Active Internet measurement to support policy research

David Clark
MIT

Matthew Luckie, Shivani Hariprasad, kc claffy
CAIDA/UCSD

August 20, 2025

Active Internet measurement to support policy research[1]

While Internet measurement findings are relevant to policymakers, economists, and advocates, direct engagement with these communities is rare. Barriers to engagement include limited technical fluency among policy audiences and a lack of visibility into what measurement experts can do. As a result, policy researchers often must discover and interpret technical work on their own. Cross-disciplinary exchange barriers are often insurmountable. Recent advances in Internet measurement infrastructure lower this barrier by making it possible to run sophisticated global tests in minutes—without deep technical expertise—and to locate existing datasets more easily. Our goal is to spark cross-disciplinary collaboration, enabling more researchers to run Internet measurements. By reducing the time needed to launch global experiments from *months to minutes*, we aim to make Internet measurement research accessible well beyond the circle of technical experts.

Active measurement requires vantage points (VPs) at the network edge. One approach is *crowd-sourced measurement*, where users run tests from their own computers. The most common example is speed testing, as with Ookla[2], MLab NDT[3], and Cloudflare[4]. People run these tests to check if they're getting the performance they paid for or to troubleshoot slow connections. However, crowd-sourced data limits the control of the researcher over the data collected. An alternative is to deploy *dedicated VPs* programmed specifically for active measurement. The most notable example is RIPE Atlas, which supports a limited set of measurements but has broad deployment. CAIDA's Ark (Archipelago) platform has fewer probes than RIPE Atlas, but supports a wider variety of basic probing functions that can be tailored and combined into complex experiments.

Balancing Capability and Safety Active Internet measurement is not risk-free. Poorly designed experiments can consume excessive resources, violate laws, or cause reputational harm. For example, probing content blocked in a jurisdiction by downloading prohibited material could create legal or diplomatic issues. Ark's design balances capability with safety. Instead of giving users raw packet-sending ability, Ark offers a Python-based programming environment with predefined measurement primitives. These primitives clearly define the type of traffic each VP will send, which allows Ark operators to constrain permissible experiments and helps VP hosts understand exactly how their vantage points will be used and risks involved. For more on the design choices of the Ark programming environment, see Luckie *et al.*'s overview [2] and project page, which also links to papers recently published using the platform [4].

Design goals and capabilities Three design goals that shaped the Ark interface:

- **Ease of use:** Measurement capabilities are exposed via Python, so researchers can run tests and receive results as Python objects, with many reusable libraries available [3].
- **Performance:** Low latency between measurement and results enables responsive, reactive experiments.

[1]Work sponsored by National Science Foundation (NSF) grant OAC-2131987, OAC-2530871, CNS-2212241. Views do not represent the official policies or endorsements of NSF.
[2]https://www.speedtest.net/
[3]https://speed.measurementlab.net
[4]https://speed.cloudflare.com/

- **Interoperable and Extensible:** Ark's software runs on diverse operating systems, works in containers, has minimal optional dependencies, and is distributed in packaged form for easy deployment.

Ark probes can perform:

- `ping`: measure latency to an IP address.
- `traceroute`: discover the sequence of router interfaces to a target IP.
- **DNS queries**: A, AAAA, TXT, SOA, NS, MX queries, over TCP or UDP, with NSID and ECS options
- **HTTP GET**: fetch web pages.
- **alias resolution**: mapping IP addresses to common routers.
- **more advanced tests**, e.g., multi-step probing and service-specific measurements.

These capabilities are programmed in Python, allowing experimenters to combine primitives into sequences. For example, a script could perform a DNS lookup to resolve a hostname, then ping the resulting IP. Programs are short and expressive. For instance, measuring latency from every Ark probe to a target IP and reporting the minimum can be done in a few lines of code [1]. One could extend such a script into a geolocation tool using triangulation. At first glance, the value of these measurement primitives may not be obvious. However, researchers have used the platform to investigate a range of important questions:

- **Global latency to a specific URL** – What is the latency from each Ark node to a specific URL?
- **Path diversity and security** – When reaching a given URL from different parts of the globe, how many Autonomous Systems (ISPs) does the path cross? This relates to resilience and resistance to route hijacking, among other concerns.
- **Web hosting distribution** – For popular websites in various countries, how many are hosted locally, how many use multiple hosting locations to provide low-latency access worldwide, and how many are hosted entirely outside the country?
- **Latency variation over time** – For a given network path, does latency vary over a day or a week? Such patterns can reveal potential congestion.
- **Anycast deployment patterns** – Some network services use anycast, where multiple servers share the same IP address and routing directs users to the nearest instance. What does the global deployment look like? Which probes reach which servers, and what patterns emerge?

References

[1] David Clark and Matthew Luckie and kc claffy. The Current State of the Art in Network Measurement. In *Telecommunications Research Policy Conference (TPRC)*, 2025. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5375475.

[2] M Luckie, S Hariprasad, R Sommese, B Jones, K Keys, R Mok, and k claffy. An Integrated Active Measurement Programming Environment. In *Passive and Active Measurement Conference (PAM)*, Dec. 2024.

[3] Matthew Luckie. scamper python module documentation, 2024. https://www.caida.org/catalog/software/scamper/python/.

[4] Matthew Lucklie. https://www.caida.org/projects/ark/programming/, 2025.