utilised by Domain Name System (DNS) servers or Content Delivery Networks (CDNs), since it allows for load balancing a service, higher availability, and lower latency for users of the service. Each PoP will receive a certain slice of traffic depending on its proximity, roughly its path length, to other networks. This slice of traffic is called its 'catchment'. Traffic can be steered to different PoPs using "path prepending". That is a practice in which a PoP announces its IP addresses with an artificially elongated path, causing the PoP to be less preferable to other networks, since its path is longer, this can be used to reduce load to that specific PoP, for example, in the case of a (D)DoS attack. In this paper, we will analyze how long it takes for a path prepend update to reach all other networks (e.g. for the prepend to converge) across the internet. It is useful to know how fast an update converges, for example in the context of (D)DoS attacks, to know how fast a measure can be deployed and how fast after deploying it will work.

Additional Key Words and Phrases: anycast, BGP, internet, catchment, verfploeter, prepend, convergence

1 INTRODUCTION

The Internet is a collection of smaller networks that are connected to each other. Each of these networks is called an autonomous system (AS). These ASes are connected in such a way that IP packets originating from one AS is routed to the destination AS, possibly through other ASes. Each machine in an autonomous system is identified using an IP address. These IP addresses belong to a certain range of IP addresses. An autonomous system can announce the fact that it is the destination for such a range using the Border Gateway Protocol (BGP) [4]. This is done by routers sitting at the border of these autonomous systems. They connect and announce routing information and routing changes to each other. The other border routers a border router is connected to are called its 'peers'. Since not every router is connected to every other router, these announcements propagate through different ASes, through the whole internet. When an update occurs, for some time, increased latency and packet loss may occur, until the routes converge to a stable state [2]. These announcements also contain the 'path' that packets from other routers need to 'walk' to reach the destination router. When an announcement propagates through a router, that router adds itself to the path. For example, a router A can announce itself to B. when B forwards that announcement to C, that announcement now has the path 'A, B', so that C knows it can reach A through B. Routers also use this path to know the 'distance' to a certain IP address range, and they can use this to determine if they should route the packets through one peer or another, depending on which peer has the shortest path to the destination. A router can also artificially elongate the path, this can be used to make other routers de-preference that announcement. The same IP address range can be announced by multiple routers in multiple physical locations (often referred to as Points of Presence, PoPs), this practice is called anycasting. Multiple announcements with different paths will reach all border routers, and every router will take the shortest, or cheapest depending on configuration, path. This can be used to reduce latency for worldwide customers, or to distribute large amounts of traffic across multiple routers to reduce load. The amount of traffic a certain PoP 'catches' is called its catchment. Single PoPs can elongate the path they announce to make other routers, as mentioned earlier, de-preference that announcement. This can be used to 'steer' traffic to other (nearby) PoPs. The announcement to other routers that the path is now "longer" has to be propagated as well. As such, it is important to know how long it takes before all other border routers receive this update, because of the aforementioned reasons, but also for operators to know if the applied traffic steering should be having an effect yet, or if it is time to implement more measures. In this paper, I will analyze the convergence time of these path updating announcements. I will also analyze the impact of path prepending on the amount of traffic a PoP receives.

This yields the following research questions with sub-questions:

RQ1: How do we measure convergence

(1) How do we systematically perform prepends at each anycast site?
(2) How do we measure whether the prepend has been received by different routers?
(3) How do we measure the number of prepends needed to significantly alter catchment?
(4) What do we classify as *full* convergence?

RQ2: How long does convergence take with path prepending updates

(1) What is the relation between prepends and catchment?
(2) How many prepends are needed to significantly alter catchment?
(3) How long does it take for the update to reach 50% convergence?
(4) How long does it take for the update to reach 95% convergence?
(5) How long does it take for the update to reach full convergence?
(6) Does the amount of prepends influence the convergence time?
(7) Does convergence time differ among sites?

The questions 1.1, 1.2, 1.3, and 1.4 will be answered thoroughly in the methodology section. The questions 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, and 2.7 will be answered in the results section.

In short, we use a tool called Verfploeter (De Vries et al. [1]) in combination with an anycast testbed to perform active measurements. It can give insight into how much a certain anycast site

TScIT 42, January 31, 2025, Enschede, The Netherlands

is preferred by other networks. We assume, and confirmed that prepending a certain site will lower its catchment. In turn, we performed prepends during measurements, and we could see a clear drop in catchment received by the site after the prepend update was sent.

We hypothesize that the more prepends a site has, the lower its catchment will become. We also hypothesize that prepends will not have an influence on the convergence time, but we do believe there will be a significant difference in convergence times between sites.

2 RELATED WORK

In this section we will discuss the literature this paper builds upon.

