TERIS: A Tool for Emulated Routing at Internet Scale

Joshua Levett, Vassilios Vassilakis, Poonam Yadav
Department of Computer Science, University of York

Abstract

In the context of Internet routing, even small changes to configurations can have devastating consequences for the overall security and stability of the global routing table. Routing protocol test environments, many of which are limited in size or rely on deep levels of technical abstraction, are currently unsuited to realistic testing at Internet scale. In this work, we present TERIS, a Tool for Emulated Routing at Internet Scale. Capturing Internet topology, we generate a feasible, scalable and representative virtual testbed, generating appropriate router and policy configurations for each ISP-level network (Autonomous System, AS) which is then deployed through Kubernetes. We demonstrate the potential for Internet routing protocol enhancement using this tool, informed by real-world Internet topology. In summary, our contribution is an improvement to existing Internet routing protocol test environments by replicating real-world Internet topology using virtual containers, such that developments and enhancements to routing protocols can be observed in representative environments prior to real-world deployment.

I. INTRODUCTION

BGP, the de-facto Internet routing protocol, connects over 77,000 Autonomous Systems (ASes), each an independent network making policy-based routing decisions and collectively forming the Internet [1]. It is the heterogeneity, within which each network operator is responsible for configuring the sending and receiving of their reachability information over BGP with the flexibility of optimising it for their own needs, that characterises the Internet. However, this is also a limitation, as without all ASes and their border routers implementing sufficient protections, a small policy change on one router can result in unintended consequences for thousands of other networks. It is particularly an issue when deliberate misconfigurations hijack traffic or make it non-routable.

A lack of real-world deployments means that the relative benefits of adopting each of these proposals have yet to be realised. There are also limited ways to test routing protocol implementations or specified policies in representative environments. These factors mean that routing equipment can respond unpredictably based on other vendor implementations,

or that small software bugs can propagate into widespread routing errors [2], neither of which are conducive to resilient Internet operation.

Much research focused on improving Internet routing protocols or device improvements is conducted in resource-constrained synthetic environments. In cases where physical routing testbeds are used, the topologies constructed are notably small, such as the two-device testbed of EPIC [3], or the SCIONLab [4] next-generation Internet testbed that contains only 56 AS-level networks. Whilst these provide high fidelity in network measurement, they are infeasible for representative testing for substantive or Internet-scale.

An alternative to physical testbeds is to mimic such environments as closely as possible through simulation or emulation. This approach substantially lowers build costs and provides benefits for scaling and configuration changes, which can be performed with comparatively little effort. Previous work has seen simulations developed on Internet topology data [4], but there is a clear divide between simulation and emulation [5]. In particular, network simulation is usually event-driven, and detail is sacrificed to constrain complexity, with the executed code representing abstractions of real-world behaviour.

In this paper, we explore the potential for container virtualisation in enabling large-scale emulations of Internet routers, allowing network equipment vendors to test the interoperability of their routing products and operators to validate their router policy configurations and potential external routing issues introduced by changes to these configurations. We construct *TERIS*, a Tool for Emulated Routing at Internet Scale, based on our preliminary work [6] and which we hope can provide insight into the potential consequences of making policy and configuration changes on the wider Internet. By replicating the existing Internet topology, we demonstrate that containerisation is a feasible approach to Internet border router emulation, and that many of the challenges faced by the 'physical' Internet can be reproduced through emulation.

This work focuses on the replication of BGP [7], a stateful inter-domain routing protocol, which requires a realistic network environment to capture the nuanced behaviours of session establishment, route propagation, and policy enforcement between ASes. Traditional event-driven simulators fall short in emulating these complexities due to their inability to replicate full TCP/IP stack interactions, persistent state maintenance, and protocol extensions such as BGPsec or multi-protocol BGP [8].

Contributions. We make three main contributions:

1) We present *TERIS*, a Kubernetes-centred Internet topology emulation tool capable of capturing router suite and configuration interoperability challenges.
2) We analyse the feasibility, scalability, and representativeness of our tool through a series of scenarios based on different viewpoints and scales of Internet topology, also measuring the resource impacts of BGP suite choice and policy configuration.
3) Finally, we make all our code and tooling publicly available[1] such that it can be used by other researchers and network operators, for instance in testing network configurations or evaluating routing policies.

[1] https://github.com/SystronLab/TERIS

II. METHODOLOGY

We capture the state of Internet topology and automate the process of generating a virtual testbed, readily deployable on orchestration platforms. Addressing the limitations of event-driven network simulation and the economically infeasible resource requirements of a physical testbed, we use container-based emulation where each container represents an AS running a real BGP routing daemon.

Containers provide lightweight, isolated network environments capable of supporting full protocol functionality, policy configurations, and multi-protocol extensions. This allows us to emulate misconfigurations, route leaks, and dynamic topological changes with high fidelity, while also enabling reproducible experiments at scale through orchestration platforms. By virtualising each AS, we can closely approximate the behaviour of the global Internet routing system, providing a safe, controlled, and representative testbed for studying BGP protocol enhancements and failures prior to real-world deployment. Our methodology is summarised in Fig. 1, and each stage of our methodology explained in this section.

