Improving anycast census at scale

Remi Hendriks
Supervisor: prof.dr.ir. R.M. van Rijswijk - Deij
Supervisor: dr.ir. R. Sommese
Committee: dr. M.H. Evert
University of Twente, Enschede, the Netherlands
Faculty: *Electrical Engineering, Mathematics, and Computer Science (EEMCS)*
Chair: *Design and Analysis of Communication Systems (DACS)*
Contact: r.hendriks@student.utwente.nl

***Abstract*—Anycast, a crucial component of the Internet, enables the sharing of a single IP address among geographically distributed servers, playing a vital role in providing essential services such as the Domain Name System (DNS). The wide adoption of anycast can be attributed to its significant benefits, particularly improved resilience. To better understand the resilience of these essential Internet services, efforts have been made to develop measuring tools capable of performing anycast censuses. One of these, MAnycast^2, uses an innovative approach in which an anycast network is used to perform a census of other anycast networks. While its results were promising, the initial version of MAnycast^2 had limitations. This work improves the MAnycast^2 measuring system, with the focus on facilitating more accurate anycast censuses. The main contributions of the new system include synchronous probing, support for TCP and UDP probing, improved resource efficiency, and service-specific DNS probing. Through multiple MAnycast^2 measurements that validate our approach, we demonstrate that these contributions significantly enhance scalability, coverage, and accuracy in performing anycast censuses.**

***Index Terms*—Anycast, Verfploeter, MAnycast^2, Internet, Measurements, DNS, System design**

I. Introduction

Anycast is the usage of a single address for multiple devices, often geographically distributed. This is widely used by, for example, content providers who want to satisfy client requests from multiple locations to ensure low-latency responses and perform load-balancing by distributing traffic over multiple anycast sites. The most prominent service that often makes use of anycast is the Domain Name System (**DNS**), which translates a domain name to an IP address when an Internet user connects to a website.

For evaluating the utility of individual anycast sites within an anycast architecture, it is important to know the catchments of these sites. The catchment of a site is the set of Internet prefixes whose traffic gets routed to that particular site by the Border Gateway Protocol (**BGP**), which inherently routes packets to the 'nearest' – in terms of BGP routing – anycast site. Mapping these catchments is useful for site capacity management and learning what would happen to the catchment distribution in case of site outages or maintenance. Verfploeter allows for finding these catchments [9], by sending out ICMP/ping echo request probes from an anycast deployment and registering at which sites the replies end up for the various addresses.

Due to its importance in enhancing resilience, anycast use is growing across the Internet. For this reason, the Verfploeter technique has been repurposed for MAnycast^2 [15], where it was used to find anycast prefixes on the Internet. This is achieved by probing targets from multiple Vantage Points (**VPs**) using an anycast source address. In the case of the target being also an anycast network, the probes will end up at multiple anycast sites, and their replies will end up at multiple VPs (the closest VP for each probed anycast site) as can be seen in Figure 1.

This technique is very promising, however it has limitations in its methodology based on the current measuring system. For example, clients are probing sequentially which meant there is a substantial time difference between the times that a probed target will receive the probes from each VP. In this work we intend on resolving these limitations by creating a new measuring system.

Furthermore, we have set of new requirements that will allow for improved MAnycast^2 measurements. Firstly, we want to extend the system with additional TCP and UDP probes to perform MAnycast^2 measurements using these protocols. This will allow us to evaluate the responsiveness of anycast prefixes to these protocols, potentially increasing discoverability as we can now detect hosts that are unresponsive to ICMP/ping but do respond to UDP/TCP. Additionally, it will enable service-

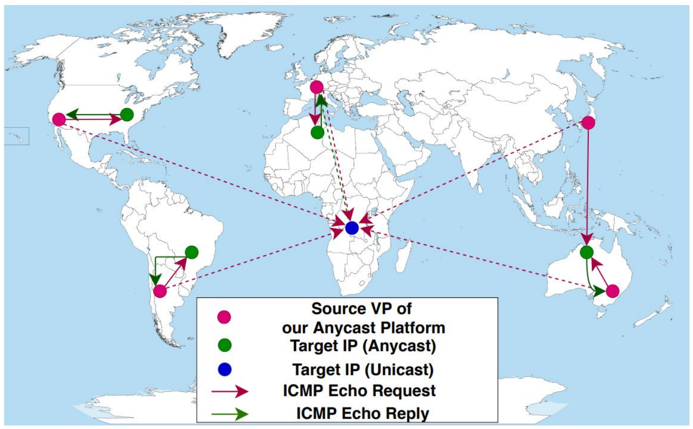

Fig. 1. MAnycast^2 overview from the paper "MAnycast^2 – Using Anycast to Measure Anycast" [15].

specific probes that use TCP/UDP to gather more information about anycast deployments. Secondly, we aim to make the system easy to deploy regardless of the Virtual Machine (**VM**) on which it is hosted, simplifying the deployment process. Thirdly, we want the system to be resource efficient, enabling deployment on VMs with limited hardware. This will make costs more manageable by allowing deployment on ‘cheaper‘ VMs, while also reducing the burdden on these machines during measurements. By achieving this, the measuring system becomes more sustainable and scalable. Finally, we intend to increase the robustness of the system to enable longitudinal data collection, ensuring that the system can run for extended periods of time.

Our research goal is to improve and extend the MAnycast^2 measuring system, such that it can be used to perform more accurate anycast censuses that provide a more detailed view of the anycast landscape on the Internet.

This goal will be solved by answering the following research questions:

1) *What changes are necessary to enhance the measuring system and address the limitations identified in the original MAnycast^2 measurements?*
2) *What additional features need to be implemented to satisfy the additional system requirements?*
3) *To what degree did the implemented changes contribute to the improvement the system?*

To evaluate whether our new system has fulfilled our research goal, and to answer our research questions, we will perform a set of MAnycast^2 measurements to validate the system and quantify the value of the improvements.

II. Background

This section provides the essential background required for understanding the thesis, for a more detailed background see Appendix A.

A. Anycast

As has been measured in the MAnycast^2 paper [15], anycast addresses make up at most 0.45% of the IPv4 range. Whilst such a small percentage may give the impression that anycast is insignificant, Internet users have a 50% chance to make use of an anycast address during daily Internet activities [11]. Anycast is used by content providers, content delivery networks (**CDNs**), cloud providers, DNS operators, operators of top- and second-level domains, and more to provide a large variety of Internet services.

Unlike unicast, which means a host exclusively using a unique IP address, anycast allows for multiple hosts to share a single address. Then, clients who contact such an anycast address get routed by the Border Gateway Protocol (**BGP**) to the nearest (in terms of routing) anycast instance. In this work we will infer whether /24 prefixes are anycast, since that is the smallest prefix that gets propagated by conventional BGP routers.

Anycast is desirable as it allows for geographic distribution of a service such that it provides:

- **Reduced latency** - Anycast enables geographic distribution of a service, resulting in reduced latency for clients by bringing content closer to them.
- **Load-balancing** - Anycast facilitates load-balancing by spreading traffic among multiple anycast sites, thereby distributing the burden and optimizing resource utilization.
- **Enhanced resilience** - Anycast enhances resilience by distributing traffic to multiple anycast sites based on traffic origin, mitigating the impact of attacks and disruptions. By leveraging BGP announcements, anycast can intelligently route traffic away from targeted sites, effectively deflecting Distributed Denial of Service (**DDoS**) attacks. Additionally, anycast architecture does not rely on a single point of failure, meaning that the availability of services remains partially intact even if a portion of the anycast infrastructure is affected by an attack or other issues.
- **Reliability** - Anycast ensures reliability by automatically routing clients to the next-nearest anycast site when an instance goes offline and its BGP announcements are withdrawn.
- **Localization** - Anycast allows manipulation of BGP announcements to limit visibility to select neighboring Autonomous Systems, enabling localization of service delivery.
- **Horizontal scaling** - Anycast provides a cost-effective and efficient method for replicating a service on the Internet, facilitating horizontal scaling.

Additionally, the possibilities and advantages of anycast have been amplified by anycast systems like PIAS and OASIS [1, 4], which allow for more fine-grained server selection mechanisms.

B. Mapping an anycast infrastructure

Since the Internet is quite unpredictable when it comes to routing, it is difficult for anycast operators to fine tune their deployments, e.g., for obtaining good load-balancing for DDoS attack mitigation. Furthermore, Internet routing is dynamic, which causes catchments of anycast sites (i.e., the set of clients who get routed to each anycast site) to be variable [3].

For these reasons, Verfploeter was developed as a state-of-the-art anycast measuring system for obtaining the catchments of an anycast deployment [9]. These catchments, combined with traffic history, can then be used, for example, to predict future load and determine the operational value of an anycast instance.

Unlike other anycast measuring techniques which probe an anycast infrastructure using a set of external vantage points that are controlled by the researcher, Verfploeter makes use of the devices on the public Internet by sending out ICMP requests and analysing the received ICMP responses, as visualized in Figure 2. This allows Verfploeter to utilize millions of devices, rather than a limited amount of vantage points. Furthermore, it provides coverage in areas where obtaining vantage points can be difficult, assuming the distribution

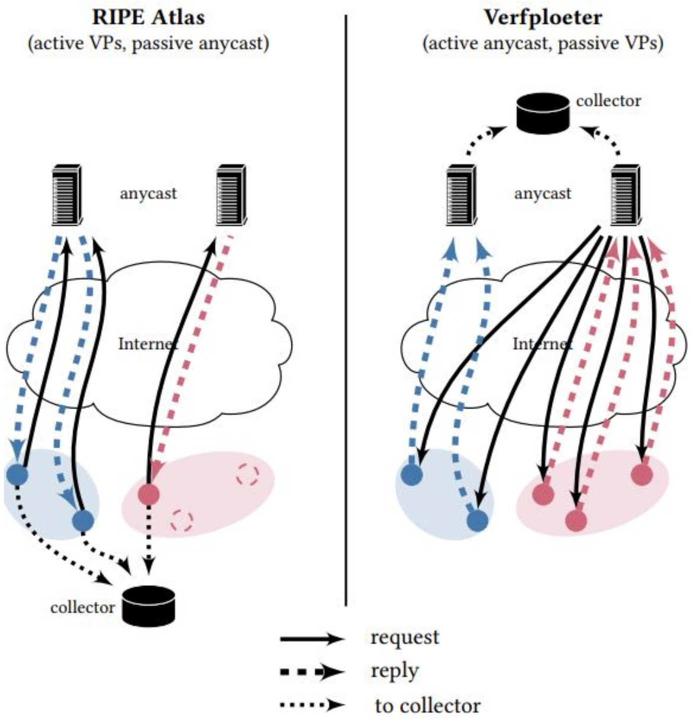

Fig. 2. How Verfploeter uses ICMP-responsive hosts on the Internet to map anycast catchments, compared to traditional techniques using RIPE Atlas from the paper "Broad and Load-Aware Anycast Mapping with Verfploeter" [9].

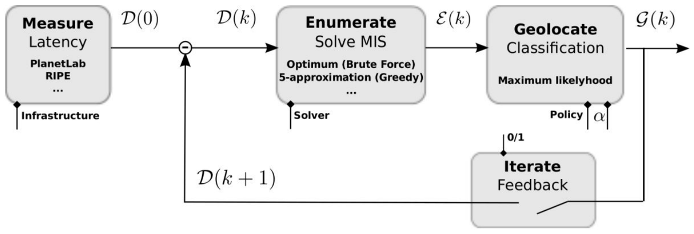

Fig. 3. The iterative workflow for the iGreedy method, from the paper "A fistful of pings: Accurate and lightweight anycast enumeration and geolocation" [7].

of ICMP-responsive hosts to be mostly unbiased. For these reasons, Verfploeter is able to achieve higher resolution data.

