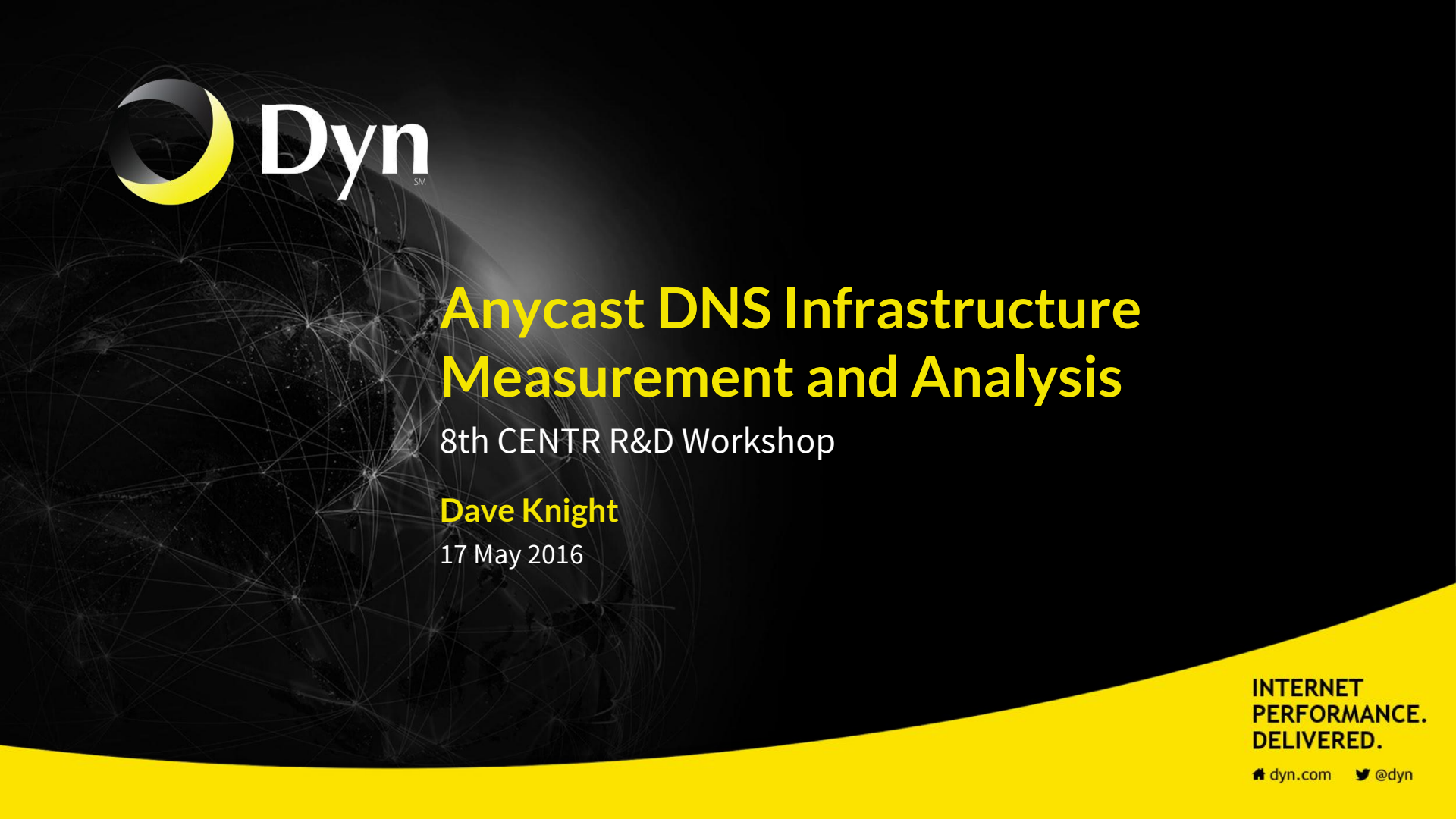


Anycast DNS Infrastructure Measurement and Analysis

8th CENTR R&D Workshop

Dave Knight

17 May 2016

INTERNET
PERFORMANCE.
DELIVERED.

dyn.com @dyn

Introduction

Dave Knight is an engineer at Dyn, working in the Infrastructure group. Engaged day-to-day with advancing the platform and the services which run atop it.

Dyn is a cloud-based Internet Performance Management (IPM) company that provides unrivaled visibility and control into cloud and public Internet resources. Dyn's platform monitors, controls and optimizes applications and infrastructure through Data, Analytics, and Traffic Steering, ensuring traffic gets delivered faster, safer, and more reliably than ever.

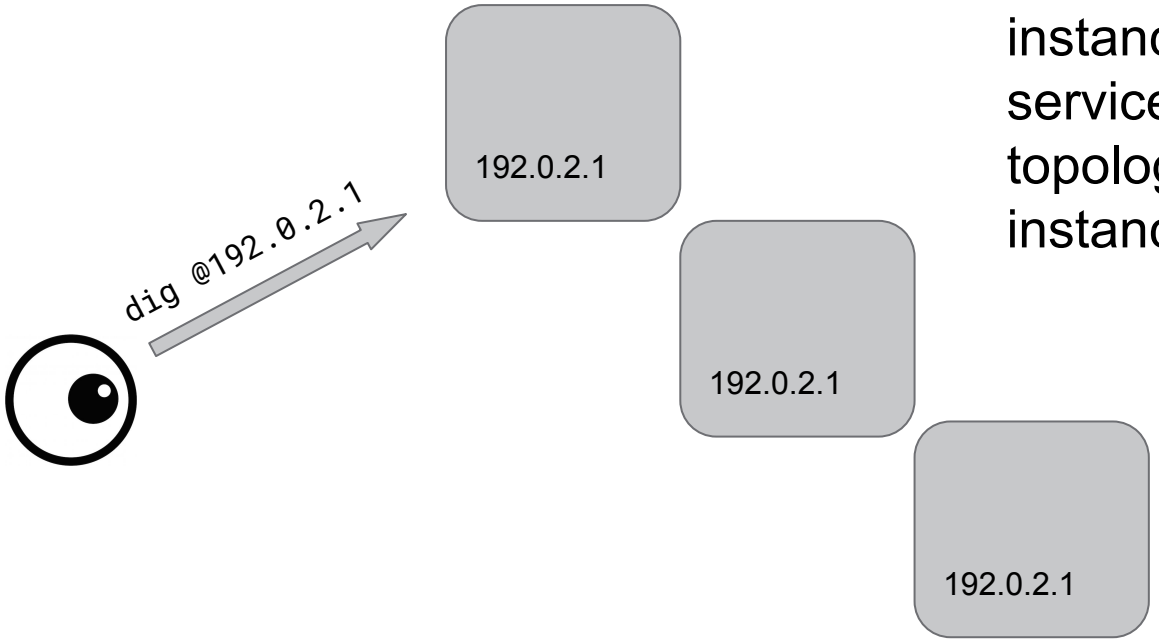
This talk concerns a technique to measure single trip times for DNS query responses from anycast service instances.

