TECHNICAL REPORT

IIT TR-07/2022

Monitoring the Registro .it Anycast DNS

L. Deri, M. Martinelli, D. Puliero, S. Ruberti, L. Vasarelli

Table of contents

Introduction

Registro .it [1] is the Registry of .it Internet domains, where .it is ISO 3166-1 code representing the country code Top Level Domain (ccTLD) assigned to Italy.

The use of anycast [2] addressing is the best solution to use in order to deploy a resilient and distributed DNS system, able to deliver low response time, and be robust to DDoS (Distributed Denial of Service) attacks. Currently the Registro .it runs DNS unicast nodes at selected locations and two different DNS anycast clouds: one has been developed by Registro .it and is fully managed by it; the other one is provided by a third party company which, at the time of this document, is named Netnod. Netnod complements the Registro .it nodes by adding many international locations on vantage points not served directly by Registro .it authoritative nameservers.

Due to the above architecture, monitoring the anycast DNS nameservers requires two different solutions. This is because:

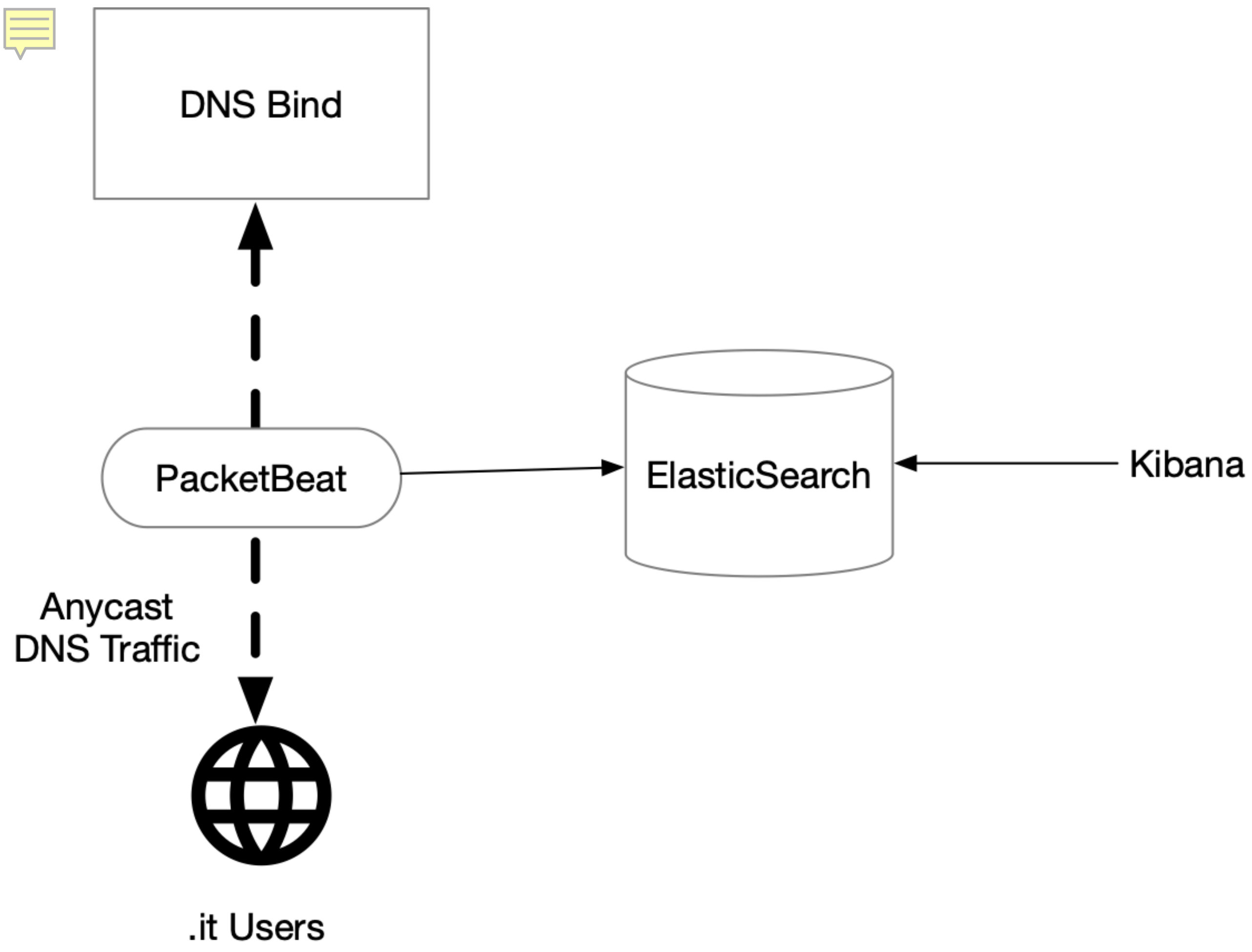
Registro .it operated nodes are fully under its control and thus it is possible to access DNS server logs that contain detailed information about the DNS traffic, or dissect DNS packets for maximum observability.