Verfploeter works by sending ICMP Echo Requests to ICMP-responsive hosts on the Internet using the anycast source address. These hosts will then send a reply to the anycast address, which will be routed by BGP to the "nearest" anycast site. The catchments are then derived by looking at the source addresses of the replies received at each anycast site. This probing technique has been used for planning the B-Root domain name server and for measuring Cloudflare's anycast CDN [17]. Since Verfploeter uses the Internet itself to perform measurements, it is a low-cost solution to measuring anycast, unlike most other catchment measuring tools that require a large number of vantage points and generate far more traffic.

C. iGreedy

As mentioned, anycast is often used on the Internet to provide critical services such as the DNS and is employed by large content providers like Google and Amazon. This makes anycast a crucial component of the the Internet, and for good reason as it improves the resilience of a service. Therefore, to understand the resilience of the Internet as a whole, it is important to gather data on the usage of anycast. In this work, we utilize two methods to externally measure anycast. The first technique we employ is iGreedy.

iGreedy was introduced in 2015 by Cicalese et al. as an enumeration and geolocation method for anycast. It is based on the *Great-Circle Distance* (**GCD**) technique, which leverages the speed of light to determine the maximum geographical distance travelled by an Internet packet [7].

This method represents one the first works on anycast that is not limited to the DNS. It is a service-agnostic methodology that enables anycast service discovery. Moreover, it can accurately enumerate and geolocate anycast replicas at city-level, provided that there are enough VPs, by relying solely on latency measurements.

Using speed-of-light constraints, they determine the maximum distance between a vantage point and an anycast replica based on measured latency. Using this, and the GCD technique, they then draw circles around multiple VPs. Next, they enumerate the minimum number of anycast instances required to ensure that there is at least one anycast instance within each circle (a single anycast instance can satisfy multiple circles when they overlap). Geolocation is then accomplished by examining the cities where anycast instances may be located and making predictions based on the population of these cities. This workflow is repeated iteratively to identify the most probable geolocations and estimate the expected number of anycast nodes. Figure 3 illustrates their iterative workflow. To validate their methodology, they compare their results with ground truths obtained from DNS root servers that disclosed their deployments. They also validate their approach using DNS CHAOS queries. The study achieved a recall rate of 50% recall and city-level geolocation accuracy of 78%.

D. MAnycast^2

The second technique we utilize is MAnycast^2, developed by Sommese et al. [15]. It makes use of Verfploeter, combined with a set of geographically distributed anycast nodes provided by the Tangled anycast testbed [18], to identify anycast prefixes on the Internet.

As mentioned, Verfploeter probes addresses on the Internet using an anycast probing system. In the case of MAnycast^2, these probes are sent from every individual anycast vantage point. Under normal circumstances, when a probed address is unicast, there is a single device behind that address, which receives a probe from every anycast node. Each of these probes generates a reply, which is directed to the nearest anycast instance. This means, that in the case of the probed address being unicast, all its replies will end up at the nearest VP that uses this shared anycast address.

However, if the probed target is also anycast, the requests from the VPs will reach different destination nodes that share

the same destination anycast address. Consequently, each of these destination nodes that receives the ICMP Echo Request will generate a reply, which will be routed to that node's nearest VP. This means that multiple VPs will receive replies in the case of an anycast target.
Therefore, all responses being received by a single VP implies that the destination address was unicast. Conversely, if responses are received by mutliple VPs, it indicates that the destination address is anycast. Figure 1 illustrates this core principle and showcases both cases.

However, there are corner cases where this principle fails.

- **False negatives** - An anycast address can be mistaken as unicast when all anycast replies are routed to a single VP. This can happen when the anycast system is deployed regionally and there is only a single VP in that region.
- **False positives** - A unicast address may be mistaken for anycast when the unicast replies are received by multiple VPs. This is no rare occurrence and happens due to dynamic routing in the Internet. For instance, if a unicast address is located in the "middle" of two VPs, such that BGP has a path to both VPs of near-equal length, its reply may reach both VPs due to load-balancing decisions made by the routing system over time.

In particular, the situation of unicast addresses ending up at 2 or 3 VPs (due to equal-cost paths) is relatively common. Therefore, in the MAnycast^2 paper, they classify an address as anycast when 4 or more VPs receive replies, while they are classified as unicast when all replies are captured at a single VP. However, when 2 or 3 VPs receive responses, there is uncertainty whether it is anycast or unicast. For these reasons, additional iGreedy measurements are performed on the potential anycast targets. This phenomenon was also observed in the anycast catchment measurements with Verfploeter by De Vries et al. [9], where they observed that due to routing flexibility, clients may utilize different anycast instances at different times.

The main advantages of MAnycast^2, compared to other anycast measuring methodologies, are that it is much faster and it does not require a large probing network, making it a cost-effective tool for monitoring anycast deployments. Moreover, the set of ping-responsive hosts that it sources on the Internet is far larger than the number vantage points that can be offered by any measuring platform. By utilizing fewer VPs for sending out probes, it mitigates the risk of network overload and potential strain on the measuring infrastructure. These properties render MAnycast^2 suitable for conducting repeated measurements, providing longitudinal data on anycast usage across the Internet.

III. Related work

In this section we will explore related work concerning IP anycast censuses. There have been multiple previous studies that have performed censuses on anycast deployments on the Internet using various techniques.

2013 - Evaluating anycast in DNS since the DNS is the most prominent use case for anycast, it has been a significant focus in anycast measurements. In 2013, Fan et al. conducted one of the first third-party anycast detection and enumeration measurements for the DNS [6]. Their work consists of using CHAOS-class DNS queries in combination with traceroute and recursive resolvers. They also proposed a new IN-class for identifying DNS resolvers. Additionally, their research revealed a malicious party that was hijacking an anycast address, highlighting another motivation for measuring anycast deployments. They found that 72% of top-level domains (**TLDs**) use anycast.

Their technique to combine traceroute with the retrieval of CHAOS DNS records, improves the accuracy of enumerating DNS anycast as it resolves ambiguities. The main limitation of their work is that it is entirely focused on the DNS, as it makes use of DNS-specific probes (CHAOS DNS records) and infrastructure (recursive resolvers as vantage points). Our work is not dependent on DNS, and can probe for any protocol.

2015 - Characterizing IPv4 anycast adoption and deployment Cicalese et al. performed multiple IPv4 anycast censuses using latency measurements [8]. The main platform they use was PlanetLab, which provides a large number of vantage points. Until their work, most anycast censuses were limited to the DNS. However, theirs was protocol-agnostic and represented the first Internet-scale measurement for anycast. Additionally, they performed service discovery and found a large range of TCP services offered using anycast. Among these services, HTTP/HTTPS was the most popular and frequently used for websites from the top-100k Alexa list. Alongside service discovery, they also conducted enumeration and geolocation to determine the number of anycast instances running behind an anycast deployment and their geographic locations, as mentioned in Section II-C. By utilizing ground truth data from CDNs that share their anycast locations, they found that their method achieved 78% accuracy in geolocating at city-level and had a recall rate of 50%.

Furthermore, they found that the usage of IP anycast had changed significantly compared to a few years prior. Which confirms the importance of periodic measuring of anycast. Such periodic censuses were also mentioned as future work, as it would reveal how anycast evolves over time.

However, their methodology had a notable drawback as they relied on latency measurements, which are not always accurate. Moreover, the accuracy of their results depended on the availability of precise geographical data for the VPs, which may not always be guaranteed. Additionally, their work required a considerable amount of traffic for their measurement to be performed, and relied on specific measuring platforms due to the requirement of a large number of VPs.

2018 - A longitudinal study of IP anycast based on their iGreedy method, as explained in Section II-C, and 3 years after their 2015 census, Cicalese et al. performed the first longitudinal study of IP anycast [11]. Unlike their previous work, this census was performed on a monthly basis for a year, from May 2016 to May 2017, using vantage points provided by the PlanetLab platform. Throughout this period, they observed dynamic changes in anycast deployments, with some companies reducing their anycast deployments while others increased them. However, when analyzing the aggregate, they found that anycast deployment remained relatively stable. Additionally, they utilized the longitudinal data to evaluate the impact of changes in anycast deployments on end-to-end latency.

Their study provides insights into changes that occur in anycast with a monthly granularity, which motivate our work.

2019 - Towards passive analysis of anycast in global routing: unintended impact of remote peering this approach by Bian et al. makes use of a machine-learning classification algorithm, using passively collected BGP data as input features [13]. They found that their method is 90% accurate in detecting anycast, and they observed that remote peering is a cause of anomalous routing behaviour that impacts at least 19.2% of anycast prefixes.

Their assumptions, on which they based their passive approach, were:

- Anycast prefixes have more upstream ASes than unicast prefixes (since anycast is announced from multiple locations, each with their own upstream ASes)
- The distance between these upstream ASes is larger in the case of anycast (since anycast is distributed and announces from distant locations)

However, the second criterion is heavily influenced by remote peering. As remote peering allows peering to an IXP without physical presence, thereby upstream ASes can have large distances whilst belonging to a unicast address.

This work is mostly overshadowed by more accurate active measurements, and lacks reproducability.

In contrast to the related work, our methodology requires very few VPs to accurately identify anycast. Furthermore, we have nearly full control over the VPs and are not limited by the constraints of measuring platforms like RIPE Atlas or PlanetLab. With this control, we can send out very specific probes, including service-specific probes. Furthermore, our approach generates far less traffic compared to the other methods, thanks to our reliance on fewer VPs. This makes our method a more responsible and efficient system to perform regular anycast censuses.

IV. Limitations of MAnycast^2

There are several limitations with MAnycast^2. Firstly, MAnycast^2 has limited utility for geolocating and enumerating anycast deployments because such data requires a large set of VPs. Secondly, MAnycast^2 has a sizeable number of false positives (unicast replies ending up at multiple VPs caused by e.g., route flips), and has a number of false negatives (an anycast infrastructure within a single VPs catchment will have all replies ending up at a single VP). These false positives and negatives lower the overall accuracy of the tool.

For these reasons, MAnycast^2 is followed up with an additional iGreedy measurement on the prefixes classified as potential anycast targets by MAnycast^2. This provides great synergy as MAnycast^2 allows for efficient and responsible measuring of large hitlists, from which it obtains a limited set of anycast targets, which are then measured using the more accurate and more intensive iGreedy measurements to filter out the false positives and obtain geolocation and enumeration data. The false negatives can be reduced by increasing the number of VPs of the MAnycast^2 infrastructure, and by locating them in diverse and geographically distant locations.

Finally, there is a limitation that affects both iGreedy and MAnycast^2, and it is caused by inconsistencies in routing on the Internet. As mentioned, anycast relies on the Border Gateway Protocol (**BGP**) for routing, which generally works as expected and results in clients getting routed to the nearest anycast instance. However, there are instances where routing protocols other than BGP are employed, resulting in unexpected and undesirable routing. The MAnycast^2 paper also reported such cases, such as the Google Public DNS resolver using their own private network. Additionally, the paper by Arnold et al. demonstrates that while BGP is highly efficient in routing and difficult to outperform, it is sometimes replaced by other routing algorithms [12]. These replacements are in place by large content providers on the Internet who have built their own private global infrastructure. These private networks are connected to the public Internet using Points of Presence (**PoPs**). Between the client and the PoP traditional routing will be employed (i.e. BGP), however the routing algorithms employed beyond the PoP is entirely up to the owner of these private networks. These inconsistencies may cause an anycast prefix to be seen as unicast by MAnycast^2, and may lead to false enumeration and geolocation data with iGreedy. This limitation, however, is out of scope for this work since we have no control over the routing policies employed on the Internet and have limited visibility into how our probes are routed.

