MONITORING MASSIVE NETWORK METRICS

Vicente De Luca
Sr. Network Engineer at Zendesk
vdeluca@zendesk.com
https://nethero.org

Problem we are trying to solve

Poll all load balancers metrics every 1min including:

- Memory utilization breakdown by linux processes
- Backend pool statistics including breakdown per members

Metric volume:

- 3k pool objects per load balancer
- each pool object contain multiple metrics (28)
- also nested member metrics (25 x N) N = amount of nodes
- total ~562k metrics per polling, per active load balancer

1st shot

Not reinvent the wheel by trying SNMP polling:

- Not all metrics available in factory MIBs (Ex. memory breakdown)
- Static creating custom MIBs don't scale (we tried)
- Hit a wall in CPU resources used by net-snmp (daemon crashes)

```
Feb 12 11:20:13 lb01 emerg logger: Re-starting snmpd
Feb 12 11:21:14 lb01 emerg logger: Re-starting snmpd
```

Vendor support provides no viable alternative for this scenario

Discovering alternatives

We realized we could:

- Get all the stats by CLI show cmds without harming CPU
- Cook a parser for extracting names,tags and values
- Use Datadog for our time series DB, dashboard panel and alerts system

How our data looks like ?

```
ltm pool POOL-INOG {
    active-member-cnt 2
    connq-all.age-edm 0
    connq-all.age-ema 0
    connq-all.age-head 0
    connq-all.age-max 0
    connq-all.depth 0
    connq-all.serviced 0
    connq.age-edm 0
    connq.age-ema 0
    connq.age-head 0
    connq.age-max 0
    connq.depth 0
    connq.serviced 0
    cur-sessions 605
    members {
        server1.inog.net:80 {
            addr 10.0.0.2
            serverside.bits-in 1289371
            serverside.bits-out 31293
            serverside.cur-conns 302
            serverside.max-conns 1000
            serverside.pkts-in 31920
            serverside.pkts-out 31289
            serverside.tot-conns 800
            session-status enabled
            status.availability-state available
            status.enabled-state enabled
            status.status-reason Pool member is available
            tot-requests 132913
        }
        ...
```

Why Datadog ?

- Wide utilized by dev/ops allowing easy correlation graphs
- Increase audience on network metrics
- No infra concerns on scaling up the amount of pushed metrics

Do I need to pay for Datadog ?

No. Similar approach should work with statsd and compatible backends suchs as InfluxDB

How (dog)statsd works?

- local agent (dogstatsd) listen to UDP messages
- expect metrics in the following format:

```
metric.name:value|type|@sample_rate|#tag1:value,tag2
```

And now, what ?

We cooked a script (python) to:

- parse the load balancer CLI show output
- extract metric name, tags and values
- craft and send an UDP packet to dogstatsd for each metric

UDP payload example:

```
netops.lb.serverside.cur_conn:143321|g|#pod:1,#netdevice:lb01,#vip:inog,#port:80,#view:public
```