Netnod instead provides access logs at 1-minute granularity, that contain a summary of the traffic exchanged by each node. Such logs are accessible through a REST API and are formatted in XML.

The rest of this document describes the two anycast monitoring solutions that have been developed by Registro .it, and positions them in terms of features and reported data.

Registro .it Anycast Nodes

Home-grown nodes operated by Registro .it are based on the DNS BIND server software that serves client requests through the anycast network operated by Registro .it. The DNS traffic is monitored using the Elastic stack and in particular by the Packetbeat [2] component that is deployed in the same node where DNS BIND runs. But there are some cases where Packetbeat operates on a different server and receives a mirror of the DNS traffic. In both cases, it has full visibility of the DNS traffic (at the IP level), correlates DNS requests/responses and pushes data, in form of JSON documents, into the local Elasticsearch database. Queries are then accessible through the Kibana user interface, that enables the creation of powerful dashboards on top of the collected metrics.



Packetbeat has been selected as it is an open source application; that means it can be adapted as necessary. In particular, due to the unique architecture of the Registro .it, the application has been modified in order to push to Elastic additional information. In particular, a key metric used to understand the effectiveness of a DNS server, is the query source. As .it DNS nameservers are deployed at IXPs, traffic can originate from peering LANs or from the Registro .it transit connections. Creating statistics on this information, can help Registro .it to request peering to entities with a large volume of queries for which there is no a peering relationship. In order to do that, at each node the .it anycast router marks IP packets using the IPv4/v6 TOS (Type of Service) byte: peering traffic has been marked with TOS 0x28 and transit with TOS 0x30.