V. Goals and challenges

In this section, we will list and introduce the goals and challenges of this work. These goals are related to realizing the requirements (which we introduced in Section I) for the improved Verfploeter measurement tool. Furthermore, we will list the challenges that we expect to encounter when realizing these goals.

A. Goals

First, we will list the goals required for an improved version of the MAnycast^2 measuring system.

G1 **Easy deployment** - One of the requirements is that the system is easy to deploy in any new future environment. We want the system to be deployable on any anycast infrastructure, which means it must be able to operate on a variety of hosts and Virtual Machines (**VMs**) with different hardware architectures and operating systems. For this reason, the system must be easy to deploy, regardless of the host.

G2 **Additional probing methods** - The system must be able to perform UDP/DNS and TCP/SYN-ACK probes in addition to ICMP/ping probes. These additional probing methods are of interest for analyzing the discoverability of anycast when probing with these protocols. It may also reveal new anycast prefixes that are unresponsive to ICMP/ping. Additionally, we want to learn more about e.g., the distribution of unicast v. anycast distributions for each protocol.

G3 **Synchronous probing** - The original MAnycast^2 measuring system performed sequential probing, which was listed as a limitation. This system must solve this limitation by allowing clients to perform their probes in parallel to avoid the effects of routing changes, such as route flips, that occur over time from affecting the measurement results. Furthermore, each target must be probed by all clients one second after each other, similar to a regular ping measurement, to avoid the probed targets from receiving a burst of requests. To evaluate the effects of synchronous versus sequential probing, we also want to retain the ability to send out probes in sequence, for which we will implement client-selective probing (i.e. we can select which clients send out probes during a measurement).

G4 **Authenticity and origin of results** - Received replies must be authenticated to be part of our measurement, to ensure the accuracy of our results. For ping probing, we can use the payload of ICMP Echo Requests, which is echoed in the ICMP Echo Response, as was done in the original Verfploeter tool. However, for TCP and UDP, we will need to implement new methods of verifying that responses belong to our measurement. Furthermore, we want to encode useful information in our requests that can be extracted from the replies, such as the VP which sent the request to which a received reply belongs and the transmission time of the request.

G5 **Future development** - It must be easy for future developers to use, understand, and modify the system. Other studies may be performed that require this probing system, and it must be easy to deploy and perform measurements for developers unfamiliar with the system. Additionally, they may need to make changes to the software to incorporate new features, which is why we want to ensure that the code is easy to understand and utilizes well-maintained dependencies.

B. Challenges

To achieve the goals discussed, we will have to overcome these key challenges.

C1 **Robustness** - The system must be able to perform in a robust manner, despite being deployed in possibly unstable environments. One of the motivations for developing this system is to make it easy to perform scheduled measurements over long periods of time, for collecting longitudinal data. This means that all the system components must be able to perform daily measurements over extended periods of time without software failures. However, there are issues such as network outages and hardware failures that can disconnect parts of the distributed system. In such cases, the system must continue to operate to the best of its ability. Furthermore, we want to enforce that only a single measurement is active at a time, as multiple measurements being performed simultaneously can jeopardize the results and cause unwanted probing rates. Finally, it must be possible to cancel measurements in the case of wrong measurements being started (e.g., due to entering the wrong hitlist as argument).

C2 **Resource scarcity** - The system must be able to scale at a low cost. This means it should be able to run on modest VMs with limited CPU, RAM, and disk space available. Therefore, we require the system to use minimal system resources and have a small storage size.

VI. Measurement system design

In this section, we will describe the system design of the probing system, keeping in mind the goals and challenges discussed in the previous section. For a detailed view, we provide the GitLab repository [20]. This repository includes the source code, Rustdoc documentation explaining the code, a README with example commands, and more (goal **G5**).

A. Design

1) Rust: The system is implemented in the Rust programming language, just like the original Verfploeter system. The main advantages of Rust are its thread- and memory-safety, which contribute to improved robustness (challenge **C1**). Additionally, Rust compiles to machine code, resulting in small binary executable sizes, low resource usage (RAM and CPU), and energy efficient code (challenge **C2**).

2) gRPC: For communication between the three components (CLI, Server, and Client), we use gRPC (Remote Procedure Calls) developed by Google. gRPC is designed to facilitate communications between components of a distributed system. In our system, a gRPC server is run in the Server component, allowing multiple Client components and a CLI component to connect. We utilize the gRPC Tonic cargo dependency to implement the gRPC components, and protobuf-compiler/protoc to serialize/deserialize the messages transferred based on our protobuf message definitions. gRPC, protoc, and Tonic are actively maintained and well-documented (goal **G5**).

3) Docker: The system can be deployed by distributing compiled binaries, but we have also implemented support for distribution and deployment using Docker. The main advantages of Docker are that it is system-agnostic (meaning that it will work regardless of the hardware and OS of the VM), and it has all configurations set in the Dockerfile (goal **G1**). By using the property of Rust that allows for small binary executable sizes, the Docker image size that is distributed is less than 100MB in size (challenge **C2**).

B. Components

Next, we will go over the components of the system individually. Figure 4 illustrates how the system components are connected.

1) Server: The Server acts as a centralized controller in the system. It keeps track of the connected clients and assigns them unique client IDs while enforcing unique hostnames. The CLI can request information about connected clients using the *client-list* command. When the CLI instructs the Server to perform a measurement using the *start* command,

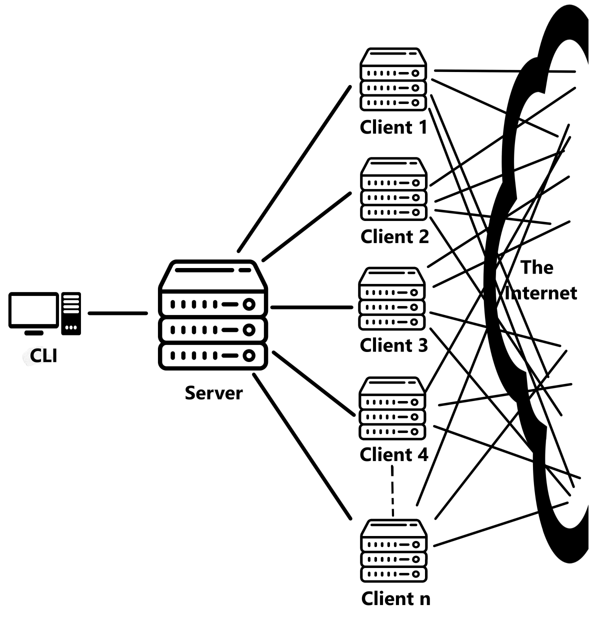

Fig. 4. The components of the system and their connections.

the Server announces to all clients that a new measurement is beginning, providing information about the measurement itself (e.g., type of measurement, probing rate). Next, the Server will synchronously distribute the hitlist to the "active" clients (i.e., the clients that are sending out probes in the measurement). These tasks will be sent out at the same rate as the measurement's probing rate. As the centralized controller, the Server has full control over the measuring process and is responsible for enforcing the desired probing rate. The measurement results are streamed back to the Server in real-time from the Clients, and these results are then streamed to the CLI. Finally, at the end of the measurement, the Server notifies all Clients that the measurement has finished, and once all Clients have made the server aware that they closed everything for the measurement, the server informs the CLI that the measurement is complete.

As the central part in our architecture, the Server must be as robust as possible, for this we implemented the following functionalities (challenge **C1**):

- Only a single measurement may be active at a time to avoid conflicting results. The Server will refuse any new measurements whilst a measurement is active.
- If a CLI disconnects during an active measurement, the Server immediately cancels the measurement by sending termination messages to the clients.
- In the event of a Client disconnecting; the Server logs the event, updates its Client list, and continues the measurement with the remaining clients to the best of its ability.
- If the Server itself disconnects, all connected clients terminate their ongoing task, and the CLI exits.
- Any wrong commands received by the CLI will be refused (e.g. invalid task type, non-existent clients selected).
- The Server lets the Clients know when the measurement starts, and when it finishes to ensure that the measurement is performed in its entirety.

2) Client: As mentioned earlier, the Client receives a stream of the measurement from the Server. The Client sends out probes as it receives targets, ensuring that the Client does not buffer the hitlist, which reduces resource usage (challenge **C2**). At the start of the measurement, the client opens sockets for sending and receiving based on the desired protocol. If the Client is "inactive" for the current task it will only listen for incoming packets and not send out probes itself. Outgoing probes contain encoded information in their fields, which gets echoed back and can be extracted from replies. All the desired information is added to the results, which are then streamed back to the server.

3) CLI: The CLI is initiated by a user with a command, either *client-list* or *start*. This command is sent to the Server, and the CLI waits for the results. The *client-list* command lists all connected Clients, displaying their unique Client ID and hostname. The *start* command is used to initiate a measurement. The user can specify the rate at which packets should be sent out, the hitlist to perform the scan on, the protocol type (ICMP, UDP, or TCP), the clients that should send out the probes, and the source address from which to send out the probes. The CLI writes the results to a file (and to command-line if specified). The CLI also offers an option to shuffle the provided hitlist before sending it to the Server.

When the start measurement command is executed, the CLI prints information about the measurement, such as the current time,hitlist size, probing rate, and an estimation of the task duration based on the probing rate and number of addresses. The user can cancel the command at any time by terminating the CLI process, which in turn terminates the measurement at the Server and Clients. If the requested command cannot be executed (e.g., due to an active task or no connected clients), the CLI prints the error message received from the Server and exits the command.

C. Measurement process

In the previous Verfploeter system, ping probing measurements were possible but had to be executed individually for each Client. In this work, we have enhanced the system to forward a probing measurement to all connected clients, allowing these probes to be executed in parallel. Additionally, the hitlist is distributed in a round-robin fashion, ensuring that all clients probe targets in the same order and with a one-second delay between each client (goal **G3**).

We have also implemented client-selective probing, which allows the CLI to specify which clients will send out probes for the measurement. By default, all clients will send out packets, but the CLI can choose a subset of clients for the measurement (goal **G3**). Regardless of the selection made, all clients will listen for incoming packets. Another configurable setting is the probing rate, which determines the number of packets sent out per second at each client.

Furthermore, we expanded the tool to support multiple probing protocols (goal **G2**):

a) ICMP/ping: Pinging was already a feature in the previous MAnycast^2 system. Ping probes are sent out with a payload that is added to the ICMP Echo Request. This payload is then returned when a probed target responds with an ICMP Echo Reply. In this payload, we encode the following information (goal **G4**):

- The task ID of the current measurement, which verifies that the received ICMP Echo Reply belongs to this measurement.
- The transmission time, which allows for calculating the Round Trip Time (**RTT**).
- The sender client ID, providing information about the VP that sent the probe triggering the reply.
- The source and destination address of the probe, enabling the detection of spoofing.

b) TCP: For our TCP probes, we utilize the TCP flags SYN and ACK. TCP is a stateful protocol, where the sender and receiver establish a session and remember the state for that session. This state is established by sending a TCP SYN packet, to which the receiver replies with a TCP SYN-ACK. Since we do not want to open unused states on the targets of our hitlist, we deviate from the protocol by sending a SYN-ACK packet instead of a SYN packet. When the target device receives this packet, it looks for an established state for this packet, which does not exist, and responds with a RST packet to terminate the exchange. By recording these RST packets, we can identify TCP-responsive hosts.

This method of probing with SYN-ACK is an improvement over the commonly used SYN probing, which opens states on the target devices, as employed by other TCP scanning tools (e.g., traceroute).