Challenges while writing the parser (python) script

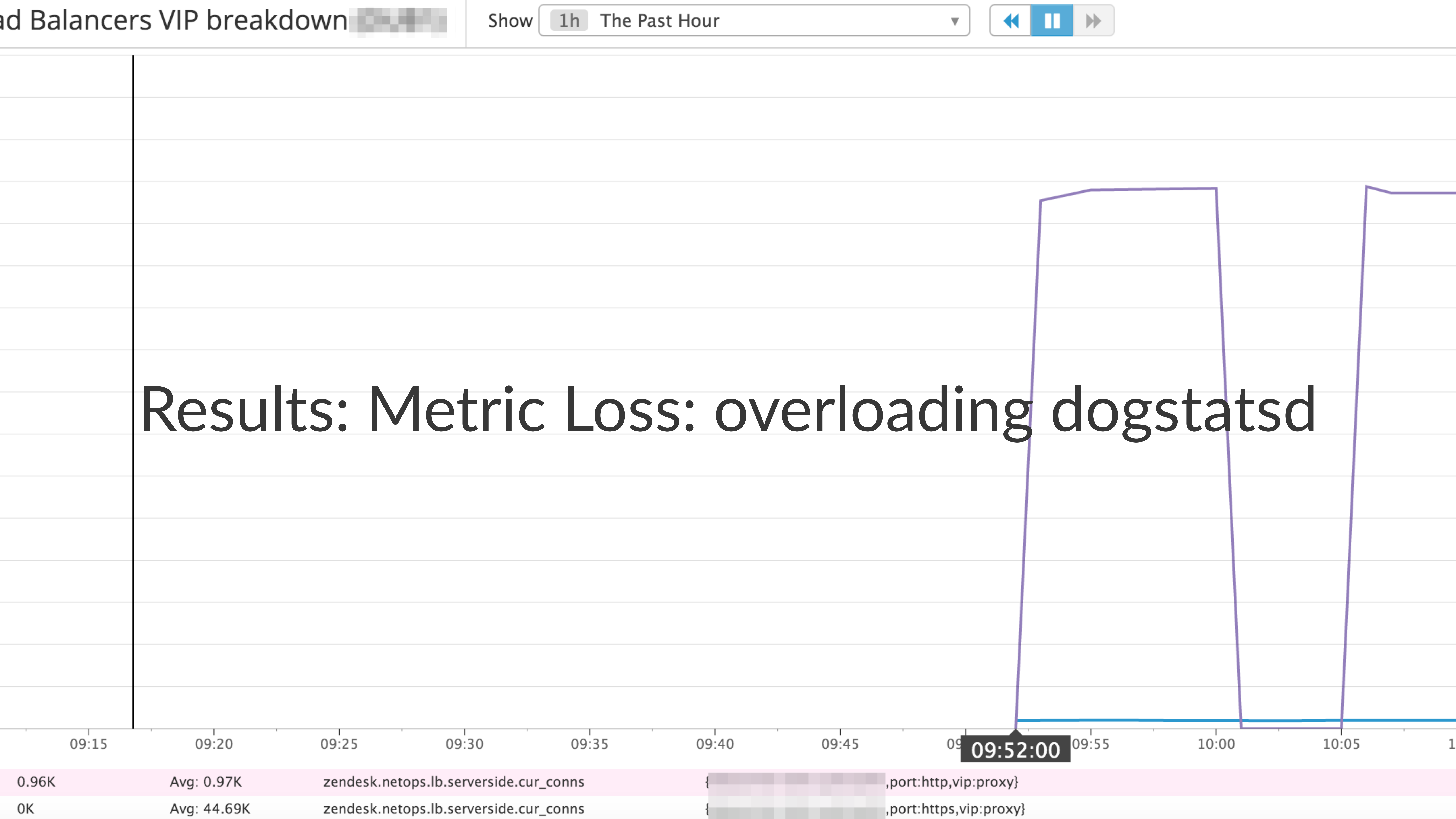
- balancing curly brackets its not an easy job
- lucky day: our load balancer output looks like JSON
- forced show output to be JSON by regex replace
- result is a python dictionary where for loops can extract name, tags and values

2nd shot

At pair of active/standby load balancers:
- bash script execute show cmd every 1 minute, compress the result and send to a linux host via netcat

At linux host:
- nc receives the data, uncompress and call a python script
- python script parse the metrics, extract tags and values
- craft and submit UDP datagrams to local dogstatsd process

Results: Metric Loss: overloading dogstatsd



ad Balancers VIP breakdown [redacted]

Show 1h The Past Hour

0.96K Avg: 0.97K zendesk.netops.lb.serverside.cur_conns {[redacted],port:http,vip:proxy}

0K Avg: 44.69K zendesk.netops.lb.serverside.cur_conns {[redacted],port:https,vip:proxy}

3rd shot (+Improvements)