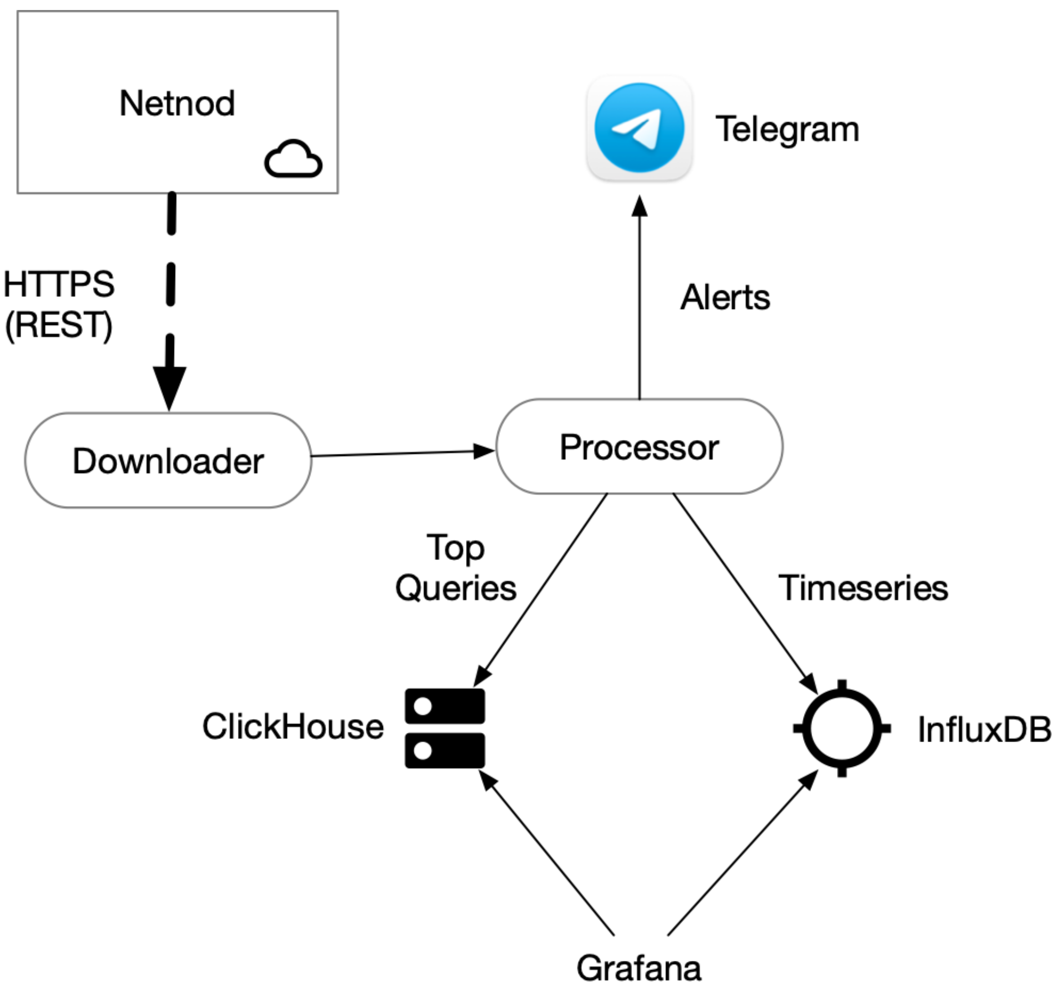
Then we have coded a Go code patch for Packetbeat (the source code is available at https://patch-diff.githubusercontent.com/raw/elastic/beats/pull/29071.diff), in order to export the IP TOS field into the JSON message sent by Packetbeat to Elasticsearch. This new field is automatically detected in the data index of Kibana, and thus available in the dashboard components with no change required. Thanks to this change, all Packetbeat instances deployed at the .it anycast DNS nodes are now able to report peering vs. transit traffic statistics in the generated dashboards.

As Elastic is a powerful albeit heavy system, it cannot be used for long term statistics (for example more than 30 days), as it would require the deployment of a complex clustering system at every DNS anycast site. For this reason, long terms metrics are kept on another system based on Munin software [4], an open source system and network monitoring application, that exports summaries through a Python script querying directly the BIND statistics channels and writes them on a database named RRD (Round Robin Database) [5]. These metric databases had been configured in the past to store data for just a few years. For this reason, an ad-hoc utility has been developed in order to modify the database structure and extend the metric storage to 10 years, while preserving existing data.

Netnod Anycast Monitoring

Netnod anycast DNS nameservers are deployed at IXPs and traffic originates only from peering LANs.

Monitoring Netnod-powered anycast DNS servers has been implemented by Registro .it from scratch by developing custom code able to collect monitoring information made available by Netnod and pushing metrics to a timeseries databases [6] and top queries to a columnar SQL [7] database as depicted below.



When an error condition (e.g. a node is down or a traffic anomaly is detected) an alert is sent to Telegram, a popular messaging application, that will then send alerts to network administrators. Finally, Grafana, a popular system for creating dashboards, will then access data stored in databases in order to create a dashboard.

The following sections describe in details the various components and report how they work internally.

Data Collection

All Netnod information can be fetched via REST API from the following endpoint:

https://www.netnod.se/dsc/api/v1/dsc/$node/$YEAR_MONTH_DAY_HOUR

where

- $node is a 3 letter abbreviation of the Netnod site that hosts the DNS server. To date the list includes AMX, AMZ, ANK, ASH, ASU, BAH, BKK, BNX, BTS, CCP, CHI, CMX, COL, CPH, DBI, DEX, ECX, FIX, FMT, GOT, GUA, GYE, HKX, JAK, JAX, JBT, JNB, JPP, KUL, LIM, LJU, LNX, LUL, LUX, MEX, MIA, MIX, MNP, MTV, MTY, NIX, ORG, PIX, PNH, PRT, QTR, RIX, RKV, ROX, SAO, SDL, SIN, SOX, SPB, STH, TAI, THI, TLL, TOK, ULA, WEL, WIE, YYC. Each node has a corresponding human-readable label (e.g. WIE corresponds to Vienna, Austria) that is then used in dashboards.
- $YEAR_MONTH_DAY_HOUR (specified in YYYYMMDDHHMM format) instead represents the hour for which we want to retrieve monitoring data. Time is specified according to the UTC timezone and at any point in time the system cannot be queried for data older than two hours with respect to the current time.

