OpenINTEL

IPv4 reverse measurements

Mattijs Jonker

Introduction

- Over **five years** ago, we started with **an idea:**

  “Can we measure (large parts) of the global DNS on a daily basis?”

- This idea led to the OpenINTEL project

  (Raffaele presented the gist of it earlier today)
- *IN-ADDR.ARPA.* is part of the global DNS, amirite?
- In this talk, I will discuss:
  - The (very recent) addition of reverse v4 measurements
  - Why, how

Reverse DNS 101

- Reverse DNS maps IP addresses to names

  … using a reversed IP address as name

  e.g., 192.168.1.15 becomes 15.1.168.192.in-addr.arpa.
- Name space managed by IANA and the RIRs
- Delegated to address space holders when the address space is assigned

Why measure?

- Check consistency with forward DNS -- especially for e-mail reverse and forward DNS mapping must be consistent (part of MTA authentication)
- Provides visibility into cloud infrastructures and network infrastructural elements, e.g.:
  - Names reflecting in which data centres clouds VPSes are hosted
  - Names of router interfaces [Chabarek13, Huffaker14]
- Gain insight in address space usage

How we perform our measurements

- The measurement process involves two stages
  1. Active measurement
  2. Streaming and persisting data

Stage I: main measurement

- We want to measure efficiently -- first find parts of the name space that are actually delegated
- Intuition: perform SOA and NS queries for /8, /16 and /24 levels (in IPv4) to find delegation points
- Yields one of the following:
  - Delegation point
  - Empty non-terminal response (RFC 8020) -- indicating no delegation exists, but names exist below
  - NXDOMAIN -- there are no names below

Stage I: main measurement

- Adapted existing OpenINTEL measurement code
- Goal: one measurement every 24h
- Challenge: do not overload authoritative servers with queries
- Solution:
  - Randomize measurement
  - Monitor traffic for the first few measurement runs

Stage I: main measurement

- We use a similar trick to ZMap, that is: leverage properties of a group of prime order
- Need a permutation over 256 and 65536 possibilities for our implementation (to randomise individual labels in an IPv4 reverse name and to randomise /16 blocks sent to worker nodes respectively)

Stage I: main measurement

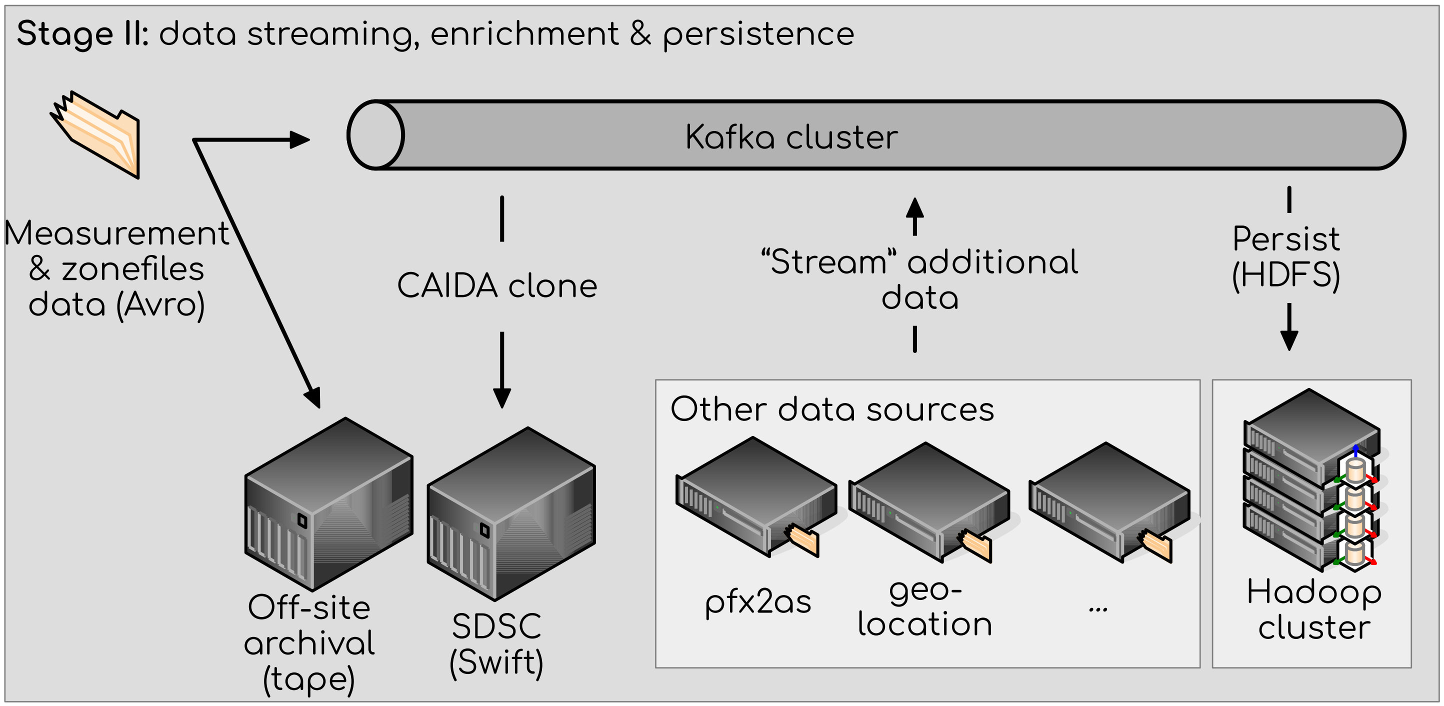
- We adapted Duane Wessels' **dnstop** to track query loads and report average and maximum queries per second
- Result: average upstream loads very reasonable (maxing out around the 100 queries/second on average)
- Modified code: *https://github.com/rijswijk/dnstop*