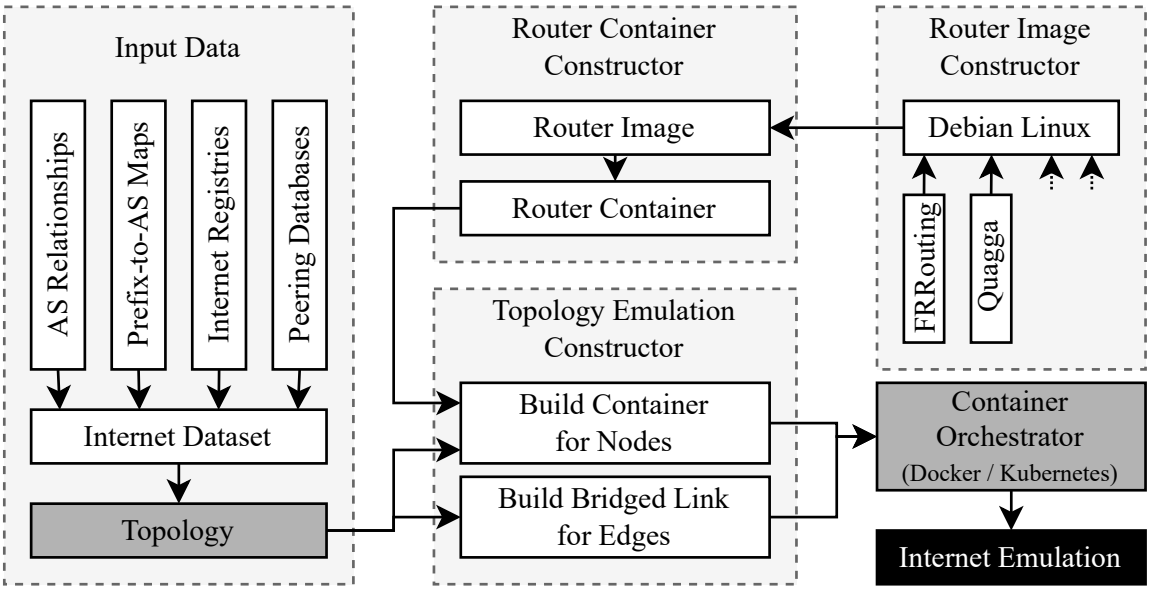


Figure 1. Overview of our methodology. We capture a snapshot of the Internet and produce a topology by fusing a variety of sources. We extend this, automating the creation of container images for each router in the topology and their associated routing configuration files.

A. Internet Topology Capture

Each AS is implemented as an individual router. Although not necessarily representative of all Internet routing dynamics, this provides significant scale and variety of routing configurations whilst helping to keep emulations from becoming unduly complex and simultaneously constrained to our available resource. We generate a graph-based Internet

topology representation using data collected from a variety of complementary sources for a fixed point in time, inspired by previous works [9], [10].

We enable the automated generation of these topology graphs by implementing loaders for CAIDA's *AS Relationships* [1] dataset, collecting a snapshot of inferred Internet topology structure for a specified day, where for each catalogued relationship we create a labelled 'edge' between two AS nodes in our graph-based representation. We additionally allow the input of data from PeeringDB snapshots [11], which provide self-reported and community generated information about ASes, such as the network type and presence at Internet Exchanges, alongside information about IXP-assigned router IP addresses. This can further be used to generate more complex routing configurations based on IXP presence or network type. We allow for the loading of IP prefixes advertised by each AS by reversing the CAIDA *Prefix-to-AS* dataset [12] or the BGP table as seen by *bgp.tools* [13]. For each AS within the topology, we collate a list of prefixes which we assume it will advertise to its neighbours.

B. Router Emulation

For each router software, we construct a dedicated container image. We demonstrate our approach using BGP based on two open-source routing software implementations: the popular but no longer maintained *Quagga* routing suite [14], which targets Unix-like platforms, and its Linux Foundation fork *FRRouting* [15], which has diverged in recent years, and notably requires a slightly different configuration format which we discuss later. Each requires a base image, which we construct using *Debian 12.5.0*, upon which we install a small number of basic network utilities and the relevant suite. It is the flexibility of this approach that means it is comparatively simple to extend *TERIS* to implement other routing suites or protocols, such as BIRD [16] and P4 [17]. Furthermore, we hope to add additional diversity by utilising images from commercial vendors.

For each router *TERIS* instantiates a list of network interfaces between direct routing neighbours based on the captured real-world Internet topology, and then creates and populates the necessary configuration files for the chosen software suite. In this paper, the diversity of configuration varies only by topological information, as presented in §II-C.

The primary advantage of this container-based emulation approach is its scalability. By reducing resource duplication, such as a router's underlying image and filesystem, and replicating only necessary elements (router memory, routing configuration and routing policies), *TERIS* aims to minimise resources required for large-scale emulation.

C. Generating Router Configurations

Each of the router containers requires a network interface through which to be able to speak to its direct neighbours, thereby creating a virtual LAN. In 'real' instances, this would be by connecting two networks physically, such as at an Internet Exchange Point (IXP). In the emulated environment, there is an added challenge in that it is imperative for later policy configuration (where each BGP neighbour needs a static IP address) that we assign addresses unique for each interface and that do not clash with the addresses used by direct neighbours. We therefore generate a unique identifier for each graph edge which is used as an index for

the network interface and to inform the generation of the subnets assigned to each such interface.

