Observing Internet Path Transparency to Support Protocol Engineering

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

Mirja Kühlewind
ETH Zurich
mirjak@tik.ee.ethz.ch

ABSTRACT

Network operators increasingly rely on the use of in-network functionality provided by middleboxes to make their networks manageable and economically viable. These middleboxes make end-to-end paths through the Internet more opaque, by making assumptions about the traffic passing through them. This in turn leads to ossification of the Internet protocol stack: new features and new protocols are difficult to deploy because middleboxes don't understand them. The first step in fixing this situation is to gather data about the nature and distribution of middlebox impairments in the Internet. This data can be used not just to understand the situation, but also provides guidance to engineer new protocols to fall back and work around these impairments dynamically. While there has been much academic work in this area, these studies suffer either from needing to make general assumptions from a relatively small and potentially biased sample of paths, or an inability to make raw data available for verification and repetition of analyses due to privacy concerns.

In this position paper we propose an Internet path transparency observatory, to form the technical basis for widespread collaborative measurement of the transparency of paths through the Internet to various types of traffic. This observatory will allow the sharing of data at a level of aggregation and anonymization so as to minimize privacy concerns, using a common vocabulary of types of impairment to enable comparison among different measurement studies. We explore related work in the area, to

1. INTRODUCTION

The end-to-end nature of the Internet architecture has steadily eroded over the past decade and a half, through the increasing deployment of middleboxes in the network to provide in-network services. Each new box that makes restrictive assumptions about the protocols passing through it further restricts our ability to deploy new protocols, protocol options, and extensions. The evolution of the stack has become extremely difficult in the face of this ossification.

Middleboxes contribute to stack ossification through two basic mechanisms: The first is *essential manipulation* of packets. An essential manipulation is something the middlebox was explicitly deployed to do. The second, *accidental manipulation*, is either a side effect of an essential manipulation, an effect of an implementation error in a middlebox, or an effect of a configuration or deployment error in a middlebox. Accidental manipulations arise from a mismatch between the actual traffic on the network and the assumptions made by the designers of the middlebox about that traffic. These tend to persist in the network, given the long development and deployment cycles of networking equipment.

Many of these boxes are necessary, though. NATs reduce address allocation pressure and support transition from IPv4 to IPv6. Proxies and caches can have performance benefits. Middleboxes, in short, make the network manageable. Therefore our eventual goal for making it possible to deossify the protocol stack can be stated in two parts: (i) reduce the accidental manipulation of packets at middleboxes as close as possible to zero, while (ii) minimizing the essential manipulation of packets for each given middlebox function. To reach this goal, the first step is to detect middleboxes and measure the impairments that they entail on legal Internet traffic leading to ossification [9].

Many of the discussions about engineering a solution to this problem, though, take place on the background of not very much data. In the best survey on the topic in the literature in recent years, Honda et al [7] look at on the order of a hundred locations in the Internet. Studies such as Netalyzr provide detection of other types of path impairments (e.g. Weaver et al [11]) from a larger set of locations, but inavailability of the raw data in these cases limits the comparability and repeatability of the study. The Alexa top million website domains list provides a convenient list of targets (e.g. as used by the authors in [10] in measuring ECN along about two million paths), but paths to web hosting providers are not particularly diverse. To get an overview of the impairments that exist and the likelihood that Internet traffic might experience connectivity or performance issues due to them, it is necessary to get a common view of a much larger set of paths and impairments in the Internet.

In this work, we propose the basis of a technical solution to this problem. We define the characteristics of a common data model for storage and analysis of middlebox impairments, showing how such a data model could be built quickly upon existing implemented systems for network traffic analysis. This position paper is best seen as a report on work in progress; we intend this foundation to be used in large-scale measurement studies for future publication, and invite other researchers that perform middlebox measurement as well as industrial organizations that may have passive measurement data that can be used to derive information on middlebox impairments to collaborate in this project.

2. DEFINING AN OBSERVATORY

Our proposed path transparency observatory will combine a variety of measurements at large scale – unidirectional active measurements of public targets, bidirectional mesh measurements between measurement nodes, and logs of path impairments generated by applications, to be able to answer questions about why specific protocols and features fail on specific networks, as well as to answer general questions about the prevalence of certain types of impairments.

At a higher level, we can reduce these questions to:

1. What is the likelihood that it will work (i.e. that all the data the option needs to function will not be changed by the path, such as through option stripping)?

