MORP4: A Dynamic Network Telescope

Iliana Xygkou*, Jithin K. Sojan*, Dhruv Rauthan*, Feng Zhu*,
Thomas Holterbach*,†, Shane Alcock‡, Brian Flanagan*, Ahmed Saeed*, Alberto Dainotti*
*Georgia Institute of Technology †University of Strasbourg ‡Alcock Network Intelligence Ltd

Abstract

A *network telescope* passively monitors traffic reaching Internet address space that is not assigned to any hosts but is advertised to the global routing system. This traffic is by definition *unsolicited*. For more than two decades, network telescopes have enabled research breakthroughs by allowing global visibility into a wide range of Internet phenomena. However, telescopes are afflicted by two main issues: progressive erosion, due to the increasing scarcity and commercial value of address space, and blacklisting. To overcome these issues, we propose MORP4, a programmable data-plane framework implementing a "dynamic" network telescope. MORP4 accurately and adaptively tracks unused space of an organization's network with configurable time and space granularity and captures only traffic directed towards unused addresses at line rate. We provide an implementation in P4 and Python/C++, and deploy it on a Tofino switch. We show that it can detect unused IPv4 address space at the finest granularity (/32) while operating at line rate as well as providing an effective approach for operating a telescope in the IPv6 domain.

1 Introduction

A network telescope, or simply *telescope*, is a research infrastructure also used for cybersecurity operations. It passively monitors traffic reaching Internet address space that is not assigned to any hosts but is advertised to the global routing system (*i.e., dark address space*). This traffic is by definition *unsolicited* (also known as Internet background radiation—IBR) and is constituted of an evolving mix of diverse traffic components originating from across the whole Internet [1].

For more than two decades, network telescope instrumentation has enabled significant research breakthroughs by allowing global visibility into a wide range of Internet phenomena [2]: the automated spread of malicious software such as Internet worms or viruses [3–6]; random spoofed source denial-of-service attacks [7]; large-scale botnet activities [8, 9]; macroscopic Internet blackouts due to natural disasters [10], network failures [11] and state censorship [12]; trends in IPv4 address space utilization [13, 14]; bugs and misconfigurations in popular applications [1], etc.

Problem: The telescopes' address space is continuously reduced and blacklisted. Two major issues affect this type of infrastructure broadly. First, due to the increasing scarcity and commercial value of IPv4 address space, the size of (even the largest) telescopes has been progressively eroding over the years [2]. As a result, some organizations have stopped operating them or reduced them to sizes that severely impact their research and educational utility [15, 16]. Another major issue experienced by telescopes is blacklisting by malicious actors: lists of network telescope address blocks have been circulating [17] and are used by these actors and hardcoded in malware to avoid probing them—again, causing a loss of the infrastructure's utility.

Key observation: Organizations' address space is *lightly* utilized. Many organizations tend to have subnet blocks that are internally assigned but are in fact *sparsely* used. Moreover, organizations interested in operating network telescopes—*e.g.,* academic and research institutions—often own significant portions of IPv4 address space. This phenomenon is particularly evident in the US, where many research institutions received generous IPv4 address block allocations in the early life of the Internet. Figure 1 shows that almost 40% of autonomous systems (ASes) for US academic organizations originate the equivalent of a /16 IPv4 block ($\approx 65k$ addresses) in BGP[1]. We analyzed one week of unsampled NetFlow records from our campus network's border traffic (2025/04/01 - 2025/04/07), filtered out spoofed addresses based on the methodology of Section 6 and found that almost 57% of its address space did not appear in any flow during that week. These addresses might be either completely unused but allocated to organization's units[2], or silent only for certain periods of time, and thus in either case not suitable to statically assign to a telescope. Sparse address space utilization is a phenomenon well known to network engineers, especially in university networks, and contrasts with the increasing unavailability of unused blocks/addresses for dedicated allocation to network telescopes.

MORP4 *dynamically* detects unused IP space. To overcome these issues, we propose MORP4, a programmable data-plane framework implementing what we call a *dynamic* network telescope. I.e., a system that, through traffic monitoring, automatically infers which portions of its network are/become unutilized in order to maximize capture of unsolicited traffic. In contrast, in a *static* telescope, the decision of what space to monitor for unsolicited traffic is typically manual and rarely updated, and thus often limited to largely (if not entirely) unutilized blocks. The first leverages intermittent utilization of space over time as well as a fine spatial granularity in identifying pockets of unused addresses, with its size constantly adjusting. The latter focuses on clearly unused blocks and

[1] 874 ASNs associated with the tag "Academic" and located in the US according to bgp.tools.

[2] This includes the rare case where public addresses are used internally but not routed towards the rest of the Internet.

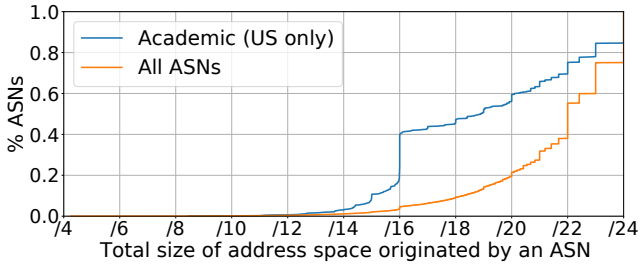

Figure 1: Distribution of the reachable address space of (a) academic organizations in the U.S. (#ASNs=847) and (b) all ASNs (#ASNs=76,646). Around 96.8% of (a) advertise on BGP less space than a /14, while only AS7377 (UCSD) advertises more space than a /10. Similarly, around 98.8% of (b) announce on BGP less space than a /14, whereas only 121 ASNs (0.16%) announce more space than a /10.

its size seldom changes. Our solution is based on deploying programmable switches at the ingress/egress points of the organization's network to adaptively track—in real time and at line rate—its unused space with configurable time and space granularity, and capture only traffic directed towards unused addresses. MORP4 enables an organization not only to "recover" unused space for capturing unsolicited traffic, but also counters blacklisting, since it dynamically monitors addresses that are intermittently used or mixed with used ones in the same address block. Furthermore, it enables the telescope concept to be applied in the IPv6 domain, where a traditional static telescope would not be effective at collecting IPv6 IBR.

Key challenge. When deploying our dynamic network telescope, a key challenge arises. A network telescope must consist only of unused addresses. However, in a dynamic telescope, some addresses that were not previously in use may suddenly become active and start exchanging traffic with entities outside of the organization's network. This traffic is confidential and out of the scope of a telescope's capturing purposes. Thus, a *correct* implementation should under no circumstances capture it. MORP4 addresses this issue through a programmable data-plane design that couples *(i)* address utilization inference and *(ii)* look-up + capture decision making in the same hardware. This solution guarantees correctness when a network's ingress/egress links can be monitored in a single location. Since this scenario is likely for most university campuses but not for every network, we also extend our solution to a distributed scenario, where we can achieve correctness under reasonable and realistic constraints.

Main contributions. Our main contributions are:

- A discussion of the limitations of network telescope state of the art, and the practicality and benefits of dynamic telescopes.
- An approach for accurate dynamic detection of unused IP space within a network; MORP4's data- and control-plane design to *(i)* monitor specified network space, *(ii)* dynamically detect which addresses are unused, and *(iii)* log unsolicited traffic towards unused addresses.
- An open-source implementation[3] of MORP4 in $P4_{16}$ for BMv2, Tofino-1 and Tofino-2 for detection of unused space at the finest IPv4 granularity (/32) and at /56 IPv6 granularity, while operating at line rate and leaving enough resources for other applications running on those switches.
- Preliminary insights from a real-world deployment of MORP4 for both IPv4 and IPv6 in an academic network.
- An implementation and deployment of our dynamic telescope in IPv6, able to capture vast amounts of IPv6 IBR traffic.

Address granularity	24	26	28	30	32
% unused	0.15%	9.47%	24.11%	38.14%	56.53%

Table 1: Analysis of the potentially unused IP addresses in our campus research and educational network which consists of (non-darknet) 165,892 IPv4 addresses (≈1.2x a /15 block) allocated over a few non-contiguous blocks. The first row denotes the address granularity at which we process the subnets to discover inactive ones, and the second row provides the portion of the total subnets which are found inactive.

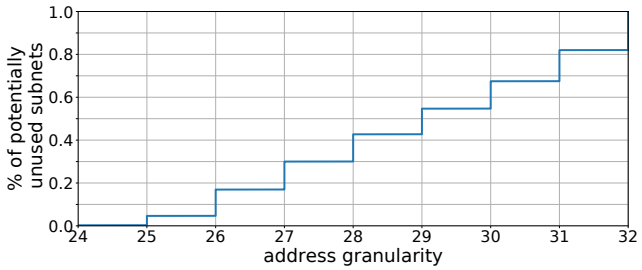

Figure 2: Analysis of the marginal gain in potentially unused IP addresses in the network of Table 1. We consistently discover higher proportion of the total unused space when we increase the processing address granularity.

2 Background and Motivation

2.1 Background

Network telescopes are not only precious research infrastructure [2], they also continue to be broadly used for security awareness and operations and are deployed across a range of diverse network types [18–21]. Research shows that diversity and size of telescopes have important implications on their ability to observe IBR and thus on the type and quality of information that can be extracted. For example, larger telescopes enable Internet outage monitoring and detection by observing, in IBR, sufficient distinct source IP addresses within a short time bin (*e.g.,* 10 min.) that originate from each ASN or from each subnational region, worldwide [22]. Large

[3] https://github.com/InetIntel/morp4-dynamic-telescope.

telescopes, like the UCSD Network Telescope [2], can collect data from enough distinct source IPs of botnets to support the rapid identification of pseudo-random patterns in their packet headers. This enables researchers to study botnet behavior as well as develop fingerprints for individual bots in some cases [6, 8, 9]. Benson et al. also show that a telescope size is proportional to its ability to be used to perform a variety of Internet-wide passive measurements [1]. Recent studies also highlight the benefits of a diverse set of addresses to monitor, *e.g.,* showing that scanning activities (precursors of full-blown exploitation attempts) often exhibit geographic locality [23–26]. Another indication of the importance of IBR for research purposes is that researchers have often gone to a great extent to permanently archive IBR traffic. For instance, CAIDA has archived a few petabytes of compressed "pcaps" of IBR traffic covering more than a decade [2]. In other cases, where archival costs were prohibitive, researchers have developed IBR-specific information reduction approaches [27, 28]. This suggests that it is preferable to capture more traffic from more monitored IP addresses and then compress it with some information loss, rather than reduce the amount of traffic captured by *e.g.,* reducing the size of a telescope.

