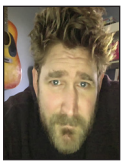
iVoyeur
Tcpdump at Scale

DAVE JOSEPHSEN



Dave Josephsen is a book author, code developer, and monitoring expert who works for Sparkpost. His continuing mission: to help engineers worldwide close the feedback loop.dave-usenix@skeptech.org

A little while ago, at my day job, we had a catastrophic DNS outage [1]. DNS (he explained, captain-obviously) is a rather essential bedfellow of SMTP. I'm not understating the relationship because I want to. Honestly, it's sort of impossible to overstate it, or even do justice to how intertwined the protocols really are.

For every SMTP conversation you initiate, you make at least twice the number of DNS queries (and usually, given MX round-robin load balancing, DKIM txt records, and failure-induced retransmission, **way** more). In fact, every time you push that send button to ACK a calendar reply or send a one-liner to thank someone, you're putting more DNS bytes on the wire than you are SMTP.

As an email service provider, sometimes it seems like our **REAL** job is to run DNS at scale. I know, bum-ba-bum *at scale*, those two cheap little words that make every story interesting. You process ACH payment transactions? Okay, that's cool I guess. Oh, you process ACH payment transactions at scale? Why didn't you say so? Any banal undertaking is keynote-worthy if you do it eleventy-billion times a day.

SaaS email service providers, as we are at my day job, carry out SMTP conversations *at scale* at the behest of our customers. All those password reset emails and coupon mailers add up evidently, and the rub, of course, is you can't speak SMTP *at scale* without first speaking DNS *at scale*. It's a very strange business model if you think about it. Step one, implement DynDNS. Step two, layer your actual product on top of it.

As a result, we have a somewhat complicated relationship with AWS in the context of DNS, as you can probably imagine. We often assist them in locating the real-world limitations of this or that service with our perfectly rational, real-world traffic patterns. They often struggle to define the word "abuse" in such a way that doesn't encompass our perfectly rational real-world traffic patterns. We are not unlike a consultant in these respects. An extremely diligent, successful, and annoying consultant.

Were you to ask AWS, I suspect they might tell you that the phrase "perfectly rational" as applied to real-world DNS traffic from a non-DNS provider is subject to interpretation. Of course, they're wrong in our case (with startling regularity), but one does have to respect how predictably on-message they are. For our part, we continue to assure them that gigabyte-sized bursts atop our already absurd cardinality of DNS traffic is a metric of nothing but success for us both, but convincing them of that fact has admittedly been an uphill walk.

We are, however, blessed with a large and seasoned reliability engineering team at SparkPost, so when this latest DNS snafu occurred, there were plenty of eyes and hands at work to mitigate the problem with various temporary nefarious kludges (you know how it is), all of which left me free to poke around what the Internet kids refer to as WTAF.

Now, you're probably thinking to yourself, *ah-hah, this is the point in the story where the auspicious (and ruggedly handsome) author of* ;login: *magazine's column on systems monitoring turns to the highly advanced monitoring platform du jour and, via some arcane means involving machine learning, expertly extracts from it the root cause of the problem*, but alas, no.

I pretty much just launched tcpdump.

Yup, good-ole tcpdump, from the '90s. I wanted to see what was going on on the wire, and, predictably, what was going on was metric tons of DNS transmission failure, but I did learn a few important things:

- The outage was affecting all types of traffic (not just DNS).
- The overwhelming amount of traffic on the wire was outbound DNS traffic to the Internet root NS servers.
- Very little of our outbound DNS traffic was being returned. In fact, we were getting one response for about every 17 queries, but we were getting *a few* responses.

