Visualization and Monitoring for the Identification and Analysis of DNS Issues

Christopher Amin, Massimo Candela, Daniel Karrenberg, Robert Kisteleki and Andreas Strikos
Réseaux IP Européens (RIPE) Network Coordination Centre
Amsterdam, Netherlands
Email: camin@ripe.net, mcandela@ripe.net, daniel.karrenberg@ripe.net, robert@ripe.net, astrikos@ripe.net

***Abstract*—The user experience of an Internet service depends partly on the availability and speed of the Domain Name System (DNS). DNS operators continually need to identify and solve problems that can be located at the end user, a name server, or somewhere in between. In this paper, we show how DNSMON, a production service for measuring and comparing the availability and responsiveness of key name servers, correlates and visualizes different types of measurements collected by RIPE Atlas vantage points worldwide. DNSMON offers an interactive view, both historic and near real-time, at different levels of detail. It has successfully revealed and allowed analysis of many operational issues, including less obvious ones.**

***Keywords*–DNSMON; DNS; monitoring; network visualization.**

I. Introduction

The Internet Domain Name System (DNS) [1][2] provides a mapping from user-visible domain names to other identifiers, such as network layer addresses. The performance of most Internet services can be perceptibly influenced by the quality and responsiveness of the DNS. Since it is a distributed system, its performance depends in turn on a number of elements, such as the authoritative name servers, caching name servers, local resolvers and the network in between. Due to the importance of this infrastructure, there have been many monitoring projects collecting data about different aspects of DNS, such as stability, security, performance, and traffic. Some of these projects also provide visualizations for administrators and decision makers — especially the DNS operators at various levels.

The visualization of Internet measurements in general has seen a growing interest in recent years. This is mainly due to the huge amount of data collected by Internet measurements projects, which cannot be fully understood and explored without meaningful visual representations. There are various free and commercially available DNS visualization tools, some examples being: DNSViz [3], which, given a domain name, visualizes the chain of name servers and certificates involved in the DNSSEC chain of trust; Flying Term [4], which offers a visualization to help administrators identify and understand DNS querying behavior due to anomalies such as misconfiguration and security events; DNSential [5], which allows users to issue custom IP address or domain name queries returning a graphical depiction of IP/domain relationships over time; RTIVS [6], which helps administrators to display and detect DNS amplification attacks; and VisualK [7], a system that visualizes how and when the clients of K-root name server migrate from one instance to another.

DNSMON [8] is a monitoring project started in 2001 to actively measure authoritative DNS servers at the root and top-level domain (TLD) level, from a large enough number of vantage points to reliably identify issues at or close to the servers themselves. While the majority of the DNS tools, including the tools cited above, focus on data collected by a specific server or through DNS resolutions from the resolver point of view, DNSMON aims to constantly monitor all the name servers belonging to entire zones — considered strategic for the functioning of the whole Internet — through performance measurements. It was initially conceived as a response to claims that root name servers performed poorly. Such claims were often based on measurements from one or — at most — a handful of vantage points and thus heavily influenced by network performance on a small number of network paths. The first implementation of DNSMON was based on a small process running on the nodes of the RIPE Test Traffic Measurement (TTM) [9] network. This process executed DNS queries and reported response times to a central server which stored them in round-robin database (RRD) [10] files. Users were able to monitor name servers and DNS zones through several automatically generated, non-interactive images. Due to the limitations of RRD and the static images, it was not possible to view results from arbitrary time periods at arbitrary levels of detail, nor to interactively access related measurements.

The project has proven its usefulness amongst operators and has evolved over time. In this paper, we will present the latest stage of DNSMON. The system is now based on the RIPE Atlas [11] network measurement project, which currently counts around 8000 active probes and 100 anchors worldwide carrying out more than 2000 network measurements per second. An anchor is a high-end machine colocated in a professional data center with defined network requirements; a probe is a small device usually located at end user sites. We decided to use measurements performed by anchors rather than probes in the interest of providing more accurate, reliable and consistent information without needing to compensate for erratic uptimes. It is also beneficial to avoid possible noise present in end user networks, since the focus is to identify issues near to core services.

Although this tool targets mostly network operators, the services they operate are sufficiently important that by facilitating analysis and quality of service improvement to diverse locations around the world, significant benefits are conferred to all users of the wider Internet.

Figure 1 shows a snapshot of the DNSMON interface. The main objectives of the system are as follows. The user selects a DNS zone z and an interval of time $\mathcal{T}$, the system shows a visual overview of the results of the measurements performed by the anchors against the set $\mathcal{S}$ of name servers belongong to z over time. The servers in $\mathcal{S}$ are detected in advance by performing an Name Server (NS) lookup on the zone, and collecting all unique IP addresses found in referenced A or AAAA records. This type of visualization