Problem statement

Anycast service distribution on the Internet presents challenges for monitoring.

It’s hard to mimic the eyeball experience, so compromises are employed.

Problem statement (2)

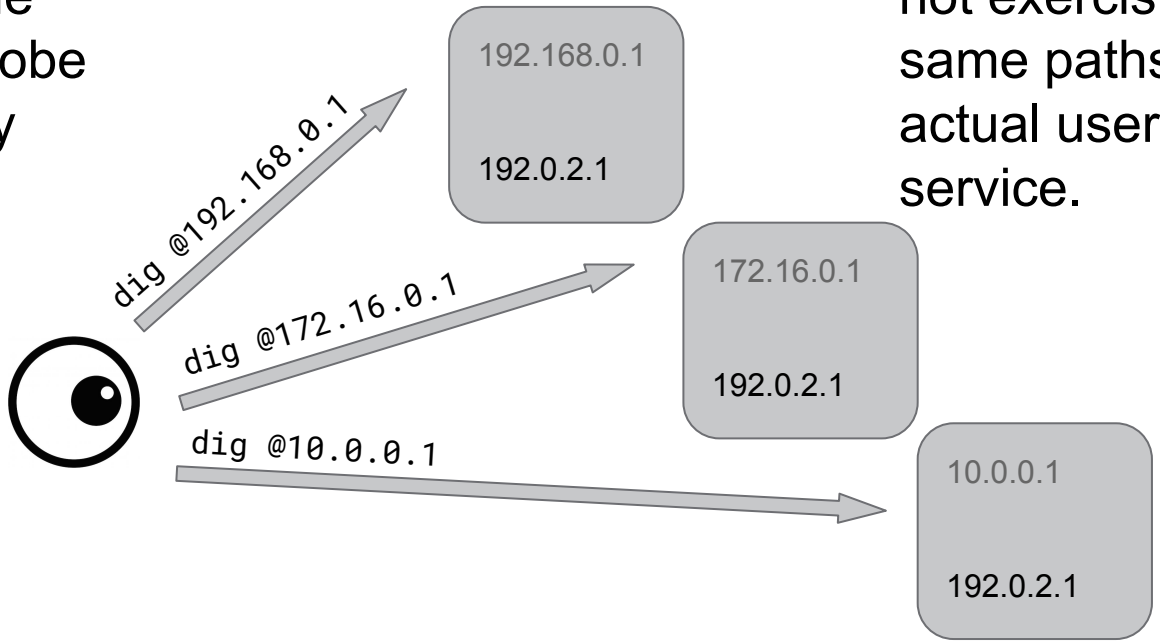


A single monitoring node cannot directly probe all instances of an anycast service, as only the topologically least distant instance is visible.

Dyn

Monitoring compromise

A single monitoring node may directly probe all instances by their **management address**...

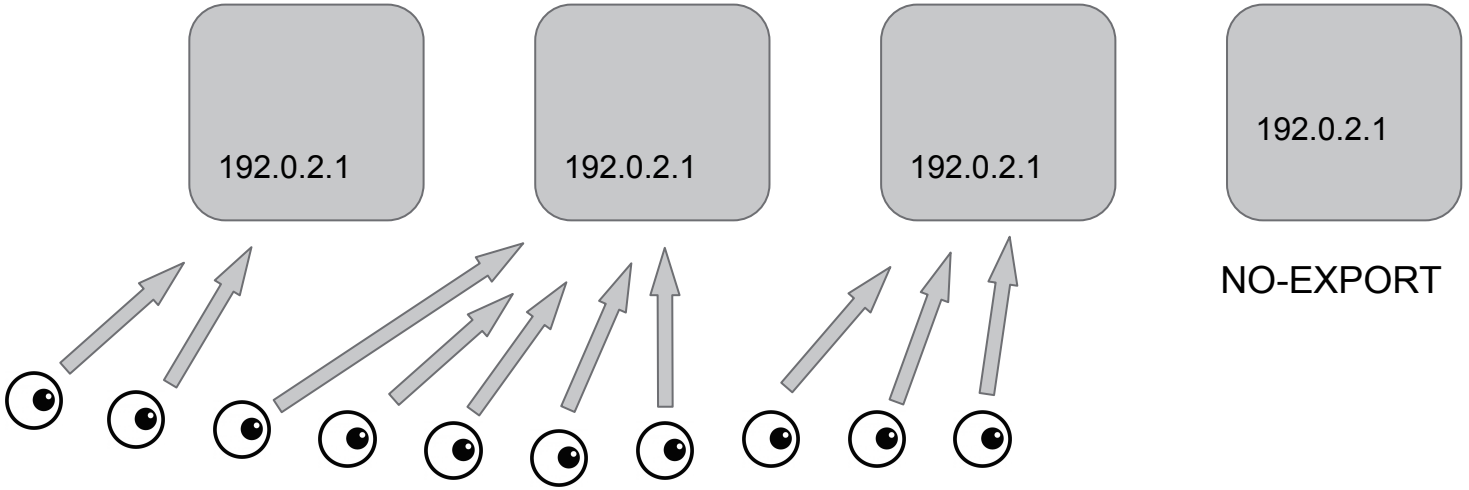


...but then it's likely not exercising the same paths as actual users of the service.

Dyn

Monitoring compromise (2)

Many monitors (RUM, RIPE Atlas, etc), well distributed in the topology may succeed in probing all service instances, but nondeterministically.



An instance of an anycast service with constrained route propagation may remain invisible to all but the most widely distributed probes.