For these probes we encode the following information (goal **G4**):

- The client ID is encoded into the destination port, and the RST reply will have this value in the source port.
- Transmission time is encoded into the SEQ field of the TCP header, which is returned in the ACK field of the RST packet. Since these fields are limited by 32 bits, we encode the 32 least significant bits of the timestamp (current time in milliseconds) into this field.
- We verify that the reply is part of our measurement by examining the TCP flags (we only want the RST flag set) and checking for valid port numbers.

For both the source and the destination ports, we use high port numbers that are not commonly used by services to ensure that it is highly unlikely that the target machine has a running service on the probed port.

c) UDP/DNS: The last probing methods involves sending out DNS A Record Requests using the UDP protocol. This is a service-specific probe used to discover DNS servers, as they respond with a DNS A Record Response. DNS A Records are typically requested by users to obtain the IP address associated with a given domain. In our probing, we send a request for a non-existing domain, to which a conventional DNS server will send back a reply containing our requested domain. Since this reply contains the domain of our request we encode our information into the subdomain of the requested domain (goal **G4**), this includes:

- The source and destination address of the request.
- The source and destination port.
- The transmission time.
- The client ID of the sender (which is also encoded in the first two bytes of the DNS transaction ID to ensure each DNS request has a unique ID).

The destination port of the UDP header is set to 53 (the default port used for DNS), and we use a static high-numbered source port.

Additionally, we listen for ICMP packets during our DNS scan. Hosts that do not have a DNS service running either ignore the request or send back an ICMP packet. These ICMP packets are used to indicate to the sender that, for example, the requested port (port 53 for DNS) is unreachable. We specifically look for type 3 (network unreachable) ICMP packets and record the code, which provides further information about why the network is unreachable. We also parse the ICMP payload. Depending on the replying host, the payload may contain our original DNS Request in its entirety or parts of the request such as the IPv4 and UDP headers, just the IPv4 header, or an empty payload. We extract as much information as possible from the payload. If certain information is unavailable, we record it as 0, which identifies unknown data.

The domain name for which we request the DNS Record can be set to any value, including a domain owned by the researchers, which allows them to capture more information on their own nameservers, we leave this for future work.

VII. Validation

In this section, we will validate that the system meets our goals by conducting several case studies that involve numerous measurements.

A. Measuring environment

We will perform our MAnycast^2 measurements using the TANGLED platform [18], which is owned by the University of Twente and allows for full control over the nodes as required for this work. Additionally, we will utilize RIPE Atlas for our iGreedy verification measurements.

1) TANGLED: TANGLED is an anycast testbed that researchers can use to run experiments utilizing geographically distributed vantage points that share an anycast address [18]. It provides a set of tools for customization and configuration of the anycast network, measurements, and data collection. Like most other anycast deployments, TANGLED makes use of BGP to take care of the routing. For this research we will be using TANGLED to perform MAnycast^2 measurements that determine whether /24 prefixes are unicast or anycast.

2) RIPE Atlas: Atlas has a large set of vantage points (12k+) with high AS-level diversity. However, since RIPE is based in Europe, its vantage points are biased towards this continent. Furthermore, Atlas employs a credit system [19] to limit the amount of traffic generated by users, reducing the

Node	Location
au-syd	Sydney, Australia
nl-ams	Amsterdam, Netherlands
us-sea	Seattle, USA
fr-par	Paris, France
uk-lnd	London, UK
de-fra	Frankfurt, Germany
sg-sin	Singapore, Singapore

TABLE I
THE LOCATION OF THE VULTR NODES.

# of VPs	Distinct /24s	Distinct ASNs
1 (unicast)	3,963,783 (99.67%)	67,596
2 (anycast*)	3,241 (0.08%)	666
3 (anycast*)	591 (0.01%)	159
4 (anycast)	820 (0.02%)	141
5 (anycast)	477 (0.01%)	87
6 (anycast)	1,633 (0.04%)	78
7 (anycast)	6,375 (0.16%)	88
Total anycast	13,137 (0.33%)	1,219

TABLE II
CLASSIFICATION AND BREAKDOWN OF /24S BY THE NUMBER OF VPS THAT RECEIVE RESPONSES.

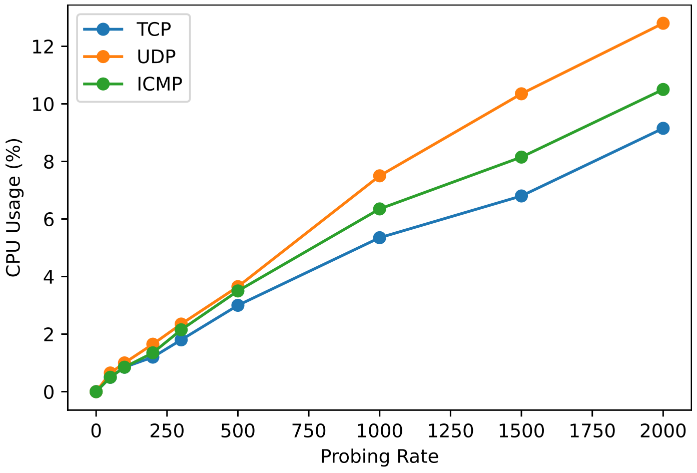

Fig. 5. CPU usage in percentages for the three different probing types at various probing rates.

impact of their measuring platform on the Internet. Despite these limitations, this platform was used in many other anycast studies [7, 10, 11, 13, 14] and offers good flexibility. Although biased towards Europe, it still provides good global coverage and geographical diversity. We will use RIPE Atlas to perform iGreedy measurements that verify anycast candidates discovered with MAnycast^2.

3) Measurement deployment: The system was deployed on Vultr nodes that are part of the TANGLED testbed. All of these nodes share an anycast address, and we performed BGP announcements to ensure that all traffic for our anycast /24 prefix would only reach these nodes. On these nodes we ran the Client component of our System. The location of these nodes can be seen in Table VII-A3. The Server and CLI component were run on a VM owned by the University of Twente.

B. Resource efficiency

As mentioned, the system was designed with resource scarcity in mind. Therefore, we perform buffering at the Server so that Clients receive new target IPs as they send them out. This means the Clients do not need to buffer the entire hitlist. Since the Server and CLI can be run at any location, including locally, we are not concerned with their resource usage. However, the Clients must be run at the anycast sites which may have limited hardware.

Figure 5 shows the resource usage for all three protocols in terms of the percentage of CPU used at various probing rates. The hardware used for testing was a x86_64 Intel Core Processor with a 2.4GHz CPU and 987 MiB memory. The chart does not include memory usage, as it remains constant at 1.3% (13 MiB) for all three protocols, regardless of the probing rate. From the chart, we can observe that UDP/DNS is the most demanding protocol, likely due to the large size of DNS packets and need to listen on two sockets (one for UDP and one for ICMP). The ICMP and TCP probes have nearly identical packet sizes, however we still see that ICMP has a slightly higher CPU usage. This could be attributed to the increased responsiveness of the ICMP protocol, resulting in more incoming packets to process.

C. *MAnycast^2 census*

The goal of our first case study is to perform a MAnycast^2 census, similar to the one conducted in the MAnycast^2 paper, using ICMP/ping. This will validate that the system is capable of conducting MAnycast^2 measurements. We will also run a follow-up iGreedy measurement to quantify the performance by examining the True Positive (**TP**) rate of our measurement. Additionally, we can compare the results of this ping scan to that of the MAnycast^2 paper to get an indication of how this new system performs compared to the old one.

As mentioned, the MAnycast^2 scan is performed at /24 prefix granularity, and the results are analyzed to identify unicast and anycast prefixes. For this purpose, we utilized the ISI hitlist [16], which is based on the most ICMP-responsive host in each /24 prefix. The hitlist consists of a total of 8.5 million addresses, each belonging to a unique /24 prefix.

The scan was conducted with a probing rate of 1,000 outgoing probes per second for each client, lasting 142 minutes. The hitlist was shuffled to avoid probing consecutive /24 prefixes that may be part of a single larger prefix all at once.

In Table II, we present the results of our ICMP scan, displaying the distribution of /24 prefixes and Autonomous System Numbers (**ASNs**) based on the number of VPs that received responses. out of the 8.5 million probed targets, 4 million responded, resulting in a responsiveness rate of 47%. The majority of the responsive prefixes were identified as unicast (99.67%). Among the 0.33% anycast prefixes, most of them had their replies reaching 7 VPs, followed by the next largest group with 2 VPs.

In the original MAnycast^2 paper, they identified prefixes replying to a single VP as unicast, while prefixes replying to two or more VPs are identified as anycast However, the paper acknowledges the presence of uncertainty when replies were received from two or three VPs, as they observed low TP rates in such cases. The TP rates were determined by conducting

# of VPs	MAnycast^2	iGreedy confirmed	Diff. (resp.)
2	3,241	1,726	46.74%
3	591	554	6.26%
4	820	811	1.10%
5	477	475	0.42%
6	1,633	1,629	0.24%
7	6,375	6,367	0.13%
Total	13,137	11,562	11.99%

TABLE III

THE NUMBER OF POTENTIAL ANYCAST TARGETS DISCOVERED BY MANYCAST^2, LISTED BY NUMBER OF VPS, ARE SHOWN ALONG WITH THE COUNT OF THOSE THAT WERE CONFIRMED TO BE ANYCAST BY IGREEDY. ADDITIONALLY, PERCENTAGE DIFFERENCE BETWEEN THE TWO IS PROVIDED.

# of VPs	Parallel	Series
1	99.67%	99.00%
2	0.08%	0.73%
3	0.01%	0.03%
4	0.02%	0.02%
5	0.01%	0.01%
6	0.04%	0.04%
7	0.16%	0.16%
Total anycast	0.33%	1.00%

TABLE IV

THE PERCENTAGE OF ADDRESSES THAT ENDED UP AT X NUMBER OF VPS FOR THE ICMP MEASUREMENT IN PARALLEL AND IN SERIES.

follow-up iGreedy measurements, which offer greater accuracy in evaluating anycast.

1) iGreedy confirmation: To assess the TP rate of our scans, we conducted a follow-up iGreedy measurement, and the results are present in Table III. Similar to the results found using the previous MAnycast^2 measuring system, we observed the lowest TP rates at two and three VPs, with TP rates of of 53% and 94%, respectively. However, these TP rates are considerably higher than those reported in the MAnycast^2 paper, where TP rates of 10% were observed for two VPs and 87% for three VPs. Overall, our results achieved an 88% TP rate, compared to the 38% TP rate obtained with the previous measuring system. This indicates a significant improvement in accuracy using the new system.

2) Synchronous probing: In the previous section, we examined the results of our MAnycast census, which was performed using synchronous probing. These results demonstrated a significant increase in the number of correctly classified anycast prefixes compared to the findings of the original MAnycast^2 paper. However, direct comparisons between these censuses are challenging due to several factors, such as a three-year gap between measurements, the ISI hitlist had different and fewer targets, and variations in the number and geographic distribution of VPs. To further explored the effects of synchronous probing, we will delve into this topic in this section.

Prior to this work, MAnycast^2 scans were conducted in sequence, where each VP would probe the hitlist one after the other. In the case of the MAnycast^2 paper, there were 13 minute-intervals intervals between probes targeted at the same destination. This interval was considered a limitation because Internet routing is dynamic over time (e.g., route flips as discussed in Section A-B3). Consequently, between these 13 minute intervals, the routing might have changed, increasing the likelihood that replies from a unicast target would traverse different paths for the various probes and ultimately reach multiple VPs. Our hypothesis is as follows: *By performing a MAnycast^2 census in parallel instead of in series, we will observe a decrease in false positives (i.e., unicast prefixes being falsely classified as anycast), thereby increasing the overall accuracy of the census.*

To test our hypothesis, we leveraged the client-selective probing feature (as explained in Section VI-C) to conduct a MAnycast census with asynchronous probing. This involved probing the hitlist seven times, where a different client was sending out probes each time.