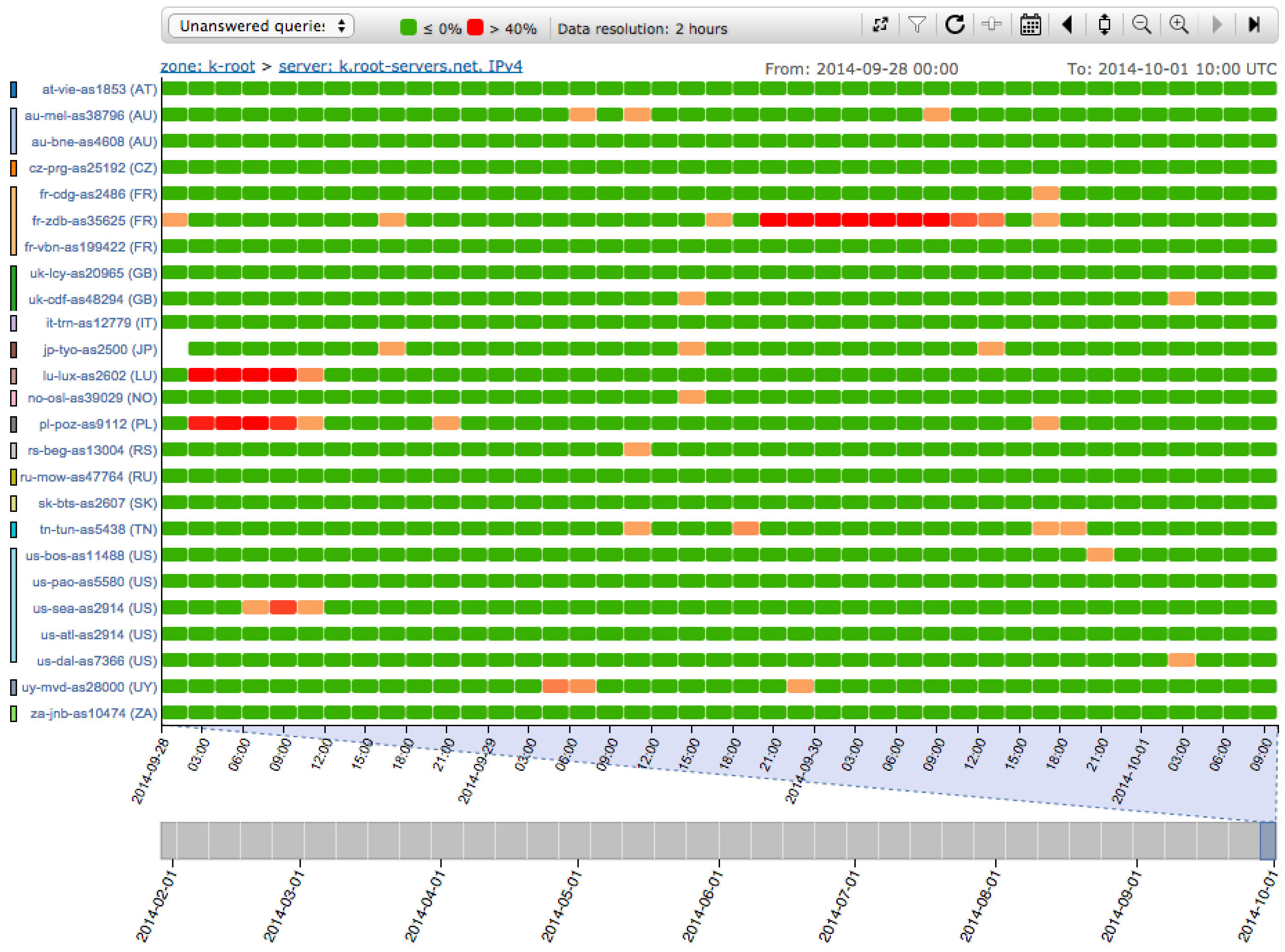


Figure 1. The main interface of DNSMON.

should be effective in giving a close to real-time overview of the quality of the entire zone at a glance and flexible and interactive enough to increase the details during analysis, while also correlating different types of measurement results and distinguishing abnormal from ordinary situations. The back end should only provide the data while the visualization and data correlation should be completely client-side, accessible from a web browser without third-party plug-ins. A typical use of our system is the following. Let γ be a geographical region in which the anchor α is deployed. Suppose that during $\mathcal{T}$ the DNS resolution of z exhibited some packet loss or high latency problem. Is there a specific name server s involved in the problem? Is the problem bound to γ? How can we exclude the possibility that the problem is related to a malfunction of α? Moreover, how did α reach s before and after the issue and what DNS responses were received?

The paper is organized as follows. In Section II, we describe the adopted visualization approach and introduce some formal terminology. Section III gives an overview of the general method for finding and solving problems using the tool, as well as some specific examples. In Section IV, we describe the collection and processing of measurement data used to support the visualization. In Section V, we outline the implementation of our tool and the technical challenges we faced. In Section VI, we present our conclusions and mention future directions.

II. Visualization Approach

For each monitored zone in DNSMON, we look up all the name servers defined for that zone, and schedule DNS measurements against them. The primary scheduled measurements are periodic Start of Authority (SOA) queries executed by a common subset of the RIPE Atlas anchors. Assumptions about the periodicity of the measurements are not hardcoded and may be varied in the future, depending on the monitoring needs.

As stated in the user requirements in Section I, the user should be able to monitor an arbitrary interval $\mathcal{T}$, up to the entire available history in the system. Therefore, the visualization should be able to represent both the potentially large number of results collected during $\mathcal{T}$ and to clearly convey the zone status without requiring interaction.