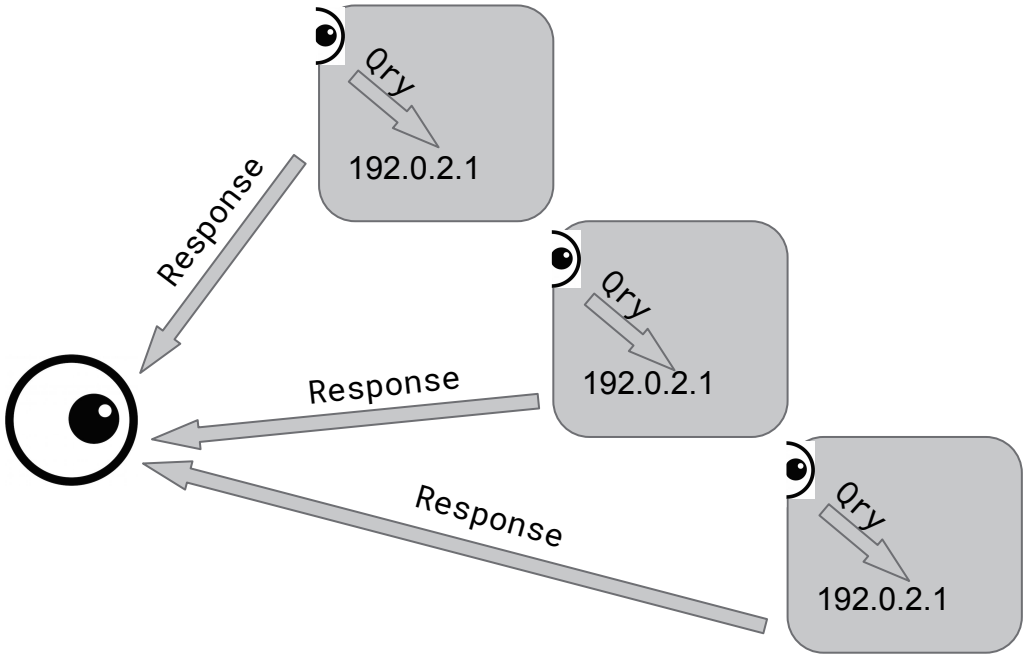
Dyn

Another compromise!

If we **generate a query local to the anycast service** instance, we can probe it directly.

If we **spoof the source address** of that query we can direct the response to our single monitoring node.

We can probe all instances of anycast service deterministically and **gather responses at one node**.



Dyn

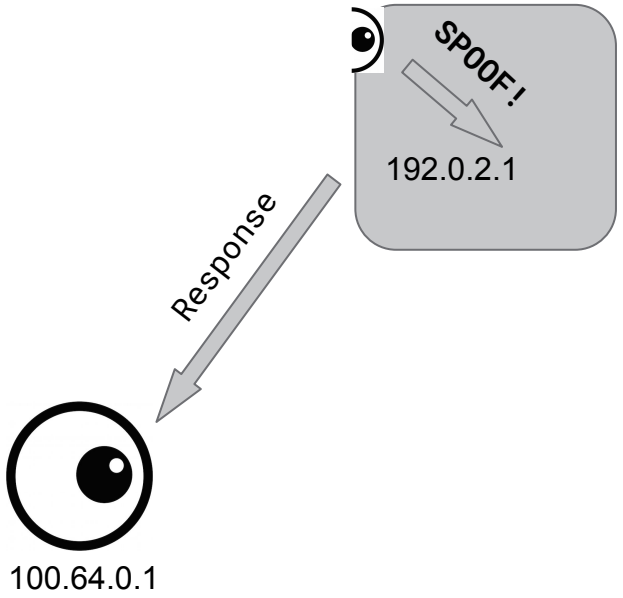
Some words on spoofing

This sounds more exciting than it actually is.

Spoofing takes place inside the server and results in a completely unsurprising packet traversing the Internet:

```
DNS Response: 192.0.2.1:53 => 100.64.0.1:54321 †
```

No violation of the provisions of BCP38, or MANRS, etc is being perpetrated here.



† Of course a packet with source and destination in ranges not intended to be publicly routed would in fact be surprising on the Internet

Dyn

Spoofing a query

```
#!/usr/bin/perl                 # Net::RawIP is easy to use

use JSON::XS;                   # Our config is JSON flavoured
use Net::DNS;                   # Need to construct a query
use Net::RawIP;                 # Need to construct a packet
use Time::HiRes qw(time);       # Need to do time in ms
use POSIX qw(strftime);
```

Spoofing a query (2)

DNS Message

```
1463295169321.dyndns.com IN SOA ? +NSID
```

UDP

```
dst port: 53
src port: 4653
data: DNS Message
```

IP

```
dst addr: 192.0.2.1
src addr: 100.64.0.1
ttl: 1
```

Encode current time when we generate the query.

Collector listens on port 4653

Collector listens at 100.64.0.1

Guard against locally unanswerable queries confusingly going elsewhere with IP TTL=1

Collecting responses

```
#!/usr/bin/ruby          # What? I like ruby...

require "socket"         # Listen for responses on a socket
require "date"           # Get time in ms
require "ipaddr"         # Check ip address validity
require "collectd"       # Send metrics to collectd
require "syslogger"      # Send diagnostics to syslog
require "dnsruby"        # Inspect DNS messages
```

Deconstructing a response

IP

```
dst addr: 100.64.0.1
src addr: 192.0.2.1
```

UDP

```
dst port: 4653
src port: 53
```

DNS Message

```
;; OPT PSEUDOSECTION:
; NSID: hivecast-11-usiad.as15135.net
;; QUESTION SECTION:
;1463406752123.dyndns.com. IN  SOA
;; AUTHORITY SECTION:
dyndns.com.  0  IN  SOA  ns0.dynamicnetworkservices.net.
hostmaster.dyndns.com. 2016051200 10800 1800 604800 1800
```

Analysing the response

The source address is the anycast nameserver we are testing

`src addr: 192.0.2.1`

NSID names the anycast instance which sent the response

`; NSID: hivecast-11-usiad.as15135.net`

QNAME contains the time when the query was generated

`;1463406752123.dyndns.com. IN SOA`

Authority Section contains the SOA SERIAL of the tested zone

`dyndns.com. .. IN SOA .. .. 2016051200 .. .. .. ..`

What use is this?

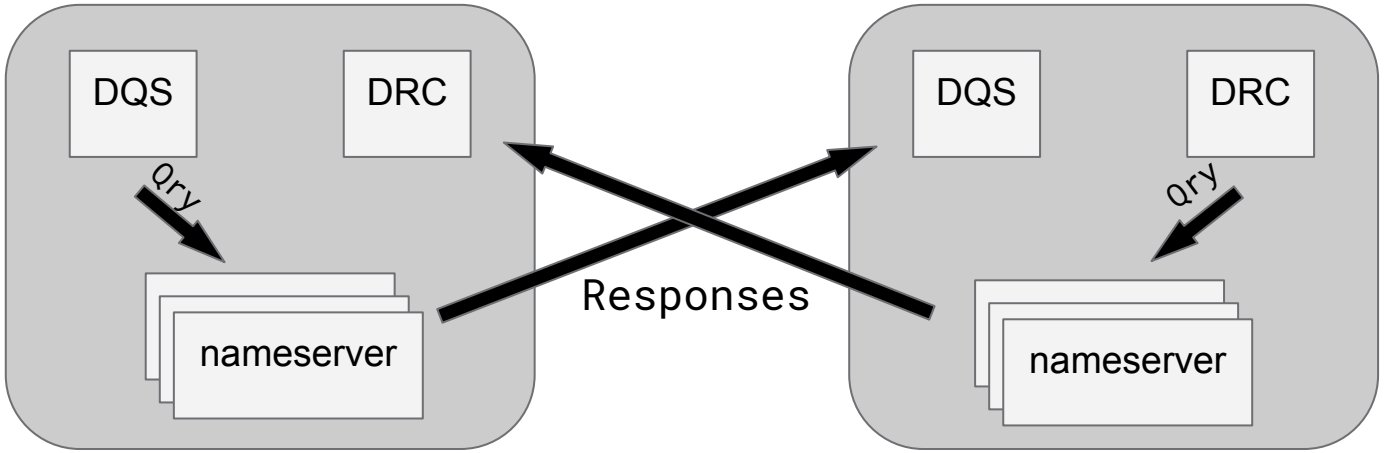
We have a heartbeat!