Data is specified in XML format. As each file contains one hour, the XML file is divided in 60 "one minute" sections. Each section contains several blocks that include information such as top queries, TCP vs UDP queries distribution, top client queries etc. All data is in clear text with the exception that IP addresses have the least significant byte set to zero in order to implement a lightweight anonymisation facility. Below you can find an example of a section of an XML file that contains top X clients queries.

```
 <array name="client_subnet" dimensions="2" start_time="1651561200"
stop_time="1651561260">
  <dimension number="1" type="All"/>
  <dimension number="2" type="ClientSubnet"/>
  <data>
   <All val="ALL" count="1734">
    <ClientSubnet val="101.7.8.0" count="19"/>
    <ClientSubnet val="114.71.100.0" count="6"/>
    <ClientSubnet val="117.16.191.0" count="4"/>
    <ClientSubnet val="121.194.9.0" count="4"/>
    <ClientSubnet val="122.129.122.0" count="1"/>
    <ClientSubnet val="124.158.186.0" count="3"/>
    <ClientSubnet val="128.112.128.0" count="1"/>
    <ClientSubnet val="128.112.129.0" count="4"/>
    <ClientSubnet val="128.119.10.0" count="3"/>
    <ClientSubnet val="128.122.0.0" count="3"/>
    <ClientSubnet val="128.135.249.0" count="2"/>
    <ClientSubnet val="128.146.1.0" count="4"/>
    <ClientSubnet val="128.195.199.0" count="2"/>
    <ClientSubnet val="128.197.253.0" count="2"/>
    <ClientSubnet val="128.200.200.0" count="1"/>
    <ClientSubnet val="128.223.60.0" count="2"/>
    <ClientSubnet val="128.230.21.0" count="2"/>
    <ClientSubnet val="128.238.1.0" count="1"/>
    <ClientSubnet val="128.239.60.0" count="2"/>
    <ClientSubnet val="128.32.206.0" count="13"/>
    <ClientSubnet val="129.105.49.0" count="2"/>
    <ClientSubnet val="129.108.9.0" count="1"/>
    <ClientSubnet val="129.137.96.0" count="1"/>
    <ClientSubnet val="129.171.150.0" count="1"/>
    <ClientSubnet val="129.171.64.0" count="3"/>
    <ClientSubnet val="129.22.104.0" count="1"/>
: </array>
```

The reader can immediately realise that data is grouped and thus it is not possible to perform low-level statistics or answer questions such as "what queries has been issued by host a.b.c.d".

In order to monitor nodes availability (e.g. some might be under maintenance or unavailable for other reasons), the downloader component takes into account and keeps track of some issues:

- Nodes unavailable
- (i.e. the download fails)
-
- Nodes with no data
  (i.e. node up but probably not serving data and thus with an empty XML).

- Nodes with partial data
  (i.e. some nodes report no traffic for specific XML sections).

Based on the issue reported, the downloader component reports the error immediately (e.g. download failed) or forwards the downloaded file to the parsing component. In case of failure, the failure time is reported so it can be used in alerts.

Data Processing

The parsing component, written in Python, is responsible for interpreting the XML file just downloaded and extracts the various sections that contain traffic statistics.

In each section there are two types of data:

- usage metrics (e.g. number of queries), that is a numerical value that needs to be store on the timeseries database.
- top X statistics (e.g. top X queries), that instead need to be stored on the SQL database.

Grafana is already used in Registro .it for data visualization and therefore we have decided to also use it in this project. For this reason, we have selected some databases that are natively accessible by Grafana without coding custom extensions that might be error prone to maintain as the various components change versions over time.

The databases we chose are:

- InfluxDB for storing timeseries.
- ClickHouse for storing non-numerical data.