The visualization methodology we adopted is presented below together with supporting motivations. We discarded solutions based on line charts or time animations. Line charts are often used, in combination with downsampling algorithms, to represent trends in large amounts of data. The purpose of the downsampling is to try to retain the visual characteristics of the original line using considerably fewer data points, which in a web application results in a helpful reduction in the number of Document Object Model (DOM) elements to be handled by the browser [12]. At the same time, line charts require great attention to the axes, especially during comparisons of different trends. Moreover, we need to represent multiple trends concurrently and this requires too much space for an intelligible representation. A time animation, instead, can solve the space problem by representing a portion of $\mathcal{T}$ at a time, but requires the user to interact with the system and it does not provide an immediate overview of the whole $\mathcal{T}$, violating two base requirements.

We decided to follow Shneiderman's mantra of "overview first, zoom and filter, then details-on demand" [13] and to adopt a matrix with two axes. Each cell represents a measured value for the visualized zone z by using colors. Since the canvas space is limited and a single cell should be big enough to be clearly distinguishable and allow the user to interact with it, a data aggregation mechanism has been adopted representing groups of results instead of single values. $\mathcal{T}$ is represented on the x-axis and is divided into sub-intervals. Each cell represents an evaluation of the aggregation of all the results collected by the system in a certain sub-interval. The aggregation drastically reduces the number of represented elements, making it possible to use a performant and purely client-side browser visualization approach. We use the term "*native resolution*" for the case in which $\mathcal{T}$ is so small that each sub-interval contains a single measurement result.

In addition to z and $\mathcal{T}$, a name server s can be specified as an input. If s is specified, the set of anchors $\mathcal{A}$ monitoring z is represented on the y-axis. In this case, a cell refers to a measurement executed by a specific anchor and involving s at a sub-interval $t \in \mathcal{T}$. Instead, if s is not specified, all the name servers in $\mathcal{S}$ are represented on the y-axis. In this second case, a cell refers to the aggregation of all the data collected by all the anchors in $\mathcal{A}$ against a specific name server in $\mathcal{S}$ at a sub-interval $t \in \mathcal{T}$. By default s is not specified, a choice aligned with the aim of fostering incremental information enrichment, but it is possible to specify it by clicking on a name server label on the y-axis. The elements on the y-axis are grouped by using colored rectangles placed on the left of the axis labels. The colors are computed with the algorithm described in [14] to ensure that they are distinguishable from each other. The IP addresses, both IPv4 and IPv6, derived from lookups of the same NS record are visually close and grouped together, allowing a comparison between protocols and isolation of possible protocol-specific routing problems, while also satisfying modern operators who are as concerned with IPv6 reachability as IPv4. The anchors are grouped by the country code [15] where they are deployed, adding the possibility of seeing geographic correlations.

The user can select a *point of view*, or rather which qualitative aspect of the measurement to visualize. In particular, in the actual release, the available points of view are: packet loss, Round Trip Time (RTT), and relative RTT. Each cell in the *packet loss and RTT views* depicts respectively the percentage of packets lost and the amount of milliseconds of RTT measured for the sub-interval represented by the cell. The *relative RTT view* shows the percentage disparity of each sub-interval relative to the minimum measured in each row, highlighting unexpectedly high RTT values, and smoothing ordinarily high latencies. As a first approach we tried to represent more than one point of view at the same time, but this solution was discarded for the following reasons: it requires different graphical metaphors resulting in an increased complexity of the scene; given the amount of data to be represented, introducing additional elements into the scene causes cluttering and performance issues; and some users may be interested in only one qualitative aspect while, since the points of view are strongly related, spotting a problem in one will often indicate the same issue in another, causing an unnecessary information overload.

The cells are colored according to a threshold pair $c1, c2$. Cells corresponding to values below $c1$ are shown in green, above $c2$ in red, and between $c1$ and $c2$ in a color from a gradient domain between green and red. When a value is not available, the coresponding cell is shown in gray. In the measurements performed by DNSMON, lower values indicate better results. For this reason the choice of the colors reflects a transition from positive low values (green) to negative high values (red) [16]. The aim of the two thresholds is to create a visual separation between values considered good and bad, reducing the noise on the matrix. This is achieved by creating two separate ranges — the expected acceptable values, and the unacceptable ones — showing the gradient only for the cells between the two. A pair $c1, c2$ is defined for each point of view, by default tollerant threshold values are set. The user can easily tune $c1, c2$ to change the sensitivity of the tool and adapt it to the specific case. In addition to the coloring, it is possible to obtain the textual values of a cell, along with other information, by hovering over it with the mouse.

The user interaction plays a major role in our visualization. It is possible to zoom the viewport in and out in order to reduce or increase the visualized interval $\mathcal{T}$. When $\mathcal{T}$ changes, the data resolution also changes. In addition, $\mathcal{T}$ can be shifted in time, thereby maintaining the same data resolution. A time overview at the bottom shows the current extent of $\mathcal{T}$ within the total monitored period for that zone. The time can be easily navigated by interacting with the control panel, with the time overview, or simply by scrolling with the scroll wheel or by using the arrow keys on the matrix. By clicking on a cell it is possible to obtain different types of correlated measurement results. This feature is very effective during issue analysis. The reader can experiment with the current version of DNSMON by visiting the address https://dnsmon.ripe.net.