Queries: 5705 new, 18354801 total

Destinations	Count	%	cum%	Delta	AvgDelta	MaxDelta
144.160.128.177	273801	1.5	1.5	48	99.6	157
144.160.229.221	272482	1.5	3.0	34	99.1	154
218.176.253.72	214129	1.2	4.1	61	77.9	111
218.176.253.104	213269	1.2	5.3	71	77.6	112
211.216.50.180	145442	0.8	6.1	52	52.9	76
211.216.50.170	144780	0.8	6.9	52	52.7	75
200.33.150.193	144355	0.8	7.7	75	52.5	112
202.98.0.73	126125	0.7	8.4	58	45.9	161
201.10.204.43	112960	0.6	9.0	41	42.4	75
201.10.132.1	112906	0.6	9.6	24	42.4	79

Stage II: storage and persistence

- Data is persisted in HDFS
  - allowing batch-based, analyses
- We stream the data to a Kafka cluster
  - enabling stream-based analysis
- Will clone data to CAIDA (WIP)



What do we have, in simple numbers

- Started measuring February 17, 2020
- This adds approx $1.1 \cdot 10^9$ data points each day (SOA, NS, PTR)
- 45% increase w.r.t. what we were already getting daily

Which data do we share

- This type of data: not yet
- No real obstacles
- Should probably think of *how?* and not *whom with?*

Case study: forward-confirmed rDNS

- Checked, for our “forward” active DNS data, which IP addresses are forward confirmed
- 1.1M / 6.08M [18%] are

Case study: multi PTR

dptr_name_count	dquery_name_count
1	1149126160
0	15137003
2	4189153
6	208533
3	185014
4	47997
5	17503
17	9146
7	6818
8	5302
9	3669
10	3049
11	2576
12	2027
13	1586
14	1466
16	1011
15	988
20	850
18	746
22	653
21	609
19	585
27	580
23	548