We introduce an additional 'dummy' interface to which each of the prefixes advertised by an ASN are routed, meaning that the complete address space advertised by a router can appear reachable in subsequent testing of an emulation (which can trivially be changed if such behaviour is not desired). We use this to derive a router's `router-id`, which is required to be neighbour-unique in the Quagga and FRRouting routing suites. We set this as the first address of a router's lowest prefix.

D. Routing Policy Configuration

As it is infeasible to derive the complete policy set for each router within an AS, we instead attempt to generate policy sets that replicate the expected behaviour of ASes depending on the AS *type* as self- or community-reported in the PeeringDB dataset. The inferencing of specific routing policies has been shown to be challenging [18], and although some information can be determined through transitive BGP community values, the meaning of the values themselves is not widely standardised and many networks remove these values upon receiving routes.

Instead, our approach of configuring network type-specific policies allows for the replication of key behaviour (such as transit networks or stub networks with no downstream ASes). Such an approach has limitations, but where richer data is available, we make it possible for others to provide their own input graph and configuration templates (or specific policies) which may in future remove this abstraction.

Base Policies. We provide an underlying configuration for all ASes to facilitate basic routing between ASes: customer networks advertise their own prefixes to their provider but receive the full routing table; peers advertise their own prefixes to all of their peers, and receive the same from their peers; and networks with no determined provider exchange the complete routing table with peers.

Extended Configurations. We also define extended policy sets for the following network types: transit networks carry traffic destined for another network – in these cases, the AS must advertise routes to prefixes it does not own to other networks such that it might appear in the BGP path; and content networks advertise their own prefixes to all peers, in addition to networks identified as customers and providers, but are configured (in our emulation) to never transit traffic destined for other networks.

E. Inter-Container Networking

Inter-container networking implementations differ between container orchestration frameworks. *TERIS* generates emulation configurations for each of *Kathará* [19] and its multi-host sibling *Megalos* [20], which differs slightly in format and capability. Kathará supports Docker-based deployments of up to approximately 1000 containers but can be used for smaller-scale testing, but Megalos is a Kubernetes-based container orchestration framework that implements (through chained Container Network Interfaces or *CNI*) eBGP routing

across multiple nodes and the deployment of virtual LANs, each representing a container-to-container connection. Therefore, with Megalos, it becomes possible to produce a large heterogeneous emulation without some of the resource restrictions imposed by the internal network spanning-tree in a single-host Docker deployment. For the remainder of this paper we discuss the Megalos implementation, for which there exist a number of scalability and architectural challenges as a consequence of the larger number of emulated hosts.

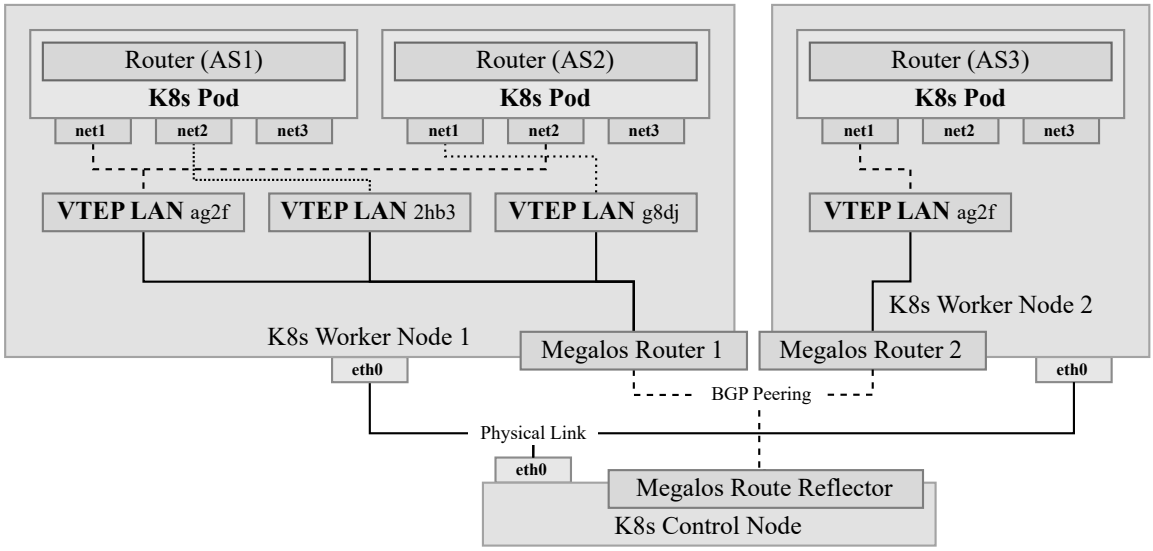


Figure 2. The emulation network architecture. Containers use virtual 'net' interfaces which correspond with different VXLAN Tunnel End Points (VTEPs). These VTEPs effectively form LANs for each router-to-router connection, using MAC-in-UDP encapsulation to allow the carriage of untampered packets, allowing the virtualisation architecture to remain invisible to the routers themselves.