III. Discovering and Analyzing Abnormalities

The visualization makes it easy for the user to spot abnormal situations and, once discovered, to interactively provide access to extra information helpful for analysis and diagnosis. The first stage of the discovery is usually to look at the matrix and attempt to discern visual patterns in the coloring of the cells. Various patterns provide clues to the presence and nature of an abnormality. For instance, the existence of one or more horizontal red lines in the zone view may answer the question of whether there is a problem that involves a specific name server — if all but one row is green then we can probably determine that there is. Similarly, in the server view, red cells spanning multiple anchors in the same country may tell us that a problem is bound to a particular region, e.g., when a routing problem involving a particular internet exchange affects nearby anchors. When there is a problematic row corresponding to a single anchor, it is essential to understand whether that is a hint of a genuine network problem, or whether the anchor itself is malfunctioning. In this case, the user can opt to see the recent results from that anchor for every name server, as shown in Figure 2(d). If similar problems are seen for other name servers then it is likely that the anchor has some general issue. Also interesting are the edges before and after a visible period of disruption, as they may provide clues as to the cause of the problem and subsequent recovery. It is useful to request extra information for the cells here — in particular, comparing traceroutes side-by-side can show information about changes in packet routing behavior. Looking at the results of approximately concurrent HOSTNAME.BIND queries indicates which specific instance of a load-balanced or anycast service is answering DNS queries. The details for the SOA queries themselves allow more in-depth analysis, e.g., it may be possible to guess that a response has passed through an intermediate cache by looking for simplifications or other forms of answer mangling.

Operators and researches can use clues in the visualization and correlated data to determine the likely root causes, and share specific views that describe the exact details.

We present the following examples of real network events, which were spotted and analyzed thanks to our tool. Technical details and names have been intentionally removed from two of the examples to obscure identities of the affected services.

In our first case, depicted in Figure 2(a), a zone was involved in a major outage. The situation was immediately visible in DNSMON, where none of the anchors were successful in executing the measurements. The start and end times of the red areas are aligned perfectly across topologically and geographically distinct anchors, which points out that all the servers were affected at the same time suggesting a global service cause. Notice the orange borders around the main red area are due to the data aggregation, where positive and negative responses fall within the same sub-interval. Further zooming in revealed the exact start and end times of the event, with a sharp transition between green cells, without packet loss, to red cells, with 100% packet loss.

In our next example, shown in Figure 2(b), the visualized matrix for a zone showed sparse failed measurement from all locations. The zone was involved in a denial-of-service attack and the overloaded name servers caused sporadic packet loss. In this type of cases, our tool provides a ready-to-use, close to real-time global overview of the importance of the attack that can help to guide decisions.

The third example involved the "fr." zone on 15 March 2014. Around 01:30 UTC a high packet loss rate pattern started appearing on DNSMON as depicted in Figure 2(c). Some of the anchors were no longer able to reach the "d.ext.nic.fr" name server because its anycast instance located in Amsterdam was out of order. Our tool played a key role in spotting the issue early on, reducing the reaction time. As visible on the matrix, the instance was quickly restored and the anchors were able to reach again the name server a few hours later the same day. It is conceivable that in other scenarios the problem could be caused by BGP routing problems — where some of the paths reaching the target end up in "routing black holes" — and all the vantage points which end up on these paths share a similar fate. In both scenarios, a comparison of the traceroutes may help to identify where the disruption is located.

TABLE I. MEASUREMENTS USED BY DNSMON, INCLUDING DETAILS OF DNS QUERY OPTIONS AND FREQUENCY.

Type	Protocol	Additional Options	Frequency
CH TXT HOSTNAME.BIND	UDP	No Retries	240s
CH TXT VERSION.BIND	UDP	NSID, IPv4 UDP Payload 1472 bytes, IPv6 UDP Payload 1232 bytes, No Retries	86400s
IN SOA	UDP	NSID, IPv4 UDP Payload 1472 bytes, IPv6 UDP Payload 1232 bytes, No Retries	300s
IN SOA	TCP	No Retries	300s
Traceroute	ICMP		300s

