The Internet Protocol Journal

September 2015 Volume 18, Number 3

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at:
www.protocoljournal.org

ISSN 1944-1134

From The Editor

Just over a year ago we relaunched *The Internet Protocol Journal* (IPJ) under a new funding model. Since that time we have published 5 issues of IPJ, designed and implemented a new subscription system, built a website, established new relationships with authors and contractors, and, most importantly, rebuilt our subscriber base with both previous and new subscribers. With some 15,000 print subscribers and 6,600 e-mail subscribers, we are recreating that all-important *community* that makes this publication unique. We've already received many messages of support and appreciation as well as suggestions for topics and article submissions. Please keep them coming!

None of the work behind IPJ would be possible without the support of numerous individuals and organizations. If you or your company would like to sponsor IPJ, please contact us for further details. Our website at **protocoljournal.org** contains all back issues, subscription information, a list of current sponsors, and much more.

The general topic of *Internet of Things* (IoT) has received much attention in recent years. One way to explain IoT is to describe it as a large collection of Internet-aware *sensors*—such as the temperature sensor in a thermostat, and associated *actuators*—such as the electronic switch that turns your heating or air conditioning unit on or off. However, Internet-aware sensors can also be used to measure details about the Internet itself, and this is the idea behind the *RIPE Atlas* project, which is described in our main article in this issue.

Just as I was finishing writing this editorial, the *American Registry for Internet Numbers* (ARIN) issued the final IPv4 addresses in its free pool. If your organization has not yet deployed IPv6, now would be a good time to start the process. You will find many articles and pointers to further information about IPv6 in back issues of IPJ, all available from our website.

Another reminder that if you received a printed copy of this journal in the mail, you should also have received a subscription activation e-mail. If you didn't receive such a message, it may be because we do not have your correct e-mail address on file. To update or renew your subscription, just send a message to **ipj@protocoljournal.org** and include your subscription ID. Your subscription ID is printed on the back of your journal.

—Ole J. Jacobsen, Editor and Publisher
ole@protocoljournal.org

RIPE Atlas: A Global Internet Measurement Network

by RIPE NCC Staff

RIPE Atlas is a global Internet measurement network that provides an understanding of the state of the IP Layer in real time. Here we describe the functionality and design of RIPE Atlas, which collects information about Internet connectivity and reachability via thousands of measurement devices around the world. Then it makes this data available to everyone by measuring the *Internet Protocol* (IP) layer of the Internet with real packets, from anywhere, at any time, by everyone, and for the benefit of all.

- *Real packets:* RIPE Atlas measures the IP layer of the whole Internet—not just the last mile or the network of a single provider. It sends real packets and observes responses so it can directly measure the performance of the IP layer. RIPE Atlas does not focus on measuring applications, which run on top of the IP layer. RIPE Atlas is not an observer of metadata like *Border Gateway Protocol* (BGP) routing traffic; there is no need to make inferences from metadata. And it does not observe any user traffic, thus avoiding many ethical concerns. RIPE Atlas supports five different types of measurements: *ping, traceroute, Domain Name System* (DNS), *Secure Sockets Layer* (SSL), and the *Network Time Protocol* (NTP).
- *From anywhere:* As of July 2015, RIPE Atlas comprises more than 8,400 active vantage points globally. These vantage points are both small hardware devices called probes that volunteers connect to their home or business networks, and *anchors,* which are generally installed in data centres and act as both enhanced probes with more measurement capacity and regional measurement targets within the greater RIPE Atlas network. Currently, more than 8,300 probes and 133 anchors are deployed in 173 countries and in 11% of all IPv6 *Autonomous System Numbers* (ASNs) and 6% of all IPv4 ASNs. Although this coverage is still far from "everywhere," it is quite an extensive network—and the architecture allows for scaling up by another order of magnitude.
- *At any time:* Measurements can be started at any time from a chosen subset of these devices and can be performed quickly, or they can be set up to run for weeks, months, or even years. This scenario allows for both quick glances for troubleshooting purposes and deeper analyses of long-term trends. In order to make results comparable, we carefully control the experimental conditions.
- *By everyone:* RIPE Atlas was conceived to collaboratively build and share a huge measurement infrastructure, rather than building individual small ones for exclusive use. Everyone can contribute to RIPE Atlas by hosting a probe or anchor, by building tools to run measurements or analyse results, by providing sponsorship funding, or by helping us distribute probes to difficult-to-reach locations. Everyone contributing to RIPE Atlas can use the network to run their own measurements from virtually all devices.