We can watch for changes in the SOA serial.

Subtract the query generation time from the current time and we get the **single trip time** for the response to get from the anycast instance to the monitor node.

This assumes excellent clock synchronisation. This can otherwise still be useful in detecting aberrant behaviour if the clocks are at least consistently dyssynchronous.

Scaling up



DNS
Query
Spoofer

DNS
Response
Collector

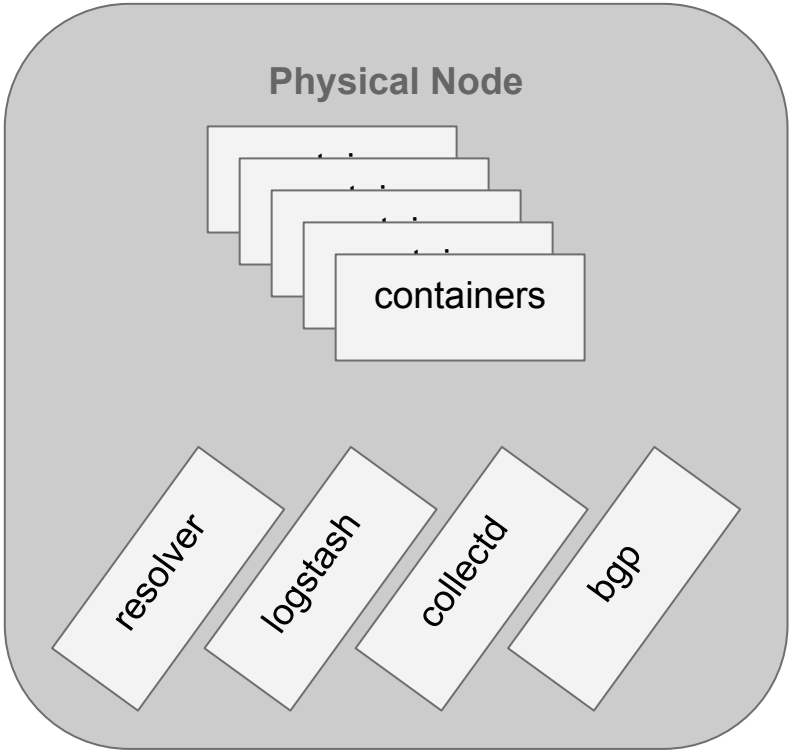
Probe all of the nameservers on a node

Send responses to collectors running on other nodes.

Build a full mesh of single trip latencies.

Dyn

Dyn's edge platform



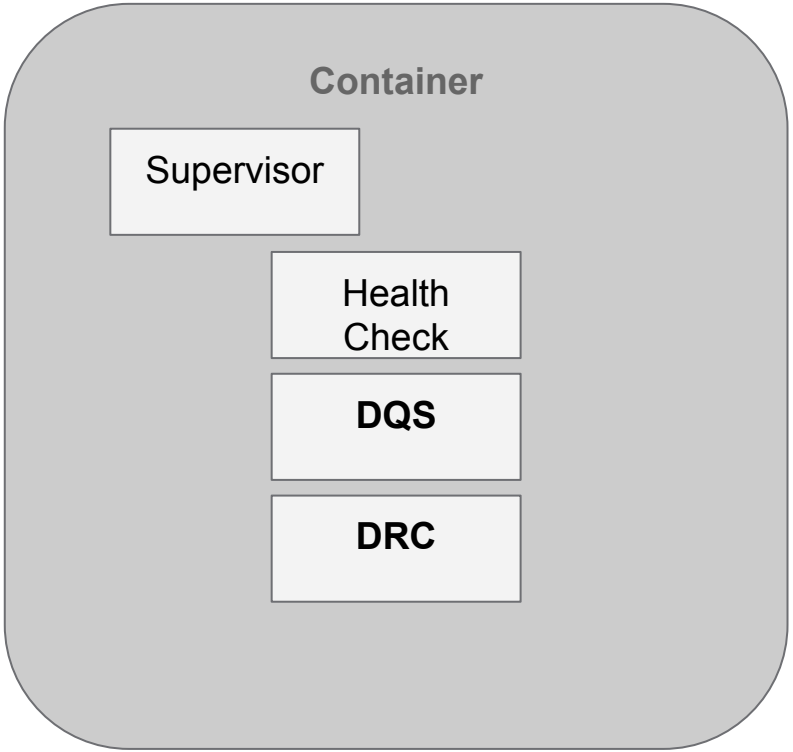
Many physical nodes around the world.

Workloads run in Docker containers.

Service workloads can advertise their service prefixes in BGP.

The platform provides various services, e.g. logging and metrics to the contained applications.

Dyn's edge platform (2)



Service and its dependencies in the container gives us a single unit of deployment.

DNS Query Spoofer and DNS Response Collector are packaged in one container.

This container is started on many nodes with the same configuration.

Configuration

Some JSON describes the targets and collectors.

DQS send queries to each of the targets on behalf of (with the spoofed source address of) each of the collectors.

Now we have a full mesh of measurements.

```
"targets": [
    {
        "name": "dyndns.com",
        "host": "108.59.165.1",
        "port": 53
    }
],
"collectors": [
    {
        "name": "hivecast-1-defra",
        "host": "80.231.25.21",
        "port": 4653
    },
    {
        "name": "hivecast-3-gblon",
        "host": "80.231.219.21",
        "port": 4653
    }
]
```

Dyn

Graphs!