The results of the census when probing in parallel and when probing in series are presented in Table IV. Firstly, we observe that by probing in series we have 99.00% of the targets reply to a single VP, compared to 99.67% targets whose replies end up at a single VP for the scan in parallel(i.e. targets inferred as unicast). This may seem like a small difference, but considering that the number of prefixed who use anycast are at most 0.45%, based on the previous MAnycast^2 work [15], this is a significant difference. We also see that by performing the measurement in series we get a significant increase in potential anycast prefixes (1.00%), compared to probing in parallel (0.33%). When we look at the breakdown based on the number of VPs receiving replies, we find that there are more prefixes whose replies end up at two or three VPs when probing in series, however for four or higher VPs we see identical percentages. This coincides with the findings of the MAnycast^2 paper where they only noticed a significant false positive rate for two or three VPs.

To confirmed that the increase in potential anycast prefixes when probing in series, are due to an increase in false positives, we performed a follow-up iGreedy measurement for these prefixes. These results were inaccurate, due to wrong geographical data for one of the Atlas vantage points, which caused many unicast addresses to be falsely enumerated as having two instances. For this reason, we compared the results of the scans in parallel and in series, by looking at the number of anycast prefixes with more than two instances, according to iGreedy. This shows that, when probing in parallel the accuracy is 78% whereas probing in series yields an accuracy of 19%. This confirms that the additional anycast prefixes found when probing in series, are in fact unicast prefixes falsely classified as anycast.

These results show that our hypothesis was correct, by probing in parallel we increase the accuracy of the measuring system, we even see an amplified effect compared to the results found in the original MAnycast^2 paper, this amplification is likely due to the intervals between measurements being 142 minutes instead of only 13 minutes.

D. Slow measuring

By implementing synchronous probing, where the hitlist is distributed in a round-robin fashion and each VP's probe for the same target is sent out one second after the other, we enable significant reduction in the the probing rate without increasing the intervals between probes. Although it may seem counter-intuitive to perform our MAnycast^2 census at a slow probing

Number of VPs	1,000 packets/second	100 packets/second	Difference
1	3,963,783	3,861,467	-2.58%
2	3,241	3,028	-6.57%
3	591	588	-0.51%
4	820	820	0.00%
5	477	522	9.43%
6	1,633	1,616	-1.04%
7	6,375	6,374	-0.02%
Total anycast	13,137	12,948	-1.44%

TABLE V
THE NUMBER OF ADDRESSES THAT ENDED UP AT X NUMBER OF VPS FOR THE FAST RATE AND SLOW RATE ICMP MEASUREMENTS, ALONG WITH THE DIFFERENCE OF THE TWO.

rate, considering our goal of facilitating efficient anycast censuses that can be performed regularly, it helps us achieve this objective. This approach aligns with the responsibility-oriented design of MAnycast^2; ensuring that it does impose significant strain on the Internet or the measuring platform on which it operates. Slowing down the scans significantly means that the generated traffic is spread thinly over time, which is a commonly adopted tactic for conducting Internet measurements ethically. Moreover, a reduced probing rate leads to decreased CPU usage (as observed in Section 5) and lower Internet speed requirements for the Client VMs. With our synchronous probing technique, we believe it is feasible to perform scans at slower rates without compromising the overall performance.

Table V presents the results of our previous "regular" measurement with a probing rate of 1,000 and a slow measurement with a probing rate of 100 packets per second. These scans took 2.3 and 23 hours, respectively. In the slow measurement, we observed a slight decrease in responsiveness, this is due to this scan being performed a month later and the hitlist being more outdated. Additionally, it is possible that the responsiveness of the hitlist is not consistent throughout the day, meaning that the fast scan could have been conducted during a more favorable time compared to the slow scan, which was spread out over an entire day.

The results of both scans is highly consistent, with the outliers being 6.57% fewer anycast prefixes at two VPs and 9.43% more anycast prefixes at five VPs. For the iGreedy-confirmed anycast prefixes from the fast ICMP scan, we found that 99.78% were also identified in the slow scan. This demonstrates that reducing the probing rate of our MAnycast^2 measuring system has negligible effects on the performance.

E. UDP & TCP probing

To validate the new probing protocols in our system, we performed MAnycast^2 with TCP SYN/ACK and UDP/DNS. These results give us insights into the individual and combined coverage for these protocols.

1) ISI scan: Using the previously mentioned ISI hitlist, we performed additional TCP and UDP scans, these results will be combined with those of the ICMP scan, such that comparisons can be made.

In Table VI, we can observe a detailed breakdown of the number of virtual private servers (VPs) that received responses for all probed /24s. The table is segmented based on the

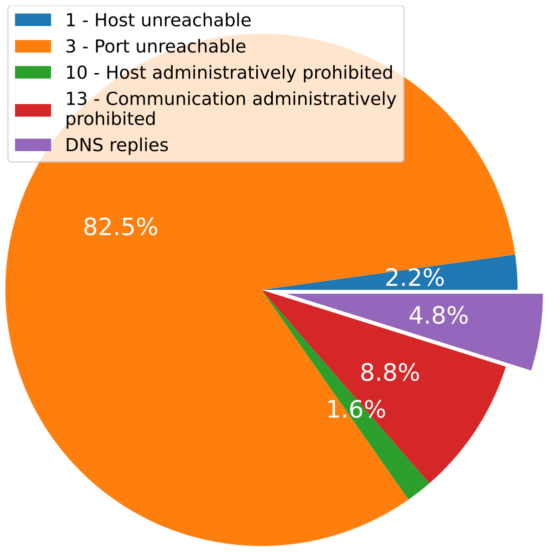

Fig. 6. The type of UDP responses received from unique source addresses (excluding rare codes received that combined make up 0.3% of the source addresses).

protocol used for the scan. The UDP scan is further categorized into three groups: UDP (all), UDP (ICMP), and UDP (DNS).

- UDP (all): This category includes prefixes that responded with either ICMP or DNS replies.
- UDP (ICMP): This category consists of prefixes that only sent ICMP responses and did not provide any DNS responses.
- UDP (DNS): This category includes prefixes that exclusively sent DNS responses without any ICMP replies.

It is important to note that there are a few rare prefixes for which we received both DNS and ICMP replies. As a result, the count for UDP (all) is not simply the sum of UDP (ICMP) and UDP (DNS). In the subsequent sections, we will delve into a more detailed analysis of the obtained results.

a) Responsiveness: In Table VI, we observe that ICMP/ping has the highest responsiveness with receiving replies from 47% of the probed targets, followed by TCP with 17% responsiveness, and then UDP which received replies from 12% of the hitlist.

A breakdown of the UDP/DNS replies can be seen in Figure 6. It shows that only 4.8% of the received responses to our UDP probes were DNS replies. The majority of the responses to the UDP probes were ICMP type 3 (destination unreachable) packets, with the most common ICMP code being port unreachable. These packets indicate that the probed target is listening on UDP but does not have port 53 open (the commonly used UDP port for DNS that we targeted).

More details about the responsiveness can be seen in Figure 7, which shows the intersections of these protocols. This figure uses an UpSet plot [22], which we will re-use multiple times, these plots are structured as follows:

At the bottom left, we see the total responses for UDP, TCP, and ICMP, along with the percentage of the hitlist that it received. Moving to bottom center/right, we find all the different intersection options for these three protocols, including the unresponsive category. On the bar chart above,

# of VPs	ICMP	TCP	UDP (all)	UDP (DNS)	UDP (ICMP)
1 (unicast)	3,963,783 (99.67%)	1,471,334 (99.46%)	972,318 (99.45%)	52,218 (96.54%)	920,159 (99.62%)
2 (anycast*)	3,241 (0.08%)	2,874 (0.19%)	979 (0.10%)	498 (0.92%)	487 (0.05%)
3 (anycast*)	591 (0.01%)	228 (0.02%)	448 (0.05%)	365 (0.67%)	93 (0.01%)
4 (anycast)	820 (0.02%)	81 (0.01%)	434 (0.04%)	345 (0.64%)	95 (0.01%)
5 (anycast)	477 (0.01%)	90 (0.01%)	291 (0.03%)	243 (0.45%)	34 (0.00%)
6 (anycast)	1,633 (0.04%)	226 (0.02%)	240 (0.02%)	189 (0.35%)	50 (0.01%)
7 (anycast)	6,375 (0.16%)	4,488 (0.30%)	3,008 (0.31%)	234 (0.43%)	2,772 (0.30%)
Total anycast	13,137 (0.33%)	7,987 (0.54%)	5,400 (0.55%)	1,874 (3.46%)	3,531 (0.38%)

TABLE VI
CLASSIFICATION OF /24S BY THE NUMBER OF VPS THAT RECEIVE A RESPONSE FOR THE DIFFERENT PROTOCOLS.

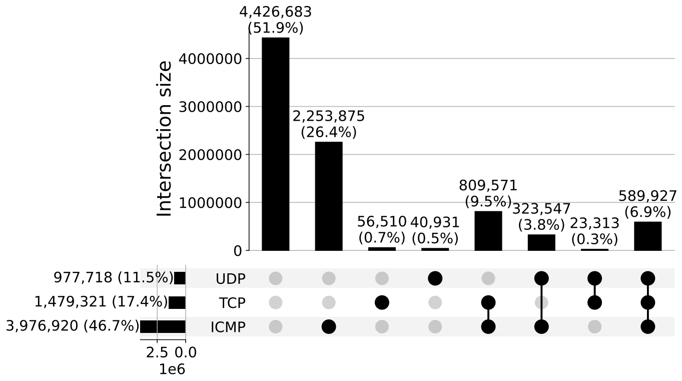

Fig. 7. /24 responsiveness intersections and counts for the three protocols.

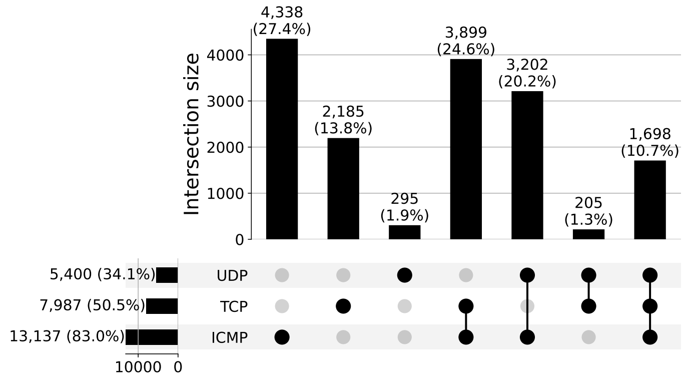

Fig. 9. /24 anycast classification intersections and counts for the three protocols.