This paper is an extension of Degen et al. [2], differing in the fact that (1) our focus is on path prepends, which is different from the prefix announcements and withdraws in the paper, although prepending the path to a prefix can be seen as a prefix announcement as well. (2) We also use a different methodology which allows for a higher resolution when measuring, allowing us to make claims in the "below 5 seconds" range instead of the "below 10 seconds" range concluded in Degen et al. [2].

We make use of Verfploeter for our active measurements, introduced by De Vries et al. [1]. It differs from other tools like RIPE Atlas by running on anycast infrastructure itself, allowing for measurement of anycast performance without requiring extensive infrastructure or data scraping. Verfploeter uses a list of live IP addresses, it knows these addresses will respond to e.g. ICMP ECHO packets. Verfploeter sends these pings from anycast sites, and keeps track of where the replies are received, the proportion of received replies (per anycast site) is called the *catchment*.

3 METHODOLOGY

3.1 How do we systematically perform prepends at each anycast site?

We announce two /24 IPv4 prefixes at every site, of which, generally, one /24 is used for the control measurements and the other /24 is used for experimental measurements. For IPv6 we announce two /41 IPv6 prefixes, the rest of the setup is identical.

The prepend updates are sent using ExaBGP, this tool sends out BGP updates to our (single) upstream router (which in turn sends them to its peers until it propagates to all border routers that receive this route). These prepends are sent from a single site. We combine ExaBGP and Verfploeter into scripts to automatically perform prepends and then analyze the catchment per site.

3.2 How will we measure whether the prepend has been received by different routers?

As shortly mentioned in the introduction, an active measurement setup is used. In terms of tooling, we use Verfploeter is used to do the active measurements. It uses a list of IP addresses that respond to, e.g. ICMP ECHO requests. Prepends are performed on an anycast testbed that consists of 32 geographically unique anycast sites. Verfploeter sends pings from every site, and keeps track of where they are received. Since these sites are anycast (e.g. they all share the same IP prefix), networks sending the reply will route to a certain preferred site, that could be geographically or economically preferred. This generates a good view of what sites are preferred by what network. For example, consider a network located in Germany, it will probably always send its reply to the site located in Frankfurt, Germany. In any given environment, every site will receive a certain proportion of traffic, this is called its *catchment*. We assume that prepending a site will de-preference it among some networks, causing it to have a lower catchment.

To measure whether a prepending a site made it less preferable to other networks, we perform control measurements without prepends, and during and/or after the control measurement, we perform the prepends and do measurements on the experiment prefix. We can then extract the amount of catchment received by every site after the prepends.

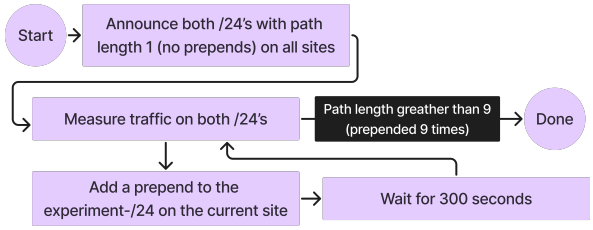

Fig. 1. Prepending algorithm used to determine changes in catchment from prepends and how many prepends are needed to alter catchment

3.3 How do we measure how many prepends we need to meaningfully alter catchment?

To measure optimally, we want to have as much change as catchment as possible. Thus, we want to know how many prepends we need to push away as much catchment as we can. To measure this, we wrote a script that prepends a site one prepend at a time. It waits for at least 5 minutes to allow for full convergence between prepends, we assume 5 minutes is enough, as found by Degen et al. [2]. After the delay, we measure the catchment for all sites using Verfploeter with a hitlist consisting of 5.8 million IP addresses (we use the ISI hitlist combined with OpenIntel live addresses). The algorithm stops after 9 prepends have been done. This is because we see no significant changes after 9 prepends, and we want to limit the burden of measurement on the internet. For a full overview of the algorithm, see Figure 1. The algorithm was repeated for every anycast site.

3.4 What do we classify as *full* convergence?

In our analysis, we flag three distinct points relating to convergence. The first point is when convergences reaches 50%, as it marks a significant halfway point towards convergence. The second point is when convergence reaches 95%, we believe this marks "practical" convergence, e.g. the update has converged for most practical use cases, like load balancing measures. The third point is when full convergence is reached, however, because we have typically observed a lot of noise in our data, we flag > 99% convergence as *full* convergence.