- *For all:* RIPE Atlas data is openly available, and everyone can benefit from the tools and analyses that others provide. By default, measurement specifications are open for inspection, and all results are accessible to everyone and can be accessed for many years. This continued accessibility means that anyone can reproduce experiments and analyses using RIPE Atlas data, and that is the only way to do real science; it fosters development and the sharing of tools. Having all data openly available also allows others to reuse existing results. RIPE Atlas data consists of a large number of results from a large number of vantage points to a large number of destinations. The topology of the IP layer is far from flat or fully interconnected; thus, by design, the data contains a lot of information about "core" parts of the IP layer. This information can be used to produce maps about the IP layer topology, and that is why we chose the name, *RIPE Atlas*.

In general, RIPE Atlas can be used to:

- Continuously monitor the reachability of a network or host from thousands of vantage points around the globe
- Investigate and troubleshoot reported network problems by conducting ad hoc connectivity checks
- Test IPv6 connectivity
- Check the responsiveness of DNS infrastructure, such as root name servers
- Execute measurements from a large number of vantage points for use in academic research

We describe specific use cases for RIPE Atlas data in more detail in the "Use Cases" section later in this article.

History and Funding

Before RIPE Atlas, the *Réseaux IP Européens Network Coordination Centre* (RIPE NCC) ran numerous other measurement platforms. RIPE Atlas replaced the older *Test Traffic Measurement Service* (TTM)[1], an active measurement tool geared towards optimising the use of transmission resources that was developed at a time when transmission capacity was much scarcer than it is now. In contrast, RIPE Atlas was developed for an environment where a stable and well-optimised network layer is more critical to the quality of service in the IP layer than squeezing the most out of scarce transmission resources.

The RIPE NCC[2] began developing RIPE Atlas in 2010 to complement its array of measurement and data-gathering tools[3] with membership-supported funding. The RIPE NCC membership continues to fund the bulk of RIPE Atlas operations, expansion, and development. Many external sponsors also support RIPE Atlas financially.

Our focus on the IP layer of the entire Internet is very much in line with the needs of our members, mainly Internet Service Providers, who provide the bulk of the funding. In recent years, RIPE Atlas has been extended to measure numerous core services, such as the DNS, in addition to the IP layer. However, this added measurement does not affect our main focus. We are also committed to keeping all measurement data accessible for as long as possible so that this data can be used for future purposes and so that anything based on it can be independently verified at any time.

Use Cases

Since the beginning of RIPE Atlas, the network has produced interesting data that everyone can use, regardless of whether or not they host a probe or anchor themselves. Over the last few years especially, the increased reach of the network has meant that researchers, operators, and other RIPE Atlas users have used RIPE Atlas data to debug network problems, analyse network behaviour, and conduct their own interesting research. Many useful visualisations have also been developed based on RIPE Atlas data. Here we list just a few examples:

- A group of researchers from Africa measured inter-domain routing to determine the best possible locations to establish Internet exchange points in the region.[4]
- A severe power outage and a variety of network outages were analysed and visualised, highlighting how the Internet routes around outages.[5, 6]
- Engineers from the *Wikimedia Foundation* and the RIPE NCC collaborated on a project to measure the latency of Wikimedia sites and improve performance for users worldwide.[7]
- Recently, the RIPE NCC developed a tool to analyse how much of a country's local traffic actually leaves the country, and the role that Internet exchange points play in keeping traffic local.[8]
- A team of researchers investigated content-blocking incidents in Turkey and Russia, as well as outlining ethical considerations when using measurement networks that involve volunteers as hosts, and giving a comparative overview of RIPE Atlas and other measurement networks.[9]

RIPE Atlas users also produce useful and stunning visualisations that make it easier to grasp the results of RIPE Atlas measurements. For example, one network operator visualised the measurements collected by the local RIPE Atlas anchor, and was able to analyse the quality of the local connectivity and topology changes and help debug network problems.[10] *CartoDB* helped us visualise network outages and other interesting network behaviour based on RIPE Atlas.[11] RIPE Atlas also helps make geolocation data of Internet infrastructure more accurate over time. We started a crowd-sourcing project in which operators can specify the geolocation of servers and other equipment based on RIPE Atlas *traceroute* data.[12]