However, large telescopes are becoming increasingly difficult to deploy in practice. Historically, research initiatives have relied on specialized infrastructure to monitor traffic directed toward large unutilized IPv4 address blocks [29, 30]. Yet, the growing rarity and commercialization of IPv4 addresses have led to a gradual reduction in the scope of even the most extensive network telescopes. Notable examples include the UCSD Network Telescope [2, 31] and the Merit Network telescope [32]. The former, initially close to a /8 in size, has seen its capacity decrease as parts of its address block were allocated or sold; the latter shrank from a /8 to about a /13 due to consistent allocations. Moreover, as mentioned earlier, large dark address blocks can be subject to blacklisting, limiting their value to studying global Internet phenomena.

IPv6 telescopes do not suffer from the address rarity problem, but they face their own unique challenge: due to the immense size of the IPv6 space, which is mostly unutilized, IPv6 scanners target active IPv6 address ranges using so-called hitlists [33]. Thus, IPv6 telescopes that rely on static and unused address ranges have limited efficacy and are of scarce utility [34].

Despite the rarity of IPv4 addresses, the reality is that their utilization is fragmented in both space and time [13, 14, 35]. Utilization fragmentation of IP addresses implies the existence of opportunities to create telescopes out of unused fragments. However, identifying fragments is challenging in practice. For instance, control over an address space is typically distributed among teams within an organization, making it challenging to track unassigned and unused addresses. One approach to overcome this challenge is to reserve blocks within the fragmented address space to be used by telescopes. However, resource reservation is challenging for both bureaucratic and technical reasons. In particular, it is challenging to identify teams that should (and are willing to) relinquish parts of their allocated address space. Moreover, the need for addresses in an organization is typically dynamic, which makes it wasteful to reserve address blocks that cannot be used at a later time. Introducing dynamicity into a network telescope creates the opportunity to leverage also any fragmentation in the time dimension—*i.e.,* the fact that addresses might be actively utilized only for certain periods of time. In this paper, we advocate for continuously tracking, *at line rate*, unused fragments of an organization's address space to use them as a part of a dynamically-sized telescope.

2.2 Why Develop a Dynamic Telescope

Expands the size of a telescope. This approach allows an organization to potentially leverage *all* unused fragments of their address space as part of a telescope. Using the same NetFlow records dataset described in Section 1, in Table 1 we show that (for the week we analyzed) the hypothetical gain of unused address space when identifying "silent" subnets increases with the address block granularity of our search. In Section 6, we show results from a first experimental deployment of MORP4 in our academic network and we show that, while the network engineers had dedicated only 5 /24s to their traditional telescope, with MORP4 they now monitor IBR towards a daily average of 100 times more /24s.

More robust to blacklisting in IPv4 and a natural solution to effectively capture IPv6 IBR. A dynamic telescope monitors space that is changing over time, largely fragmented in many blocks of different size, and likely intermingled with actively used addresses (*i.e.,* the actual targets of malicious actors), thus making it practically infeasible for attackers to blacklist. Depending on the configuration and the specific scenario, a dynamic telescope will also monitor addresses previously used (*e.g.,* a few hours or a week before), again rendering blacklisting practically impossible. For IPv6, by monitoring inactive addresses that are part of subnets that also contain active addresses, a dynamic telescope is able to capture scanning traffic generated using IPv6 address hitlists—which is typically the method for undertaking IPv6 scanning [34, 36]. In Section 6, we show how our IPv6 dynamic telescope captures from partially active address blocks two orders of magnitude more IBR traffic than from a much larger number of entirely inactive blocks.

Can generate more scientifically and operationally valuable data. By monitoring a mix of *(i)* largely unused address blocks—and thus less interesting to attackers based on previous scans—and *(ii)* addresses part of populated address blocks, a dynamic telescope not only captures relevant IBR that other telescopes might miss, but also offers the opportunity to continuously compare IBR directed towards these

two different categories of space. This data labeling offers a useful pre-classification of IBR, potentially separating noisy traffic from more interesting traffic.

2.3 Why a programmable data-plane solution

There is no specialized software solution that can meet our objectives out of the box—*i.e.,* implement a dynamic network telescope with simple configurations. Moreover, we argue that such a software solution would not be ideal, even if it might be feasible to implement. A dynamic telescope needs to perform two essential functions at wire speed: *(i)* detect and mark internal active IP addresses by monitoring outbound traffic, and *(ii)* look up inbound packets and capture those that are destined to internal inactive addresses.[4] These two operations require, respectively, a write and a read on a shared state (per internal IP address) that need to be performed in the order in which packets enter the switch. This is because when a new host becomes active (with a previously unutilized IP address) the state change triggered by its first outbound packet must occur *before* we attempt lookups for any subsequent inbound packets destined to that host (expected within *O(ms)*). Decoupling these two functions would introduce a synchronization problem that could lead to capturing traffic towards an active IP address—which, as mentioned in Section 1 needs to be carefully avoided.[5] Instead, performing these operations in the data plane, at line rate, is a natural solution that guarantees such operations are performed in order. Doing the same (coupling) in software would require these two functionalities to be implemented within the same thread in order to avoid non-deterministic delays caused by OS scheduling. This model does not fit well a software-based architecture, where—to cope with line rates potentially in the order of Tbps—we would have to incorporate high parallelism and specialized NICs as well as a load distribution node, thus increasing complexity as well as the cost of both the deployment and maintenance of such a system.

While we can avoid synchronization issues in a scenario with all ingress/egress traffic monitored in one location—*i.e.,* one or multiple ingress/egress links, but with all taps feeding into a single switch—we cannot avoid the presence of a different distributed problem when a network presents ingress/egress points in multiple locations, thus requiring multiple switches. In such case, due to asymmetric routing (*e.g.,* the possibility that the first outbound packet and the subsequent inbound packet will traverse two different switches), the above-mentioned state needs to be synchronized across all switches. However, the challenges introduced by this scenario are relatively easy to control, since they are tied to the latency and packet loss experienced by packets traveling within the network. In Sections 3.2 and 5 we show we can achieve correctness when assuming reasonable upper bounds for packet loss and delay (through buffering and limited intelligence in the capturing host and sync messages between switches as well as the capturing host).

[4] By contrast, marking an address as *inactive* can be performed with time dynamics of hours or days. We thus implemented it in an external controller.

[5] We operate under the assumption that IBR packets are sent only to monitored addresses that do not generate any outbound traffic (see Sec. 8).

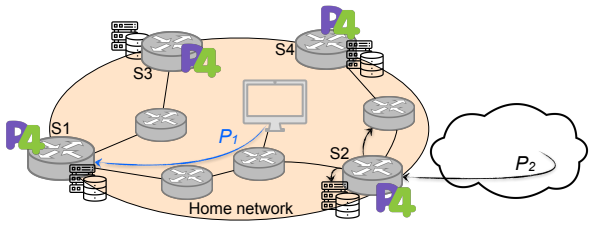

Figure 3: Overview of MORP4. P1 is an outbound packet. P2 is an incoming packet destined towards an unused address.

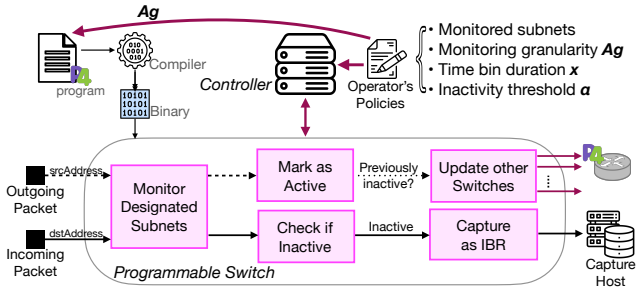

Figure 4: Workflow of a packet traversing MORP4. The operator's policies define which prefixes should be monitored. (a) Outgoing packet: The switch checks whether the source IP address belongs to the monitored space and if so, it will set the state of the corresponding $/A_g$ subnet to active and will notify the rest of the MORP4 switches. (b) Incoming packet: The switch checks whether the destination IP address belongs to the monitored space and if so, it will examine whether it's destined to an active or inactive subnet. In the latter case, a copy of the packet is sent to the capture host.

Finally, working with programmable hardware that offers guarantees of operating directly in the data plane at line rate, enables the easy introduction of other features. For example, we could introduce a feature to mitigate the impact of a DoS flooding attack against our telescope infrastructure (other telescopes experience these [2]), where the switch rate-limits the traffic that is forwarded to the capturing host while selectively sampling a relevant portion of IBR to capture (we present a sample solution in Appendix B).

3 MORP4: System Design

Figure 3 shows an overview of the deployment of MORP4. Programmable switch(es) are deployed at the ingress and egress points of the network to observe the traffic exchanged between that network and the rest of the Internet. MORP4 tracks the activity of IP addresses, at a user-defined subnet

granularity. A subnet can either be *active* or *inactive*: we mark a subnet as active when we observe at least one of its hosts generating traffic and inactive otherwise. Traffic destined to inactive subnets is directed to a *capture* host through a dedicated monitoring port at the switch. Programmable switches and capture hosts constitute the data plane of MORP4. On the other hand, the control plane runs on an SDN controller which maintains the set of inactive addresses over time and keeps the switches synchronized. The workflow of MORP4 is shown in Figure 4. Each switch tracks all monitored subnets, relying on the controller to determine if a subnet has been inactive for a configurable time duration. Upon detecting a packet generated by an inactive subnet, the switch changes its local state and updates all other switches.

To reduce the overhead of tracking subnets, detection logic is distributed between switches and the controller. Naively, each switch could independently declare a subnet inactive by tracking the last time it generated packets. However, such an approach would require individual switches to maintain a large amount of state. Instead, we discretize time into short bins of length x, with each switch tracking one bit per subnet to indicate whether the subnet has been active or inactive during that time bin. The controller collects the state of all subnets from all switches at the end of each time bin, allowing the controller to track subnet activity over longer period of time. The controller can declare a subnet inactive if it has consistently been labeled inactive for a timeout duration $T_o >> x$. Each switch keeps track of two flags per subnet: *Long Term State (LTS)*, which indicates whether the subnet had any activity over the past T_o seconds, and *Short Term State (STS)*, which indicates whether the subnet has been active over the past x seconds. A switch logs IBR traffic *iff* both bits are 0, which is determined based on the collective state of all switches tracked by the controller. Tracking distributed state can create race conditions. We first describe the behavior of MORP4 when it has only one switch (Section 3.1). Then, in Section 3.2, we discuss how MORP4 scales to many switches and how we mitigate the effect of race conditions.