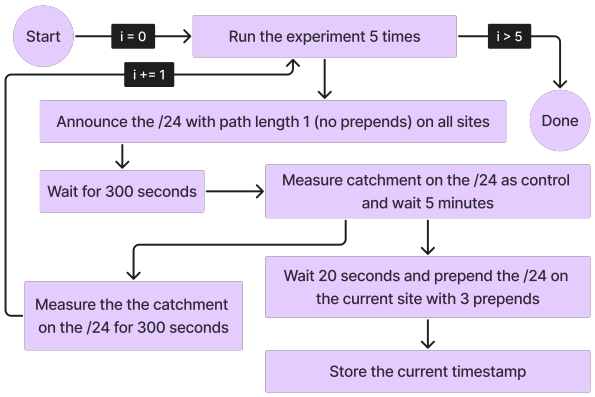

Fig. 2. New convergence measurement algorithm

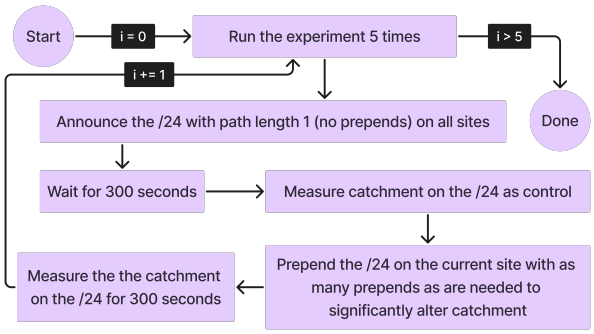

Fig. 3. Original convergence measurement algorithm

3.5 How do we measure convergence?

Thus, to measure convergence, we measure how fast our catchment drops and then stays stable at that lower level. Measurements were done using the approach described by the flowchart in Figure 2. A repeated Verfploeter measurement is started, and after some delay D ($20 \leq D \leq 30$), a single anycast site is prepended with 3 (three) prepends. We assume the amount of prepends does not alter convergence time, and will prove so in the Results section. The repeated Verfploeter measurements continuously measures the hitlist for 5 minutes, thus, with at least 270 seconds of measurement *after* the prepend occurs. We do this by measuring the catchment with 5 second intervals. To get a high resolution of convergence, we want to measure fast. However, a high sending rate could not be used. Instead, a reduced and partitioned hitlist is used. The hitlist is partitioned using the CAIDA prefix2as dataset, if we see a large prefix is always routed the same (e.g. a certain site always catches the whole prefix), only a singular address from the prefix is used. This creates an even slimmer hitlist, while still conserving a fair distribution. This approach comes from Schomp and Al-Dalky [3]. The reduced hitlist is then shuffled and divided into 5. We run the experiment five times per site, and every run gets its corresponding slice of the hitlist. This reduces the hitlist size from 5.8 million addresses to about 841.000 addresses. This algorithm is repeated for every site.

To measure whether the amount of prepends causes significant differences in convergence time, we ran the algorithm described above for prepends of length 1, 2 and 3 on six distinct sites.

To measure convergence from this catchment data, we plot the cumulative proportions for every site, taking the catchment for that site of the control measurement as 0% convergence. We then take maximum amount of catchment, for that site, in our experiment measurement, after the timestamp of the prepend, as 100% convergence. Be aware that 100% convergence is never possible in five minutes as routes could become stuck or otherwise never update in some border routers worldwide, though, practically we know that measuring for 300 seconds is enough to reach very close to 100% convergence, as per Degen et al. [2].

3.6 Original approach

One might be left wondering why the results from "How many prepends do we need to meaningfully alter catchment?" were not used in the measurements. This is because during the analysis of the results of our original convergence measurement algorithm, we had found multiple issues. The first issue is that the catchment measurements do not start fast enough, as there is latency in parsing the hitlist and sending the first pings. This led us to another issue. The second issue we encountered is that our upstream did not accept route updates that went from 0 prepends to 4 or more prepends in one go. This means that other routers never receive our update. The prepends will have to go incrementally like in the algorithm visualized in Figure 1. We used 4 or more prepends as this was the amount of prepends needed to push away most of the traffic for certain sites.

For consistency sake, the original algorithm is visualized in Figure 3.

4 RESULTS

The following section presents an analysis of the catchment data collected by the two algorithms. First we will present the relation between catchment and the amount of prepends. Then, we will present the analysis of how many prepends we need to push away catchment. Lastly, we will present the analysis of convergence time.

4.1 Prepends and catchment relation

We observed a clear relation between the amount of prepends and the catchment of a site. A larger number of prepends to a site directly correlates with a lower catchment. Prepends done on the site DE-FRA are visualized in Figure 4. Do note that, for clarity purposes, the plot in this figure does not show sites that are not affected by the prepend, only the prepended site (DE-FRA) and sites that significantly increase in catchment are shown. Key observations are that (1) catchment is spread out, when DE-FRA is prepended, networks move preference other sites. (2) Catchment does not reach zero, no matter the amount of prepends and (3) after 2 prepends there are no significant changes in catchment for this site. Lastly, (4) we see an increase in traffic to the site IN-BOM, which is geographically