Having been throttled in pretty much every way possible, I thought this looked very much like some sort of throttling, so I wrote this small shell script to broaden my sample set and verify my initial observations.

```
#!/bin/sh
L='<local IP of the machine>'
P=$(tcpdump -c 5000 -vvv -s 0 -l -n)
echo
echo -n "outbound:"
echo "${P}" | grep "${L}.>" | wc -l
echo -n "inbound:"
echo "${P}" | grep "> ${L}" | wc -l
echo -n "top-ten dst"
echo "${P}" | grep '> ..domain' | cut -d\ -f7 | sort | uniq -c |
sort -n | tail -n10 | sort -nr
```

It uses `tcpdump` to sample 5000 packets, measuring the ratio of inbound to outbound packets while dumping a list of the top-10 destination IP addresses. In the grand old days of *actual* computers in *actual* nearby closets, this would have been unnecessary, but I'm sure you've noticed, as I have, just how much of contemporary SRE work consists of convincing upstream service providers not only that a problem exists but that it is, in fact, *their* problem. In this regard, my low-tech shell script was highly successful, and AWS ran off with the baton, only to return some time later to inform us that we'd discovered a new limit in their infrastructure, namely conntrack memory allocations in the VGWs (virtual gateways).

I'll admit, it felt just a tiny bit troglodyte to have, in that time of crisis, turned to tcpdump, but I happen to know that I'm in fine company, because not just one but TWO talks at Monitorama this year had favorable things to say about our loyal old packet-inspecting friend.

Julia Evans' talk [2], entitled "Linux Debugging Tools You'll Love," seemed custom-curated to preach to my personal choir, but really it's Douglas Creager's presentation [3] I'd like to talk to you about.

How shall I phrase this diplomatically? Google builds zany stuff. That's the word. Zany. I feel like that's an arguably uncontroversial, if not objective, observation that we can both agree upon in 2017, and I'm not judging. I sincerely feel like there's nothing *wrong* with their particular brand of zaniness, I mean...it's just what they *do*, and they're very good at it, and it seems to make them happy.

Every time I see a speaker from Google giving a talk on a subject I've never heard of, I scooch down in my seat a little bit to get comfortable, close my laptop lid, and expectantly fold my hands together in preparation for whatever zany systems engineering antics they've gotten up to this time. I know I'm not alone in this. They made a distributed file system over HTTP with 64 MB chunks? Huh. They've overloaded cgroups into a deployment strategy? Cool. They built a compressionless metrics database because they have an infinitely large MapReduce cluster lying around? Okay.

Anyway, my point is, I admit to being a little surprised by Doug's Monitorama talk, wherein he outlines one methodology employed by the "Internetto" team at Google to monitor the edge of Google's network. In that capacity, they've employed visualizations and other analysis tools on good-ole humble libpcap packet captures.

I won't spoil the talk for you, but Doug is a proponent of *continuously* capturing and evaluating TCP headers as a means of monitoring application performance health. In other words, Doug thinks we should all be running tcpdump on every public-facing machine, all the live-long day (though, I suspect he might not phrase it that way). To that end, Google is evidently using service-based libpcap to capture every header of every packet in every end-user interaction on the wire.

Rather than taking a flow-based approach where the sum of all traffic is sampled at a switch, Google runs local pcaps, and RPC ships the data upstream to be centrally processed (no doubt via MapReduce). They extract throughput and latency numbers and do some simple arithmetic, eventually reducing each connection to a JSON blob describing that interaction, complete with Boolean flags like *BufferBloat:true* to indicate common performance problems detected on the wire.

It's an excellent talk which, obviously, should have been called "Tcpdump at Scale," but otherwise was perfect in every way, and if you only have time to see one talk from Monitorama this year, it would be my pick. I'm obviously biased, but I love the approach and sincerely hope it catches on among all sorts of service providers, CDNs, and, especially, IaaS shops that don't have any idea what is transpiring on their networks. Just imagine your cloud provider notifying *you* of wire latency for once (or at least believing you when you report it to them?).

Take it easy.

References

[1] C. McFadden, "How We Tracked Down Unusual DNS Failures in AWS": https://www.sparkpost.com/blog/undocumented-limit-dns-aws/.

[2] J. Evans, "Linux Debugging Tools You'll Love," Monitorama PDX 2017: https://vimeo.com/221062212.

[3] D. Creager, "Packet Captures Are Useful," Monitorama PDX 2017: https://vimeo.com/221056132.