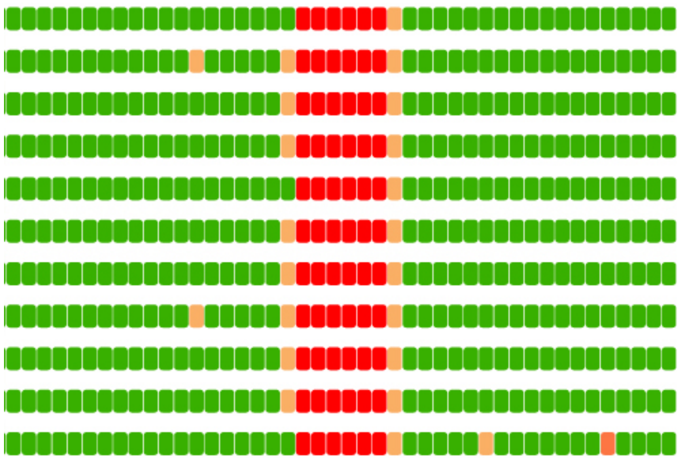
IV. DATA COLLECTION AND AGGREGATION

DNSMON uses the RIPE Atlas measurement network as its data source — in particular, measurement results from RIPE Atlas anchors. The exact set of anchors [17] used in DNSMON is chosen to have as much network and geographical coverage as possible. Currently, anchors are present in all the inhabited continents, especially Europe and the United States, and improving diversity of coverage is a main goal of the project. Each anchor runs the same set of measurements towards the same targets and periodically reports the results back to the RIPE Atlas infrastructure. These results are forwarded and stored in an Apache HBase [18] database that is hosted on an Apache Hadoop [19] cluster. This combination gives us the computational power that we need to process, aggregate and analyze a lot of data, as well as the high availability and good performance required for serving data to interactive clients.

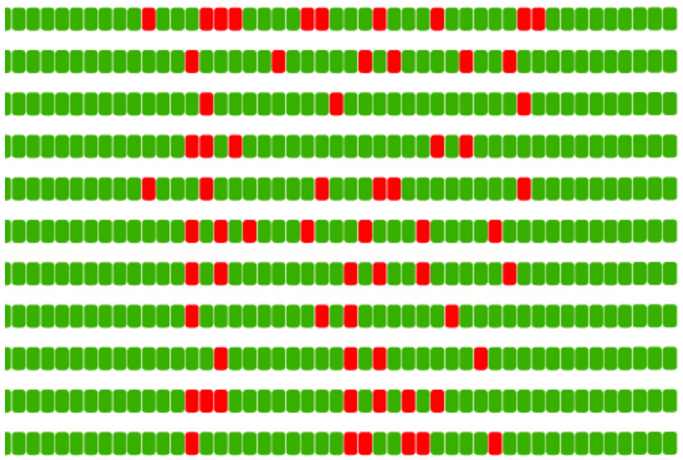
Whenever we want to start monitoring a new zone, we follow these steps: 1) we get the current NS records for this zone; 2) for each record we resolve all available IPv4 and IPv6 addresses; and 3) we start a predefined set of periodic measurements against each address.

The system checks regularly for any changes to the set of the name servers for each zone and, after a manual check, updates the periodic measurements.

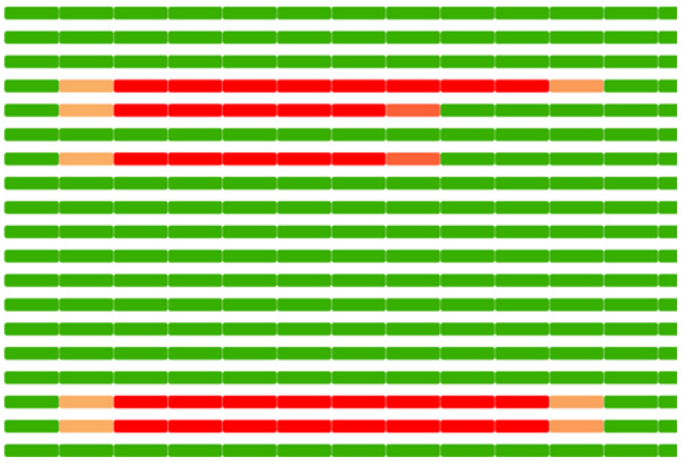
RIPE Atlas exposes a range of options per measurement type. The set of measurements and options used by DNSMON is shown in Table I. HOSTNAME.BIND [20] queries reveal which instance of the target service is responding, and are carried out frequently to make it easier to pinpoint routing changes. VERSION.BIND [21] queries reveal the software version of DNS servers, which is unlikely to change often, so they are only carried out once per day. These names are used because of their widespread adoption, but for servers that do not support them, ID.SERVER [20] and VERSION.SERVER would be used instead. The core measurements used by DNSMON are UDP and TCP SOA queries, and are used as the basis for the response times and loss rates in the visualization. The VERSION.BIND and the UDP SOA measurements are configured with the NSID [22] option enabled, which will prompt some servers to include an instance identifier. This may cause the response to be bigger, so we set a UDP response payload size in order to avoid fragmentation issues: 1472 bytes for IPv4 to fit a common MTU of 1500 bytes, minus 28 bytes for the IPv4 and UDP headers; and 1232 bytes for IPv6, to fit an MTU of 1280 bytes, minus 48 bytes for the IPv6 and UDP headers. We perform traceroute measurements mainly



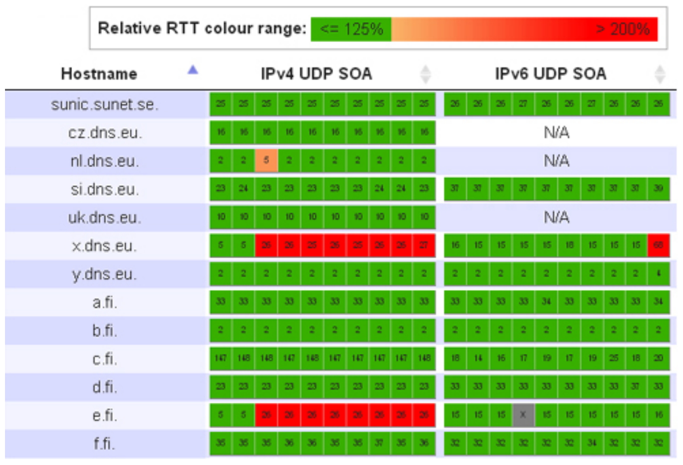
(a)