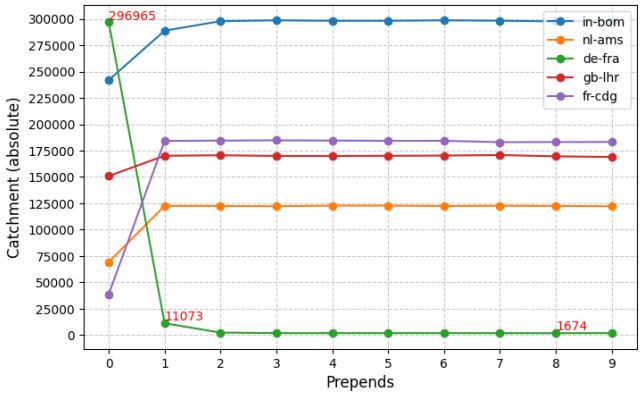

Fig. 4. Prepend catchment relation, prepends performed on DE-FRA

distant from DE-FRA. Possible explanations for these observations are mentioned in the discussion section.

We do observe outliers, a clear outlier is the site IL-TLV. This site does not receive a lot of catchment to begin with, and our prepend measurements only measure noise, we see catchment variations of 1% per prepend measurement, while on DE-FRA catchment was reduced by over 98% after 2 prepends. All observed changes on IL-TLV are negligible and non-significant when normalized. We see this behavior (though, to a lesser extent) on other sites like US-HNL and ZA-JNB (among others), these sites do respond to path prepends but the change in catchment is not as significant as other sites. In the case of US-HNL, the catchment is only 5.5% lower after prepending, no matter the amount of prepends, though the change is measurable unlike on IL-TLV.

We performed these measurements on our IPv6 prefixes as well, using the same set-up. We observed the same relation. We also observed more of the outliers as mentioned in the paragraph above. We observe this on sites that have a very low catchment to begin with, and there are more of these low catchment sites on IPv6 than on IPv4.

4.2 How many prepends are needed to significantly alter catchment

We have found that on most sites, a single prepend would cause at least an 80% drop in catchment, an example of this is shown in Figure 6 where we show the normalized (0-100) catchment of the site JP-NRT on IPv4. On a most sites, 3 prepends is enough to reach a stable amount of catchment, however, on a handful of sites, more prepends are needed. As before, an example of this is visualised in Figure 5 where the site JP-NRT is prepended on IPv4. As mentioned above, pushing away all catchment is not always possible as seen in Figure 4 and Figure 5. The amount of prepends needed to push away as much catchment as possible is shown in Table 1. This is both for IPv4 and IPv6. Sites, like IL-TLV, whose catchment stay consistent despite prepends are marked as "N.A.". Observations are that (1) we see there is a difference in prepends needed per site, and (2) we see that there is a difference between the number of prepends that are needed on the same site between IPv4 and IPv6.

Site	Needed prepends IPv4	Needed prepends IPv6
pl-waw	1	1
us-sjc	1	1
us-lax	1	1
in-del	1	1
us-dfw	1	2
au-mel	1	3
se-sto	1	3
cl-scl	1	3
us-hnl	1	N.A.
in-bom	2	1
kr-icn	2	1
au-syd	2	2
us-ewr	2	2
gb-lhr	2	2
br-sao	2	2
za-jnb	2	3
us-mia	2	3
us-sea	2	3
de-fra	2	3
sg-sgp	2	3
jp-itm	2	4
us-ord	3	1
mx-mex	3	1
in-blr	3	2
nl-ams	3	3
us-atl	3	3
es-mad	3	3
ca-yto	3	3
fr-cdg	4	3
gb-man	4	N.A.
jp-nrt	5	3
il-tlv	N.A.	N.A.

Table 1. Table showing the amount of prepends needed to push away as much catchment as possible

4.3 Convergence results

Do note that due to time limitations, convergence analysis has only been done for IPv4.

We announced our experiment prefix with 3 prepends at once on every site, one site at a time. 3 prepends were used because this was the maximum amount of prepends that could be done at once. We normalize the values of the catchment data we collect, by segmenting it in segments of 5 seconds. We take the segment with the maximum amount of catchment we receive on the site as "0% convergence", this is always a segment before the prepend. We then take the segment with the minimum amount of catchment we receive as "100% convergence", this is always a segment after the prepend occurs. We plot a step graph for every site, with a purple vertical marking at $t = 0$, this is the timestamp where the prepend occurs. We mark three key points, 50% convergence, 95% convergence and full convergence. We observe that sites can be grouped into three categories. (1) The fast convergers, these are