3.1 Single-switch scenario

MORP4 splits responsibilities between the controller and the switch: in a nutshell, the controller makes the decision to turn an active address to inactive, whereas the switch is responsible for detecting when an inactive address should be turned to active. Figure 5 shows how the switch and controller interact with each other in a time-binned fashion to coordinate and achieve adjustable monitoring granularity. Initially, the LTS of all subnets is set to 1, preventing MORP4 from logging any traffic at initialization. On the other hand, we set STS to 0 for all subnets. If a switch detects traffic generated by a subnet, the switch sets the LTS and STS of the subnet to 1. The controller periodically checks for any activity from each address by querying the switch in discrete timebins of

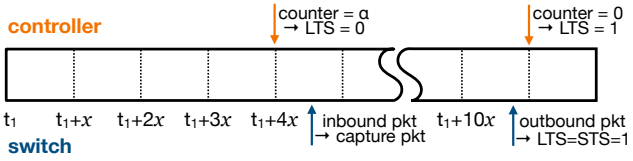

Figure 5: Evolution of the state of an address where $\alpha = 4$ ($T_o = \alpha \cdot x$). After α timebins of no record of outgoing packets, the controller declares the address as inactive. As a result, the switch starts logging incoming packets towards that address. However, when the switch observes at least one outgoing packet, it will immediately set the state of the address to active. Finally, the controller retrieves the STS value for that timebin; since it is active, the controller resets its counter and, because the address was previously inactive, it sets LTS = 1.

x seconds each, where x is fixed and configurable (*e.g.,* $100s$ in our deployment). The controller retrieves the STS bit of all addresses every time bin, and resets the STS at the switch only if it was set, which enables the switch to track activity during the new timebin. Moreover, the controller sets the LTS of a subnet to 0 if its STS was 0 for α consecutive time bins, where $\alpha = T_o/x$. When a previously inactive subnet becomes active (STS = 1), the controller sets LTS = 1 to prevent race conditions.

A subnet that has its LTS and STS set to 0 is considered inactive. Traffic directed to an inactive address is unsolicited and thus IBR. However, since an address might become active at any point in time, the switch will immediately set both LTS and STS to 1—and stop logging any inbound traffic towards it—as soon as it observes a packet originating from it. On the one hand, the delay in declaring a subnet inactive has a maximum value of $T_o + x + \varepsilon$, where ε is the communication delay between the controller and the switch. This latency is configurable and acceptable, since for telescope applications it is not critical that addresses are added to the inactive pool quickly. An operator might instead want to be conservative in picking a T_o large enough to avoid logging packets somehow related to the activity of the host that was using an address before becoming inactive. On the other hand, by coupling activity detection and packet capture in the same data plane, our design ensures that an active subnet is detected at line rate, *i.e.,* before the triggering packet leaves the switch. In a scenario with only one switch, this configuration guarantees correctness, *i.e.,* that incoming response packets to a host that suddenly started using a previously inactive IP address, are *not* logged by the switch.

In the Appendix, in Figure 15, we show the FSM describing the transitions between the MORP4 states of a monitored subnet. We define a system state as (c_state, LTS, STS), where c_state is the state of the subnet in the controller. We highlight two intentional design choices of MORP4: *(i)* the system cannot reach the state $(1,0,0)$ in which the controller con-

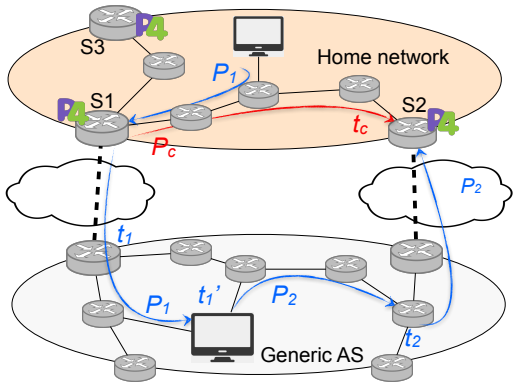

Figure 6: In the multi-switch scenario MORP4 switches send control packets to prevent capturing non-IBR traffic in the case of asymmetric routing.

siders the subnet active but the switch does not,[6] and *(ii)* the switch ensures that STS will always be set to 1 at the exit of a triggering packet regardless of when the switch receives the packet. In the latter case, the most intricate scenario is if the switch receives the triggering packet when the controller has received STS for α timebins, and is in the process of resetting the subnet's LTS. To prevent the controller from "erasing" this activation, we reset STS only if it was previously set (LTS may still be reset by the controller but it will be set again in the next timebin when the controller receives STS = 1).[7]

3.2 Multi-switch scenario

Networks with multiple ingress/egress points will require the deployment of a MORP4 switch at each ingress/egress point. We illustrated in Section 3.1 that as long as both outbound and inbound traffic traverses a single switch, no incoming response packets will be logged by the *switch*. However, in a multi-switch scenario, it is now necessary to keep the state of all subnets synchronized across all switches, especially when considering that a packet might egress the network from one point and ingress from another point. It is thus necessary to update every switch's state for that address to active *before* they might observe an inbound packet due to the address' sudden activity.

We address this problematic case with a strategy that errs on the side of potentially missing some IBR traffic to ensure that we avoid logging non-IBR traffic. The strategy has two elements: *(i)* the switches send control packets at line-rate in the data plane without involving the controller as soon as a subnet becomes active, and *(ii)* the capture host holds IBR traffic in memory for a configurable amount of time and records it as IBR only after it has ensured that the packets are destined towards an inactive subnet.

Our basic approach is to allow each MORP4 switch to notify other switches directly through a special control packet when a subnet becomes active. In particular, upon detecting a packet generated by a host in the subnet, a switch will set the LTS and STS of the subnet to 1. Moreover, it will send control messages to all other switches which will set their LTS and STS to 1 for that subnet.

We illustrate this mechanism through a sample scenario in Figure 6: the control packet p_c is generated directly by the switch $S1$ and is sent at the same time that the packet p_1 (which triggered the detection) is forwarded. The assumption is that the sum of the time t_1 for p_1 to reach a host outside of the monitored network (which will traverse at least one inter-domain link), the time t_1' for the destination host to process p_1 and send a response packet p_2, and the time t_2 for the corresponding response packet p_2 to reach back any of the other MORP4 switches (again, at least another inter-domain link), *e.g.,* $S2$, is longer than the time t_c for p_c to travel from $S1$ to $S2$. In our implementation we send p_c three times to take into account accidental packet loss. While the likelihood that $t_1 + t_1' + t_2 < t_c$ is remote, there is no mathematical guarantee it will not happen—*e.g.,* because of extreme delays or packet loss within the home network. To account for the possible delays and loss, each capture host holds IBR traffic in memory for a configurable amount of time. If a subnet becomes active, the capture host is notified and purges all in-memory packets destined to the active subnet. Once that delay expires, IBR traffic is permanently stored at the capture host. We note that only asymmetric communication (outbound traffic through one switch and inbound traffic through a different switch) calls for this IBR buffering solution.

Finally, in a multi-switch scenario, at each time bin, the controller will read the STS associated with each address from each MORP4 switch in order to keep track of the addresses use over time: if at least one switch observed outgoing packets from an address, the controller will reset its timeout counter and switch its corresponding state to active if previously inactive. Conversely, if there have not been any outgoing packets for α time bins in any switches, the controller will force the address state to inactive in all switches. While delayed logging allows us to handle most race conditions, there is one particular scenario that requires careful consideration.

Consider the scenario in Figure 7. In particular, the controller collects all states from all switches after which an inactive subnet becomes active. While the switches are notified, the controller is not notified. Thus, the controller sends its update message to set the LTS state of the subnet to 0 (*i.e.,* it is inactive). MORP4 handles this inconsistency in the state of the subnet by only allowing the controller to reset the STS of

[6] In a given timebin x_i (or transition to x_i), *c_state* can be equal to 1 in only two cases:
1. STS = 1 in x_{i-1} and STS will be reset only after $c\_state = 1$ following the transition path $(0,1/0,1) \rightarrow (1,1/0,1) \rightarrow (1,1,1) \rightarrow (1,1,0)$, or
2. $c\_state = 1$ in x_{i-1} implying that LTS = 1 in x_{i-1}.

[7] This will have an effect only when there is no other packet from that subnet for the next α timebins and we delay considering it inactive by one timebin.

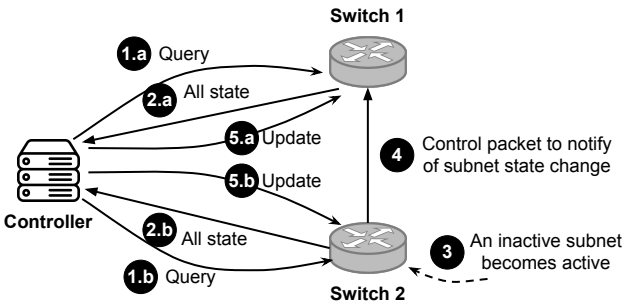

Figure 7: Messages between the controller and two switches. An inactive subnet becomes active after the controller has queried the switches' state. Switches set the LTS and STS of the subnet to 1 after steps 3 and 4. The controller sets the LTS of the subnet to 0 after step 5. However, MORP4 requires both LTS and STS to be 0 to log IBR traffic for a subnet.

the subnet if it was set when the controller queried the switch. The previously inactive subnet will have STS value of 0 at the time of query (step 2 in the figure), and thus this is the value that is returned to the controller. On the switch, both values are set to 1 when activity is detected on the subnet (step 3 and 4). When the controller sends an update (step 5), it only sets the LTS value to zero because, from the controller's perspective, the STS was 0 and MORP4 does not allow the controller to modify the STS in this case. Thus, each switch is now in a state where the STS of the subnet is set to 1 while the LTS is set to 0. However, since MORP4 logs IBR traffic only if both LTS and STS are 0, non-IBR traffic is prevented from being captured for the active subnet even when inconsistencies arise between the controller STS and the local STS on the switch.

3.3 Implementation

We implement the core data-plane pipeline of MORP4 in $\sim$ 430 lines of $P4_{16}$ [37] code and the control plane in $\sim$ 400 lines of *Python* code or $\sim$ 1100 lines of *C++* code. Our implementation supports a maximum of 2^{22} subnets. When considering the finest possible granularity of /32 for an IPv4 subnet, this limit translates into the ability of tracking each individual address in a set of prefixes cumulatively as large as a /10 block. This choice is consistent with the size of the space that ASes originate in practice, which we show in Section 1. We implemented two versions of MORP4 for Intel Tofino-1 and Tofino-2 switches respectively [38, 39]. Tofino-1 and -2 have 12 and 20 pipeline stages correspondingly. MORP4 uses at most 8 stages when deployed with the rate limiting option, leaving room for other applications to run on the switch.

