PATHspider: A tool for active measurement of path transparency

Iain R. Learmonth
University of Aberdeen, Scotland
iain@erg.abdn.ac.uk

Brian Trammell
ETH Zurich, Switzerland
trammell@tik.ee.ethz.ch

Mirja Kuhlewind
ETH Zurich, Switzerland
mirja.kuehlewind@tik.ee.ethz.ch

Gorry Fairhurst
University of Aberdeen, Scotland
gorry@erg.abdn.ac.uk

ABSTRACT

In today's Internet we see an increasing deployment of middleboxes. While middleboxes provide in-network functionality that is necessary to keep networks manageable and economically viable, any packet mangling – whether essential for the needed functionality or accidental as an unwanted side effect – makes it more and more difficult to deploy new protocols or extensions of existing protocols. For the evolution of the protocol stack, it is important to know which network impairments exist and potentially need to be worked around. While classical network measurement tools are often focused on absolute performance values, we present a new measurement tool, called *PATHspider* that performs A/B testing between two different protocols or different protocol extensions to perform controlled experiments of protocol-dependent connectivity problems as well as differential treatment. *PATHspider* is a framework for performing and analyzing these measurements, while the actual A/B test can be easily customized. This paper describes the basic design approach and architecture of *PATHspider* and gives guidance how to use and customize it.

CCS Concepts

•**Networks** → **Network experimentation; Network measurement;** *Transport protocols;*

Keywords

Active network measurement, path transparency, transport protocol features, network experimentation, measurement tools

1. INTRODUCTION

Network operators increasingly rely on in-network functionality to make their networks manageable and economically viable. These middleboxes make the end-to-end path for traffic more opaque by making assumptions about the traffic passing through them. This has led to an ossification of the Internet protocol stack: new protocols and extensions can be difficult to deploy when middleboxes do not understand them [8]. This paper presents a new software measurement tool, *PATHspider*, for active measurement of Internet path transparency to transport protocols and transport protocol extensions, to generate raw data at scale to determine the size and shape of this problem.

The A/B testing measurement methodology used by *PATHspider* is simple: We perform connections from a set of observation points to a set of measurement targets using two configurations. A baseline configuration (A), usually a TCP connection using kernel default and no extensions, tests basic connectivity. These connections are compared to the experimental configuration (B), which uses a different transport protocol or set of TCP extensions. These connections are made as simultaneously as possible, to reduce the impact of transient network changes.

PATHspider is a generalized version of the *ecnspider* tool, used in previous studies to probe the paths from multiple vantage points to web-servers [14] and to peer-to-peer clients [7] for failures negotiating Explicit Congestion Notification (ECN) [10] in TCP.

As a generalized tool for controlled experimental A/B testing of path impairment, *PATHspider* fills a gap in the existing Internet active measurement software ecosystem. Existing active measurement platforms, such as RIPE Atlas [11], OONI [5], or Netalyzr [9], were built to measure absolute performance and connectivity between a pair of endpoints under certain conditions. The results obtainable from each of these can of course be compared to each other to simulate A/B testing. However, the measurement data obtained from these platforms provide a less controlled view than can be achieved with *PATHspider* given coarser scheduling of measurements in each state.

In this paper we present the design approach and architecture of *PATHspider*. Source code and documentation for *PATHspider* is available at
`https://pathspider.mami-project.eu/`

PATHspider as a tool is still work-in-progess. However, we are currently using it to test the deployment and deployability of different transport protocols and TCP extensions on the open Internet, including TCP Fast Open[3], Multipath

ANRW '16, July 16 2016, Berlin, Germany

DOI: http://dx.doi.org/10.1145/2959424.2959441

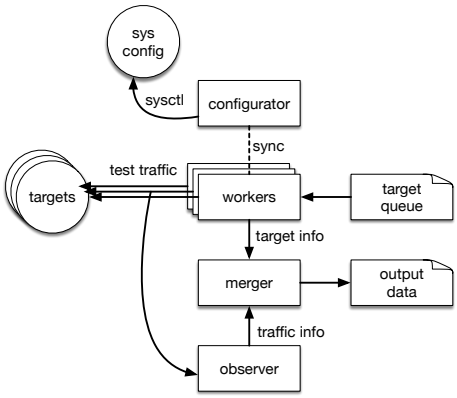

Figure 1: Block diagram illustrating control flow and flow of data between *PATHspider* components.

TCP[6], UDP options[12] and UDP with zero checksum[4]. To collect these kind of measurement data at scale, we are currently deploying *PATHspider* on existing measurement testbeds such as CAIDA Ark and MONROE.

Given *PATHspider* 's modular design and implementation in Python, plugins to perform measurements for any transport protocol or extension are easy to build and can take advantage of the rich Python library ecosystem, including high-level application libraries, low-level socket interfaces, and packet forging tools such as Scapy [2]. While presenting no new measurement results, we present an approach for running controlled experiments of path impairments in the Internet, and offer *PATHspider* as a framework for doing so.

2. ARCHITECTURE

The *PATHspider* architecture has four components, illustrated in figure 1: the configurator, the workers, the observer and the merger. Each component is implemented as one or more threads, launched when *PATHspider* starts.