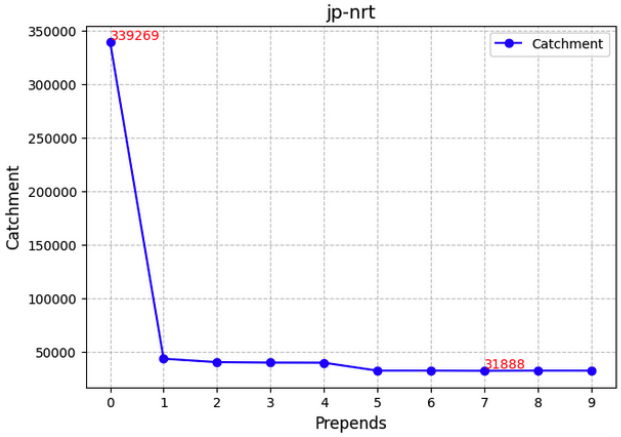

Fig. 5. Prepend catchment relation, prepends performed on JP-NRT

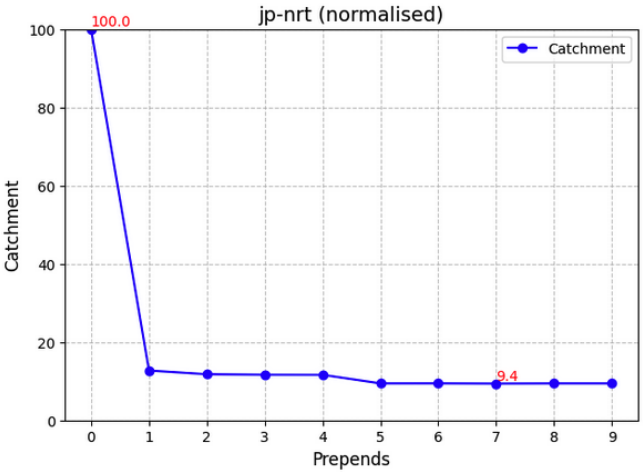

Fig. 6. Prepend catchment relation, prepends performed on JP-NRT, normalized plot

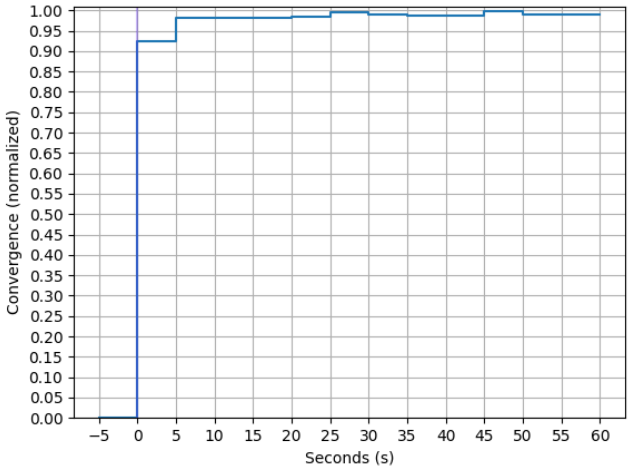

Fig. 7. Step plot describing the convergence time on DE-FRA, a fast converger

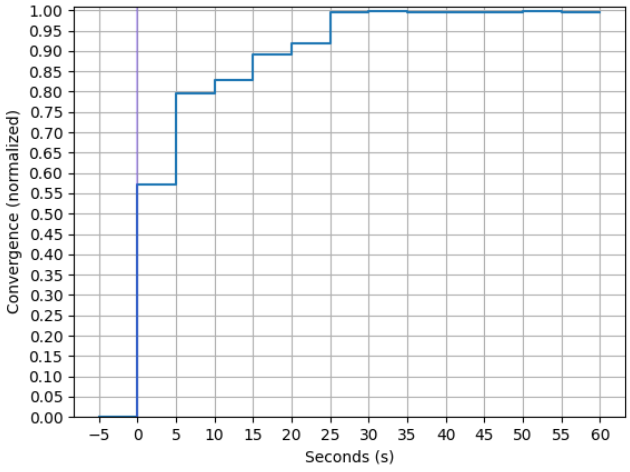

Fig. 8. Step plot describing the convergence time on US-EWR, a slow converger

sites that converge fast, ≥ 95% within 10 seconds, an example of this is the site DE-FRA as seen in the convergence step plot in Figure 7. (2) The slower convergers, these are sites that take >20 seconds to converge to 95%. There are only a handful of these sites, with an example of this being shown in Figure 8. These sites converge slowly but their convergence can still be accurately measured. (3) The noisy sites, these are sites that seem to converge fast or slow, but that are too noisy to accurately measure. An example of a noisy site and the noise can be seen in Figure 9, where the convergence for the site ZA-JNB is plotted. The convergence "bounces" after the prepend. We observe that these are often (though not always) sites that do not respond well to path prepends, like we mentioned earlier for IL-TLV and US-HNL, where catchment is not affected or only slightly altered with prepends. From looking at the plot, we established the noisy sites are ZA-JNB, ES-MAD, US-SEA, MX-MEX, US-HNL, AU-MEL, CL-SCL, IL-TLV, GB-MAN, and BR-SAO.