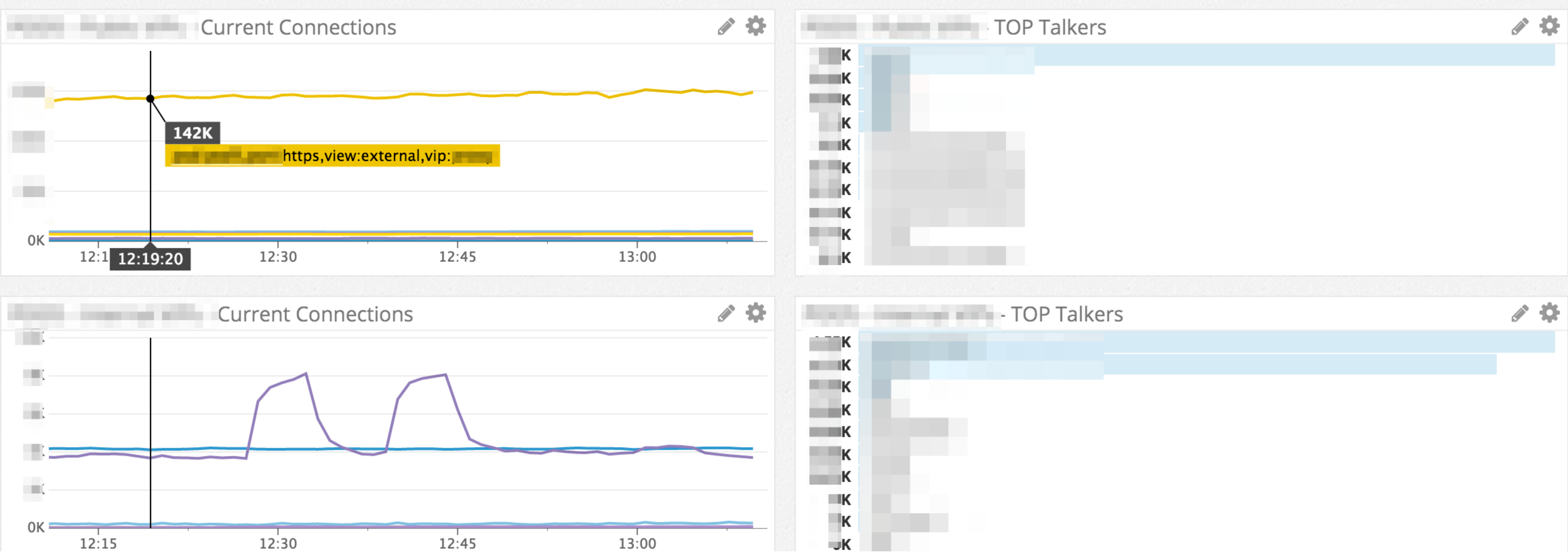
At pair of active/standby load balancers:
 - bash script now cheks if unit is active before submit metrics via netcat (reduced by half workload on dogstatsd)

At linux host:
 - Filter any non need metrics (all permanent zeroed values)
 - Splay: Python send the UDP packets in large blocks, sleeping a few before submiting next block