Metrics sent to Collectd are viewable in a Grafana dashboard with templated queries

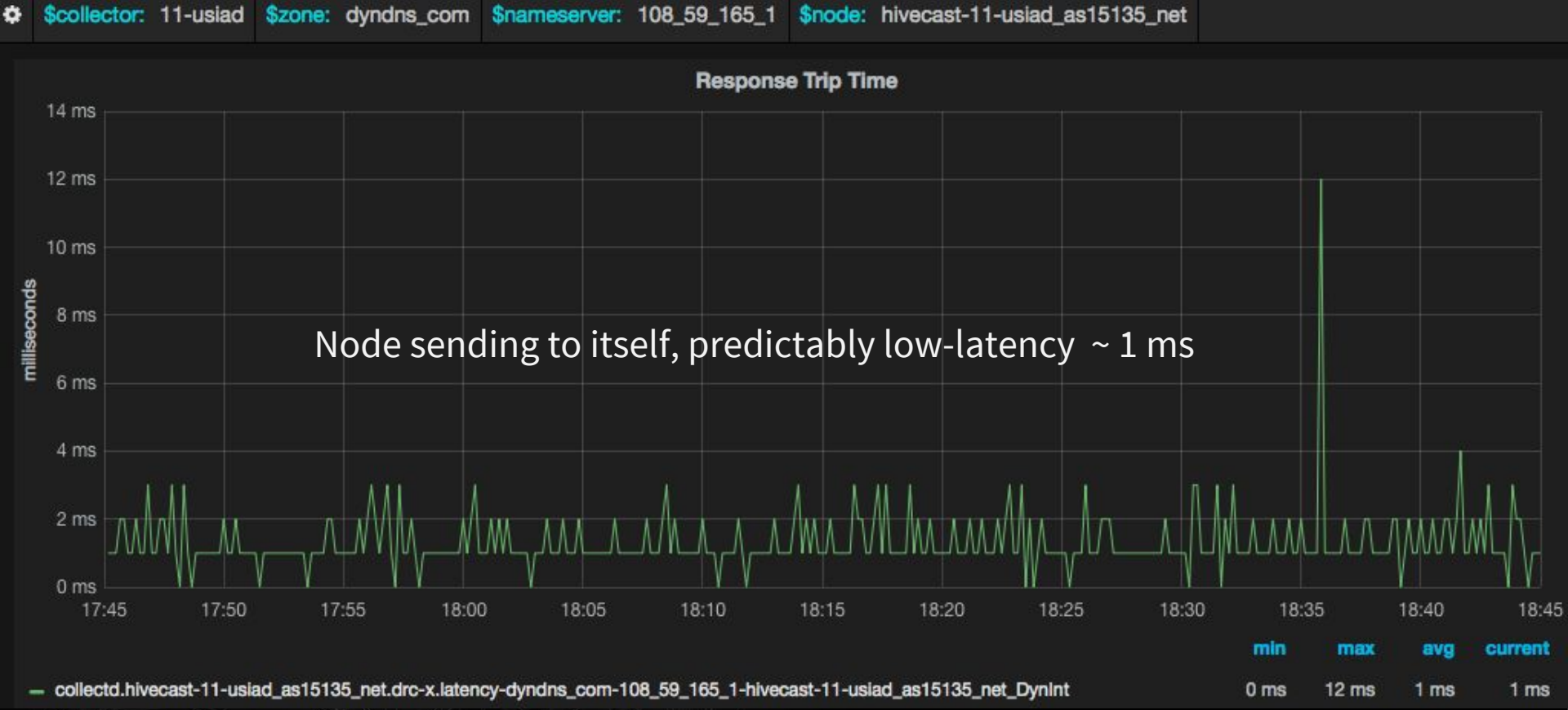
[**collector**].drc-x.latency-[**zone**]-[**nameserver**]-[**node**]-[container]
= single trip time in milliseconds

Dyn

Nice and unthreatening

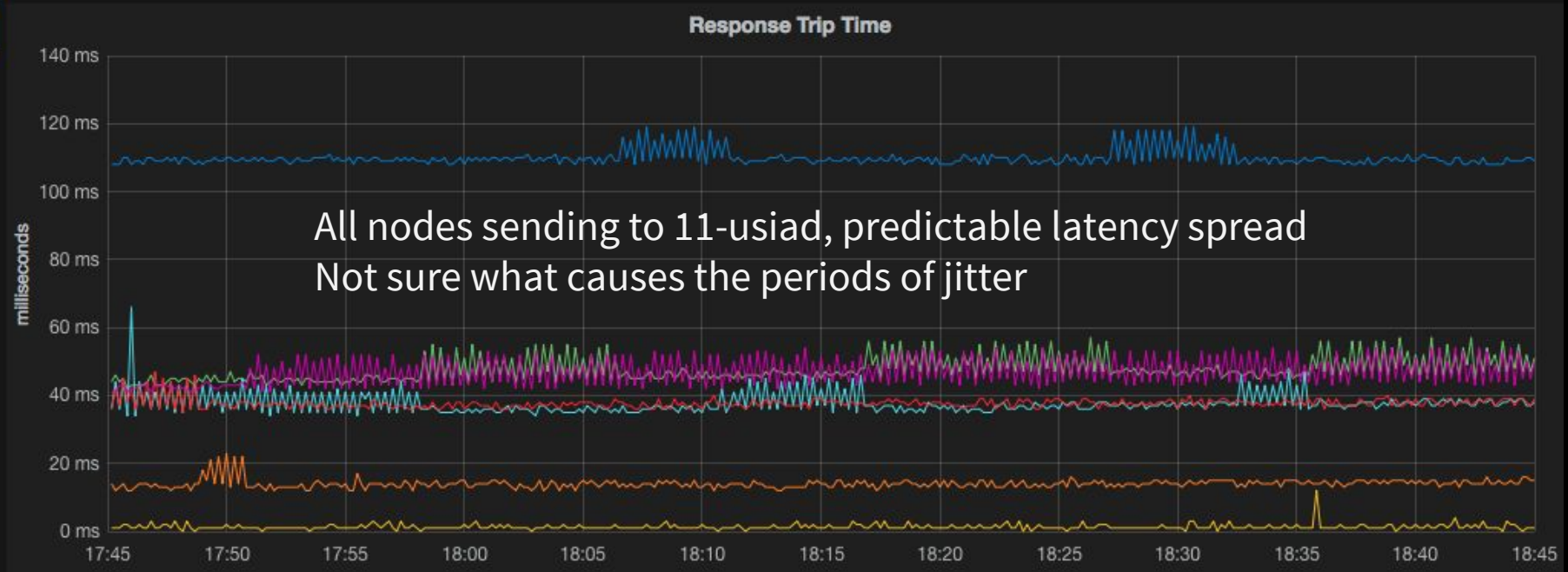
Cherry picked an hour when nothing odd is happening
Let’s look at these relationships

- hivecast-11-usiad ⇒ hivecast-11-usiad
- All probes ⇒ hivecast-11-usiad
- hivecast-11-usiad ⇒ All collectors
- All probes ⇒ All collectors


$collector: 11-usiad
$zone: dyndns_com
$nameserver: 108_59_165_1
$node: hivecast-11-usiad_as15135_net
Response Trip Time
14 ms
12 ms
10 ms
8 ms
6 ms
4 ms
2 ms
0 ms
milliseconds
Node sending to itself, predictably low-latency ~ 1 ms
17:45
17:50
17:55
18:00
18:05
18:10
18:15
18:20
18:25
18:30
18:35
18:40
18:45
min
max
avg
current
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt
0 ms
12 ms
1 ms
1 ms