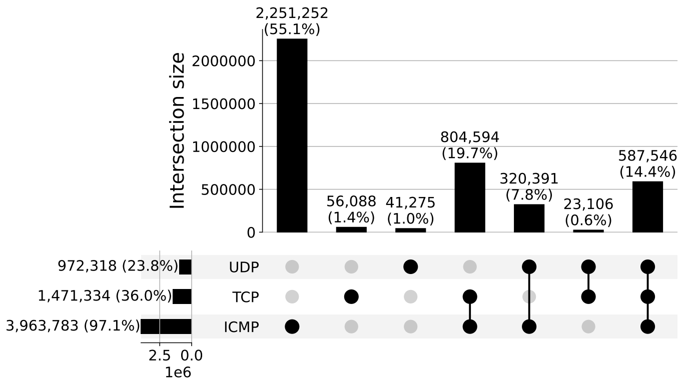

Fig. 8. /24 unicast classification intersections and counts for the three protocols.

we see the count and percentage of the hitlist for each intersection. 51.9% of the hitlist was unresponsive during the measurements, while 26.4% responded only to ICMP/ping. Very few responded solely to TCP (0.7%) or UDP (0.5%). 9.5% responded to both TCP and ICMP, 3.8% responded to both UDP and ICMP, 0.3% responded to both UDP and TCP, and finally, 6.9% responded to all protocols.

Interestingly, we observe that the bars that do not include ICMP only add up to 1.5%, indicating that ICMP/ping alone revealed the majority of the probed targets.

b) Unicast: Figure 8 shows that ICMP/ping identified 97.1% of the unicast addresses, with the majority (55.1%) of unicast addresses responding to ICMP only. By performing additional TCP and UDP scans, we discovered an additional 2.9% of unicast addresses, meaning they bring little value to discovering unicast prefixes. However, the intersections of these two protocols with ICMP are significant, indicating their value in providing more insights into the probed unicast targets. Specifically, one can run service-specific probes that make use of TCP and UDP to determine the type of services provided by these unicast prefixes.

c) Anycast: Figure 9 shows the overlap in discovered anycast prefixes between ICMP, TCP and UDP. It is important to note that these are potential anycast targets and may include false positives. Similar to the unicast graph, we observe that ICMP revealed the majority of anycast prefixes (83%). However, we did discover an additional 14% with TCP, which is not insignificant. In total, 51% of the anycast prefixes were responsive to TCP. This high responsiveness for TCP among anycast prefixes is surprising since TCP is conventionally considered unsuitable for anycast due to its stateful property. We suspect that many of these anycast prefixes implement TCP state sharing, which allows sharing of TCP sessions information between anycast sites, ensuring continuity for client TCP sessions when they switch to a different anycast site. Future work is to investigate the adoption of TCP state sharing.

Regarding UDP, the responsiveness among potential anycast targets is 34%, of which 2% were discoverable only with UDP. Among the anycast prefixes that were discoverable with UDP, 1,874 were found using DNS responses, this means that of the total 15,822 potential anycast /24 prefixes 11.84% respond to DNS queries, reaffirming our prior research on the common deployment of DNS with anycast. It remains unclear how many of the TCP and UDP anycast prefixes are true positives, as iGreedy is performed using ICMP. We leave modifying iGreedy to also work with TCP and UDP to future work.

	ICMP			TCP			UDP		
# of VPs	MAnycast^2	iGreedy	TP	MAnycast^2	iGreedy	TP	MAnycast^2	iGreedy	TP
2	3,241	1,726	53.26%	2,874	971	33.79%	979	557	56.89%
3	591	554	93.74%	228	88	38.60%	448	408	91.07%
4	820	811	98.90%	81	57	70.37%	434	407	93.78%
5	477	475	99.58%	90	90	100%	291	287	98.63%
6	1,633	1,629	99.76%	226	226	100.00%	240	240	100.00%
7	6,375	6,367	99.87%	4,488	4,479	99.80%	3,008	3,005	99.90%
Total anycast	13,137	11.562	88.01%	7.987	5.911	74.01%	5,400	4,904	90.81%

TABLE VII
THE NUMBER OF ANYCAST PREFIXES FOUND USING MANYCAST^2, WITH THE NUMBER CONFIRMED BY IGREEDY, AND THE TRUE POSITIVE (**TP**) RATE ASSUMING THAT IGREEDY IS THE GROUND TRUTH, FOR ALL THREE PROTOCOLS.

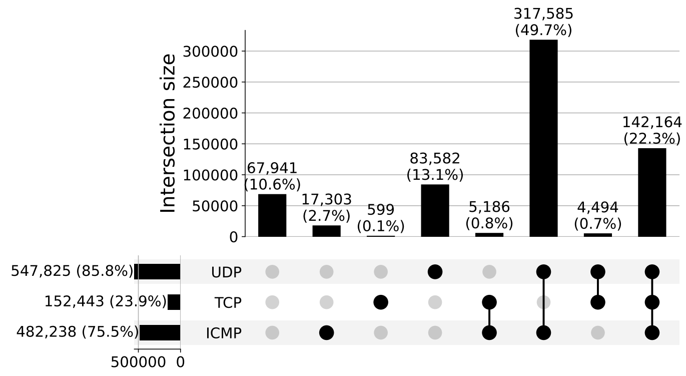

Fig. 10. Responsiveness distribution for the SLD hitlist across all three protocols.

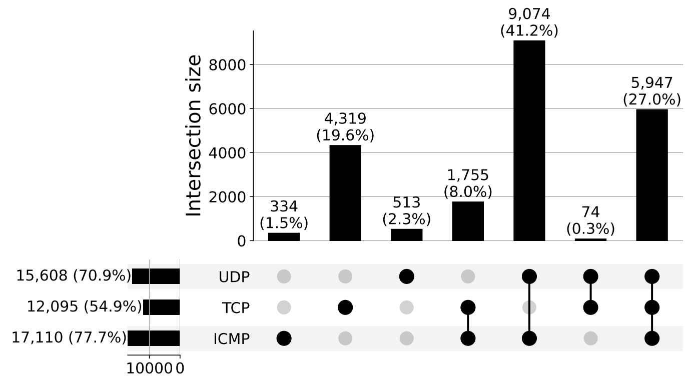

Fig. 11. Distribution of Anycast classification among the SLD hitlist across all three protocols.

For the anycast targets that were found, we ran iGreedy to validate the results. These results and TP rates can be seen in Table VII. It should be noted that iGreedy is performed with ICMP/ping packets, and some of the anycast targets that were revealed with TCP and/or UDP/DNS may not respond to ping. Therefore, the TP rates for these protocols are likely higher. Particularly, the TP rate for TCP is very low, which may attributed to anycast sites being responsive to TCP exclusively. Assume that the TP rate of our MAnycast^2 scan is consistent (which does seem to be the case based on repeated measurements) and given that that both our MAnycast^2 ICMP/ping and iGreedy measurement are reliant on ping, we can assume that the accuracy rate for ICMP (88%) also holds for TCP. This would mean that the missing 14% of the TCP anycast targets are in fact anycast prefixes that do not respond to ICMP.

2) Nameserver scan: Due to the very low responsiveness of UDP, and the even lower responsiveness of DNS (most UDP replies were ICMP port unreachable), we performed an additional anycast census. This scan involved probing a hitlist that contains Second-Level Domain (SLD) nameservers, obtained from OpenINTEL's SLD authoritative nameserver list from February 17, 2023 [21]. The hitlist consisted of 638,839 targets.

In contrast to the MAnycast^2 scan on the ISI hitlist, where the majority of responses to our DNS probes were ICMP port unreachable replies, we found that the vast majority of targets responded with DNS (98.5% of the UDP replies were DNS replies). Therefore, in this section, we will not distinguish between ICMP and DNS responses for this census.

Figure 10 shows the distribution of responsiveness for the nameservers scan across all three protocols. We observed a very high response rate, with only 10.6% being unresponsive. UDP exhibited the highest responsiveness at 85.8%, followed by ICMP at 75.5%. Similar to the previous scan, TCP showed a low responsiveness (23.9%), and TCP alone revealed very few additional targets (0.1%). Surprisingly, a significant portion (13.1%) of the nameservers only responded to UDP requests and did not respond to ICMP or TCP. This suggests that the ability to perform DNS probes holds value in discovering nameservers.

Figure 11 presents the distribution of potential anycast targets from the nameserver scan across all three protocols. Similar to the ISI scan, we found that ICMP is the most responsive (78%). Additionally, TCP alone revealed a substantial number of prefixes (20%) with has good responsiveness (55%). For UDP, we observed high responsiveness (71%), but only 2% were discoverable with UDP only.

We also conducted an ICMP iGreedy scan on these targets, as we did for the ISI scan, and obtained TP rates of 99.55% for ICMP, 86.71% for TCP, and 96.30% for UDP. Overall the TP rates were higher, which can be attributed to to the hitlist containing fewer unicast addresses that have a chance of ending up at multiple VPs.

When investigating the 20% that were classified as anycast by TCP alone, we found that for the majority of those prefixes the other protocols identified it as unicast. The ICMP iGreedy scan also revealed those prefixes as anycast, confirming that they are in fact anycast. We suspect that these anycast prefixes contain sites that only service users within a certain address space (region-specific). However, this is speculative, and further research is needed to butter understand this anomaly. Additionally, it is unclear why this region-specific responsiveness would not apply to TCP.

F. Limitations

The scans were conducted with 7 VPs, which is fewer than the 10 VPs used in the MAnycast^2 paper. The reduced number of VPs may lead to an increased false negative rate (i.e. anycast prefixes being falsely classified as unicast), as there is a higher chance of multiple anycast sites falling within the catchment of a single VP. Additionally, the geographical distribution of the sites is sub-optimal, especially us-sea and sg-sin had very high catchments due to a lack of VPs in Asia and the American continents.

Another limitation lies with the ISI hitlist used, which was approximately 3.5 to 4 months old at the time of our measurements. This time gap caused the overall responsiveness to be lower. Additionally, the hitlist was based on ICMP-responsiveness, further reducing the responsiveness of the UDP and TCP protocols.

VIII. Discussion

In this section, we will discuss various observations that we made during our measurements.

A. Variable port scanning

During our UDP/DNS and TCP measurements, we observed a significant increase in the number of addresses classified as anycast compared to the ICMP/ping measurements. In particular, a large proportion of the targets had their replies end up at two VPs. For TCP, we saw 3.74% of the targets reply to more than one VP, and for DNS we found 6.92% (excluding ICMP replies to our UDP probes as ICMP does not utilize port numbers). In contrast, the ping showed only 0.33% ending up at more than one VP. Since most of the addresses that were ending up at multiple targets for the DNS/TCP scans, were only ending up at a single target in our ping scan, we suspected that the port numbers were affecting routing behaviour.

To investigate this further, we conduction additional TCP and UDP scans using static ports in our probes. The results of these scans were notably different and aligned more closely with the results of the ping scan. With static ports, we found that 0.54% of TCP replies and 3.45% of DNS replies reached more than one VP. This finding supports our hypothesis that variable port numbers affect routing decisions, which leads to falsely classifying unicast addresses as anycast.

We also ran our measurements using only static source ports, which yielded identical results to the previous static port scan. From these measurements we conclude that only the source port in our probes (which are the destination port in the replies) lead to false negatives. This behaviour means our design choice to encode the client ID in the destination port number, for our TCP probes, will likely not affect the accuracy of the measurement.

Tables VIII and IX present the percentage of probed targets on the ISI hitlist who have their replies ending up at various numbers of VPs. Notably, at two VPs there is a very large difference between the static and variable port scan. However, as the number of VPs increases the effect becomes less noticeable, and for 4 or more VPs, both scans yield nearly identical results.

Number of VPs	Static	Variable
1	99.46%	96.26%
2	0.19%	3.09%
3	0.02%	0.27%
4	0.01%	0.07%
5	0.01%	0.01%
6	0.02%	0.02%
7	0.30%	0.30%

TABLE VIII
TCP - Percentage of TCP replies from unique source addresses that end up at X number of VPs for the static and variable source port scan.