We observe that all sites except one reach a 50% convergence within 5 seconds. The outlier is the site US-SEA, this is a noisy site. Although it is not a key point we mark, we observe that >80% convergence is reached within 5 seconds on most sites as well. In Figure 8 the site US-EWR is shown, which is one of the slowest to converge among the non-noisy sites. We see that it reaches a convergence of over a majority of networks within 5 seconds. We do not reference the site GB-LHR here as its convergence is too slow to show in a concise plot.

We do not observe significant differences for the 50% convergence mark between sites. However, we do observe significant differences for the 95% and full convergence marks between sites. A plot visualizing the convergence times per site for 50%, 95% and full convergence

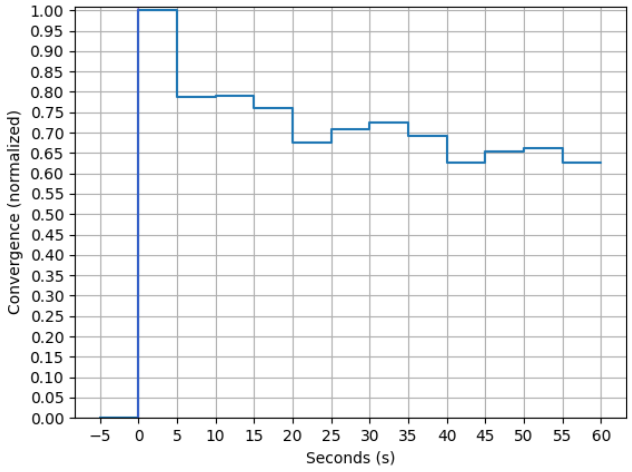

Fig. 9. Step plot describing the convergence time on ZA-JNB, a noisy site

is shown in Figure 10, 11, and 12 respectively. An interesting observation is that the site IN-BOM has a very high convergence time in both plots. This could be due to measurement or analysis error. Since we run each site 5 times, we can analyze separate runs. Analyzing each run individually we find this site has a full convergence time of at most 15 seconds, so we attribute this outlier to measurement and/or analysis error.

	Including outliers		Excluding outliers	
	Mean	Median	Mean	Median
50% convergence	5.3	5.0	5.0	5.0
95% convergence	22.8	10.0	16.19	10.0
full convergence	38.1	20.0	30.0	20.0

Table 2. Table of results of our convergence analysis

We find a mean time of 5.3s for 50% convergence, 22.8s for 95% convergence, and a mean of 38.1s for full convergence across all sites including all outliers. We also find median times of 5.0s for 50% convergence, 10.0s for 95% convergence, and 20s for full convergence.

If we remove the outliers, which are the sites IN-BOM and the noisy sites, we find a mean of 5.0s, 16.19s, and 30.0s for 50%, 95%, and full convergence respectively, and we find medians of 5.0s, 10.0s, and 20.0s for 50%, 95%, and full convergence respectively. These results are summarized in Table 2.

4.3.1 Effect caused by the amount of prepends. We performed 1, 2 and 3 prepends on six (6) distinct sites to test if there is a significant difference in convergence time caused by the amount of prepends. We observed no difference in 50% convergence times, all sites with all prepend counts had a 50% convergence within 5 seconds. The data is not normally distributed, so an ANOVA could not be used. A Kruskal-Wallis Test was used and found, with $p_{95} = 0.290$ and a $p_{full} = 0.308$, that our the convergence times do **not** significantly vary between prepend counts. A boxplot showing the convergence