Dyn

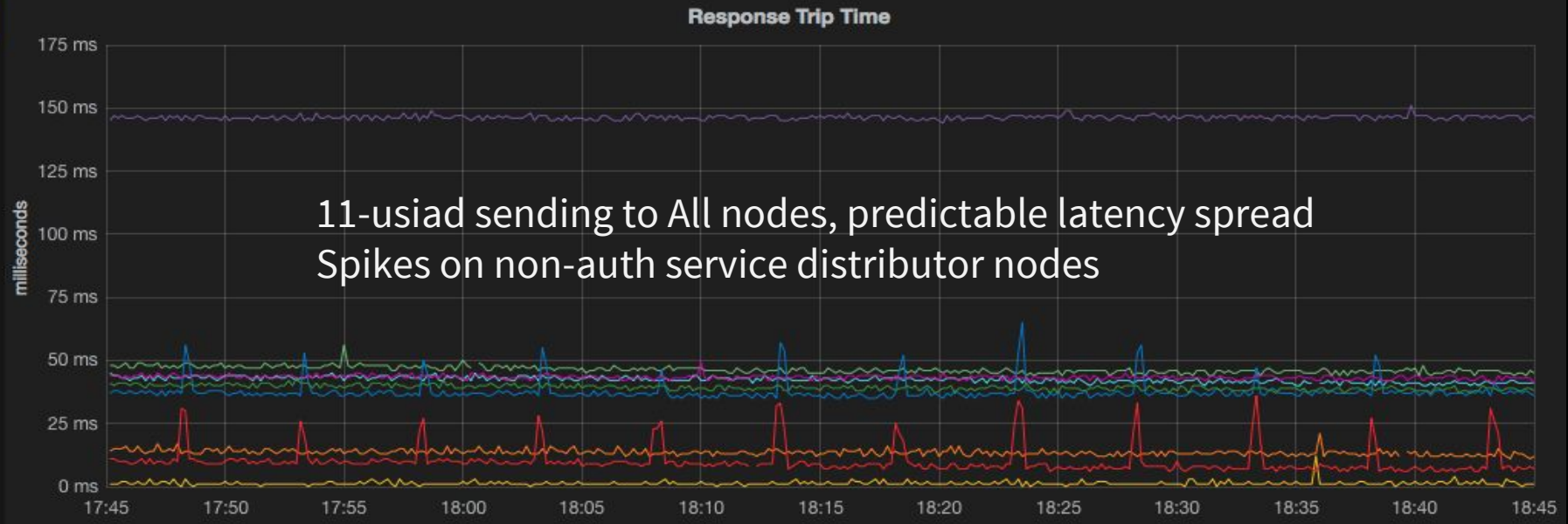
$collector: 11-usiad | $zone: dyndns_com | $nameserver: 108_59_165_1 | $node: All



	min	max	avg	current
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-1-defra_as15135_net_DynInt	42 ms	57 ms	48 ms	51 ms
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt	0 ms	12 ms	1 ms	1 ms
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-13-uslax_as15135_net_DynInt	34 ms	66 ms	38 ms	38 ms
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-15-usmia_as15135_net_DynInt	12 ms	23 ms	14 ms	15 ms
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-3-gblon_as15135_net_DynInt	35 ms	47 ms	38 ms	39 ms
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-5-hkhkg_as15135_net_DynInt	108 ms	119 ms	110 ms	109 ms
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-7-nlams_as15135_net_DynInt	41 ms	54 ms	47 ms	50 ms

Dyn

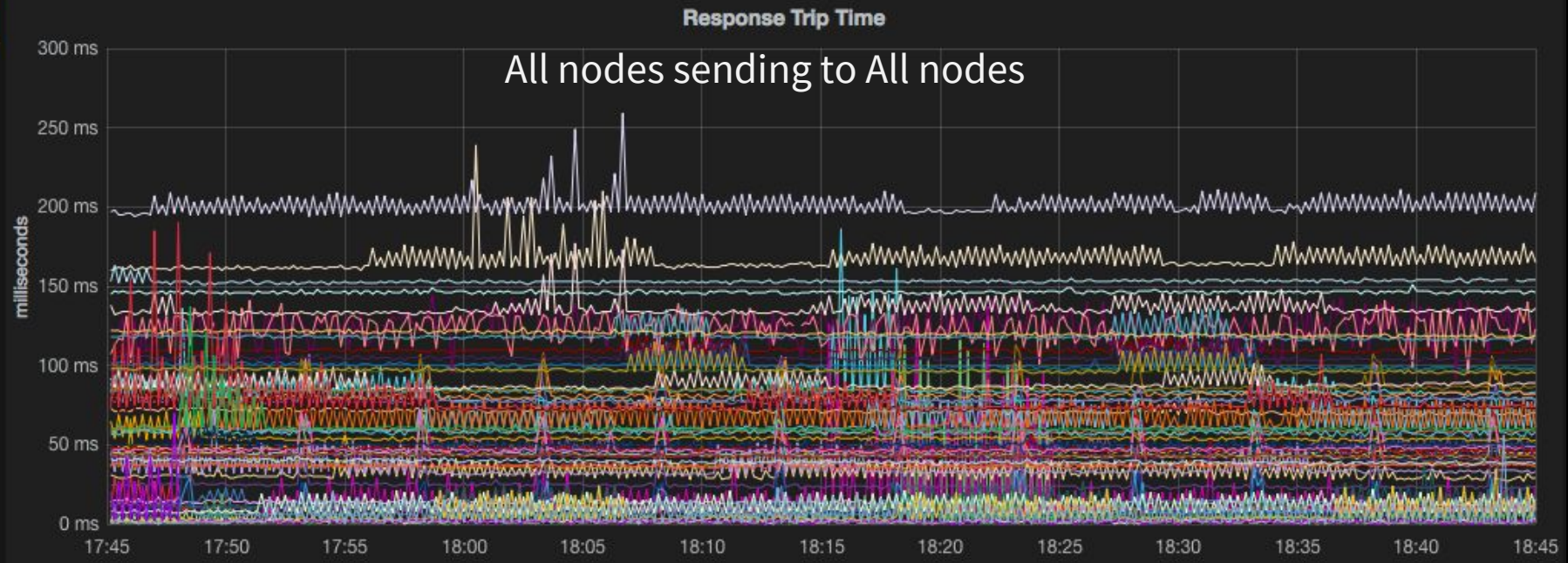
$collector: All | $zone: dyndns_com | $nameserver: 108_59_165_1 | $node: hivecast-11-usiad_as15135_net



	min	max	avg	current
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt		56 ms	46 ms	45 ms
collectd.hivecast-11-usiad_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt	0 ms	12 ms	1 ms	1 ms
collectd.hivecast-13-uslax_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt		45 ms	42 ms	41 ms
collectd.hivecast-15-usmia_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt		21 ms	13 ms	12 ms
collectd.hivecast-17-usnbn_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt		36 ms	10 ms	7 ms
collectd.hivecast-19-ussnn_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt	35 ms	65 ms	38 ms	36 ms
collectd.hivecast-3-gblon_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt	41 ms	49 ms	43 ms	41 ms
collectd.hivecast-5-hkhkg_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt	144 ms	151 ms	146 ms	146 ms
collectd.hivecast-7-nlams_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt	37 ms	43 ms	39 ms	38 ms