Our design uses two tables, implemented as arrays of registers, to store the LTS and STS values for each subnet. Subnets from the same monitored prefix are stored sequentially: their position in both arrays is computed as an offset from a base index. We use a third (TCAM) table, the *Monitored prefixes table*, to *(i)* check if the source (destination) address of an outbound (inbound) packet entering the switch belongs to the monitored prefixes and, if so, *(ii)* to obtain the base index and offset needed to locate the corresponding subnet's registers in the LTS and STS tables. The main resource that MORP4 requires is the SRAM for the LTS and STS tables register arrays whose size depends on A_g (2 bits per monitored $/A_g$ subnet), and thus can be adjusted accordingly to satisfy specific memory constraints. Note that this is an average across stages. In reality, with $A_g = 32$, the LTS and STS tables almost saturate the maximum size of stateful objects dedicated to two respective stages (91.43% in Tofino-1 and 69.6% in Tofino-2), indicating that we cannot monitor space more than a /10 unless in the implementation we duplicate these tables, at the cost of using two more stages. However, even at the finest granularity, MORP4 uses only $\sim 8.4M$ SRAM bits to monitor 2^{22} addresses in total and allows for extensions and other applications to run on the switch. We note that in the Tofino-2 implementation the maximum size of a stateful object decreased. Thus, to still be able to operate at address granularity $A_g = 32$ we halve each of the LTS and STS register arrays.

We implemented the rate limit feature using meters provided by the P4 language for rate-limiting. For the first step, we define the *Global Inactive* meter, and for the second step we define an *Inactive* array of meters of length equal to $2^{14}(\sim 16k)$, *i.e.,* the number of /24 subnets for a set of monitored prefixes cumulatively summing up to the equivalent of a /10 address block. Similarly to computing the index for the LTS and STS tables, *(i)* the controller pre-computes the *inactive_base_index* by setting $A_g = 24$ and inserts it as an additional value in the *Monitored prefixes table*, and *(ii)* the switch derives the index for the *Inactive* array of meters by setting $A_g = 24$ and replacing *base_index* with *inactive_base_index*.

In the multi-switch scenario, switches exchange control packets to synchronize their state of active/inactive IP addresses. Instead of implementing a reliable protocol for the exchange of control packets, we rely on duplication to improve the resilience of the system to loss. In particular, switches send three replicas of the same control packet. Moreover, in the multi-switch scenario, the capturing machine needs to buffer traffic for *buffer_time* (*e.g.,* 1 second) and examine again if it is indeed destined to an unused address before we finally store it. For this, we leverage DPDK [40] to process packets at very high speed, and queues to buffer these packets on the capture server. We built an application that binds to the Mellanox interface connected to the switch and we use two different logical cores to maintain the state of the monitored subnets in a *state* array (much like the switch's *LTS* array), and to buffer the packets. The first core receives the packets from the switch. If a packet is a *control packet*, we update the state of the corresponding subnet to active and drop the packet. Otherwise, we consider the packet to be po-

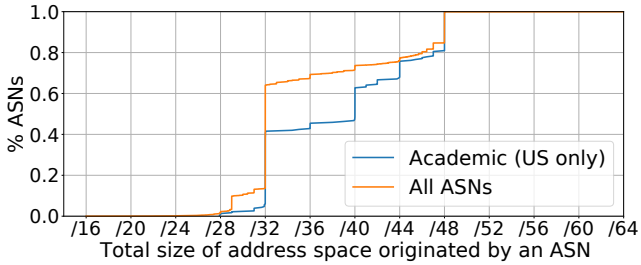

Figure 8: Distribution of the reachable IPv6 address space of (a) academic organizations in the U.S. (#ASNs=231) and (b) all ASNs (#ASNs=34590). For (a), around 99.6% of them advertise on BGP IPv6 space between a /28 and /48, while for (b), around 98.2% of the ASNs announce on BGP IPv6 space between a /28 and /48. We exclude from the graph 10 (0.03%) ASNs that announce less IPv6 space than a /64.

tential IBR and record its capture timestamp (*arrival_time*) as part of its private data. Then, we pass the packet to the second core through a ring buffer. The second core retrieves packets from the ring buffer and repeatedly checks whether *buffer_time* has passed since the reception of the packet, *i.e.,* $current\_time - arrival\_time \geq buffer\_time$. When the condition becomes true, we check the state of the destination subnet of the packet. If the state for that subnet remains inactive, we write the packet into a pcap file, else we drop it.

4 Extending MORP4 to IPv6

Figure 8 shows that the vast majority of ASes announce, using BGP, IPv6 address space that varies between a /28 and /48 in size. In contrast to IPv4, it is infeasible, and in most networks meaningless, for MORP4 to monitor at the finest address granularity $A_g = 128$. Best common practices recommend that ISPs delegate /48 and /56 blocks to business and residential CPEs (Customer Premises Equipment) respectively [41]. Padmanabhan et al. consistently observed the prevalence of /56 and /60 block assignments [42].

Here we describe an implementation to monitor a /32 IPv6 network at $A_g = 56$ granularity. We choose /32 as the size of the network because 87% of the US Academic ASes and 84.8% of all ASes advertise space using BGP that is *at most* equivalent to a /32. To achieve the granularity of the typical prefix length assigned to CPEs, $A_g = /56$, we begin by directly translating the IPv4 solution of MORP4 to IPv6, which allows us to monitor a /32 network at $A_g = /54$ granularity. We then increase the number of LTS and STS tables from 1 to 4 each, so that we increase the maximum total number of monitored blocks from 2^{22} to 2^{24}, thus achieving $A_g = 56$ granularity. The 55^{th} and 56^{th} bits of the IPv6 address determine which LTS/STS table holds the address' state, and the remaining previous host bits dictate the index within the table. From the resource perspective, the 6 additional tables require 6

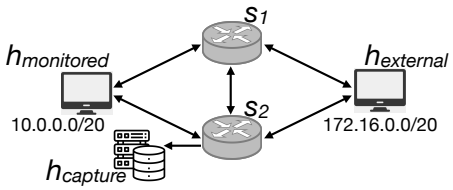

Figure 9: Simulation of the asymmetric communication. $h_{external}$ sends IBR traffic from IP_1 to IP_{test} through s_2 until it receives a trigger packet through s_1 from $h_{monitored}$ with IP_{test} to IP_1. Then, $h_{external}$ switches source address and sends user traffic from IP_2 to IP_{test} through $s2$. s_1 will send control packets to s_2 when it processes the trigger packet from IP_{test}.

more stages in the Tofino pipeline, leaving on Tofino-1 no capacity for our current IBR rate-limiting implementation. While an IPv6 deployment has the potential to require all pipeline stages on Tofino-1, the operator can trade that for the size of the monitored IPv6 space—especially considering that most networks will not actively use their entire IPv6 allocation—*and* the subnet granularity (A_g).

5 System Evaluation

In Section 3.2, we presented the solution of MORP4 for the multi-switch scenario, which requires a distributed protocol to coordinate the state of all participating switches. We opt for a lightweight approach that does not ensure reliable delivery of coordination messages, instead relying on duplication of control packets. In this section, we evaluate the robustness of MORP4 in the multi-switch scenario. In particular, we identify scenarios in which MORP4 may erroneously consider user traffic as IBR due to inconsistent switch states across the system. The possibility of such inconsistency can arise when control packets are lost or excessively delayed. Thus, we evaluate MORP4 under a wide range of network conditions.

We use Mininet [43] to simulate a network consisting of two hosts, one representing the monitored network ($h_{monitored}$) and another representing the rest of the Internet ($h_{external}$). Each host connects to two border switches, s_1 and s_2. The switches run MORP4 on the BMv2 software switch [44] and are connected to each other. The link between s_1 and s_2 is used both for exchanging address activity notifications between the MORP4 instances running on each switch and for load balancing traffic between the $h_{monitored}$ and $h_{external}$ networks.

All links in the network have a bandwidth of 1Gbps. $h_{monitored}$ and $h_{external}$ are assigned the prefixes 10.0.0.0/20 and 172.16.0.0/20, respectively. We use the Mausezahn traffic generator [45] to make each host generate UDP traffic at a rate of 80Mbps through each of its links: $h_{monitored}$ selects a random source address from its subnets 10.0.8.0/21 and 10.0.4.0/22 (excluding the subnet 10.0.0.0/22) and a random destination address from the prefix of $h_{external}$. Conversely, $h_{external}$ selects a random source address from its own prefix

and a random destination address from the aforementioned two subnets of $h_{monitored}$. We use the Linux traffic control tool (tc) with the netem queuing discipline [46] to introduce delay, jitter, and packet loss on the link between the two switches and emulate network impairments that could cause delay and/or loss of the control packets.

We focus on the scenario where a specific inactive IP address (IP_{test}) that belongs to the subnet 10.0.0.0/22 was excluded from the traffic generation. In the scenario, IP_{test} receives traffic through s_2 (*i.e.,* traffic that would be considered IBR if no response is generated by IP_{test}). Then, IP_{test} generates response traffic that egresses the network through s_1. Thus, s_1 will need to inform s_2 about the change of the state of IP_{test}. Due to our focus on this specific scenario, our simulation includes a single capture host $h_{capture}$ connected to s_2 and running tcpdump to capture all forwarded traffic (IBR and control packets) from s_2 into a pcap file. Our setup is shown in Figure 9. We vary the network delay between the two switches from 200 to 1400*ms*, network jitter from 0 to 100*ms*, and loss rate from 0 to 10%. We repeat each scenario 20 times and report the number of failed cases. We consider as failure any case that logged at least one user packet, reflecting our strict objective of capturing IBR-only traffic. We configure MORP4 to send three duplicates of every control message, and the capture host to buffer traffic for 1 second before considering it to be IBR.

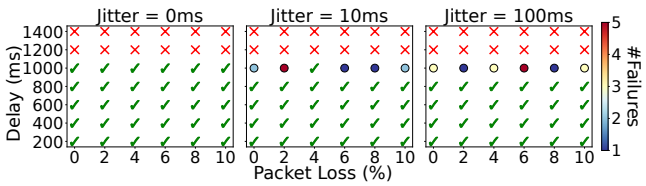

Figure 10: Simulation with #control_packets=3 and buffer_duration=1*s*: for each combination of jitter (subplot), delay (y axis), packet loss (x axis), we run 20 experiments. The green check mark (✓) and the red cross (×) mean that user traffic was captured in **none** or **all** experiments, respectively. In all other cases a circle's color denotes the number of occurrences user traffic was captured.