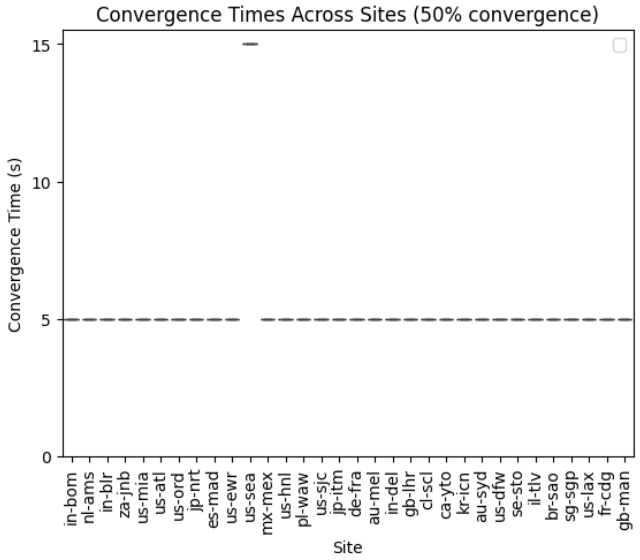

Fig. 10. Plot visualizing convergence time per site for 50% convergence

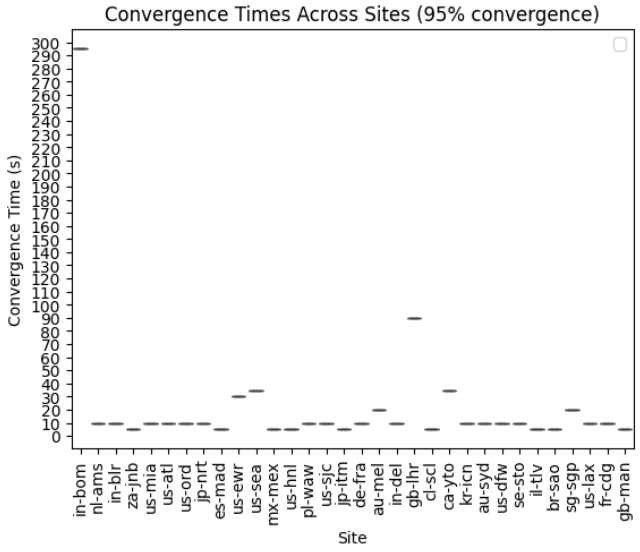

Fig. 11. Plot visualizing convergence time per site for 95% convergence

times between prepends is shown in Figure 13 does however show significant differences.

4.3.2 Differences between runs. Since we ran each site five (5) times with a 1/5th hitlist, we can compare these results to see if there is a significant difference in convergence time between runs on the same site. The convergence times for these runs are shown in table 3 The 50% and 95% convergence time are the same across all runs. We do see differences in the full convergence times. A Kruskal-Wallis test says, with $p = 0.4060$ that there is **no** significant difference between runs for the full convergence time.

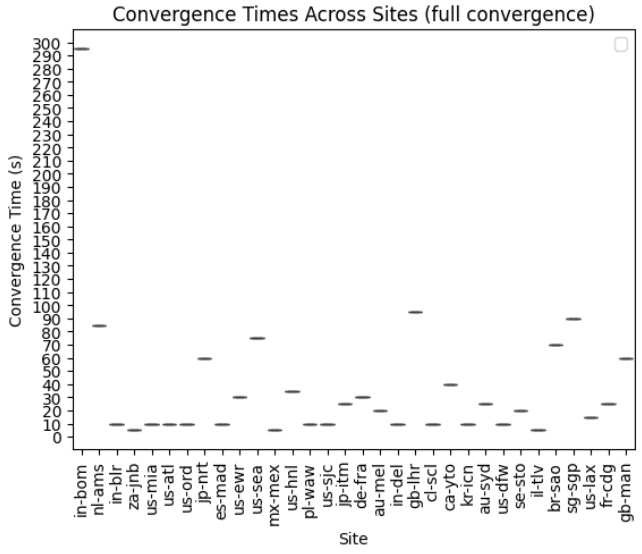

Fig. 12. Plot visualizing convergence time per site for full convergence

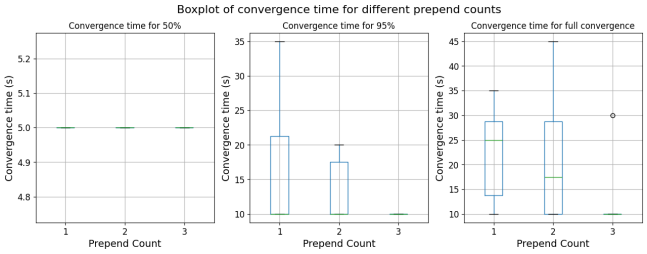

Fig. 13. Boxplot showing convergence times between different prepend counts

run	50% convergence	95% convergence	full convergence
1	5	10	35
2	5	10	30
3	5	10	50
4	5	10	30
5	5	10	20

Table 3. Convergence times for five runs of prepends on DE-FRA

5 DISCUSSION

In this section, we will interpret the results, its implications and talk about the limitations of the research.

5.1 Prepend catchment relation

We have seen that, as hypothesized, the more prepends a site has, the lower its catchment will become. We have also seen that there is a limit to the amount of catchment we can lower using prepends, for example in the case of DE-FRA. We suspect this is due to traffic engineering, like so called local preferences and static routing. These are practices in which a router always routes to, or is much more likely to route to the same other router, no matter the path length or other types of preferences. This is often used to lower operating costs since routing to certain parts of the world can be expensive. We also saw that catchment "spreads out", if the path to a site is longer it will move to other sites. This means that path prepending can successfully be used for load balancing. We did however also see that catchment can also spread out to sites that are not geographically close, for example when DE-FRA was prepended and catchment spread partially to IN-BOM. We believe this happens due to a practice called remote peering, where a network, like an Internet Service Provider, peers with networks it is not geographically close to.