Number of VPs	Static	Variable
1	96.55%	93.08%
2	0.90%	4.08%
3	0.68%	0.97%
4	0.59%	0.60%
5	0.49%	0.48%
6	0.33%	0.35%
7	0.45%	0.45%

TABLE IX
DNS - Percentage of DNS replies from unique source addresses that end up at X number of VPs for the static and variable source port scan.

The reason why the destination port in the replies affects routing decisions is likely due to load balancers employing equal-cost multi-path routing based on the destination port to determine the outgoing link. This means that when the destination ports are variable, replies are more likely to be sent out over different links at these load balancers. Conversely, with static destination ports it is more likely that replies will take the same path. Having these non-static paths is generally not problematic when there is a clear nearest VP, but for hosts that are located somewhere in the "middle" of multiple VPs, the chances of their replies ending up at multiple VPs are amplified.

Based on these observations, we conclude that the source port for outgoing UDP/TCP probes must be static when probing for anycast discovery. However, for measuring anycast catchments, it can be useful to have variable destination ports as it increases the likeliness of observing all possible catchments for addresses whose packets can end up at multiple anycast sites.

B. ICMP replies for UDP

As mentioned, we actively listen for ICMP packets during our UDP/DNS probes. These ICMP replies have provided us with a number of insights.

1) **Increased discoverability of unicast** - We discovered a significant number of unicast addresses that did not respond to ICMP/ping or TCP but replied with ICMP port unreachable to our UDP requests.
2) **Middle boxes** - In multiple scanned /24 prefixes that belong to a shared larger prefix, such as a /23, we observed that all requests targeted towards the larger

Host	ICMP	TCP	UDP (ICMP)	UDP (DNS)
au-syd	1.64%	1.33%	1.32%	3.04%
de-fra	15.17%	14.25%	15.64%	19.55%
fr-par	4.19%	4.21%	4.14%	5.89%
nl-ams	8.17%	7.21%	8.09%	11.04%
sg-sin	21.44%	26.11%	30.37%	20.46%
uk-lnd	12.44%	14.14%	13.59%	14.81%
us-sea	36.95%	32.76%	26.85%	25.21%

TABLE X
THE CATCHMENT DISTRIBUTIONS FOR THE VARIOUS PROTOCOLS.

prefix triggered replies from a single source address. This indicates middle boxes, such as a firewall, at the border of these prefixes, blocking our UDP requests and responding with ICMP.

3) **Network unreachable** - Our VPs received internal replies from Vultr, using addresses reserved for private networks. These replies indicated that there was no known route to our destination address.
4) **Rate-limited replies** - Whilst we typically received approximately seven replies for each probed target (as expected since we send seven packets to each target, one from each VP), we observed that ICMP network unreachable replies yielded only around four replies per probed target. When we conducted the scan where all targets are probed at the same time (rather than with one-second intervals), we obtained around 1.6 replies per probed target. This suggests that the targets or middleboxes likely implement rate-limiting, responding only once to UDP probes targeting the same address/port within a certain time period.
5) **ICMP codes** - As shown in Figure 6, we found that the most network unreachable responses were for "Port unreachable", indicating that the device is listening for UDP packets but not on our requested port. The second most prevalent code we encountered is "Communication administratively prohibited".

C. Catchments variable for the different protocols

The catchment distributions for the various protocols used in our measurements are shown in Table X. we compare the catchments of each VP for ICMP, TCP, ICMP responds to UDP/DNS probes, and DNS.

When examining the catchments distributions, we observe that the DNS catchment is an outlier where the catchments are more evenly distributed across the VPs. In contrast, ICMP, TCP, and UDP (ICMP) show similar catchment patterns, except for sg-sin and us-sea. The reasons for the discrepancy at these VPs are not entirely clear, but we suspect it may be due to a large number of targets in China that do not respond to ICMP/ping but do respond to TCP (likely due to the presence of the Chinese Great Firewall). These observations highlight the usefulness of these new probing methods for discovering hosts in these countries where ping-responsiveness is generally low.

IX. CONCLUSION

In conclusion, our research aimed to enhance and extend the MAnycast^2 measuring system to enable more accurate anycast censuses, providing a comprehensive understanding of the anycast landscape on the Internet. Through our efforts, we successfully developed a robust, resource-efficient, and easy-to-deploy measuring system with several notable features, including synchronous probing, client-selective probing, and additional probing methods such as TCP, UDP, and DNS.

One of our key contributions was the implementation of synchronous probing, which significantly improved the accuracy of the system compared to the previous MAnycast^2 version. By conducting comparative studies, we confirmed that synchronous probing yielded an impressive true positive (TP) rate of 88%. This rate represents a substantial improvement over the 38% TP rate achieved by the previous system. Furthermore, the changes made to the system to enable synchronous probing allowed us to perform the census using a much lower probing rate, effectively distributing traffic over time without compromising performance, thereby significantly reducing the strain on the measuring infrastructure.

Additionally, we performed an anycast census using TCP and UDP/DNS probing methods on the ISI hitlist. The results indicated that TCP probing identified an additional 2.2k potential anycast prefixes (14% of the total), while DNS probing revealed that 12% of the anycast prefixes were responsive to DNS requests. These findings underscored the value of these supplementary probing methods in identifying new anycast prefixes and conducting service discovery on anycast infrastructures.

In summary, our efficient and scalable MAnycast^2 measuring system can be readily deployed on low-cost virtual machines. It allows for responsible MAnycast^2 measurements with increased accuracy, making it suitable for performing frequent anycast censuses to obtain a longitudinal view of anycast usage on the Internet. Moreover, the system's extensibility allows for the easy addition of new features, such as more service-specific probes, for future work.

REFERENCES

[1] Hitesh Ballani and Paul Francis. "Towards a global IP anycast service". In: *ACM SIGCOMM Computer Communication Review* 35.4 (2005), pp. 301–312.
[2] M. Caesar and J. Rexford. "BGP routing policies in ISP networks". In: *IEEE Network* 19.6 (2005), pp. 5–11. DOI: 10.1109/MNET.2005.1541715.
[3] J Abley and K Lindqvist. *RFC 4786: Operation of anycast services*. 2006.
[4] Michael J Freedman, Karthik Lakshminarayanan, and David Mazieres. "OASIS: Anycast for Any Service." In: *NSDI*. Vol. 6. 2006, pp. 10–10.
[5] Yakov Rekhter, Tony Li, and Susan Hares. *A border gateway protocol 4 (BGP-4)*. Tech. rep. 2006.
[6] Xun Fan, John Heidemann, and Ramesh Govindan. "Evaluating anycast in the domain name system". In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 1681–1689.
[7] Danilo Cicalese et al. "A fistful of pings: Accurate and lightweight anycast enumeration and geolocation". In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE. 2015, pp. 2776–2784.

[8] Danilo Cicalese et al. “Characterizing IPv4 anycast adoption and deployment”. In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. 2015, pp. 1–13.

[9] Wouter B De Vries et al. “Broad and load-aware anycast mapping with verfploeter”. In: *Proceedings of the 2017 Internet Measurement Conference*. 2017, pp. 477–488.

[10] Lan Wei and John Heidemann. “Does anycast hang up on you?” In: *2017 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE. 2017, pp. 1–9.

[11] Danilo Cicalese and Dario Rossi. “A Longitudinal Study of IP Anycast”. In: *SIGCOMM Comput. Commun. Rev.* 48.1 (Apr. 2018), pp. 10–18. ISSN: 0146-4833. DOI: 10.1145/3211852.3211855. URL: https://doi.org/10.1145/3211852.3211855.

[12] Todd Arnold et al. “Beating BGP is Harder than we Thought”. In: *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. 2019, pp. 9–16.

[13] Rui Bian et al. “Towards passive analysis of anycast in global routing: Unintended impact of remote peering”. In: *ACM SIGCOMM Computer Communication Review* 49.3 (2019), pp. 18–25.

[14] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. “Taming anycast in the wild internet”. In: *Proceedings of the Internet Measurement Conference*. 2019, pp. 165–178.

[15] Raffaele Sommese et al. “Manycast2: Using anycast to measure anycast”. In: *Proceedings of the ACM Internet Measurement Conference*. 2020, pp. 456–463.

[16] USC/ISI. *USC/ISI ANT datasets*. 2020. URL: https://ant.isi.edu/datasets/ip_hitlists/ (visited on 01/20/2023).

[17] Wouter B. de Vries, Salmān Aljammāz, and Roland van Rijswijk-Deij. “Global-Scale Anycast Network Management with Verfploeter”. In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. 2020, pp. 1–9. DOI: 10.1109/NOMS47738.2020.9110449.

[18] Leandro M Bertholdo et al. “Tangled: A cooperative anycast testbed”. In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2021, pp. 766–771.

[19] RIPE Atlas. *RIPE Atlas docs - Credits*. 2022. URL: https://atlas.ripe.net/docs/getting-started/credits.html (visited on 01/20/2023).

[20] Remi Hendriks. *GitLab repository*. 2023. URL: https://gitlab.utwente.nl/s1978047/manycast-extended (visited on 06/12/2023).

[21] University of Twente. *OpenINTEL*. 2023. URL: https://www.openintel.nl/ (visited on 06/10/2023).

[22] Wikipedia.org. *UpSet plot wikipedia page*. 2023. URL: https://en.wikipedia.org/wiki/UpSet_Plot (visited on 06/10/2023).

Appendix A
Extended background

A. Anycast

As has been measured in the MAnycast^2 paper [15], anycast addresses make up at most 0.45% of the IPv4 range. Whilst such a small percentage may give the impression that anycast is insignificant, Internet users have a 50% chance to make use of an anycast address during daily Internet activities [11]. Anycast is used by content providers, content delivery networks (**CDNs**), cloud providers, DNS operators, operators of top- and second-level domains, and more to provide a large variety of Internet services.

Unlike unicast, which means a host exclusively using a unique IP address, anycast allows for multiple hosts to share a single address. Then, clients who contact such an anycast address get routed by the Border Gateway Protocol (**BGP**) to the nearest (in terms of routing) anycast instance. In this work we will infer whether /24 prefixes are anycast, since that is the smallest prefix that gets propagated by conventional BGP routers.

Anycast is desirable as it allows for geographic distribution of a service such that it provides:

- **Reduced latency** - Anycast enables geographic distribution of a service, resulting in reduced latency for clients by bringing content closer to them.
- **Load-balancing** - Anycast facilitates load-balancing by spreading traffic among multiple anycast sites, thereby distributing the burden and optimizing resource utilization.
- **Enhanced resilience** - Anycast enhances resilience by distributing traffic to multiple anycast sites based on traffic origin, mitigating the impact of attacks and disruptions. By leveraging BGP announcements, anycast can intelligently route traffic away from targeted sites, effectively deflecting Distributed Denial of Service (**DDoS**) attacks. Additionally, anycast architecture does not rely on a single point of failure, meaning that the availability of services remains partially intact even if a portion of the anycast infrastructure is affected by an attack or other issues.
- **Reliability** - Anycast ensures reliability by automatically routing clients to the next-nearest anycast site when an instance goes offline and its BGP announcements are withdrawn.
- **Localization** - Anycast allows manipulation of BGP announcements to limit visibility to select neighboring Autonomous Systems, enabling localization of service delivery.
- **Horizontal scaling** - Anycast provides a cost-effective and efficient method for replicating a service on the Internet, facilitating horizontal scaling.

Additionally, the possibilities and advantages of anycast have been amplified by anycast systems like PIAS and OASIS [1, 4], which allow for more fine-grained server selection mechanisms.