Our results are shown in Figure 10. As expected, MORP4 responds well to delays up to 800 milliseconds, even with a high jitter and loss rate. Thus, MORP4 should be robust for most practical networks (*i.e.,* aside from transit or hyperscaler networks). With higher jitter and loss rate, the number of failed scenarios increases. As expected, MORP4 fails when the delay between switches is larger than the buffer_duration value set at the capture host. In Appendix C, we evaluate MORP4 under a different configuration of buffer_duration and #control_packets and show that one control packet does not suffice to guarantee the expected system performance.

6 Real world deployment

For our initial experimental setup, we deployed MORP4 (in a single-switch configuration) in a U.S. university's campus network. The university's monitorable address space consists of 167,172 IPv4 addresses (allocated over a few non-contiguous blocks and roughly equivalent to 1.2 times a /15 block), and 1 /32 IPv6 block. The network engineers and security team also operate a static IPv4 telescope constituted of 5 nonadjacent /24 blocks, which we use in the following as a reference pre-existing solution (labeled as `dark`) for comparison. We exclude these 5 /24s from the space monitored by MORP4 and we monitor the rest of the IPv4 space (165,892 IP addresses) at the maximum granularity ($A_g = 32$), and the IPv6 space at granularity $A_g = 53$.[8]

We use an Edgecore Wedge100BF-32X Tofino switch [47] to deploy MORP4 in *monitoring* mode, where 8 100Gbps fiber taps forward to the switch a copy of all the traffic entering and exiting the campus network's border routers. We configure MORP4's controller to retrieve the state of the monitored blocks every $x = 100$ seconds, and to determine a block as *inactive* after a timeout of 6 hours, ($T_o = 21600$ seconds), *i.e.,* after $\alpha = 216$ periods without a sign of activity. The switch forwards IBR to a capture server equipped with a 100Gbps Mellanox ConnectX6 network interface. In this section we show results based on running the system between 2025/03/28 and 2025/04/30 (34 days) with a $\sim$ 2 days gap around April 10 (April 9^{th} 2PM UTC until April 11^{th} 5PM UTC) and 2 reboots on 2025/03/29 and 2025/04/07.[9]

Crosschecks. First, we verify that our system behaves as expected. Specifically, we verify that it *(i)* tracks the activity of all monitored blocks and *(ii)* only captures traffic destined towards inactive space. For this purpose, we employ two sources of *truth*: *(i) unsampled* NetFlow records that we obtain by forwarding *all* the traffic that the switch receives to a server running a NetFlow Exporter and Collector; *(ii)* ARP records that our network administrators collect every 30 minutes from a large subset of the campus' routers and switches, covering 86.4% of our monitored space. We use ARP records as a validation source orthogonal to traffic capture as well as to identify potential internal (outbound) source IP spoofing.

In Figure 11 we show the timeline of active IPs for the 5-day period 2025/04/01 - 2025/04/05 based on the different sources of data: for ARP and NetFlow data we apply MORP4's activity inference criteria with the same T_o and x values.[10] We collect three sets of IPs: *(i) arp_active* – IPs

[8] Even though a Tofino switch can accommodate $A_g = 56$ for IPv6, in this first experimental deployment we use only one switch to host co-existing IPv4 and IPv6 implementations, which limits the monitoring granularity to /53 due to resource constraints.

[9] We had to stop the system on April 9 and restart it on April 11th. Recall that MORP4 requires at least 6 hours of continuous operation to detect inactive addresses and capture unsolicited traffic towards them. Thus, system reboots cause partial data around those times.

[10] The NetFlow line starts from a smaller number as it needs to gradually

with an active ARP entry, *(ii)* $netflow\_active$ – IPs with a NetFlow record with an external destination address, and *(iii)* MORP4_*active* – IPs that have been marked as active by MORP4. For the latter, we process the logs of the controller, which for every query dumps the STS/LTS states of the monitored subnets. Since the MORP4 data and NetFlow records are based on the same exact traffic, we confirm that MORP4 marks as active all the IPs that have a NetFlow record, *i.e.,* $netflow\_active \subseteq$ MORP4_*active*. However, we observe that MORP4 detects 29 more active IPs than NetFlow; we investigate this discrepancy and discover that our NetFlow generator had been dropping a very small percentage of packets due to suboptimal configuration that caused core overload and RSS threads saturation.[11]

Spoofing. As a second, orthogonal check, we compare MORP4 with ARP. Due to ARP data not covering 13.6% of the monitored space, we exclude those addresses from our comparison and produce the line "MORP4 (in ARP)" in Figure 11. Interestingly, MORP4 detects 69.9% (40,237) more active addresses than ARP (58*k*). After manual investigation, we find that this difference is due to packets where the source IP address field has been spoofed. For example, with the help of unsampled NetFlow records, we identify a spoofing episode during 2025/04/01 - 2025/04/05: we observe 73,973 IP addresses that have an outbound flow of a single UDP packet with source port 53 and destination port 12345. The subnets for which we have ARP data, cover 65,592 of these addresses and show that 39,146 of these IPs never negotiated an ARP entry. These 39,146 likely spoofed IPs constitute 97% of the additional IPs that MORP4 detects compared to ARP (the remaining 3% is likely spoofed traffic for addresses that were legitimately used at different time bins). Using the ARP data in conjunction with manual analysis, we discover another property of these spoofed IP addresses: most of them are visible as active by MORP4 only for at most 10 non-consecutive 100s time bins in the whole 34-days observation period. This data point indicates that, while covering a large number of addresses, the spoofing behavior we observed is limited to very short-lived events and it does not significantly compromise the ability of a dynamic telescope to "collect" a large number of inactive IPs. Nevertheless, spoofing behavior is an interesting phenomenon that we intend to investigate in future work. Besides, through cooperation with the network administrators and leveraging ARP/NetFlow data, it should be relatively easy to identify and counter these unauthorized behaviors in the future.

Dynamic size of the (IPv4) telescope. We then look at which IPv4 addresses are detected as inactive by MORP4 over time. Specifically, Figure 12 shows both daily values—*i.e.,* count of

observe active IPs. ARP and MORP4 have been operating since before 2025/04/01 and therefore already start with a set of active IPs.

[11]These missed active IPs were active only during one timebin of 100 seconds, and potentially sent a minimal number of packets such that NetFlow failed to record a flow for them.

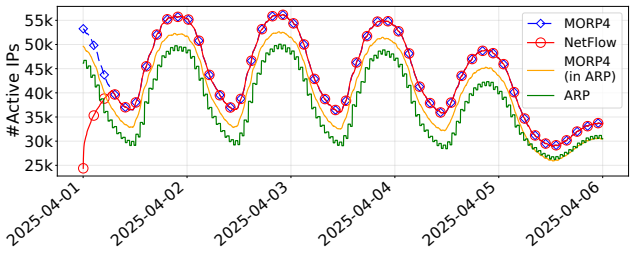

Figure 11: Number of active IPs over time based on *(i)* MORP4 logs, *(ii)* unsampled NetFlow data, *(iii)* MORP4 logs limited to the available ARP /24 blocks, and *(iv)* ARP records.

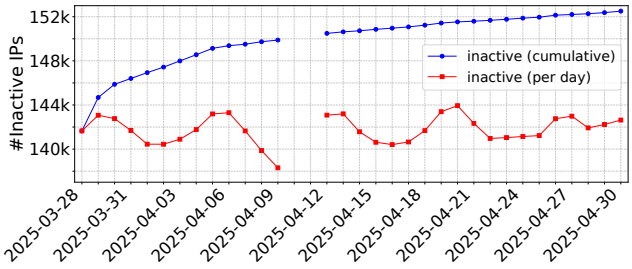

Figure 12: Number of inactive IPs in our deployment.

IP addresses that were detected as inactive at least once during a given day (x axis)—and cumulative values—*i.e.,* count of IP addresses that were detected inactive at least once up to the considered day. Overall, these numbers show that the vast majority of the IP addresses of this institution contribute with capturing IBR (we verified not only that they are at a certain point inactive but that they receive actual unsolicited traffic): 152,504 out of 165,892 after 34 days. Interestingly even after only 24 hours we find 141,645 addresses contributing to capture IBR. If compared to the space the campus had reserved for telescope use (5 /24s, *i.e.,* 1,280 addresses) this is a staggering number, showing that through MORP4 it is possible to obtain a large (dynamic) telescope. Finally, comparing the amount of IBR traffic respectively captured by MORP4 and the 5 darknets allocated by the institution confirms such proportions: ∼37.3B packets (∼2.5TB) by MORP4 vs ∼356.7M packets (∼23.14GB) by the 5 /24 darknets. The benefit of MORP4 is further highlighted by the greater number of source IP addresses, /16s, and ASNs that we observe only in the IBR towards the space monitored by MORP4 compared to the IBR towards only the darknets. We find that 15,243 /16s (from 17,278 ASNs) are present exclusively in MORP4's IBR as opposed to only 40 /16s (from 33 ASNs) visible only in the darknets. We find 26,485 /16s (from 13,610 ASNs) in common between the two IBR datasets.

Looking at data at the finest system resolution provides another interesting data point: the number of IPs marked as inactive in each individual 100s time bin never drops below 46,880 during the entire 34-days period. Since the amount of traffic captured by MORP4 over time changes based on the

number of inactive addresses, certain analyses of the captured traffic might need to take that into account in order to normalize their results. MORP4 can return this number at each time bin, therefore with a granularity in the order of minutes.

Lack of utilization across space and time. Given these successful results, we then perform a preliminary analysis of *if* and *how* two distinct properties of (un)utilization contribute to our dynamic telescope during the observation period: *(i)* sparse utilization purely in the space dimension—*i.e.,* how much space has been inactive for the whole 34 days period and how fragmented it is; and *(ii)* sparse utilization over time—*i.e.,* how long addresses that are used at least once and have been inactive at least once, tend to stay inactive. This analysis is not exhaustive but intends to provide a first empirical intuition of the characteristics of the phenomena that can contribute to a dynamic telescope.

Since our focus here is to characterize properties of actual utilization, in this analysis we try to remove as much as possible any "noise" caused by spoofed traffic (*i.e.,* if an address appeared as active only due to spoofing, we instead count it as inactive). To remove as much spoofing as possible, we use MORP4's logs to detect which IPs have been detected as active only during at most (non-consecutive) 10 timebins, and out of those we keep only the IPs with no ARP record despite belonging to an ARP-covered subnet. This, combined with the IPs that MORP4 never detects as active, yields 81,785 inactive IPs. That is, approximately 53% of the addresses detected at least once as inactive (and used) by MORP4 (49% of the whole monitored address space) have been unutilized for the whole 34-days period. Interestingly, this unutilized space appears largely fragmented, since it contains only 1 entirely inactive /24 and only 31 entirely inactive /25 subnets. For the remaining 70,720 sometime-inactive IPs that have been detected as active at least once (*i.e.,* 47% of the IPs used by MORP4), we look at the distribution of inactivity periods in Figure 13. We find that 70% of the inactivity periods last less than 24 hours (an inactivity period starts once an IP address is declared inactive by MORP4—*i.e.,* after 6 hours of inactivity) and 5% last more than 5 days. These data also suggests that choosing a configuration setting of T_o of a week or greater can significantly limit the size of a dynamic telescope, at least in a setting with similar characteristics. In future work, we plan to characterize these behaviors in greater detail and within multiple networks.