Both databases are open source and require no license or node clusters to operate at the scale request by this project. These databases have been selected as:

- InfluxDB is a popular database used for very long time in the monitoring industry that is robust and efficient.
- ClickHouse is a very fast columnar database able to store billion of records in very little space.

Both databases are not designed to be ingested with individual data. This means that the parser tool creates some temporary files where data is stored and then imported in batches. InfluxDB files are written using the native Line protocol format, whereas ClickHouse files in SQL.

The processing tool is invoked after the downloader completion and it parses all the successfully downloaded files. At the end of the processing, metrics and data are appended to temporary files stored on the filesystem. When all the downloaded files have been processed, a new component imports the data into the two databases and then deleted the temporary files.

These steps are repeated once an hour, triggered by the crontab facility.

Detecting Metric Anomalies

Numerical metrics (e.g. number of queries per hour) are checked both against value thresholds and comparing them with past values. In order to limit the number of reported issues, the system uses only the lower threshold set to zero but there is no upper-bound for top numerical values.

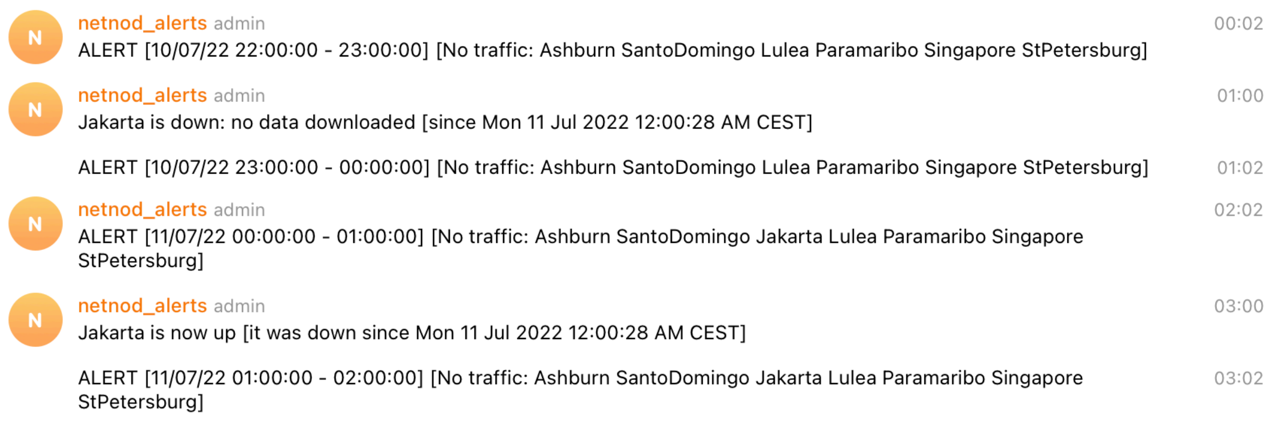
Due to the heterogeneous natures of DNS nodes, using thresholds to trigger alerts when metric X exceeds value Y was not an option. This is because each node has a different behavior and also because Netnod, probably due to traffic engineering needs, often redistributes traffic across nodes close in distance (e.g. Amsterdam 1 and 2). In order to address these issues, the monitoring system uses an algorithm based on exponential smoothing [8] in order to detect anomalies. This algorithm has been adapted to the monitoring needs both in terms of smoothing strategy and issue repetition. This means that:

- The typical confidence bands used in smoothing algorithms have been increased by 20% to avoid triggering alerts when values slightly exceed the threshold.
- Low values are ignored and not matched for anomalies. This is because some nodes receive very few queries per minute (< 10) and thus with so little data we can create many false positives when at minute X you have 2 queries and a minute X+1 4 queries, that is double the value but still close to noise.
- In order to avoid creating per-minute anomalies that be of no interest as the nature of DNS traffic is very spiky, we are using learning intervals of one day, and perform per-hour value checks. This allowed us to reduce the number of alerts and consider anomalous an hourly interval and not the individual minute.

In summary, an alert is triggered on a metric whenever its value is zero (i.e. no traffic is reported), or if in the last hour the reported value is falling outside the lower/upper prediction boundaries.