Dyn

$collector: All | $zone: dyndns_com | $nameserver: 108_59_165_1 | $node: All



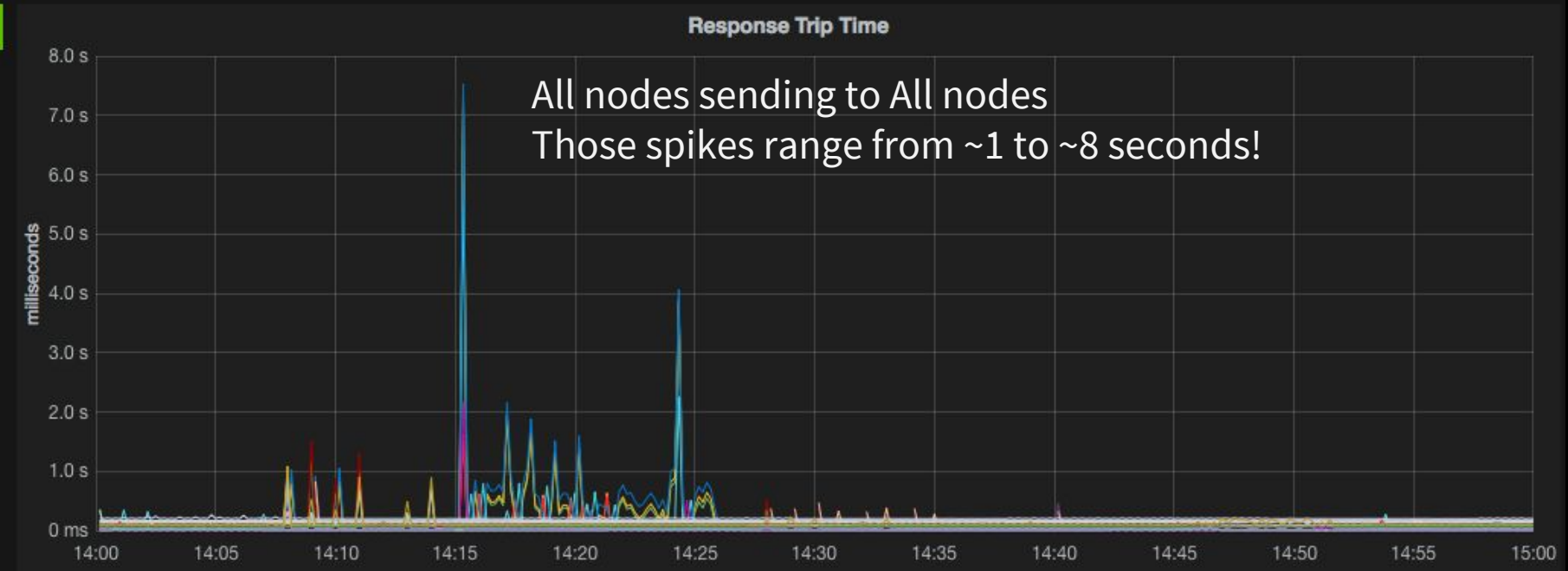
	min	max	avg	current
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-1-defra_as15135_net_DynInt	0 ms	116 ms	7 ms	1 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt		56 ms	46 ms	45 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-13-usiax_as15135_net_DynInt	76 ms	186 ms	83 ms	78 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-15-usmia_as15135_net_DynInt	56 ms	91 ms	59 ms	59 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-3-gblon_as15135_net_DynInt	7 ms	31 ms	10 ms	8 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-5-hkhkg_as15135_net_DynInt		119 ms	101 ms	99 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-7-nlams_as15135_net_DynInt	5 ms	137 ms	18 ms	24 ms

Dyn

Weird…

Now let’s look at an hour with curiously massive spikes

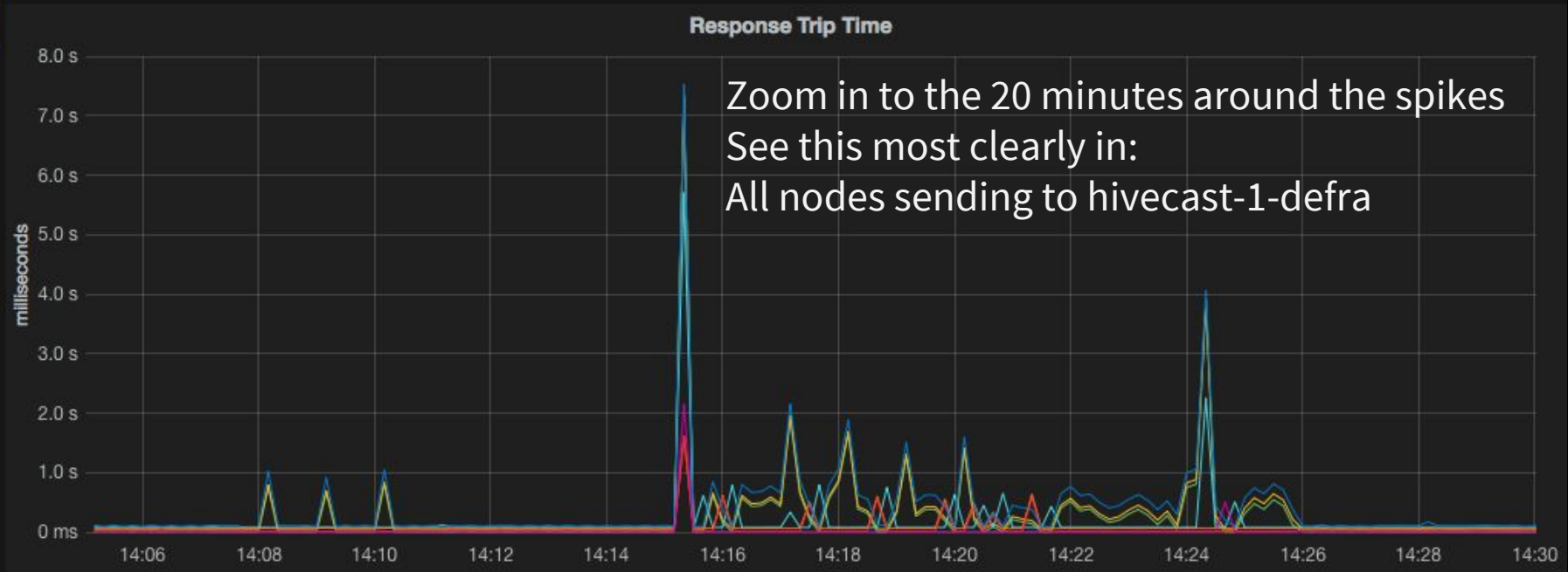
$collector: All | $zone: dyndns_com | $nameserver: 108_59_165_1 | $node: All