For each AS within the topology graph we produce a single Kubernetes pod. These are natively assigned an 'IP-per-pod' address accessible through the onboard `eth0` interface, which we then disable in the router configurations to prevent route pollution (where routes between routers might default to a path through the Kubernetes host). Each topology edge becomes a distinct virtual LAN, implemented as a Megalos VXLAN segment, connecting two or more containers. Each VXLAN segment is identified using a VXLAN network identifier, a four-character code comprised of numbers and letters (base-36) and providing a theoretical maximum far in excess of the true number of Internet topology adjacencies (Fig. 2).

We use the largely redundant `ethX` interface of the pods to collect telemetry from each container. For this we use cAdvisor (https://github.com/google/cadvisor), a resource usage and performance monitor for Kubernetes clusters. This collects data from each container using daemons running in the same Kubernetes cluster which we collate and export as a single API endpoint we export using a Prometheus (https://prometheus.io) instance. Prometheus is itself compatible with a number of data visualisation and analysis frameworks.

III. RESULTS AND EVALUATION

In this section we evaluate the feasibility, scalability and representativeness of our approach. Based on the topology graph captured in §II-A, we develop a series of subgraphs to demonstrate each of these characteristics, each of which are then input into the graph-to-emulation pipeline from §II. To deploy these scenarios, we use a Kubernetes cluster comprising of two interconnected desktops running Linux Mint 22.1 with 256GB RAM and 48-core Intel Xeon processors.

A. Demonstrating Feasibility

We demonstrate the feasibility of the *TERIS* approach using three different network configuration scenarios based on the same underlying topology subgraph, validating *TERIS* emulations deploy and perform as expected, and the behaviour exhibited matches our expectations. We further ensure that the emulations are not restricted by unwanted external factors, such as host memory or CPU throttling. Similarly, we also use solely the *Quagga* routing suite to avoid potential issues with configuration incompatibility across scenarios.

In each case we implement solely peer-based relationships between connected ASes, with no route filtering. Whilst this is highly unrealistic, it serves as a useful baseline to demonstrate that routing information is exchanged between peers.

In each scenario, the emulated routers take a short time to reach BGP convergence – which is achieved within a few seconds of the successful emulation deployment (by propagation of routing announcements) – and upon doing so we observe largely static CPU usage, and stable memory usage, which varies between each of the different protocol implementations. We also note that this is substantially beneath the resource limits of the host machines.

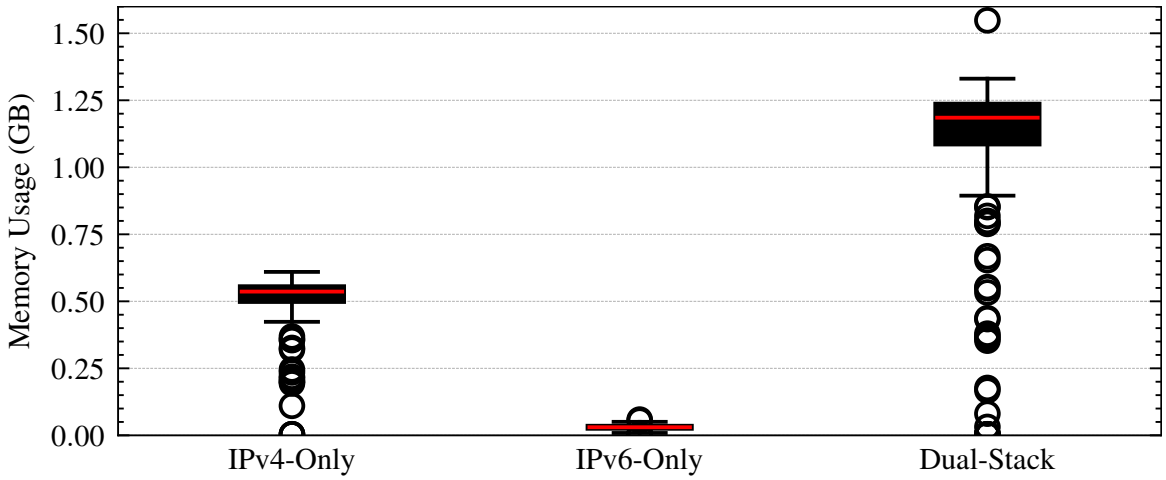


Figure 3. Per-container memory usage for different IP addressing implementations. These boxplots show IPv4 implementation contributing most to the variation in resource requirement and reduced requirements for IPv6 topology, a factor of the reduced adoption of IPv6 in real-world Internet routing.

Scenario 1: IPv4-only addressing. We use a router image without IPv6 addressing enabled and do not implement any IPv6-based routing configuration, meaning that only