2. What is the likelihood that trying to use it will cause connectivity failure (by dropping packets using the protocol or protocol feature, or worse, as in the case of the old routers that ECN [8] would reboot)?

3. Is there a measurable performance penalty to the use of an option or protocol as opposed to some other option or protocol (e.g. through slow-pathing, different treatment at the queues, etc, etc, etc.)?

We note that question (1) requires information from both endpoints of a path, while questions (2) and (3) can also be answered by comparing two different tests taken from a single vantage point performing active measurements against remote endpoints. At first glance, this would appear to make efforts to answer questions (2) and (3) scale much better than (1): compare for example Honda et al [7] or TCP HICCUPS [5], which address questions of packet modification by looking at hundreds of paths, versus prior work on ECN [2, 10] looking at ECN-related connectivity issues on millions of paths. The latter work gives little insight into the details of packet modification, but scales better due to its use of public websites as opposed to network research testbeds for target selection.

Assumptions about the actions of remote endpoints – either that they will implement protocols as specified, or based on empirical evidence about non-standard implementations – can substitute for control of remote endpoints. For example, tracebox [6] uses packet fragments returned in ICMP Time Exceeded messages to substitute routers along the path for controlled endpoints to get information about IP header modifications along the path.

In order to provide the broadest possible basis for further research and protocol engineering research, any observatory must meet a few additional requirements as well: First, the representation of path impairments in the observatory must be independent of the implementation of the testing tools involved, in order to encourage broad participation. Second, the representation of paths must account for limitations in the precision of path: sometimes full traceroutes will be available, sometimes only addresses, prefixes, or Autonomous System (AS) numbers. Third, path impairments must be described in such a way that the tests are repeatable over different parts of the Internet and at different points in time, supporting independent verification, comparison of impairments over different access networks, and longitudinal studies.

2.1 Observatory Data Model

On the basis of these questions and these requirements, we propose a design centered around a data model based on *packet patterns*, which are essentially templates for sequences of packets sent and received. Data elements in the observatory are expressed as sequences of packets that are used as evidence for a given impairment (or lack thereof), along with data extracted from these templates.

A packet pattern p is associated with a path designator P, which contains some identifier for the initiator and target of a measurement, and may optionally contain an ordered list of identifiers of nodes and/or networks known or suspected to be on the path, whether observed by traceroute, BGP looking glasses, and/or taken from some other source. Given the known issues with extracting topology data from either data-plane [1] or control-plane observations, only the source and destination information are taken to be authoritative in further analysis of data associated with a given path designator. Identifiers may be in terms of network-layer addresses, prefixes, or AS numbers. Path designators may also be completely pseudonymous for privacy reasons, as well. A path pseudonym can be used to refer to a given path without locating it in the Internet topology; data sources which agree on a pseudonymization algorithm can compare data directly, while all users of the repository can use pseudonymized observations for aggregate impairment analysis.

A path transparency observation consists of a P, p, t tuple, where t is a temporal scope defining when a given observation was taken and therefore when the inferences from that observation are assumed to be valid. Time is expressed in arbitrary precision, in order to accommodate the different precisions available; it is assumed that observations will be taken from devices whose clocks are synchronized via Network Time Protocol (NTP), Global Positioning System (GPS), or other signals.

An observatory, then, is simply a database of these observations, which can accept tuples from a variety of source, and from which Internet-wide conclusions can be drawn, as long as these sources are using the same packet patterns to represent the same types of impairments.

Using packets as a least common denominator has a couple of important advantages. The definition of the packet patterns p can be easily matched against captured packet traces, allowing extraction of data both from controlled, dedicated active measurements as well as from passive measurements. Since many of the impairments we care about for the purposes of transport stack ossification occur in the network and transport layer headers, these traces need not contain payload. The definition of the packet patterns can also be easily used to generate test traffic for active probing along a path. This is the basis of the repeatability and comparability provided by the observatory.

We now turn to the design of a language for expressing packet patterns. We leave the complete specification of this language to future work, but explore the requirements and the related work as a basis for this future work.

2.2 Toward a Packet Pattern Language

First and foremost a packet pattern language must be able to express the common elements of a given path transparency test, while leaving out irrelevant details as well as source and destination information captured by the path designator. Second, since we want to use the observatory to check for future as well as present transport features and options, the language must allow the expression of arbitrary byte strings in packets (i.e., strings that do not appear in any presently deployed protocol).