In terms of outliers we now know that sites with low catchment do not respond to prepends as well as sites with higher catchment. This could be attributed to the fact that the these low-catchment sites receive mostly engineered or very local traffic, both of which are not easily pushed away.

5.2 Convergence time

We have seen than on average and without outliers, a path prepending update fully converges within 30 seconds, and that 50% convergence is reached within 5 seconds. We also observe that convergence is reached among a majority of networks within 5 seconds, even among the slow converging sites. This means measures in terms of DDoS mitigation or load balancing using path prepends can be applied and verified quickly, and the effect of these measures should be noticed within 5 seconds.

5.3 Outliers and differences between sites

We observed significant differences in convergence time between sites. And were able to categorize sites into the groups fast convergers, slow convergers and noisy sites. We speculate this can be attributed to two things. (1) Possible measurement errors and noise, for example in the case of the site IN-BOM and the noisy sites mentioned in the results section. We know all noisy sites did not respond well to path prepends. (2) How well connected a site is to other networks, especially those with modern equipment. In the case of DE-FRA, we see fast convergence and believe this can be attributed to its connection at the DE-CIX Frankfurt Internet Exchange (IX).

5.4 Limitations and future work

5.4.1 Effect caused by the amount of prepends and site. We saw possible significant difference between in convergence times for different amounts of prepends. To properly analyze this effect more data is needed. We saw no significant difference in convergence times for the same site, and we saw differences between convergence times between sites. Ideally, these experiments are repeated more often, so proper statistical distributions can be recognized and analysis can be applied. This is a future work recommendation.

5.4.2 Partitioned hitlist. To get a higher resolution without a high probing rate, a partitioned hitlist was used. This could cause bias, since large blocks of addresses that route are only taken as a single

address by design, even though they could account for higher (practical) convergence. An example of this would be if Google makes up 90% of your usual traffic, by partitioning the hitlist we reduce its influence on the convergence to a fraction of its actual value, even though it practically makes up for a big chunk of the traffic. If a partitioned hitlist is used in future experiments, weights should be added to each partition to reflect its original size.

5.4.3 Resolution of measurement. The lowest positive bound of our measurement is 5 seconds, this is because every Verfploeter run takes 5 seconds. In the future, it could be productive to have a better resolution, since we see 50% convergence on all sites and even 90% convergence on most sites within 5 seconds. This could be achieved using a higher probing rate, though this will fill the TX/RX-queue of the network card sending and receiving the probes, leading to packet loss and thus a failed measurement.

5.4.4 IPv6 Convergence. Due to time limitations, the data for IPv6 convergence was not analyzed. The setup for measurement is identical and is easily created from an existing IPv4 measurement setup.

6 CONCLUSIONS

In this work, we analyzed the convergence time for path prepending updates. We developed methodologies that can systematically prepend sites and measure convergence of these prepends. An active measurement setup using an anycast testbed in combination with Verfploeter was used to perform measurements, using ExaBGP to perform path prepends.

We find that on average, 50% convergence takes $\leq$ 5 seconds, 95% convergence takes 16.19 seconds, and full convergence takes 30 seconds. We also see that convergence among a majority of networks is reached within 5 seconds on most sites. Our results show that the number of prepends might impact convergence time and that more research on the topic is needed. Furthermore, we find that convergence time differs between sites. Knowing how long these updates should take is vital for estimating the effectiveness of measures like DDoS mitigation and load balancing, and these results show that 5 seconds convergence time is enough to divert traffic in case of a DDoS attack. However, we recommend using an estimate of 15 seconds to take into account different convergence times among sites and upstream providers.

In the future, we recommend these measurements be repeated more often to enable proper statistical analysis. We also recommend devising an algorithm that allows for measurements with a higher resolution, so more accurate convergence times can be found. Moreover, if a partitioned hitlist is used, we recommend assigning a weight to each prefix to be used in analysis, so a big network of IP addresses has a higher weight on convergence than a smaller network. Lastly, we recommend measuring and analyzing convergence times for IPv6 and comparing them to convergence times on IPv4.

REFERENCES

[1] 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*. ACM, London United Kingdom, 477–488. https://doi.org/10.1145/3131365.3131371

[2] Bernhard Degen, Mattijs Jonker, Roland Van Rijswijk-Deij, and Raffaele Sommese. 2024. An Empirical Characterization of Anycast Convergence Time. In *Proceedings of the Applied Networking Research Workshop on zzz*. ACM, Vancouver AA Canada, 23–30. https://doi.org/10.1145/3673422.3674890

[3] Kyle Schomp and Rami Al-Dalky. 2020. Partitioning the internet using Anycast catchments. *ACM SIGCOMM Computer Communication Review* 50, 4 (Oct. 2020), 3–9. https://doi.org/10.1145/3431832.3431834

[4] Y. Rekhter, T. Li, and S. Hares. 2006. A Border Gateway Protocol 4 (BGP-4).