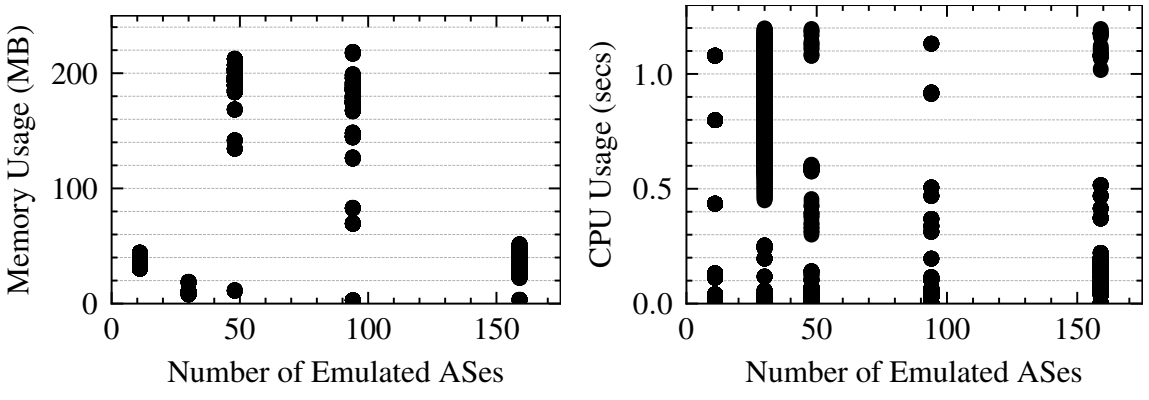
IPv4-based routing and reachability exchanges are possible. All emulated ASes participate in IPv4 peering and therefore this serves as a useful test of connectivity, which is validated by ensuring at least one path exists from any given AS to all others. The memory resource usage of this scenario is presented in Fig. 3.

Scenario 2: IPv6-only addressing. We implement a similar scenario wherein IPv4-based routing has been replaced with IPv6. We only implement peerings between ASes where one exists in the real topology, reducing the volume of interconnections as a product of the lower adoption of IPv6 within real-world ASes. The nature of IPv6 addressing also means that fewer prefixes are exchanged as the address pool is less segmented than its IPv4 equivalent. As shown in Fig. 3, the combination of a reduced routing table and fewer direct neighbours has a notable impact in the reducing in resource.

Scenario 3: Dual-stack addressing. We implement a dual-stack approach which could be considered most representative of the 'real' Internet (and what is used hereafter), combining the IPv4 and IPv6 configurations, with reuse of the same VXLAN where two routers are peered through both IPv4 and IPv6 to reduce resource overhead. The increase in the number of network interfaces on each router leads to an increase in the overall resource demand, shown in Fig. 3.

B. Demonstrating Scalability

We address the challenge of scalability by undertaking a series of emulations with topologies of different sizes to identify the relationship between emulation size and resource requirements. From the topology graph in §II-A, we extract the interconnected (not necessarily complete) subgraph for the first n highest-degree ASes, with increasing n. This naturally means the first subgraphs are more complete than those that come with higher n.



(a) Memory usage. (b) CPU usage.

Figure 4. Memory and CPU usage per container for increasing topology sizes. This shows increased usage where the number of stub ASes is disproportionately low and therefore more route permutations, alongside fairly stable CPU usage irrespective of emulation size.

We use a 'no-policy' BGP configuration wherein all local and received IP prefixes are advertised to all neighbours and all ASes can act as *transit* providers, with no preconfigured local preferences. Notably both Fig. 4(a) and Fig. 4(b) show neither the per-container memory

or CPU resource usage increases at a linear or greater rate. Significantly, the mean per-container resource usage is 63% lower than that achieved in previous work [5]. Importantly, the latter figure shows CPU usage remaining relatively stable irrespective of the topology size.

C. Demonstrating Representativeness

Finally, we consider the representativeness of our emulations, which we validate by comparing aspects of our emulation with that of the real-world equivalents.

Scenario 1: Realistic routing policies. In this first scenario we implement two configurations: the 'no policy' BGP configuration used in §III-B; and secondly the extended configurations implementation discussed in §II-D. This latter configuration means only a small subset of ASes provide transit for other networks, and thus traffic routes must traverse the Internet hierarchy of tier-1 (the transit core), tier-2, tier-3 and 'stub' networks. As shown in Fig. 5, the impact of this also reflects on the memory requirements of each AS as the number of sent and received routes has reduced significantly, reducing the size of the routing table. We perform validation by comparing the output BGP table to equivalents obtained through the *bgp.tools* [13] looking glass.

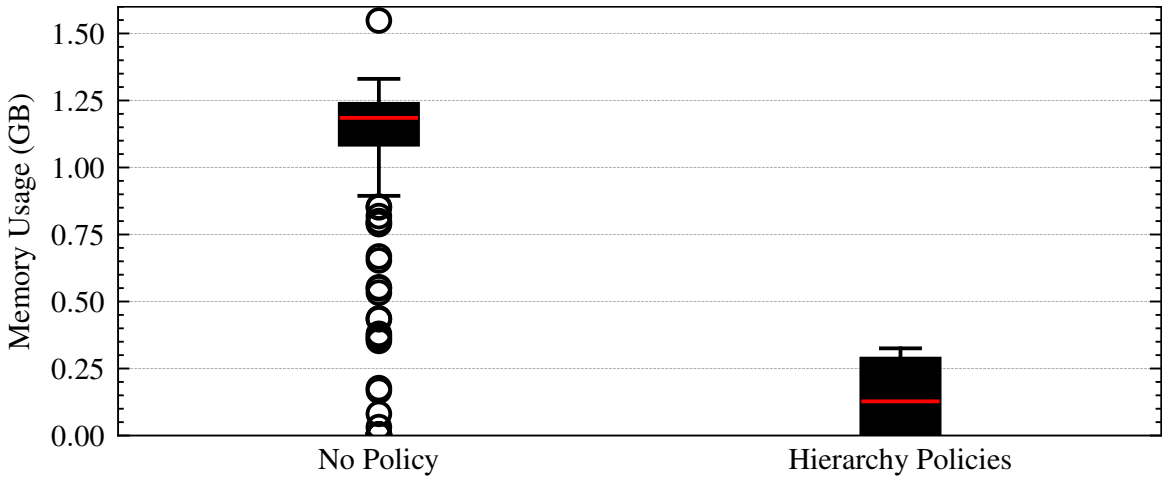


Figure 5. Memory usage per container before and after the implementation of policies enforcing the Internet hierarchy, such as non-transitory, customer and provider relationships. Emulation is dual-stack IPv4 and IPv6.