These requirements would at first glance appear to point toward a language based on regular expressions, though these are not particularly suitable to parsing the types of encodings common in packet headers (type-length-value, ASN.1, and so on). The problem of extracting patterns of information from packets, however, is a well-studied one, so there are many places from which a packet pattern language can take inspiration, if not implementation.

Properties of single packets can be expressed as filtering expressions: the packets which match a filter for a property are said to have that property. The Berkeley Packet Filter language is the most widespread way to define filtering expressions over single packets. It includes primitives for a wide variety of protocols, and does provide support for testing against arbitrary patterns at arbitrary parts of the packet. It remains under active development in the Linux kernel [4].

Similarly, the *de facto* standard for exchanging information about patterns of packets seen in a single flow in the security monitoring community is the Snort rule language [1]. Snort adds the ability to split patterns across multiple packets, but as its detection engine makes assumptions about the underlying protocols, it is probably not suitable to expressions covering not-yet-invented protocols. A more promising approach from the security community is taken by Scapy [3]. Since it was made to craft packets for security testing, it can express patterns not foreseen in the specifications and implementations of the protocols under test. It is, however, oriented toward the generation of packets, and is bound to a single implementation environment.

At the other end of the spectrum would be to define a single execution environment and API, and to require packet patterns to be written as programs in a domain-specific language written to that API. Here pattern definitions can be included in the observatory by reference (e.g., to a specific commit in a specific Git repository). This would have the disadvantage of forcing an execution environment on collaborators, but would ensure comparability across multiple observations at the implementation level.

3. CONCLUSION AND OUTLOOK

Facing the increasing ossification of the Internet due to middleboxes that make restrictive assumptions about the traffic passing though them, we propose a path transparency observatory to collect, correlate, and analyze middlebox impairment measurements and thereby provide a common view on path transparency in the Internet. The next obvious step is to begin implementation of both the observatory and measurements to populate it, and the solicitation of participation by both the research and engineering communities in the use of the observatory.

The authors have chaired the "How Ossified is the Protocol Stack?" proposed research group within the Internet Research Task Force (IRTF) at the IETF meeting in Prague in July 2015. The aim of this research group is to bring together researchers and protocol stack and application engineers to share insights taken from both Internet measurement and experience with new protocols and protocol features, as well as tools and data that can be used to build a deeper understand-

[1] see `http://manual.snort.org/node27.html`

ing of the deployability of new protocols and protocol features in the Internet.

4. ACKNOWLEDGMENTS

This work is partially supported by the European Commission under grant agreement FP7-318627 mPlane; this support does not imply endorsement of the content.

5. REFERENCES

[1] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 153–158, New York, NY, USA, 2006. ACM.

[2] S. Bauer, R. Beverly, and A. Berger. Measuring the state of ECN readiness in servers, clients,and routers. In *Proc. Internet Measurement Conference (IMC)*, 2011.

[3] P. Biondi. Packet generation and network based attacks with Scapy. In *CanSecWest/core05*, May 2005.

[4] J. Corbet. Extending extended BPF. *Linux Weekly News*, July 2014.

[5] R. Craven, R. Beverly, and M. Allman. Middlebox-cooperative TCP for a non end-to-end Internet. In *Proceedings of ACM SIGCOMM 2014 Conference*, Chicago, IL, USA, August 2014.

[6] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet. Revealing Middlebox Interference with Tracebox. In *Proceedings of the 2013 Internet Measurement Conference*, IMC '13, pages 1–8, Barcelona, Spain, 2013.

[7] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it still possible to extend TCP? In *Proc. of IMC 2011*, IMC '11, pages 181–194, New York, NY, USA, 2011. ACM.

[8] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, IETF, Sept. 2001.

[9] B. Trammell and J. Hildebrand. Evolving Transport in the Internet. *IEEE Internet Computing*, September 2014.

[10] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger. Enabling internet-wide deployment of explicit congestion notification. In *Proc. Passive and Active Measurement (PAM)*, New York, March 2015.

[11] N. Weaver, C. Kreibich, M. Dam, and V. Paxson. Here be web proxies. In *Proc. Passive and Active Measurement (PAM)*, Los Angeles, March 2014.