Alerting Subsystem

In case of failure (e.g. a download failed or an anomalous metric value) an alert is triggered. As the popular Telegram application is already in use for distributing alerts for other projects, it has also been used for reporting Netnod .it anycast alerts. At the end of each hourly data, processing alerts, if any, are reported in Telegram using a bot application we coded, that delivers messages to relevant people. Below you can see a typical alert log.



```
netnod_alerts admin                                                                                00:02
ALERT [10/07/22 22:00:00 - 23:00:00] [No traffic: Ashburn SantoDomingo Lulea Paramaribo Singapore StPetersburg]
netnod_alerts admin                                                                                01:00
Jakarta is down: no data downloaded [since Mon 11 Jul 2022 12:00:28 AM CEST]
ALERT [10/07/22 23:00:00 - 00:00:00] [No traffic: Ashburn SantoDomingo Lulea Paramaribo Singapore StPetersburg]   01:02
netnod_alerts admin                                                                                02:02
ALERT [11/07/22 00:00:00 - 01:00:00] [No traffic: Ashburn SantoDomingo Jakarta Lulea Paramaribo Singapore
StPetersburg]
netnod_alerts admin                                                                                03:00
Jakarta is now up [it was down since Mon 11 Jul 2022 12:00:28 AM CEST]
ALERT [11/07/22 01:00:00 - 02:00:00] [No traffic: Ashburn SantoDomingo Jakarta Lulea Paramaribo Singapore   03:02
StPetersburg]
```

The following principles have been applied:

- Alerts are not sent individually (i.e. per node) but per hour in order to reduce the number of messages.
- As these alerts are stateful (e.g. node X is down, node X is now up again) the system keeps track of this state change. In the above figure the Jakarta node went down at 1 AM, and back at 3 AM, and the alerting system reported the duration of the outage.

It is common to see failures in nodes as some of them are (probably) under periodic maintenance. A typical message is:

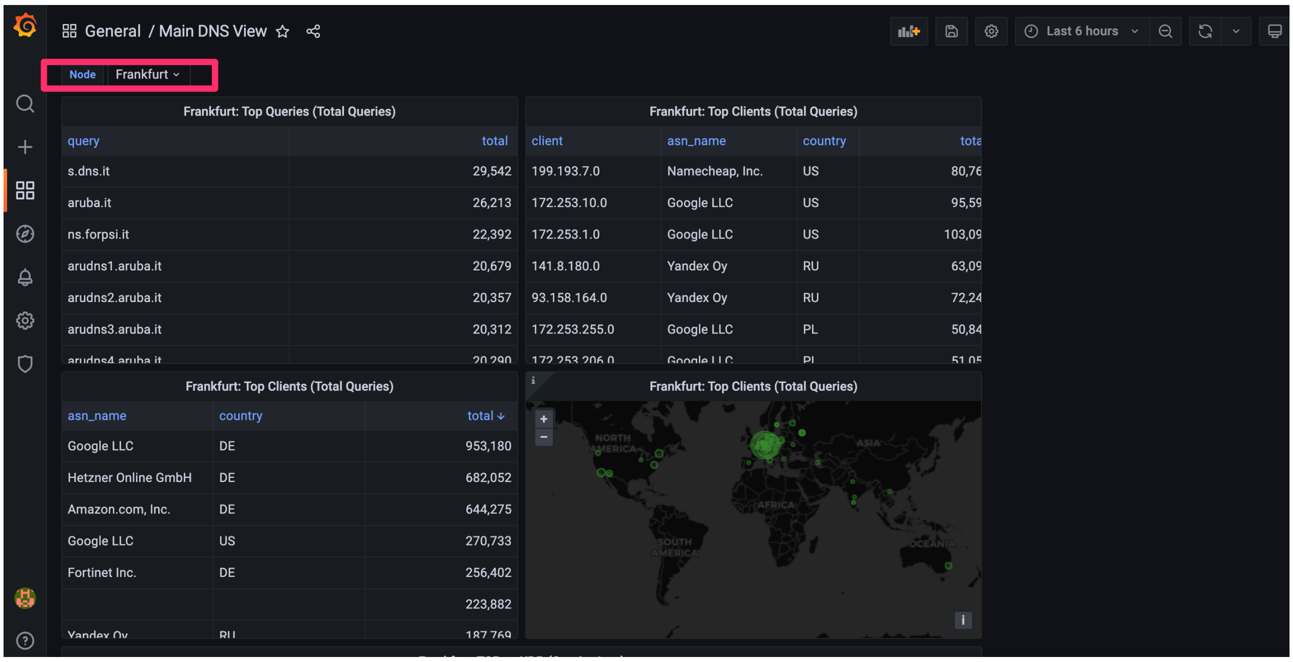
ALERT [10/07/22 20:00:00 - 21:00:00] [No traffic: Ashburn SantoDomingo Jakarta Lulea Paramaribo Singapore StPetersburg][Anomaly: Luxembourg]