Scenario 2: Country-level validation. We consider the topologies of three countries with increasing volumes of domestically registered ASes: Zambia (11 ASes), Luxembourg (96 ASes), and Norway (282 ASes). For these, we extract the country-level subgraph from the Internet topology in §II-A and use this to inform the generation of an emulation, using hierarchical policies, for each.

We use these topologies to validate whether we can recreate the same country-level topology graph using data collected from the highest-degree AS within each country (which

Table I
HOST MEMORY USAGE

Country	Containers	Memory Usage (MB)
Zambia	11	56.12
Luxembourg	96	854.97
Norway	282	1581.30

should have a maximal viewpoint), which we validate using the perspectives found in the *bgp.tools* [13] looking glass.

These scenarios also support our scalability findings in §III-B, demonstrating the resource requirements from our emulation approach are better than linear, as shown in Table I.

Scenario 3: Different routing suites. In this scenario we demonstrate the representativeness of our approach because of its support for different routing suites and operating system (container) images. We reuse the Zambia and Luxembourg topologies and for each implement equivalent BGP policy configurations in the format required by the FRRouting suite.

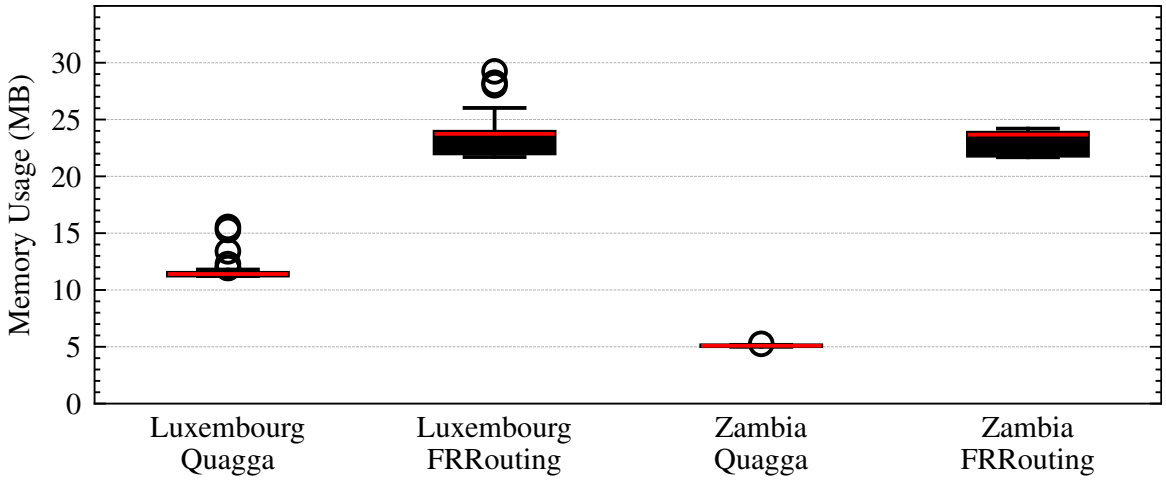


Figure 6. Memory usage per container. In this boxplot, we show the range of per-container memory usage for Luxembourg and Zambia, under the two routing suites.

As would be expected, these converge to a near-identical graph, with minor differences in cases where the tie-breaker in the BGP route selection algorithm has chosen a different preference, but in each such case the route selected previously is still received. In Fig. 6 we see that the resource requirements of the FRRouting suite are higher than that of the Quagga alternative, potentially indicative of the minimum resource requirements of FRRouting instances. Significantly, even in the FRRouting case, this remains a 47% reduction in memory resource requirement relative to previous work [5].

IV. RELATED WORK

The importance of realistic network testbeds have been widely recognised in the study of network protocols. We focus specifically on achieving *realism*, essential for ensuring that a testbed is both relevant and applicable; and on addressing the challenges of testbed design and

scalability. Ontology-driven approaches, such as the Internet Yellow Pages [9] have shown the feasibility of integrating multiple distinct Internet datasets into complex knowledge graphs that provide meaningful context for Internet routing data. Here, we consider only a small subset of this wider Internet data, focusing on the elements necessary to replicate inter-AS relationships and model the macroscopic view of the BGP routing system, sacrificing data richness in a trade-off with data volume reduction (with *TERIS* representing the BGP topology in only 74 MB) and improved explainability, where problems can be traced to their source data. Our approach is designed to be highly interoperable with richer topology sources provided in the `.graphml` format.