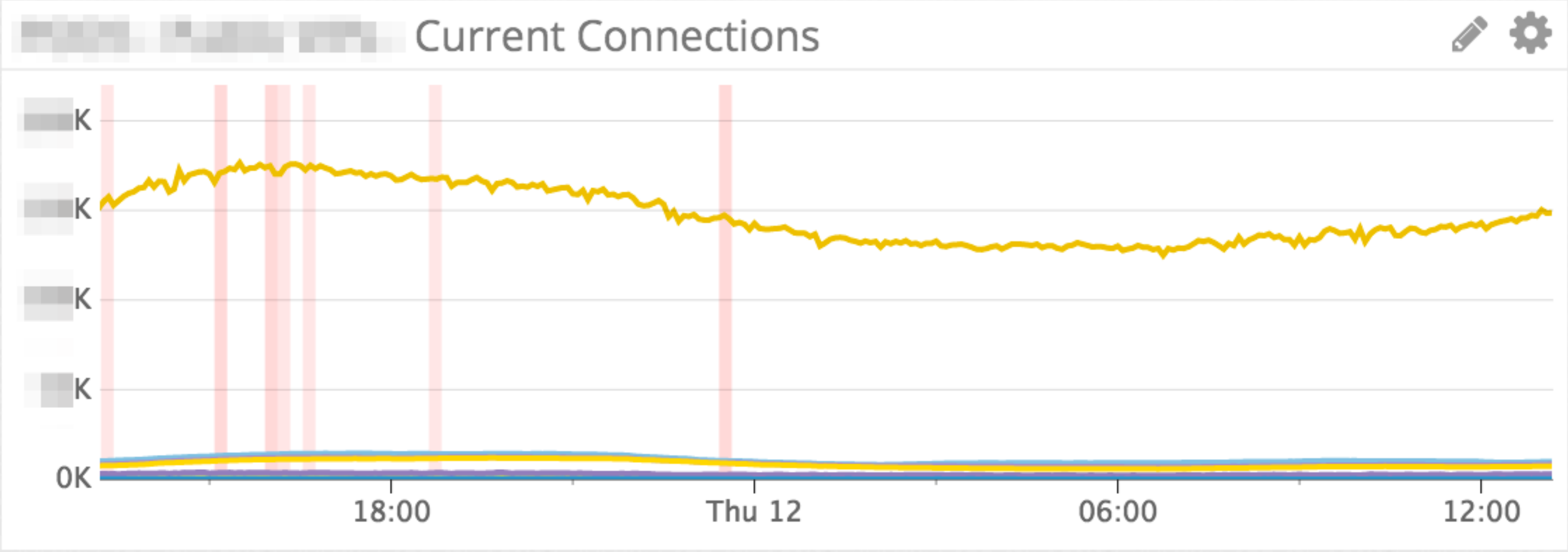
Results

How our data looks now ?

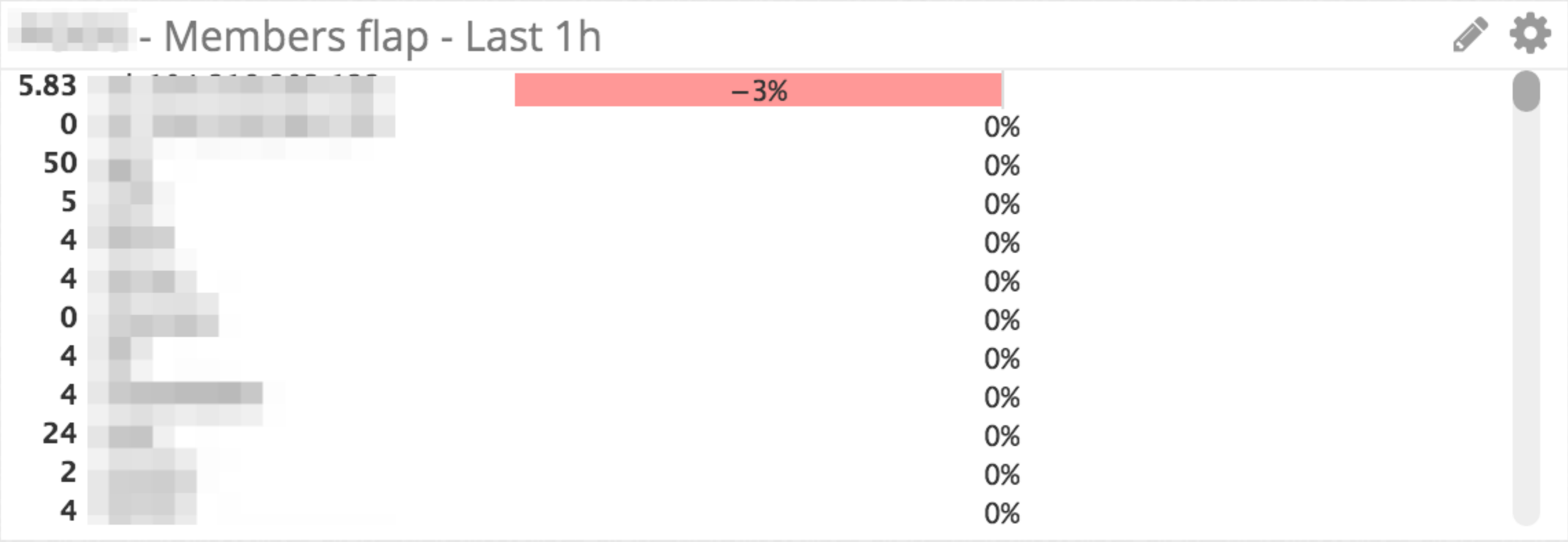
Current Connections / Top Talkers (every 1 min)


[redacted] Current Connections
142K
[redacted] https,view:external,vip: [redacted]
0K
12:1
12:19:20
12:30
12:45
13:00
[redacted] TOP Talkers
K
[redacted] Current Connections
0K
12:15
12:30
12:45
13:00
[redacted] - TOP Talkers
K

Events overlay (Network Changes)