	min	max	avg	current
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-1-defra_as15135_net_DynInt		7.297 s	100 ms	1 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt		7.342 s	149 ms	52 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-13-uslax_as15135_net_DynInt	76 ms	5.706 s	121 ms	77 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-15-usmia_as15135_net_DynInt	59 ms	1.608 s	77 ms	72 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-3-gbion_as15135_net_DynInt	8 ms	1.595 s	25 ms	12 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-5-hkhkg_as15135_net_DynInt		7.523 s	223 ms	109 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-7-nlams_as15135_net_DynInt	3 ms	2.149 s	14 ms	5 ms

Dyn

$collector: 1-defra | $zone: dyndns_com | $nameserver: 108_59_165_1 | $node: All

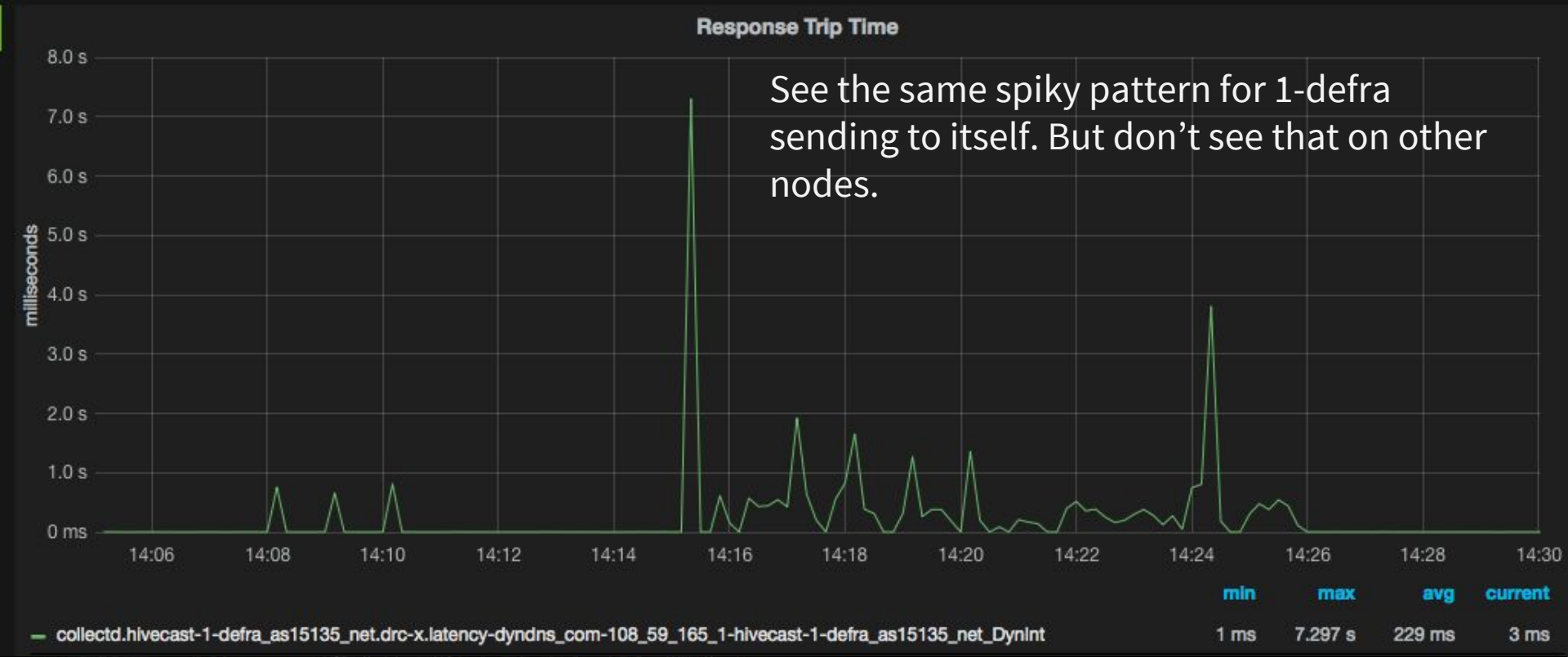


Zoom in to the 20 minutes around the spikes
See this most clearly in:
All nodes sending to hivecast-1-defra

	min	max	avg	current
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-1-defra_as15135_net_DynInt	1 ms	7.297 s	229 ms	3 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-11-usiad_as15135_net_DynInt	47 ms	7.342 s	280 ms	50 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-13-uslax_as15135_net_DynInt	76 ms	5.706 s	163 ms	77 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-15-usmia_as15135_net_DynInt	59 ms	1.608 s	92 ms	74 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-3-gblon_as15135_net_DynInt	9 ms	1.595 s	44 ms	17 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-5-hkhkg_as15135_net_DynInt		7.523 s	380 ms	107 ms
collectd.hivecast-1-defra_as15135_net.drc-x.latency-dyndns_com-108_59_165_1-hivecast-7-nlams_as15135_net_DynInt	3 ms	2.149 s	23 ms	5 ms

Dyn

$collector: 1-defra | $zone: dyndns_com | $nameserver: 108_59_165_1 | $node: hivecast-1-defra_as15135_net



See the same spiky pattern for 1-defra sending to itself. But don't see that on other nodes.

Dyn

Weird... (2)

Our traditional monitoring hasn't reported a problem with this node, but clearly there is something to look at here.

These measurements are very new and we have barely started to get to grips with them, or determine that they are useful in general operations at this point.

We have a lot to look at.

Closing thoughts

Limitations

Only useful for UDP

Currently only IPv4 is implemented

No authentication

Further work

Compare with traditional measurements

Address known limitations

Publish the tools

Further explore the observations

Advantages

A new tool in the box

Auto discovery, monitors don't need to know of anycast instances in advance

Probing can scale horizontally (though maybe not with a full mesh)

No state means no timeouts, this may reveal previously hidden weirdness

Can measure latency in a single direction


QUESTIONS?
dknight@dyn.com
Dyn

THANK YOU!