Testing network and Internet protocols has led to the construction of numerous testbeds with varying fidelity and scalability. Physical testbeds such as SCIONLab [21] and PEERING [22] offer realism and high fidelity, but are constrained by the equipment and cost required for controlled topology variation or scalable experimentation. VM-based testbeds [23]–[26] offer benefits due to their scalability and isolation, and have demonstrated their feasibility in accurately replicating production networks [27], [28] – however, they remain considerably more resource-intensive than our lightweight container-based approach. The closest work, AutoNetkit [5], utilises a similar graph-driven approach to topology representation, but deploys using the discontinued Netkit tool, requiring 37 GB RAM to replicate 1158 routers and 1470 links – nearly six times the memory requirements of *TERIS* on the same size of topology.

Other approaches use containerisation, such as IoT-focused UiTiOt [29], or replications of production topology and BGP configuration using Kathará [30], [31] – demonstrating the flexibility of container-based emulation. *TERIS* applies these benefits to the inter-AS topology and at increased scale.

V. CONCLUSION

In this work, we have presented *TERIS*, a new Internet emulation approach that captures real-world Internet topology to produce a scalable and representative container-based emulation environment. This approach enables the testing and identification of interoperability challenges between different protocol implementations and allows researchers to validate new routing strategies on a realistic Internet-scale testbed. We have evaluated the feasibility, scalability, and representativeness of *TERIS*, demonstrating a 63% reduction per-container resource usage compared to previous work. Overall, *TERIS* provides a foundation for testing routing suites and configurations at Internet scale.

Acknowledgements

This work is supported, in part, by EPSRC and DSIT under grants: EP/X040518/1, EP/Y037421/1, and EP/Y019229/1.

References

[1] CAIDA, "The CAIDA AS relationships dataset, Jan 2025." [Online]. Available: https://www.caida.org/catalog/datasets/as-relationships/

[2] T. Strickx, "How Verizon and a BGP optimizer knocked large parts of the Internet offline today," Jun. 2019. [Online]. Available: https://blog.cloudflare.com/how-verizon-and-a-bgp-optimizer-knocked-large-parts-of-the-internet-offline-today/

[3] M. Legner, T. Klenze, M. Wyss, C. Sprenger, and A. Perrig, "Epic: every packet is checked in the data plane of a path-aware Internet," in *Proceedings of the 29th USENIX Conference on Security Symposium*, ser. SEC'20. USA: USENIX Association, 2020.

[4] S. Tabaeiaghdaei, S. Scherrer, J. Kwon, and A. Perrig, "Carbon-aware global routing in path-aware networks," in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, ser. e-Energy '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 144–158.

[5] S. Knight, H. Nguyen, O. Maennel, I. Phillips, N. Falkner, R. Bush, and M. Roughan, "An automated system for emulated network experimentation," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. Santa Barbara California USA: ACM, Dec. 2013, pp. 235–246.

[6] J. Levett, V. Vassilakis, and P. Yadav, "Building block for the Internet digital twin: Scalable and automated internetwork emulator," in *2025 9th Network Traffic Measurement and Analysis Conference (TMA)*, 2025, pp. 1–4.

[7] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," Internet Engineering Task Force, Request for Comments RFC 1654, Jul. 1994. [Online]. Available: https://datatracker.ietf.org/doc/rfc1654

[8] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "A survey on network simulators, emulators, and testbeds used for research and education," *Computer Networks*, vol. 237, p. 110054, 2023.

[9] R. Fontugne, M. Tashiro, R. Sommese, M. Jonker, Z. S. Bischof, and E. Aben, "The wisdom of the measurement crowd: Building the Internet Yellow Pages a knowledge graph for the Internet," in *Proceedings of the 2024 ACM on Internet Measurement Conference*, ser. IMC '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 183–198.

[10] S. Anderson, L. Salamatian, Z. S. Bischof, A. Dainotti, and P. Barford, "iGDB: connecting the physical and logical layers of the Internet," in *Proceedings of the 2022 ACM on Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 433–448.

[11] PeeringDB, "The CAIDA ucsd PeeringDB dataset, Jan 2025." [Online]. Available: https://www.caida.org/catalog/datasets/peeringdb/

[12] CAIDA, "RouteViews prefix to AS mappings dataset for ipv4 and ipv6, Jan 2025." [Online]. Available: https://www.caida.org/catalog/datasets/routeviews-prefix2as/

[13] Port 179 Ltd, "bgp.tools," 2018. [Online]. Available: https://bgp.tools/

[14] P. Jakma and D. Lamparter, "Introduction to the Quagga routing Suite," *IEEE Network*, vol. 28, no. 2, pp. 42–48, 2014.

[15] Linux Foundation, "FRRouting project," 2017. [Online]. Available: https://frrouting.org/

[16] BIRD Team, "The BIRD Internet Routing Daemon," 2000. [Online]. Available: https://bird.network.cz/

[17] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, jul 2014.

[18] S. Kastanakis, V. Giotsas, I. Livadariu, and N. Suri, "Replication: 20 years of inferring interdomain routing policies," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, ser. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 16–29.

[19] G. Bonofiglio, V. Iovinella, G. Lospoto, and G. Di Battista, "Kathará: A container-based framework for implementing network function virtualization and software defined networks," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.