(b)



(c)



Relative RTT colour range: <= 125% > 200%

Hostname	IPv4 UDP SOA										IPv6 UDP SOA									
sunic.sunet.se.	25	25	25	25	25	25	25	25	25	25	26	26	26	27	26	26	27	26	26	26
cz.dns.eu.	16	16	16	16	16	16	16	16	16	16	N/A									
nl.dns.eu.	2	2	5	2	2	2	2	2	2	2	N/A									
si.dns.eu.	23	24	23	23	23	23	23	24	24	23	37	37	37	37	37	37	37	37	37	39
uk.dns.eu.	10	10	10	10	10	10	10	10	10	10	N/A									
x.dns.eu.	5	5	26	26	25	26	25	26	26	27	16	15	15	15	15	18	15	15	15	68
y.dns.eu.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4
a.fi.	33	33	33	33	33	33	33	33	33	33	33	33	33	33	34	33	33	33	33	34
b.fi.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
c.fi.	147	148	148	147	148	147	147	147	147	148	18	14	16	17	19	17	19	25	18	20
d.fi.	23	23	23	23	23	23	23	23	23	23	33	33	33	33	33	33	33	33	37	33
e.fi.	5	5	26	26	26	26	26	26	26	26	15	15	15	X	15	15	15	15	15	16
f.fi.	35	35	35	36	36	35	35	37	35	36	32	32	32	32	32	32	34	32	32	32

(d)

Figure 2. Examples of problem analysis. In (a), (b) and (c) the main matrix is represented, with x-axis (time) and y-axis (anchors) removed to obscure identities. (a) An outage where all anchors had severe problems reaching the server, causing a thick vertical band. (b) Partial denial-of-service attack, shown as a scattered vertical band. (c) Outage affecting a subset of anchors, with solid horizontal rows. (d) Recent measurements by an anchor.

for investigative purposes as outlined in Section III. At the time of writing this paper, DNSMON monitors 44 zones, by measuring in total 268 servers from 47 anchors worldwide. We have created 2,773 periodic measurements in total, and we store and analyze around 714,000 results daily.

Once collected, the DNS raw results [23] are processed into a streamlined format, containing only the fields necessary for the visualization. The format contains: IDs for the anchor and periodic measurement; the measurement time; an RCODE [24], which may be an error code; and the round-trip time of the query, if completed. Built upon this format are aggregations over 10 minutes, 2 hours and 1 day, i.e., spaced by a factor of twelve. The decision of which levels to use was driven by the requirements of the visualization. There are two types of aggregation (by name server, and by anchor and name server), both of which contain: the number of queries in the period; the number of responses by RCODE, to allow the visualization to distinguish error responses; and the 5th, 50th (median) and 95th percentile of the RTT of all queries that received a reply. In general, aggregations are calculated and stored permanently in HBase tables after a grace period to allow all results to arrive. In order to enable presentation before this point, provisional aggregations are generated on demand and temporarily cached.

V. Implementation and Technical Challenges

In addition to the data collection and aggregation infrastructure, the implementation of DNSMON is split into two other main aspects: a visualization front end and a data API.

The *visualization front end* is a self-contained Web application which can be embedded in any HTML page. It allows the user to view and navigate the interactive matrix. The main interface is presented in Figure 1. It is composed of four main elements: the control bar, the matrix canvas, the data correlation panel, and the time overview bar. We detail their functionality below.

The *control bar* is a toolbar located in the upper part of the interface containing a set of controllers. The first two components from the left are the point of view selector and the colors legend. By changing the point of view, both the matrix and the legend update accordingly. The values for the color thresholds $c1, c2$ specific for the selected point of view are represented in the legend. They can easily be tuned by means of a slider bar appearing by clicking on the legend's labels or on the appropriate button; the change is reflected in real time in the color of the cells. The personalized thresholds are stored locally in the browser to be reused on the next access. The buttons on the right side of the control panel are mainly focused on the time and data navigation, replicating all the gestures accepted by the canvas. The button with the funnel icon allows the user to specify filters on the data, e.g., to exclude: answers containing DNS errors, UDP or TCP, or an IP protocol version. One of the main uses of DNSMON is the continuous monitoring of a zone, for which the visualization must be constantly updated with the latest measurement results without any interaction, e.g., on a wall-mounted monitor where it is essential to be able to see a current overview of a zone a glance. This objective can be achieved by using the *full screen* and *auto update* features accessible from buttons on the control panel. The auto update function keeps the amount of time monitored constant, as well as the data resolution. Newly collected values are introduced as cells on the right side of the matrix, while the old cells are smoothly shifted to the left.