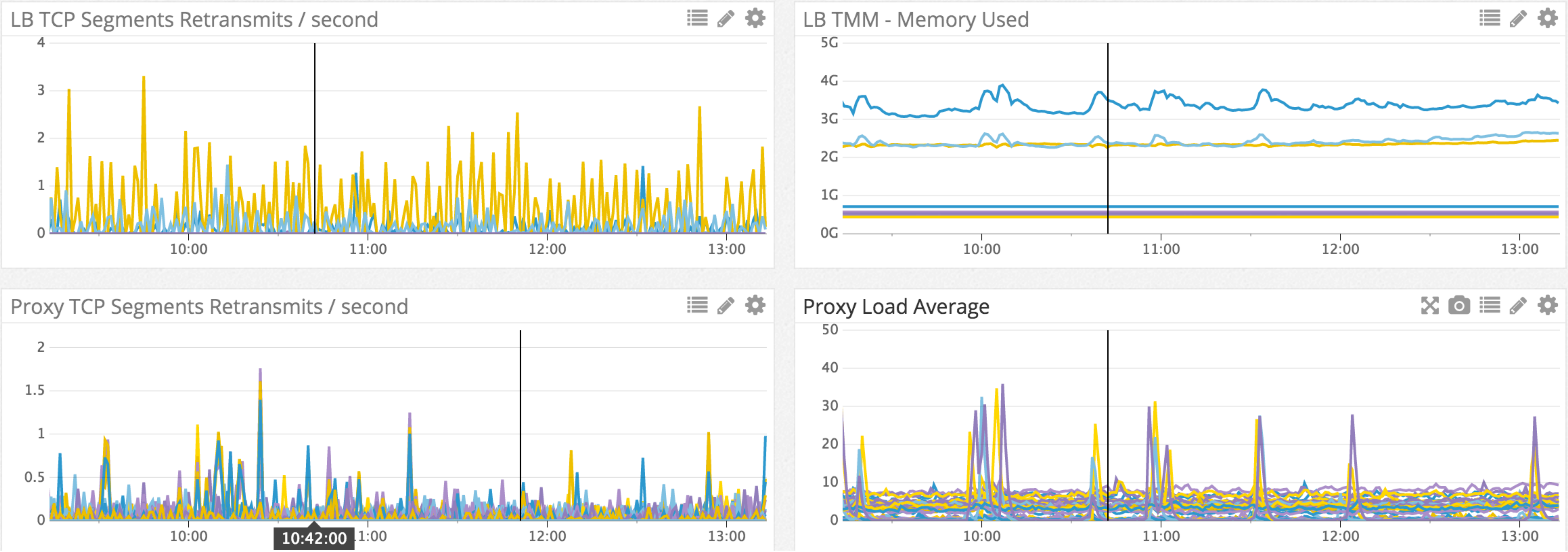
Members flap



[redacted] - Members flap - Last 1h

5.83	[redacted]	−3%
0	[redacted]	0%
50	[redacted]	0%
5	[redacted]	0%
4	[redacted]	0%
4	[redacted]	0%
0	[redacted]	0%
4	[redacted]	0%
4	[redacted]	0%
24	[redacted]	0%
2	[redacted]	0%
4	[redacted]	0%