[20] M. Scazzariello, L. Ariemma, G. Di Battista, and M. Patrignani, "Megalos: A scalable architecture for the virtualization of large network scenarios," *Future Internet*, vol. 13, no. 9, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/9/227

[21] J. Kwon, J. A. García-Pardo, M. Legner, F. Wirz, M. Frei, D. Hausheer, and A. Perrig, "SCIONLab: A next-generation Internet testbed," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2020. [Online]. Available: https://netsec.ethz.ch/publications/papers/icnp2020_scionlab.pdf

[22] B. Schlinker, T. Arnold, I. Cunha, and E. Katz-Bassett, "Peering: virtualizing BGP at the edge for research," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 51–67.

[23] W. T. Mnyandu, A. Terzoli, and H. Kobo, "Emulating LTSP clients with Mininet and QEMU: Enabling scalable testing for educational deployments," in *2024 4th International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 2024, pp. 30–37.

[24] K. Yao, W. Sun, M. Alam, M. Xu, and V. Devabhaktuni, "A real-time testbed for routing network," in *Testbeds and Research Infrastructure. Development of Networks and Communities*, T. Korakis, M. Zink, and M. Ott, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 256–270.

[25] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 253–264.

[26] Sonic Foundation, "Software for Open Networking in the Cloud (SONiC)," 2023. [Online]. Available: https://sonicfoundation.dev/

[27] H. H. Liu, Y. Zhu, J. Padhye, J. Cao, S. Tallapragada, N. P. Lopes, A. Rybalchenko, G. Lu, and L. Yuan, "CrystalNet: Faithfully emulating large production networks," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 599–613.

[28] Z. Gao, A. Abhashkumar, Z. Sun, W. Jiang, and Y. Wang, "Crescent: Emulating heterogeneous production network at scale," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 1045–1062. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/gao-zhaoyu

[29] C. D. L. Bao, N. L. Trong, and Q. Le-Trung, "UiTiOt: A container-based network emulation testbed," in *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*, ser. ICMLSC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 161–166.

[30] M. Polverini, I. Germini, A. Cianfrani, F. G. Lavacca, and M. Listanti, "A digital twin based framework to enable "what-if" analysis in BGP optimization," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–6.

[31] T. Caiazzi, M. Scazzariello, and L. Ariemma, "VFTGen: a tool to perform experiments in virtual fat tree topologies," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 718–719. [Online]. Available: https://opendl.ifip-tc6.org/db/conf/im/im2021demo/213179.pdf

Appendix A
Ethics

Despite emulation being the primary focus of this work, the experiments shown have relied on data collected from a range of external datasets. For each dataset, we have ensured that their licence or Acceptable Use Agreement (AUA) provides the rights required to integrate their data into our graph-based representation, and in this work we have ensured all such datasets are acknowledged through citations in addition to compliance with individual licence terms. As such, we believe this work does not raise any ethical issues.

Appendix B
Artifacts

To enable the extension and reproduction of our work, we make available the code required to form the topology we create in §II-A, our codebase for the emulation generation tooling, our measurement configuration, router configuration template files, and our analysis scripts. Whilst the nature of emulation means exact results are not possible to recreate, results similar or exhibiting the same patters should be fully reproducible.

- **Data set:** Each of the CAIDA AS Relationships dataset [1], the CAIDA PeeringDB snapshot [11], and one of either the CAIDA Prefix-to-AS dataset [12] or the bgp.tools ASNs and Table data [13].
- **Run-time environment:** For smaller single-host emulations with up to $\tilde{1}000$ containers it is possible to use Kathará [19] alongside a standard Docker Engine installation. For larger emulations a Kubernetes environment, potentially clustered across multiple hosts is required, with the Megalos CNI implemented.
- **Hardware:** This varies depending on the scale of the emulation desired, but for our contributions we use 2 high-end desktop machines each with 256GB RAM and 48-core Intel Xeon processors.
- **How much disk space required (approximately)?:** 20GB.
- **How much time is needed to prepare workflow (approximately)?:** 2-3 hours, but potentially longer if less familiar with Kubernetes.
- **How much time is needed to complete experiments (approximately)?:** Less than 6 hours for each of our experiments, but longer with larger topologies.
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** BSD-3

How to access. We make available our code, required configuration templates and supporting documentation in a GitHub repository[2].

Installation. We provide comprehensive dependency installation instructions in the repository `README.md`. Fundamentally, we require a Docker or Kubernetes environment with associated container daemons and a deployment mechanism. For Kubernetes-based emulations we also have specific CNI templates. We also provide instructions for the setup of Prometheus and cAdvisor, two tools supporting the measurement of each experiment shown in this paper, but not required for Internet emulations more generally.

Experiment workflow. After installation of all of the required components with datasets in the `source-data/` directory, use the `00-foundations.ipynb` Jupyter notebook to compile an Internet topology graph (or load your own in a `.graphml` format) and follow the steps in the notebook to

[2]Link not included for anonymity.

create each of the topologies used in this paper. We provide supporting instructions for the extraction of data from a connected Prometheus instance using a Python script, as well as supporting notebooks for measurement analysis.

Evaluation and expected results. For each emulated topology, it should be possible to use `kathara connect <ASN>` to access the console of any of the emulated routers. Using the Python script to extract emulation data exports it in plaintext to a chosen output file (e.g. `results.txt`). This can then be loaded into one of the analysis notebooks to reproduce the versions of each of the figures presented in this paper.