An effective IPv6 telescope. In the IPv6 domain, using a large block of unused space as in static telescopes, is of limited utility. A dynamic telescope offers the opportunity to collect IBR towards address blocks that are instead partially used. For this reason, to evaluate our IPv6 dynamic telescope deployment based on MORP4, we focus on comparing IBR it captured at entirely unused IPv6 blocks against IBR it captured in partially used blocks. We compare blocks at /48 granularity since it is a common granularity to separate IPv6 address blocks, and we leverage the ability of our setup to track inactivity at /53 granularity (within such /48 blocks).

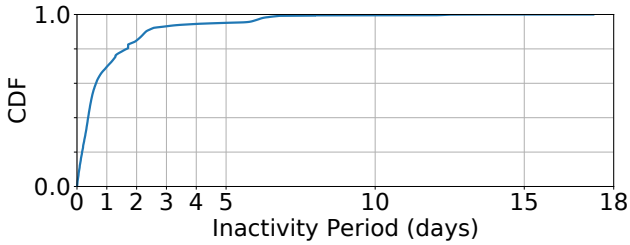

Figure 13: CDF of the inactivity periods of sometime-inactive IPs. These durations are measured starting after MORP4's 6-hours inactivity time-out (T_o).

Specifically, out of the campus' /32 IPv6 address block advertised on BGP, the network engineers have enabled internal routing only for 8,328 /53 blocks, within 272 /48 blocks. After running MORP4, we find 17 /48 blocks out of 272 (6.3%) with active /53s. We then compare what MORP4 was able to capture from these 17 /48 blocks against what it captured from the remaining 255.

We first zoom in on what activity MORP4 detects in the 17 partially active /48 blocks. We find that out of the 291 /53 routed blocks they cover, 192 are inactive for the whole time, 51 never timeout to inactive state, and the remaining 48 are only sometimes in the inactive state. Interestingly, 185 /53 blocks contribute with actual (IPv6) IBR.

We then compare the characteristics of such IBR (*Active-48s-IBR*) towards the 17 partially active /48 blocks with IBR towards the totally dark 255 /48 blocks (*Dark-48s-IBR*). We obtain ∼25M packets (∼2GB) of *Active-48s-IBR*, and only ∼69K packets (∼6MB) of *Dark-48s-IBR*. The magnitude difference of over two orders in the amount of IBR captured (also in contrast to the much larger collective size of the entirely dark blocks), highlights the effectiveness of using a dynamic telescope for IPv6 IBR collection.

Finally, we observe a striking difference in the number of source /64s, /48s, and ASNs that we find in *Active-48s-IBR* compared to *Dark-48s-IBR*. E.g., cumulatively, in *Active-48s-IBR* we observed 4066 source /48s (from 138 ASNs) that were absent in *Dark-48s-IBR*, compared to 17 /48s (from 2 ASNs) seen only in *Dark-48s-IBR*, and an overlap of 59 /48s (from 34 ASNs). In future work we plan to characterize the IBR collected in the completely dark /48s, which might *e.g.,* be directed towards /48s that are adjacent to the active ones.

7 Related Work

In 2006, Cooke et al. propose *Dark Oracle* to detect unused and unreachable addresses within a network using external (BGP) and local routing data (OSPF), and host configuration data (DHCP) [48]. However, the authors admit that access to DHCP data can be challenging, and their system's

classification accuracy is highly dependent on retrieving the updated data sources (which can be inaccurate or unstable) promptly. Mizoguchi et al. recover unused IP addresses by requesting the unassigned addresses from the DHCP server, and assign them to network sensors [49, 50]. This approach is constrained within a DHCP segment and its expansion towards more diverse collected intelligence requires the placement of additional sensors in different segments of the network.

In 2010, Shimoda et al. propose *DarkPots*, a system of virtualized honeypots utilizing the unused addresses of a network which have been detected through monitoring the produced traffic [51]. DarkPots mirrors the traffic to: *(i)* the *Vacancy Checker*, which identifies active addresses and removes them from a list of unused addresses provided by the network administrators, and *(ii)* the *Forwarder*, which forwards the traffic to honeypots if it is destined to an address within the updated list of unused addresses. However, (due to synchronization delays with the *Vacancy Checker*) the *Forwarder* might not become immediately aware of an address becoming active and misdirect legitimate traffic to honeypots and potentially interfere with the connection of the internal host. Moreover, it is not clear whether an address is added back to the list of unused addresses if it is no longer assigned to a host. The authors extend their work in [52] to detecting unused pairs of IP addresses and ports, to include into their analysis scope even active hosts. They focus on TCP flows initiated by an external host and classify them as malicious when they observe no TCP SYN/ACK responses for a specified time interval. However, if an active host becomes temporarily unavailable (*e.g.,* due to reboot) and is unable to respond to the TCP SYN packet before the timeout expires, the honeypot will take over an otherwise legitimate session. In 2023, Pauley et al. propose DScope, a cloud-native telescope that dynamically rotates short-lived, cloud-rented IP addresses to observe attacks targeting cloud infrastructure [53].

In IPv6 space monitoring, Czyz et al. create the first large IPv6 network telescope by announcing on BGP the /12 blocks allocated to the 5 RIRs and capturing packets towards addresses that are not matched by more specific BGP prefixes [34] . Other studies utilize completely dark static IPv6 space of various sizes (/20, /48) [54–56]. In 2022, Richter et al. introduce a "distributed telescope," utilizing firewall logs from approximately 230,000 servers across a major CDN [57]. More recent work uses controlled exposure methods (*e.g.,* BGP announcements, DNS triggers, responsive or reactive honeypots, etc.) to attract and study scanning behavior [58–63]. MORP4 provides an orthogonal perspective by utilizing actual host activity to detect unused subnets and study IBR in operational networks, avoiding potential bias from artificially attracting scans.

8 Discussion and future work

MORP4 enables researchers to significantly expand the number of IP addresses that can contribute to a network telescope, including IP addresses that are only occasionally used by real machines, without compromising on the confidentiality of non-IBR traffic. Our successful pilot deployment is in a network with a single ingress/egress location, but we have also demonstrated that MORP4 is able to achieve correctness in a network with multiple ingress/egress locations (under realistic network conditions). We are already working with other network operators to duplicate our deployment of MORP4 in their network, and we also plan to release the code for MORP4 as open source software.

With our pilot deployment, we found that MORP4 was able to capture significant quantities of IPv6 IBR traffic that was destined for inactive subnets within a larger, partially active /48 block; much more than the amount of IBR traffic that was observed across the /48 blocks that were entirely unused.

MORP4 makes some assumptions that were met by our deployment network, such as the property that all incoming traffic without a corresponding outgoing traffic is IBR. Our network administrators confirmed that this assumption is valid for the subnets we monitored. Even if unusual, though, it is possible for a host to respond to an inbound packet after staying completely silent for a long period of time (*i.e.,* $\geq T_o$), in which case only such inbound packet would be erroneously captured (*e.g.,* a TCP SYN). Another potential but rare source of error would be for the switch to experience a packet loss specifically for the first outbound packet that triggers the transition of an IP from inactive to active. However, MORP4 allows administrators to exclude subnets that might expect to receive non-IBR traffic or could receive traffic that is too sensitive to risk capturing by mistake.

MORP4 assumes a non-malicious monitored network. In particular, all machines in the network will only use the IP addresses they are assigned and not spoof other addresses that belong to the same network. Spoofing can therefore cause MORP4 to incorrectly infer that the spoofed addresses are active. While cooperation with network administrators can be effective in mitigating the effect of spoofing on MORP4, we intend to further study these in-network spoofing episodes in detail to better understand them and potentially discover automated methods for detecting them with MORP4 itself.

In addition, we intend to perform a detailed characterization of the qualitative and quantitative benefits of traffic captured by dynamic telescopes compared to static ones, as well as a characterization of differences in IBR traffic seen for addresses that are intermittently used and/or adjacent to active addresses compared with the IBR observed for completely unused address blocks. We also plan to conduct more extensive research on IPv6 dynamic network telescopes, as our results suggest that their deployment might be a highly promising way to capture and analyze representative IBR.

References

[1] Karyn Benson et al. “Leveraging Internet Background Radiation for Opportunistic Network Analysis”. In: *Proceedings of the 2015 Internet Measurement Conference*. IMC ’15. Tokyo, Japan: Association for Computing Machinery, 2015, pp. 423–436. ISBN: 9781450338486. DOI: 10.1145/2815675.2815702.

[2] Alexander Männel et al. “Lessons Learned from Operating a Large Network Telescope”. In: *Proceedings of the ACM SIGCOMM 2025 Conference*. SIGCOMM ’25. São Francisco Convent, Coimbra, Portugal: Association for Computing Machinery, 2025, pp. 826–841. ISBN: 9798400715242. DOI: 10.1145/3718958.3754347.

[3] Stuart Staniford et al. “The top speed of flash worms”. In: *Proceedings of the 2004 ACM Workshop on Rapid Malcode*. WORM ’04. Washington DC, USA: Association for Computing Machinery, 2004, pp. 33–42. ISBN: 1581139705. DOI: 10.1145/1029618.1029624.

[4] C. Shannon and D. Moore. “The spread of the Witty worm”. In: *IEEE Security & Privacy* 2.4 (2004), pp. 46–50. DOI: 10.1109/MSP.2004.59.

[5] D. Moore et al. “Inside the Slammer worm”. In: *IEEE Security & Privacy* 1.4 (2003), pp. 33–39. DOI: 10.1109/MSECP.2003.1219056.

[6] David Moore, Colleen Shannon, and k claffy k. “CodeRed: a case study on the spread and victims of an internet worm”. In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*. IMW ’02. Marseille, France: Association for Computing Machinery, 2002, pp. 273–284. ISBN: 158113603X. DOI: 10.1145/637201.637244.

[7] David Moore, Geoffrey M. Voelker, and Stefan Savage. “Inferring Internet Denial-of-Service Activity”. In: *10th USENIX Security Symposium (USENIX Security 01)*. Washington, D.C.: USENIX Association, Aug. 2001.