B. BGP

Anycast relies on the Border Gateway Protocol to perform its routing. Though there are rare exceptions to this rule, e.g. large privatized content providers using their private networks to route traffic. BGP was first proposed in 1989 as an improvement to the Exterior Gateway Protocol, since then there have been multiple versions and currently BGP-4 is widely used [5]. This routing mechanism operates on AS-level, an Autonomous System (**AS**) is a group of IP prefixes that are ran by one or more network operators that share a routing policy. These ASes are interconnected with a set of neighbouring ASes with their own connections, thereby creating a network of ASes. These ASes propagate announcements through this network by sharing their list of known AS links, along with a mapping of AS number to IP prefixes that are located in a specific AS. These announcements are done with BGP, and by listening to these announcements routers learn the topology of the network, which it keeps track of in a routing table. As mentioned, the shortest IP prefix that conventional BGP routers will accept for their routing tables, are those with size /24.

1) Path selection: When a client exchanges messages with an anycast address, its packets are routed through the Internet and BGP determines which anycast site will be selected for this client. BGP does this based on which anycast site is "closest" to the client (closest in terms of AS hops). But, there are exceptions to this rule. For instance, it might select a farther away path in terms of AS hops (which may mean getting connected to a different anycast site) based on certain policies.
These policies can be configured locally at each router. Whilst this makes BGP path selection very complex and unpredictable, there are commonly used path selection algorithms [2].

Firstly, there are certain business relationships that ISPs may have with each other (which determine the cost of exchanging packets):

- **Customer-provider** - One ISP pays another to forward its traffic.
- **Peer-peer** - Two ISPs have an agreement in place that allows them connect directly (often without payment).
- **Backup** - Two ISPs have a link between them that is only used when primary links are unavailable.

For these reasons, routers may select a longer path to avoid customer-provider links which are more expensive. Aside from economical reasons, a router may also select a path for load-balancing reasons, service quality reasons, security reasons, etc.

To facilitate these policies certain attributes can be added to BGP announcements. An example of this is the LocalPref value, which allows ISPs to give rankings to links.

2) Unpredictability: BGP routing does not always behave as expected, it often looks solely at the number of AS hops, but since AS hops can have greatly varying delays it is not always the case that the route with the least number of AS hops is the fastest [12]. Furthermore, the varying goals of ISPs and the freedom with regards to path selection policies make it difficult to predict how packets will be routed.

According to the operation of anycast services RFC [3], which describes the best practices for operating an anycast service, it is advised that anycast operators make use of the *NO_EXPORT*, *NOPEER*, and other policies in BGP announcements when they want an anycast node to be only visible in a limited set of ASes, i.e. the propagation scope. This also has an effect on anycast routing that may be difficult to measure, as it would require a vantage point inside of these propagation scopes to discover anycast instances that are hidden in this fashion.

3) BGP and anycast: The paper by Wei et al. investigated the stability of BGP when it comes to anycast routing [10], and whilst they found that anycast is mostly stable there are cases where *route flipping* occurs (i.e. a client's connection to an anycast address switching between different anycast sites). One of the causes for this phenomenon is that BGP routing can suddenly change due to maintenance, link failures, or load balancing reasons and these changes cause the catchments of anycast sites to be altered.

C. Mapping an anycast infrastructure

Since the Internet is quite unpredictable when it comes to routing, it is difficult for anycast operators to fine tune their deployments, e.g., for obtaining good load-balancing for DDoS attack mitigation. Furthermore, Internet routing is dynamic, which causes catchments of anycast sites (i.e., the set of clients who get routed to each anycast site) to be variable [3].

For these reasons, Verfploeter was developed as a state-of-the-art anycast measuring system for obtaining the catchments of an anycast deployment [9]. These catchments, combined with traffic history, can then be used, for example, to predict future load and determine the operational value of an anycast instance.

Unlike other anycast measuring techniques which probe an anycast infrastructure using a set of external vantage points that are controlled by the researcher, Verfploeter makes use of the devices on the public Internet by sending out ICMP requests and analysing the received ICMP responses, as visualized in Figure 2. This allows Verfploeter to utilize millions of devices, rather than a limited amount of vantage points. Furthermore, it provides coverage in areas where obtaining vantage points can be difficult, assuming the distribution of ICMP-responsive hosts to be mostly unbiased. For these reasons, Verfploeter is able to achieve higher resolution data.

Verfploeter works by sending ICMP Echo Requests to ICMP-responsive hosts on the Internet using the anycast source address. These hosts will then send a reply to the anycast address, which will be routed by BGP to the "nearest" anycast site. The catchments are then derived by looking at the source addresses of the replies received at each anycast site. This probing technique has been used for planning the B-Root domain name server and for measuring Cloudflare's anycast CDN [17]. Since Verfploeter uses the Internet itself to perform measurements, it is a low-cost solution to measuring anycast, unlike most other catchment measuring tools that require a large number of vantage points and generate far more traffic.

D. iGreedy

As mentioned, anycast is often used on the Internet to provide critical services such as the DNS and is employed by large content providers like Google and Amazon. This makes anycast a crucial component of the the Internet, and for good reason as it improves the resilience of a service. Therefore, to understand the resilience of the Internet as a whole, it is important to gather data on the usage of anycast. In this work, we utilize two methods to externally measure anycast. The first technique we employ is iGreedy.

iGreedy was introduced in 2015 by Cicalese et al. as an enumeration and geolocation method for anycast. It is based on the *Great-Circle Distance* (**GCD**) technique, which leverages the speed of light to determine the maximum geographical distance travelled by an Internet packet [7].

This method represents one the first works on anycast that is not limited to the DNS. It is a service-agnostic methodology that enables anycast service discovery. Moreover, it can accurately enumerate and geolocate anycast replicas at city-level, provided that there are enough VPs, by relying solely on latency measurements.

Using speed-of-light constraints, they determine the maximum distance between a vantage point and an anycast replica based on measured latency. Using this, and the GCD technique, they then draw circles around multiple VPs. Next, they enumerate the minimum number of anycast instances required to ensure that there is at least one anycast instance within each circle (a single anycast instance can satisfy multiple circles when they overlap). Geolocation is then accomplished by examining the cities where anycast instances may be located and making predictions based on the population of these cities. This workflow is repeated iteratively to identify the most probable geolocations and estimate the expected number of anycast nodes. Figure 3 illustrates their iterative workflow. To validate their methodology, they compare their results with ground truths obtained from DNS root servers that disclosed their deployments. They also validate their approach using DNS CHAOS queries. The study achieved a recall rate of 50% recall and city-level geolocation accuracy of 78%.

E. MAnycast^2

The second technique we utilize is MAnycast^2, developed by Sommese et al. [15]. It makes use of Verfploeter, combined with a set of geographically distributed anycast nodes provided by the Tangled anycast testbed [18], to identify anycast prefixes on the Internet.

As mentioned, Verfploeter probes addresses on the Internet using an anycast probing system. In the case of MAnycast^2, these probes are sent from every individual anycast vantage point. Under normal circumstances, when a probed address is unicast, there is a single device behind that address, which receives a probe from every anycast node. Each of these probes generates a reply, which is directed to the nearest anycast instance. This means, that in the case of the probed address being unicast, all its replies will end up at the nearest VP that uses this shared anycast address.

However, if the probed target is also anycast, the requests from the VPs will reach different destination nodes that share the same destination anycast address. Consequently, each of these destination nodes that receives the ICMP Echo Request will generate a reply, which will be routed to that node's nearest VP. This means that multiple VPs will receive replies in the case of an anycast target.
Therefore, all responses being received by a single VP implies that the destination address was unicast. Conversely, if responses are received by mutliple VPs, it indicates that the destination address is anycast. Figure 1 illustrates this core principle and showcases both cases.

However, there are corner cases where this principle fails.

- **False negatives** - An anycast address can be mistaken as unicast when all anycast replies are routed to a single VP. This can happen when the anycast system is deployed regionally and there is only a single VP in that region.
- **False positives** - A unicast address may be mistaken for anycast when the unicast replies are received by multiple VPs. This is no rare occurrence and happens due to dynamic routing in the Internet. For instance, if a unicast address is located in the "middle" of two VPs, such that BGP has a path to both VPs of near-equal length, its reply may reach both VPs due to load-balancing decisions made by the routing system over time.

In particular, the situation of unicast addresses ending up at 2 or 3 VPs (due to equal-cost paths) is relatively common. Therefore, in the MAnycast^2 paper, they classify an address as anycast when 4 or more VPs receive replies, while they are classified as unicast when all replies are captured at a single VP. However, when 2 or 3 VPs receive responses, there is uncertainty whether it is anycast or unicast. For these reasons, additional iGreedy measurements are performed on the potential anycast targets. This phenomenon was also observed in the anycast catchment measurements with Verfploeter by De Vries et al. [9], where they observed that due to routing flexibility, clients may utilize different anycast instances at different times.

The main advantages of MAnycast^2, compared to other anycast measuring methodologies, are that it is much faster and it does not require a large probing network, making it a cost-effective tool for monitoring anycast deployments. Moreover, the set of ping-responsive hosts that it sources on the Internet is far larger than the number vantage points that can be offered by any measuring platform. By utilizing fewer VPs for sending out probes, it mitigates the risk of network overload and potential strain on the measuring infrastructure. These properties render MAnycast^2 suitable for conducting repeated measurements, providing longitudinal data on anycast usage across the Internet.

F. UDP & TCP

With anycast there is no guarantee that when a client exchanges packets with an anycast address that all exchanges is with the same anycast instance. For this reason, anycast is mostly used for stateless services such as DNS.

As mentioned, we will be expanding MAnycast^2 with additional probing tools, these will be UDP and TCP probes (till now it has only been performing ICMP probes). UDP is a stateless protocol and can be probed just like ICMP, which is also stateless. However, TCP is stateful which may seem counter-intuitive to anycast. Yet it does get used in combination with anycast, to give an example, there has been efforts made for providing DNS-over-TLS and DNS-over-HTTPS (both require TCP) for security reasons.

Appendix B
Spoofing

In the results of our DNS probing (using the ISI hitlist) we see 11k responses from Google's public DNS (address: 8.8.8.8), this is due to probed targets forwarding our packet to Google's DNS server. When this happens there are two cases; either they spoof our address as source in the forwarded request such that Google replies to us immediately, or they forward it using their own source address and then forward the reply to us. The first case we detect as we see the destination address of our probe not matching up with the source address of the reply, however the second case is undetectable for our system as of now. However, it can be detected if DNS A Record Requests are sent using a domain owned by the researchers, we leave this for future work.

We see this same behaviour for Cloudflare (1.1.1.1) approximately 2k, Google (8.8.4.4) approximately 800, Cisco's OpenDNS (208.67.222.222) approximately 500. In total we found 24k spoofed replies from 1.1k unique source addresses for DNS.

For the UDP probed targets that responded with ICMP, we found 1.3 million spoofed replies from 164k unique source addresses. This is largely due to middle boxes blocking our DNS probes and replying with e.g. destination port unreachable using their own source address. We also capture a large number of replies from very few source addresses that indicate they have no path to our desired target address, most of these are privately reserved addresses which indicate they are from within the Vultr sites from which we probe. For both cases of UDP responses we see that a significant part are spoofed packets when comparing to the other measurements.

For our ping probes we saw 23k spoofed replies from 2.7k unique source addresses, and for TCP we are unaware of spoofed packets as we cannot see the destination address of the probe in the captured replies. Though we did capture 2.7k replies from 389 unique source addresses that were not on our hitlist. Therefore, we know there was at least 2.7k spoofed replies in our TCP measurement.

Spoofing is often used in Internet attacks (e.g. DDoS attacks), and for this reason many Autonomous Systems (**AS**) detect and block outgoing spoofed traffic (since they know the possible source addresses within their AS). Future work is to share our data with AS operators.