that contains

- The alert time.
- List of nodes (if any) with zero traffic reported.
- List of nodes (if any) with anomalies in metric behaviour in the past hour.

Traffic Dashboards

As already discussed, traffic dashboards have been implemented on top of the Grafana tool. The main page after login reports on the top right corner the node name that can be changed at any time by selecting the node in the drop-down menu.



Frankfurt: Top Queries (Total Queries)

query	total
s.dns.it	29,542
aruba.it	26,213
ns.forpsi.it	22,392
arudns1.aruba.it	20,679
arudns2.aruba.it	20,357
arudns3.aruba.it	20,312
arudns4.aruba.it	20,290

Frankfurt: Top Clients (Total Queries)

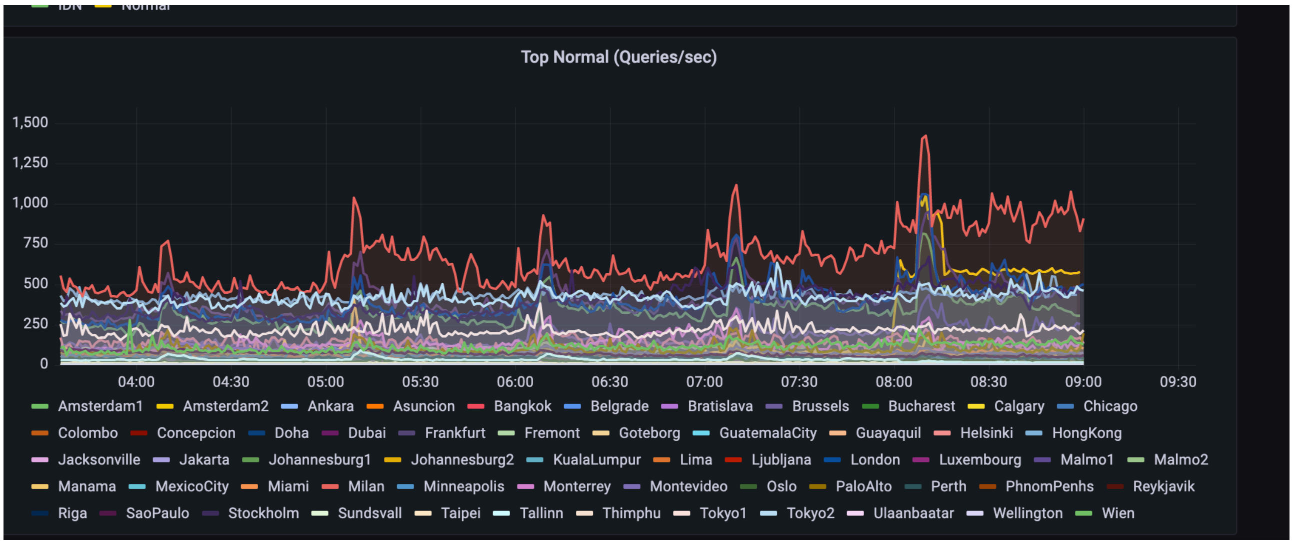
client	asn_name	country	tota
199.193.7.0	Namecheap, Inc.	US	80,76
172.253.10.0	Google LLC	US	95,59
172.253.1.0	Google LLC	US	103,09
141.8.180.0	Yandex Oy	RU	63,09
93.158.164.0	Yandex Oy	RU	72,24
172.253.255.0	Google LLC	PL	50,84
172.253.206.0	Google LLC	PL	51,05

Frankfurt: Top Clients (Total Queries)