The *matrix canvas* is the main element of the interface. It displays the interactive matrix requested by the user. As described in Section II, the matrix has two axes. The x-axis always represents the selected time interval $\mathcal{T}$, while the y-axis can represent name servers or anchors. All the labels on the y-axis are interactive. Hovering over a label with the mouse provides the user with extra information about the element. A particular name server s can be specified by clicking on a server label, the matrix will automatically switch to the new representation. Alternatively, by clicking on an anchor label, the related anchor page can be opened. In addition to the zoom and shift functionalities, the user can *select a subset of the cells* in order to increase the data resolution on a sub-matrix. We paid particular attention to providing the user with feedback during interaction. While selecting, in addition to the usual selection rectangle, the selected cells are colored in a blue gradient resembling the original tonalities; this gives the user a precise perception of which cells are involved in the selection while keeping the same visual pattern in the background. The transition between two statuses of the matrix is animated.

The *data correlation panel* is hidden by default, appearing when the user clicks on a cell. It provides access to the raw data and to correlated measurement results. In the upper part of the panel, a series of links allow the user to download JSON files from the data API containing the HOSTNAME.BIND, UDP SOA, TCP SOA, traceroute, and VERSION.BIND measurement results collected in the sub-interval represented by the cell. In the lower part of the panel, tabs provide access to: a human-readable DNS response; HOSTNAME.BIND answers collected just before and after the selected cell; and traceroutes collected just before and after the selected cell.

The presentation of the traceroutes is designed to facilitate the kind of analysis shown in Section III by allowing side-by-side comparison, and visually differentiating each line. Traceroute output is enriched with links to get information about IP addresses from RIPEstat [25].

The *time overview bar* is a crucial part of the system, placed at the bottom of the interface. It is a timeline including a resizable range slider that allows the user to select a temporal interval of interest. When the user moves the range slider, the matrix is animated accordingly, downloading or filtering out measurement results. In order to make the interaction easier, a snap to grid is placed along the timeline. With this solution it is easy to precisely select multiple whole days: the selection slider is rounded to the closest tick, accompanied by an animation simulating a magnetic effect. All the components of the system front end are constantly synchronized, providing a consistent perception of time to the user.

The front end is written in JavaScript and HTML. The whole visualization is rendered in the browser using D3.js [26], a Scalable Vector Graphics library. The architecture is based on the Model-View-Controller design pattern [27], while a data layer provides a unified access point to different datasets. The data layer is internally composed of specialized sublayers providing fundamental functionalities, such as format isolation, caching, error handling, and connection management,

The development of DNSMON followed a rigid series of tests. In addition to unit tests, we used virtual machines to test how the web application behaved on different operating system and browser combinations. We had an internal beta stage during which we deployed a *data collection system* gathering usage data, performance scores and errors from the clients. This allowed us to study the users' interaction. The results of our study, together with the introduced improvements, are reported as follows.

1) Usually the user moves in time smoothly; after each interaction only a small portion of the matrix represents new cells. The same applies when the auto update function is active. We designed a client-side cache to minimize redundant calls to the data API, selecting which portion of the matrix to retrieve and combining the results with ones from the cache. Analyzing this solution using the performance logging system, we found that we were able to save up to 88% in network bandwidth.
2) The user is often not decisive during mouse gestures or input selection, trying to correct them incrementally, resulting in useless data calls and redraws. We introduced a layer able to temporarily prevent data calls and to give fake user feedback during the interaction. This gives the user the impression that the gestures are applied in real time, while the redraw with the correct data is done when an anti-flood timer expires.
3) We noticed high memory usage during heavily interactive sections due to the large amount of instantiated cells not yet garbage collected, which prompted us to opt for an ad hoc management of the browser's garbage collector in combination with an Object Pool solution [28].

Due to the large amount of data to display, performance and user experience were two big challenges during the implementation of DNSMON. Different browsers — and different versions of the same browser — are able to handle different numbers of DOM elements, which is why we paid particular attention to always providing the visualization with the right data resolution in order to not overload the browser with an excessive number of data points. With the same test environment described above, we established the number of data points β that the major browsers were able to display in our tool without adversely affecting the user experience and rendering time. In addition to this value, the visualization autonomously computes the maximum number of cells ρ displayable on each row, based on the total number of rows available in the scene, the minimum width possible for a cell, and β. The widget communicates this value to the data API, which provides the best data resolution covering $\mathcal{T}$ with at most ρ data points per row. However, the animations and the real-time feedback are much more expensive than the drawing of the cells. For this reason we introduced a *low-profile mode* able to limit the quality and number of the visual effects while keeping the usability intact. The web application is able to automatically switch to low-profile mode based on the browser version and the actual computational load.

The *data API* is a JSON REST interface. The client provides domain-specific parameters (*zone*, *target*, *TCP/UDP*), which the server translates to periodic measurement IDs and returns aggregated, native or raw data. A major design principle for the API is that it requires minimal knowledge and input from the client. For instance, no a priori knowledge of available resolutions is required for querying, meaning that no client-side changes are required if a new aggregation level is added at a later date, e.g., in order to represent a decade of data. The current levels are listed in the output so that the client can predict a change of resolution when zooming. The client can omit an explicit end time to get the latest data, and possibly specify a length of time instead of an explicit start. The API makes use of a server-side cache. The cached periods are fine grained so as to avoid redundant, overlapping back end queries and cache entries.