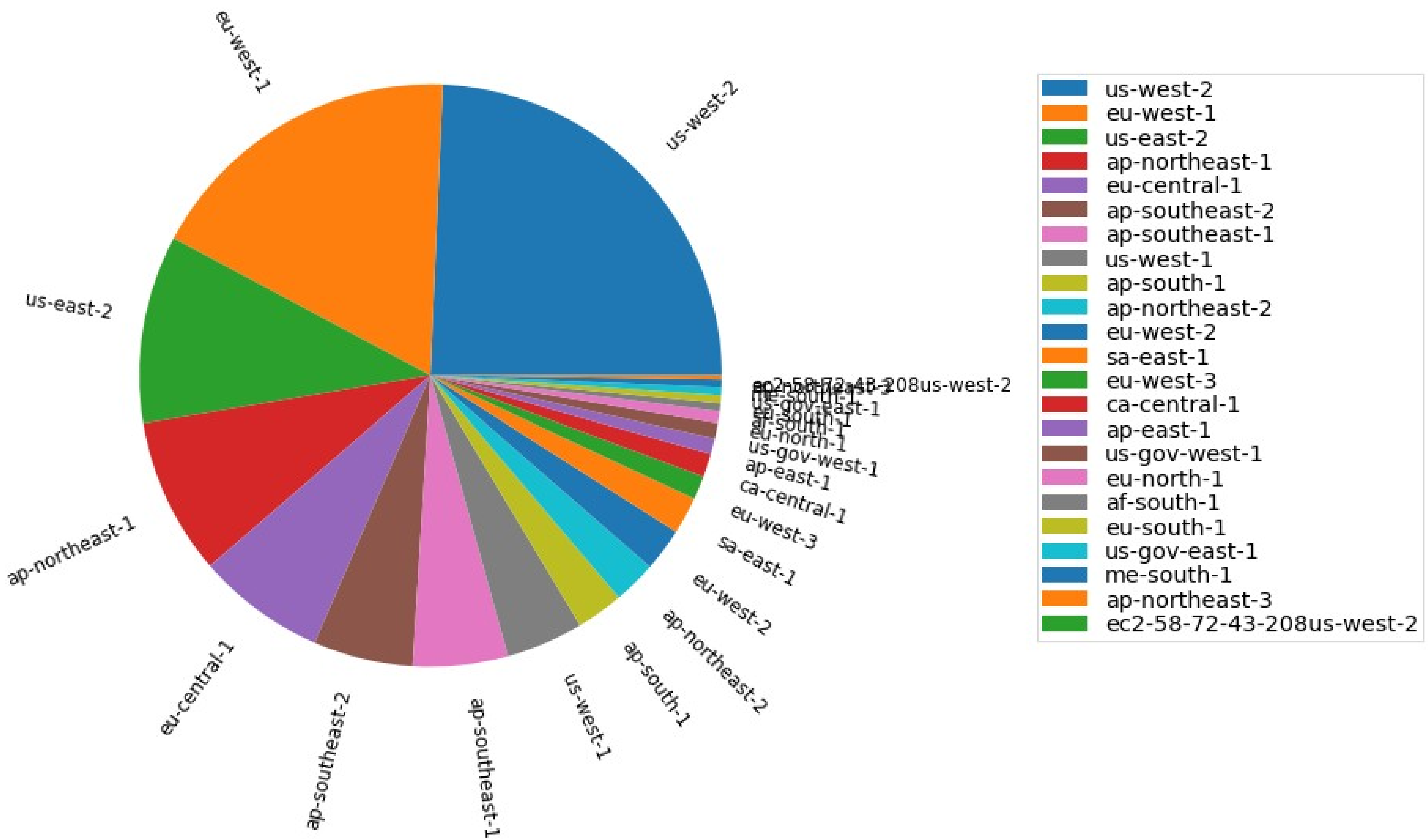
only showing top 25 rows

Case study: multi PTR

query_name	dptr_name_count
164.27.211.124.in...	2326
71.184.197.146.in...	2291
62.128.233.15.in-...	2250
119.148.241.15.in...	2250
95.20.241.15.in-a...	2250
63.160.233.15.in-...	2250
47.12.42.61.in-ad...	2013

dig +tcp -t ptr 71.184.197.146.in-addr.arpa @208.67.222.222

Cast study: Amazon EC2



Future work

- Verify consistency against existing work by CAIDA (Young Hyun)
- Check missing empty non-terminals on name servers that do not conform to RFC 8020
- Make data public: comments, thoughts?

Questions ?