[8] Alberto Dainotti et al. “Analysis of a "/0" stealth scan from a botnet”. In: *Proceedings of the 2012 Internet Measurement Conference*. IMC ’12. Boston, Massachusetts, USA: Association for Computing Machinery, 2012, pp. 1–14. ISBN: 9781450317054. DOI: 10.1145/2398776.2398778.

[9] E Raftopoulos et al. “How Dangerous Is Internet Scanning? A Measurement Study of the Aftermath of an Internet-Wide Scan”. In: *Traffic Monitoring and Analysis Workshop (TMA)*. Vol. 9053. Apr. 2015, pp. 158–172. DOI: https://doi.org/10.1007/978-3-319-17172-2_11.

[10] Alberto Dainotti et al. “Extracting benefit from harm: using malware pollution to analyze the impact of political and geophysical events on the internet”. In: *SIGCOMM Comput. Commun. Rev.* 42.1 (Jan. 2012), pp. 31–39. ISSN: 0146-4833. DOI: 10.1145/2096149.2096154.

[11] Karyn Benson et al. “Gaining insight into AS-level outages through analysis of internet background radiation”. In: *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*. CoNEXT Student ’12. Nice, France: Association for Computing Machinery, 2012, pp. 63–64. ISBN: 9781450317795. DOI: 10.1145/2413247.2413285.

[12] Alberto Dainotti et al. “Analysis of country-wide Internet outages caused by censorship”. In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement*. IMC ’11. Berlin, Germany: ACM, 2011, pp. 1–18. ISBN: 978-1-4503-1013-0. DOI: 10.1145/2068816.2068818.

[13] Alberto Dainotti et al. “Estimating internet address space usage through passive measurements”. In: *SIGCOMM Comput. Commun. Rev.* 44.1 (Dec. 2014), pp. 42–49. ISSN: 0146-4833. DOI: 10.1145/2567561.2567568.

[14] Alberto Dainotti et al. “Lost in Space: Improving Inference of IPv4 Address Space Utilization”. In: *IEEE Journal on Selected Areas in Communications* 34.6 (2016), pp. 1862–1876. DOI: 10.1109/JSAC.2016.2559218.

[15] *UCSD Network Telescope*. https://www.caida.org/projects/network_telescope/.

[16] CAIDA. *STARDUST Workshop series: 2012, 2019, 2021*. "https://www.caida.org/workshops/?workshopserieslisting=DUST&show_all=1".

[17] Manos Antonakakis et al. “Understanding the Mirai Botnet”. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. ISBN: 978-1-931971-40-9.

[18] Eric Pauley, Paul Barford, and Patrick McDaniel. “The CVE Wayback Machine: Measuring Coordinated Disclosure from Exploits against Two Years of Zero-Days”. In: *Proceedings of the 2023 ACM on Internet Measurement Conference*. IMC ’23. Montreal QC, Canada: Association for Computing Machinery, 2023, pp. 236–252. ISBN: 9798400703829. DOI: 10.1145/3618257.3624810.

[19] Daniel Wagner et al. “How to Operate a Meta-Telescope in your Spare Time”. In: *Proceedings of the 2023 ACM on Internet Measurement Conference*. IMC ’23. Montreal QC, Canada: Association for Computing Machinery, 2023, pp. 328–343. ISBN: 9798400703829. DOI: 10.1145/3618257.3624831.

[20] Aniket Anand et al. “Aggressive Internet-Wide Scanners: Network Impact and Longitudinal Characterization”. In: *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT 2023. Paris, France: Association for Computing Machinery, 2023, pp. 1–8. ISBN: 9798400704079. DOI: 10.1145/3624354.3630583.

[21] Liz Izhikevich et al. “Cloud Watching: Understanding Attacks Against Cloud-Hosted Services”. In: *Proceedings of the 2023 ACM on Internet Measurement Conference*. IMC ’23. , Montreal QC, Canada, Association for Computing Machinery, 2023, pp. 313–327. ISBN: 9798400703829. DOI: 10.1145/3618257.3624818.

[22] Zachary S. Bischof et al. “Destination Unreachable: Characterizing Internet Outages and Shutdowns”. In: *Proceedings of the ACM SIGCOMM 2023 Conference*. ACM SIGCOMM ’23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 608–621. ISBN: 9798400702365. DOI: 10.1145/3603269.3604883.

[23] Raphael Hiesgen et al. “Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope”. In: *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 431–448. ISBN: 978-1-939133-31-1.

[24] Zakir Durumeric, Michael Bailey, and J. Alex Halderman. “An Internet-Wide View of Internet-Wide Scanning”. In: *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 65–78. ISBN: 978-1-931971-15-7.

[25] Philipp Richter and Arthur Berger. “Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope”. In: *Proceedings of the Internet Measurement Conference*. IMC ’19. Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 144–157. ISBN: 9781450369480. DOI: 10.1145/3355369.3355595.

[26] Agathe Blaise et al. “Detection of zero-day attacks: An unsupervised port-based approach”. In: *Computer Networks* 180 (2020), p. 107391. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107391.

[27] Merit Network, Inc. *Orion Network Telescope*. https://www.merit.edu/research/projects/orion-network-telescope/. 2025.

[28] M. Kallitsis et al. *Zooming Into the Darknet: Characterizing Internet Background Radiation and its Structural Changes*. arXiv, https://arxiv.org/pdf/2108.00079. 2021.

[29] Vinod Yegneswaran, Paul Barford, and Dave Plonka. “On the Design and Use of Internet Sinks for Network Abuse Monitoring”. In: *Recent Advances in Intrusion Detection*. Ed. by Erland Jonsson, Alfonso Valdes, and Magnus Almgren. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 146–165. ISBN: 978-3-540-30143-1.

[30] Michael Bailey et al. “The Internet Motion Sensor - A Distributed Blackhole Monitoring System”. In: *Network and Distributed System Security Symposium*. 2005.

[31] CAIDA. *The UCSD Network Telescope*. https://www.caida.org/projects/network_telescope/. 2023.

[32] Merit Network, Inc. *Merit Darknet IPv4*. http://software.merit.edu/darknet/.

[33] Oliver Gasser et al. “Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists”. In: *Proceedings of the Internet Measurement Conference 2018*. IMC ’18. Boston, MA, USA: Association for Computing Machinery, 2018, pp. 364–378. ISBN: 9781450356190. DOI: 10.1145/3278532.3278564.

[34] Jakub Czyz et al. “Understanding IPv6 internet background radiation”. In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC ’13. Barcelona, Spain: Association for Computing Machinery, 2013, pp. 105–118. ISBN: 9781450319539. DOI: 10.1145/2504730.2504732.

[35] Sebastian Zander, Lachlan L.H. Andrew, and Grenville Armitage. “Capturing ghosts: predicting the used IPv4 space by inferring unobserved addresses”. In: *Proceedings of the 2014 Conference on Internet Measurement Conference*. IMC ’14. Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 319–332. ISBN: 9781450332132. DOI: 10.1145/2663716.2663718.

[36] Grant Williams and Paul Pearce. “Seeds of Scanning: Exploring the Effects of Datasets, Methods, and Metrics on IPv6 Internet Scanning”. In: *Proceedings of the 2024 ACM on Internet Measurement Conference*. IMC ’24. Madrid, Spain: Association for Computing Machinery, 2024, pp. 295–313. ISBN: 9798400705922. DOI: 10.1145/3646547.3688449.

[37] The P4 Language Consortium. *$P4_{16}$ Language Specification, Version 1.2.5*. Tech. rep. P4 Language Consortium, Oct. 2024.

[38] Intel Corporation. *$P4_{16}$ Intel® Tofino™ Native Architecture – Public Version*. https://github.com/barefootnetworks/Open-Tofino/blob/master/PUBLIC_Tofino-Native-Arch.pdf. Accessed: 2025-09-15. 2021.

[39] Intel Corporation. *Open P4Studio: Intel® P4 Studio Software Development Environment*. https://github.com/p4lang/open-p4studio. Accessed: 2025-09-15. 2025.

[40] The Linux Foundation. *Data Plane Development Kit (DPDK)*. https://www.dpdk.org.

[41] Jan Žorž et al. *Best Current Operational Practice for Operators: IPv6 prefix assignment for end-users - persistent vs non-persistent, and what size to choose*. Tech. rep. RIPE-690. RIPE NCC, Oct. 2017.

[42] Ramakrishna Padmanabhan et al. "DynamIPs: analyzing address assignment practices in IPv4 and IPv6". In: *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '20. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 55–70. ISBN: 9781450379489. DOI: 10.1145/3386367.3431314.

[43] *Mininet: An instant virtual network on your laptop (or other PC)*. https://mininet.org/. Accessed: 2025-09-15.

[44] P4 Language Consortium. *BMv2: Behavioral Model version 2 P4 software switch*. https://github.com/p4lang/behavioral-model. Accessed: 2025-09-15.

[45] *netsniff-ng: Linux network analyzer and networking toolkit*. https://github.com/netsniff-ng/netsniff-ng. Accessed: 2025-09-15.

[46] *tc-netem — Network Emulator*. https://man7.org/linux/man-pages/man8/tc-netem.8.html. Linux manual page. Accessed: 2025-09-15.

[47] *DCS800 - Edgecore Networks*. Accessed: 2025-04-23. Edgecore Networks. URL: https://www.edge-core.com/product/dcs800/.

[48] Evan Cooke et al. "The dark oracle: perspective-aware unused and unreachable address discovery". In: *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*. NSDI'06. San Jose, CA: USENIX Association, 2006, p. 8.

[49] Seiichiro Mizoguchi, Yoshiaki Hori, and Kouichi Sakurai. "Monitoring Unused IP Addresses on Segments Managed by DHCP". In: *2008 Fourth International Conference on Networked Computing and Advanced Information Management*. Vol. 1. 2008, pp. 510–515. DOI: 10.1109/NCM.2008.245.

[50] Seiichiro Mizoguchi et al. "Darknet Monitoring on Real-Operated Networks". In: *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*. 2010, pp. 278–285. DOI: 10.1109/BWCCA.2010.82.

[51] Akihiro Shimoda, Tatsuya Mori, and Shigeki Goto. "Sensor in the Dark: Building Untraceable Large-Scale Honeypots Using Virtualization Technologies". In: *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*. 2010, pp. 22–30. DOI: 10.1109/SAINT.2010.42.

[52] Akihiro Shimoda, Tatsuya Mori, and Shigeki Goto. "Extended Darknet: Multi-Dimensional Internet Threat Monitoring System". In: *IEICE Transactions on Communications* E95.B.6 (2012), pp. 1915–1923. DOI: 10.1587/transcom.E95.B.1915.