VI. Conclusions and Future Work

In this paper, we described a network visualization methodology for monitoring and analyzing the availability and performance of DNS zones worldwide. The incremental information enrichment provided by our graphical metaphor enables the operator to easily move from an overview of the overall quality of a zone or server, to the resolution of single DNS queries, while making it easy to correlate signals from different types of network measurements, e.g., traceroutes. The data correlation occurs in a continuous and persistent way, describing in detail the evolution of a DNS quality issue by providing both real-time and historic measurement results that would otherwise need to be manually collected separately. The flexibility of our visualization, including the ability to switch between different points of view of the same scenario, supports, with a unified interface, the investigation of various issues involving different types of quality degradation.

We demonstrated the effectiveness of this tool at solving real DNS operational problems, by describing strategies and examples in Section III. As future work, we plan to introduce automatic pattern recognition to visually highlight correlations across different targets or vantage points in order to identify a common cause of a hidden problem. Further investigation is warranted into more comprehensive visualization of anycast and load-balanced instances, correlation with routing data, and the extension of the service to any user-defined zone.

References

[1] P. V. Mockapetris, "Domain names - implementation and specification," IETF, RFC 1035, 1987.

[2] J. Postel, "Domain name system structure and delegation," IETF, RFC 1591, 1994.

[3] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, "A case for comprehensive dnssec monitoring and analysis tools," Presented at SATIN 2011, UK, http://conferences.npl.co.uk/satin/papers/satin2011-Deccio.pdf, [retrieved: 05, 2015].

[4] P. Ren, J. Kristoff, and B. Gooch, "Visualizing dns traffic," in Proceedings of the 3rd international workshop on Visualization for computer security. ACM, 2006, pp. 23–30.

[5] SRC Cyber, "DNSential Domain Name Service Visualization Tool," http://www.srccyber.com/pdf/C07-050614_DNSentinel.pdf, [retrieved: 05, 2015].

[6] H. Yu, X. Dai, T. Baxley, and J. Xu, "A real-time interactive visualization system for dns amplification attack challenges," in Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (Icis 2008), ser. ICIS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 55–60.

[7] G. Di Battista, C. Squarcella, and W. Nagele, "How to visualize the k-root name server," in Graph Drawing. Springer, 2012, pp. 191–202.

[8] RIPE NCC, "DNSMON," http://dnsmon.ripe.net/, [retrieved: 05, 2015].

[9] ——, "Test Traffic Measurement Service," http://www.ripe.net/data-tools/projects/archive/ttm/, [retrieved: 05, 2015].

[10] T. Oetiker, "Round-robin database," http://oss.oetiker.ch/rrdtool/, [retrieved: 05, 2015].

[11] RIPE NCC, "RIPE Atlas," http://atlas.ripe.net/, [retrieved: 05, 2015].

[12] S. Steinarsson, "Downsampling time series for visual representation," Master's thesis, Faculty of Computer Science, University of Iceland, 2013.

[13] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualization." Proc. IEEE Symposium on Visual Languages '96, 1996, pp. 336–343.

[14] G. Kistner, "Generating visually distinct colors," http://phrogz.net/css/distinct-colors.html, [retrieved: 05, 2015].

[15] International Organization for Standardization, "ISO 3166-1 alpha-2," https://www.iso.org/obp/ui/, [retrieved: 05, 2015].

[16] M. Hemphill, "A note on adults' color–emotion associations," The Journal of genetic psychology, vol. 157, no. 3, 1996, pp. 275–280.

[17] RIPE NCC. RIPE Atlas anchors used by DNSMON. https://atlas.ripe.net/dnsmon/probes. [retrieved: 05, 2015].

[18] A. HBase, "The apache hadoop project," http://hbase.apache.org/, [retrieved: 05, 2015].

[19] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010, pp. 1–10.

[20] S. Woolf and D. Conrad, "Requirements for a Mechanism Identifying a Name Server Instance," RFC 4892 (Informational), Internet Engineering Task Force, Jun. 2007.

[21] I. S. Consortium et al., Bind 9 administrator reference manual. Bind, 2005.

[22] R. Austein, "DNS Name Server Identifier (NSID) Option," RFC 5001 (Proposed Standard), Internet Engineering Task Force, Aug. 2007.

[23] RIPE NCC, "RIPE Atlas - raw data structure documentation," https://atlas.ripe.net/docs/data_struct/#v4610_dns, [retrieved: 05, 2015].

[24] Eastlake, et al., "RFC 2929. Domain Name System (DNS) IANA Considerations," http://tools.ietf.org/html/rfc2929, [retrieved: 05, 2015].

[25] RIPE NCC, "RIPEstat," https://stat.ripe.net/, [retrieved: 05, 2015].

[26] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," Visualization and Computer Graphics, IEEE Transactions on, vol. 17, no. 12, 2011, pp. 2301–2309.

[27] G. E. Krasner, S. T. Pope et al., "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," Journal of object oriented programming, vol. 1, no. 3, 1988, pp. 26–49.

[28] M. Kircher and P. Jain, "Pooling pattern," Proceedings of EuroPlop 2002, 2002, pp. 497–510.