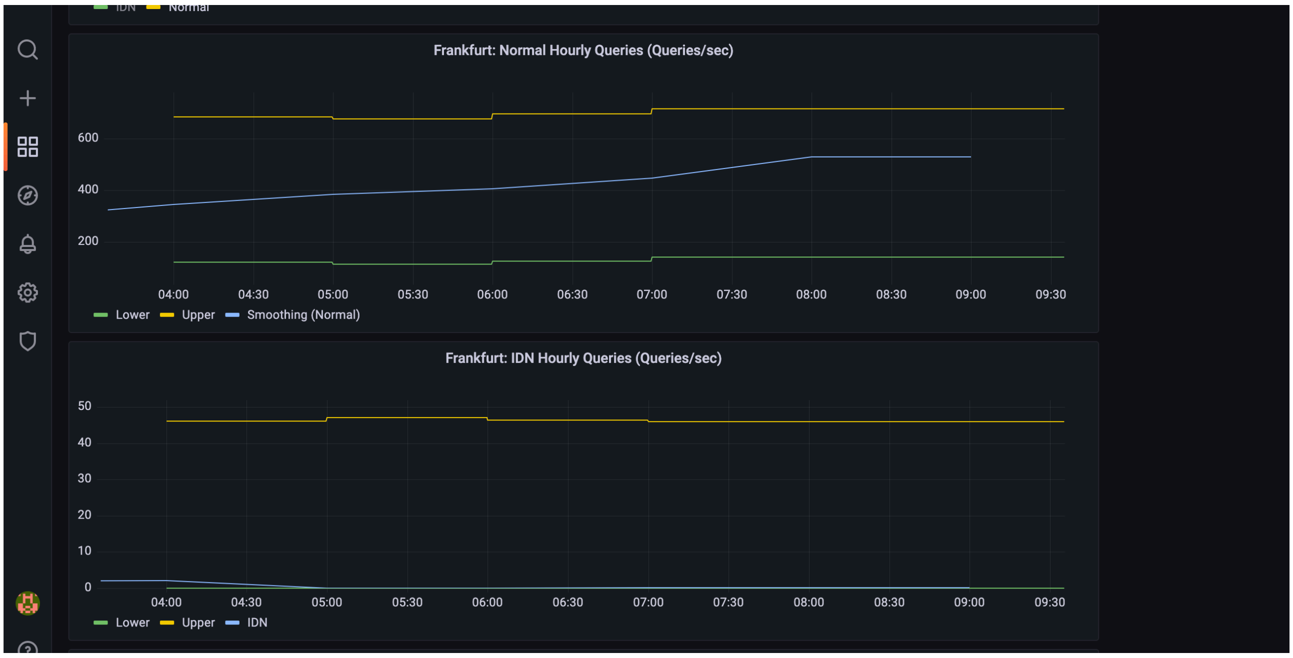
asn_name	country	total ↓
Google LLC	DE	953,180
Hetzner Online GmbH	DE	682,052
Amazon.com, Inc.	DE	644,275
Google LLC	US	270,733
Fortinet Inc.	DE	256,402
		223,882
Yandex Oy	RU	187,769

Frankfurt: Top Clients (Total Queries)

Table information is fetched from ClickHouse whereas non tabular data, such as the geomap of client queries or traffic metrics, are based on data stored in InfluxDB.



Although reports are per node, each node page also contains a chart (reported above) that includes the traffic of all active nodes.



The above image depicts query reports with boundaries. The smoothing algorithm predicts the lower (green) and upper (yellow) values, and whenever the smoothed value (blue), based on observations read from Netnod data, exceeds the boundaries an anomaly is detected.

References

[1] Registro .it, https://www.nic.it

[2] CloudFlare, What is Anycast DNS? | How Anycast works with DNS, https://www.cloudflare.com/en-gb/learning/dns/what-is-anycast-dns/

[3] Elastic Inc, Packetbeat: Network Analytics Using Elasticsearch, https://www.elastic.co/beats/packetbeat

[4] https://munin-monitoring.org/

[5] RRDtool - Round Robin Database Tool, https://github.com/oetiker/rrdtool-1.x

[6] Influx Data, InfluxDB Database, https://www.influxdata.com

[7] ClickHouse Inc, Fast Open-Source OLAP DBMS, https://clickhouse.com

[8] Exponential Smoothing, https://en.wikipedia.org/wiki/Exponential_smoothing