Infrastructure correlation



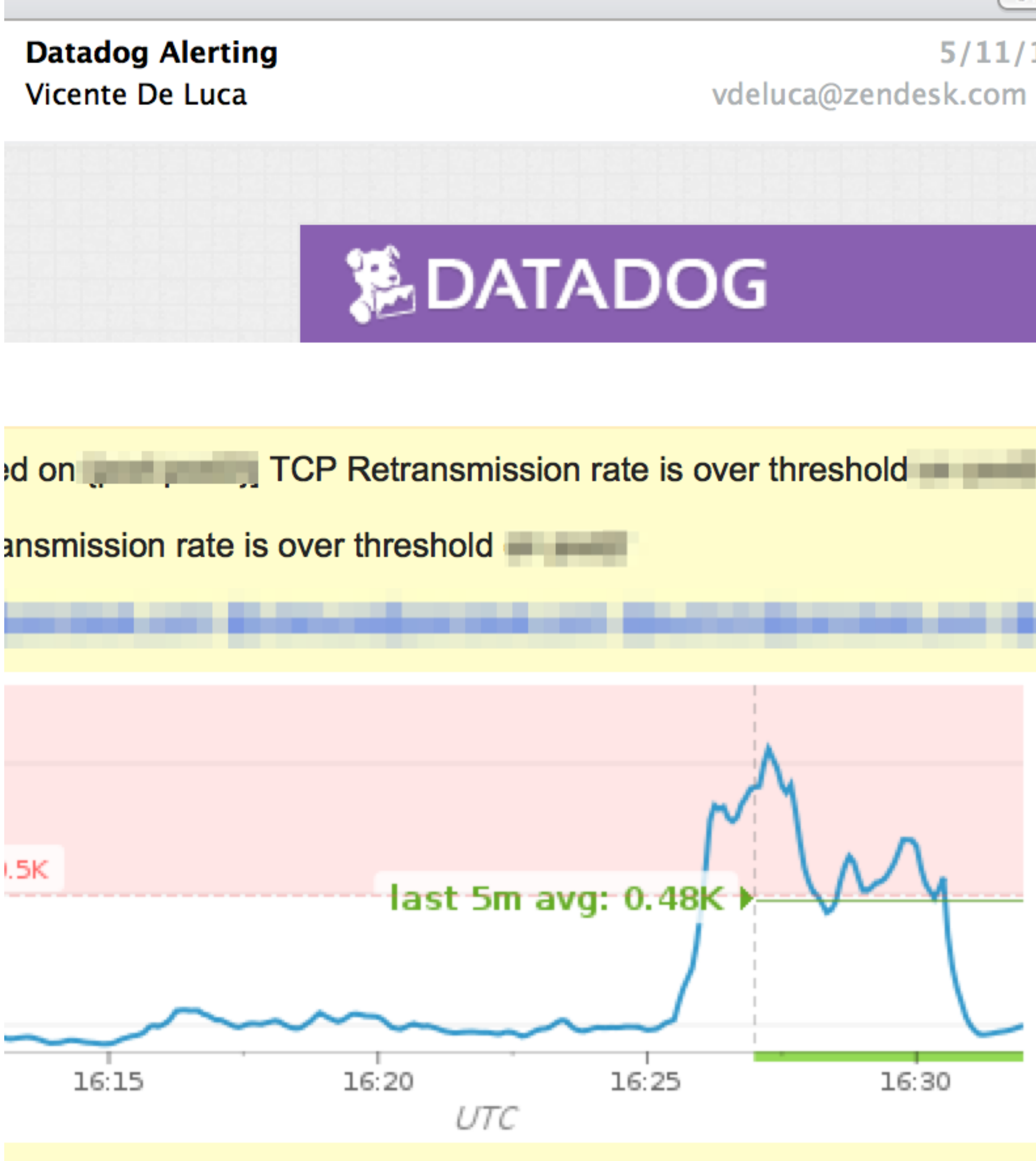
Results

Alerting

- Email
- Page on call
- all usual means

Triggered by:

- Configured Thresholds
- Outlier detection algorithms



Datadog Alerting 5/11/1

Vicente De Luca vdeluca@zendesk.com

DATADOG

ed on [redacted] TCP Retransmission rate is over threshold [redacted]

ansmission rate is over threshold [redacted]

[redacted]

Python script execution output

```
2016-05-12 12:56:58.231643 - lb01 - #31383 Metrics processed in 2.79145097733
2016-05-12 12:57:11.526415 - lb01 - #2222 Metrics processed in 0.179049015045
2016-05-12 12:57:26.471943 - lb01 - #2222 Metrics processed in 0.209563970566
2016-05-12 12:57:57.489603 - lb01 - #31383 Metrics processed in 2.79893708229
2016-05-12 12:58:17.208466 - lb01 - #31383 Metrics processed in 2.70802783966
2016-05-12 12:58:30.733715 - lb01 - #2222 Metrics processed in 0.22886300087
2016-05-12 12:58:45.601631 - lb01 - #2222 Metrics processed in 0.184427976608
2016-05-12 12:59:15.377633 - lb01 - #31383 Metrics processed in 2.51854896545
2016-05-12 12:59:28.962007 - lb01 - #2222 Metrics processed in 0.179329872131
```

Advantages

- Dynamic discovery for new pools
- Easy metric correlation between network, servers or applications
- Anomaly (outliers) detection algorithms
- Derisive CPU consumption compared to SNMP

QUESTIONS?

Special thanks for contributors here

- Cassiano Aquino (caquino@zendesk.com)
- Stephen O'Neill (soneill@zendesk.com)