For each target hostname and/or address, with port numbers where appropriate, *PATHspider* enqueues a job, to be distributed amongst the worker threads when available. Each worker performs one connection with the "A" configuration and one connection with the "B" configuration. The "A" configuration will always be connected first and serves as the base line measurement, followed by the "B" configuration. This allows detection of hosts that do not respond rather than failing as a result of using a particular transport protocol or extension. These sockets remain open for a post-connection operation.

Some transport options require a system-wide parameter change, for example enabling ECN in the Linux kernel. This requires locking and synchronisation. Using semaphores, the configurator waits for each worker to complete an operation and then changes the state to perform the next batch of operations. This process cycles continually until no more jobs remain. In a typical experiment, multiple workers (on the order of hundreds) are active, since much of the time in a connection test is spent waiting for an answer for the target or a timeout to fire.

In addition, packets are separately captured for analysis by the observer using Python bindings for libtrace [1]. First, the observer assigns each incoming packet to a flow based on the source and destination addresses, as well as the TCP, UDP or SCTP ports when available. The packet and its associated flow are then passed to a function chain. The functions in this chain may be simple functions, such as counting the number of packets or octets seen for a flow, or more complex functions, such as recording the state of flags within packets and analysis based on previously observed packets in the flow. For example, a function may record both an ECN negotiation attempt and whether the host successfully negotiated use of ECN.

A function may alert the observer that a flow should have completed and that the flow information can be matched with the corresponding job record and passed to the merger. The merger extracts the fields needed for a particular measurement campaign from the records produced by the worker and the observer.

3. EXTENSIBILITY

PATHspider plugins are built by extending an abstract class that implements the core behaviour, with functions for the configurator, workers, observer, and matcher.

There are two configurator functions: `config_zero` and `config_one`, run by the configurator to prepare for each attempted connection mode. Where system-wide configuration is not required, the configurator provides the semaphore-based locking functions. This makes the workers aware of the current configuration allowing the connection functions to change based on the current configuration mode.

There are three connection functions: `pre_connect`, `connect` and `post_connect`. `connect` is the only required function. The call to this function is synchronised by the configurator. The `pre_connect` and `post_connect` functions can preconfigure state and perform actions with the connections opened by the `connect` function without being synchronised by the configurator. This can help to speed-up release of the semaphores and complete jobs more efficiently. These actions can also perform data gathering functions, for example, a traceroute to the host being tested.

Plugins can implement arbitrary functions for the observer function chain. These track the state of flows and build flow records for different packet classes: The first chain handles setup on the first packet of a new flow. Separate chains chains for IPv4, IPv6, TCP and UDP packets to allow different behaviours based on the IP version and transport protocol.

The final plugin function is the merger function. This takes a job record from a worker and a flow record from the observer and merges the records before passing the merged record back to *PATHspider*.

4. CONCLUSIONS

We have presented *PATHspider*, a new open-source tool for controlled experiments of path transparency impairments. *PATHspider* is a part of an emerging platform being built within the EU H2020 MAMI project. It will feed its observations into a path transparency observatory for integrated analysis, and future integration of mPlane [13] will allow orchestration of large-scale, long-running observations.

By enabling larger-scale, more diverse measurements of impairments to path transparency, we expect *PATHspider* to help inform development of new protocol mechanisms and protocols, and to reduce the burden of deploying them.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421, and was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0268. The opinions expressed and arguments employed reflect only the authors' views. The European Commission is not responsible for any use that may be made of that information. Further, the opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

5. REFERENCES

[1] S. Alcock, P. Lorier, and R. Nelson. Libtrace: A packet capture and analysis library. *SIGCOMM Comput. Commun. Rev.*, 42(2):42–48, Mar. 2012.

[2] P. Biondi. Scapy: explore the net with new eyes. Technical report, EADS Corporate Research Center, http://www.secdev.org, 2005.

[3] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. TCP Fast Open. RFC 7413 (Experimental), Dec. 2014.

[4] G. Fairhurst and M. Westerlund. Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums. RFC 6936 (Proposed Standard), Apr. 2013.

[5] A. Filasto and J. Appelbaum. Ooni: Open observatory of network interference. In *FOCI*, 2012.

[6] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (Experimental), Jan. 2013.

[7] E. Gubser. Explicit Congestion Negotiation (ECN) support based on P2P networks. ftp://ftp.tik.ee.ethz.ch/pub/students/2015-FS/SA-2015-05.pdf.

[8] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is It Still Possible to Extend TCP? In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 181–194, New York, NY, USA, 2011. ACM.

[9] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating The Edge Network. In *Internet Measurement Conference (IMC)*, 2010.

[10] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001. Updated by RFCs 4301, 6040.

[11] R. N. Staff. RIPE Atlas: A Global Internet Measurement Network. *Internet Protocol Journal*, 18(3), September 2015.

[12] J. Touch. Transport options for udp. Internet-Draft draft-touch-tsvwg-udp-options-02, IETF Secretariat, January 2016. http://www.ietf.org/internet-drafts/draft-touch-tsvwg-udp-options-02.txt.

[13] B. Trammell, P. Casas, D. Rossi, A. Bar, Z. Houidi, I. Leontiadis, T. Szemethy, and M. Mellia. mPlane: an intelligent measurement plane for the Internet. *Communications Magazine, IEEE*, 52(5):148–156, 2014.

[14] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger. Enabling internet-wide deployment of explicit congestion notification. In *Passive and Active Measurement Conference*, pages 193–205, New York, USA, 2015.