[53] Eric Pauley, Paul Barford, and Patrick McDaniel. "DScope: A Cloud-Native Internet Telescope". In: *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 5989–6006. ISBN: 978-1-939133-37-3.

[54] M. Ford, J. Stevens, and J. Ronan. "Initial Results from an IPv6 Darknet13". In: *International Conference on Internet Surveillance and Protection (ICISP'06)*. 2006, pp. 13–13. DOI: 10.1109/ICISP.2006.14.

[55] ChenHuan Liu et al. "IPv6-Network Telescope Network Traffic Overview". In: *2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC)2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. 2021, pp. 1–4. DOI: 10.1109/ICEIEC51955.2021.9463724.

[56] John Ronan and David Malone. "Revisiting and Revamping an IPv6 Network Telescope". In: *2023 34th Irish Signals and Systems Conference (ISSC)*. 2023, pp. 1–6. DOI: 10.1109/ISSC59246.2023.10162033.

[57] Philipp Richter, Oliver Gasser, and Arthur Berger. "Illuminating large-scale IPv6 scanning in the internet". In: *Proceedings of the 22nd ACM Internet Measurement Conference*. IMC '22. Nice, France: Association for Computing Machinery, 2022, pp. 410–418. ISBN: 9781450392594. DOI: 10.1145/3517745.3561452.

[58] Hammas Bin Tanveer et al. "Glowing in the Dark: Uncovering IPv6 Address Discovery and Scanning Strategies in the Wild". In: *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 6221–6237. ISBN: 978-1-939133-37-3.

[59] Liang Zhao, Satoru Kobayashi, and Kensuke Fukuda. "Exploring the Discovery Process of Fresh IPv6 Prefixes: An Analysis of Scanning Behavior in Darknet and Honeynet". In: *Passive and Active Measurement: 25th International Conference, PAM 2024, Virtual Event, March 11–13, 2024, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2024, pp. 95–111. ISBN: 978-3-031-56248-8. DOI: 10.1007/978-3-031-56249-5_4.

[60] Liang Zhao, Satoru Kobayashi, and Kensuke Fukuda. "Insights into the IPv6 scanning landscape: An extended study using passive darknets and responsive honeynets with prefix exposing techniques". In: *IEICE Transactions on Communications* (2025), pp. 1–14. DOI: 10.23919/transcom.2025CEP0010.

[61] Yue Xin et al. "Two-Phase Scanning in IPv6 - First Observations from a Reactive IPv6 Network Telescope". In: *Proceedings of the ACM SIGCOMM 2025 Posters and Demos*. ACM SIGCOMM Posters and Demos '25. Coimbra, Portugal: Association for Computing Machinery, 2025, pp. 103–105. ISBN: 9798400720260. DOI: 10.1145/3744969.3748443.

[62] Isabell Egloff et al. "A Detailed Measurement View on IPv6 Scanners and Their Adaption to BGP Signals". In: *Proc. ACM Netw.* 3.CoNEXT3 (Sept. 2025). DOI: 10.1145/3749215.

[63] Hammas Bin Tanveer et al. "Unveiling IPv6 Scanning Dynamics: A Longitudinal Study Using Large Scale Proactive and Passive IPv6 Telescopes". In: *Proc. ACM Netw.* 3.CoNEXT3 (Sept. 2025). DOI: 10.1145/3749221.

A Ethical Considerations

Telescopes, including MORP4, are typically deployed in universities. Data produced by telescopes are typically shared with external researchers to better understand the characteristics of the IBR. Naturally, such a comprehensive data collection effort in such a sensitive setting raises privacy concerns. MORP4 was carefully designed to avoid collecting any user data or any data generated or destined to an active machine in the monitored network. We received the approval of the Institutional Review Board (IRB) for the deployment of MORP4 in our network. We worked closely with the university's network engineering and security teams. Although MORP4 can identify the set of all active IP addresses, we do not collect any information that connects a specific IP address to a specific user/machine.

B DoS attack mitigation

The size of the state tracked by MORP4 is completely dependent on the size of the monitored network (*e.g.,* number of tracked subnets) and not the volume of IBR traffic. However, telescope infrastructure can occasionally experience denial of service (DoS) due to an overwhelming amount of traffic towards one or more of its addresses. The capacity of capture hosts is provisioned for some estimates of the volume of IBR traffic. Thus, their network capacity can be overwhelmed. While that behavior is not intrinsically problematic, as traffic exceeding the capacity of the host will be dropped, we build a rate limiting mechanism in MORP4 to protect against such scenarios, allowing it to salvage IBR data while discarding DoS traffic. In particular, we introduce an optional IBR rate-limiter, which limits the aggregate rate of collected IBR traffic as well as the maximum capturing rate at the /24-subnet level. The rate limit is applied to the packets before they are replicated and forwarded to the capture host. The first limit ensures that the total replicated IBR traffic does not overwhelm the capture host. Additionally, we use the second limit to isolate the traffic destined to different /24 subnets so that a heavily probed subnet does not saturate the available capturing rate and thus does not prevent logging packets towards other subnets.

C Additional Evaluation of MORP4's Robustness

We evaluate the robustness of MORP4 by examining an extreme configuration where it does not duplicate its control packets (*i.e.,* #control_packets=1) and the buffer size at the capture host is small, $buffer\_duration = 0.5s$. We repeat the simulations described in Section 5 and show the results in Figure 14. Naturally, such an extreme configuration produces more failures (*i.e.,* the capture host recording non-IBR traffic). However, a careful examination of the results shows that such failures occur at extremely high loss rates (*i.e.,* $\geq 8\%$) when the delay between the two switches is smaller than $buffer\_duration$. Our simulations demonstrate the robustness and flexibility of MORP4, that allows network operators to configure it to meet the characteristics of their network.

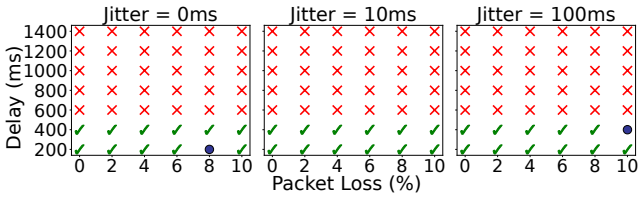

Figure 14: Simulation with #control_packets=1 and buffer_duration=0.5s: for each combination of jitter (subplot), delay (y axis), packet loss (x axis), we run 20 experiments. The green check mark (✓) and the red cross (×) mean that user traffic was captured in **none** or **all** experiments, respectively. The circle denotes that only 1 out of the 20 experiments failed.

D MORP4 state transition

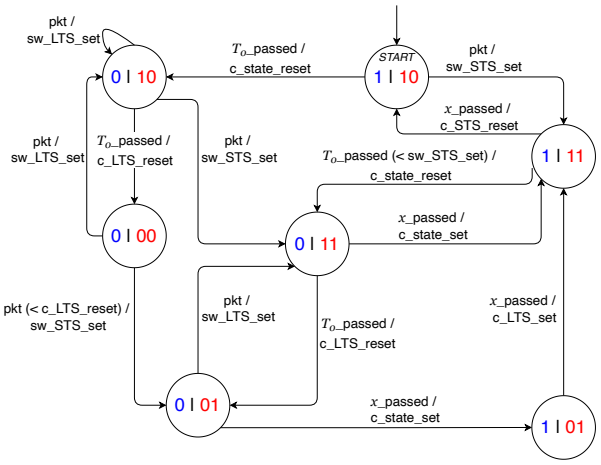

Figure 15: Finite state machine describing the transitions between the system states for a monitored subnet. A system state is $(c\_state|LTS\ STS)$, where c_state is the state of the subnet in the controller.

In Figure 15, we show how MORP4 transitions between its different states for a specific subnet when an event occurs and a MORP4 component applies an action. The events and actions that affect MORP4's state are listed in Table 2. MORP4 always starts in state (1, 1, 0) to prevent from capturing non-IBR traffic during initialization. The diagram shows how MORP4 transitions between different states when

Event or Action	Explanation
x_passed	x seconds (timebin) passed
T_o_passed	T_o seconds passed
pkt	switch receives an outbound packet
sw_STS_set	switch sets STS to 1
sw_LTS_set	switch sets LTS to 1
c_STS_reset	controller resets STS to 0
c_LTS_reset	controller resets LTS to 0
c_LTS_set	controller sets LTS to 1
c_state_reset	controller resets *c_state* to 0
c_state_set	controller sets *c_state* to 1

Table 2: Explanation of the events and actions that are annotated in Figure 15 of the FSM of MORP4.

either a timebin has expired (x_passed or T_o_passed, if α timebins of inactivity for the monitored subnet have passed) or the switch receives an outbound packet originated from the monitored subnet (pkt). Extra attention is required in the transitions $(1,1,0) \rightarrow (1,1,1) \rightarrow (0,1,1)$ and $(0,1,0) \rightarrow (0,0,0) \rightarrow (0,0,1)$: both of the middle states in these transitions are transient. In the first transition, while in state $(1,1,0)$ the controller receives STS = 0 from the switch for the α^{th} time, *i.e.,* the T_o has passed, and **before** it resets its *c_state* to 0, the switch receives an outbound packet and updates its STS to 1 changing the state to $(1,1,1)$. Right afterwards, the controller resets its *c_state* based on its outdated information and MORP4 transitions to $(0,1,1)$. In the second transition, while in state $(0,1,0)$, *i.e.,* after the controller has reset its *c_state* to 0 because T_o has passed, and before the controller resets the LTS to 0, the switch receives a packet and updates its LTS to 1 staying in $(0,1,0)$. Then, the controller resets the LTS to 0 (while the packet is still in the switch before the STS stage) moving to $(0,0,0)$, and right afterwards the switch updates its STS to 1 reaching the state $(0,0,1)$. In the multi-switch deployment, there is one additional action: when a packet sets the STS from 0 to 1, it triggers the generation of control packets to the other border switches.

E Acknowledgments

This work is supported by the National Science Foundation under Grants OAC-2419826, NeTS-2345827 and CNS-2212098, and partially by the Onassis Foundation - Scholarship ID: F ZT078-1/2023-2024. We would also like to thank our institute's network engineering team, especially Jacob Smith and Earl Barfield, for their help with the on-campus system deployment and their continuous support, Bhaskar Pardeshi, for his help with the development of the DPDK buffering solution, and Thomas Papastergiou, for his help with the initial switch setup. Finally, we would like to thank our shepherd, Ioana Livadariu, and the anonymous reviewers for their valuable feedback.