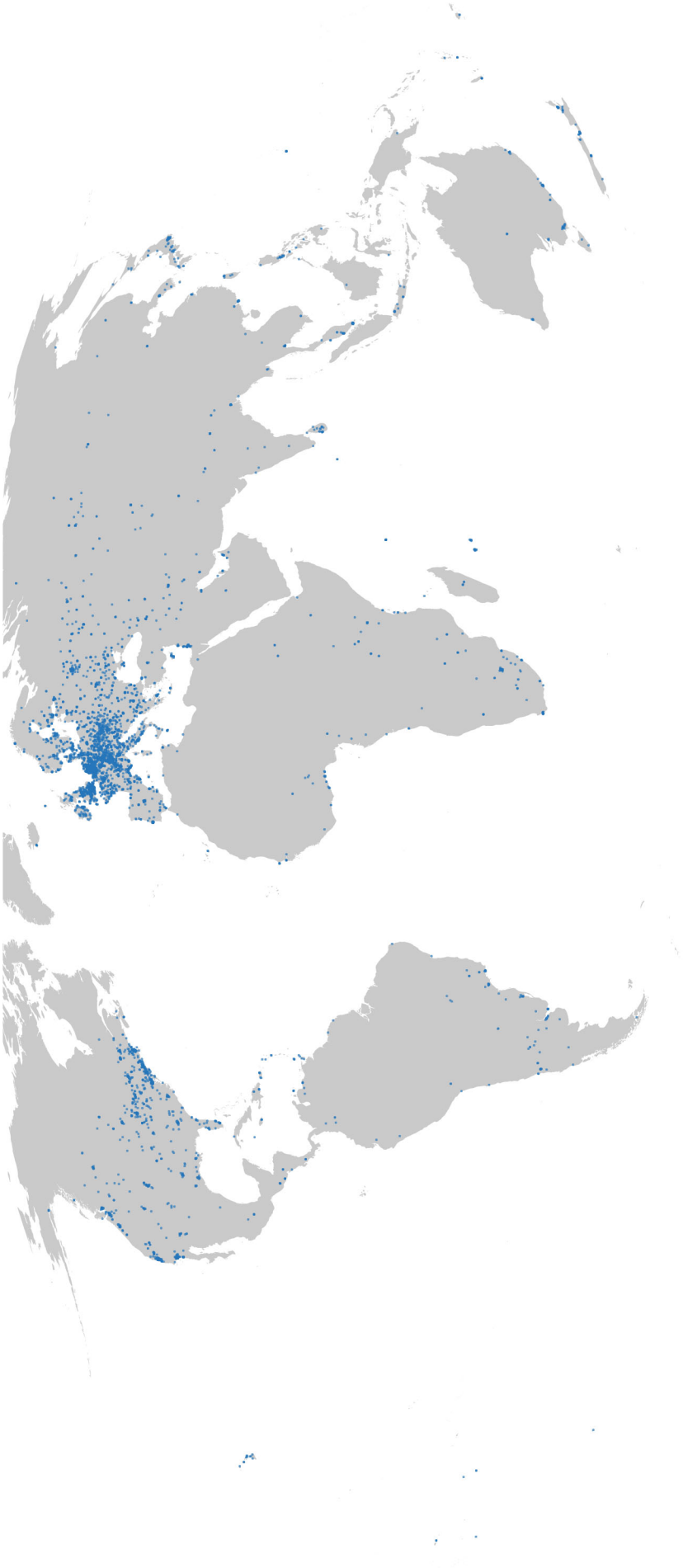


Figure 1: RIPE Atlas Probes Connected on 2015-07-20 at 05:42:26 UTC [23]

The first-ever *RIPE Atlas Hackathon*, held in March 2015, also produced a great variety of visualisations based on RIPE Atlas measurement data, including visualised *traceroute* consistencies over time, a display connecting large-scale probe disconnections to Twitter feeds and weather data, and a map of probe data by country.[13]

Overall Design

We defined a few fundamental principles early on that influenced the system design: our choice to use dedicated hardware devices as vantage points, the ability to scale up, the distributed and collaborative nature of the deployment, and the related security considerations.

Hardware vs. Software Vantage Points

Even though it incurs measurable costs, we chose dedicated hardware devices as vantage points for many reasons.

The hardware devices are "install and forget," and in this sense they are not prone to disappearing after an operating system upgrade or a home router replacement, for example. They are easy to deploy because they act as just another device on the network, and after installation they can run uninterrupted, 24/7. Hardware probes also provide a well-defined environment, so the results are much more comparable than widely varying platforms. All in all, these conditions support the goals of achieving datasets with long time series.

We also prefer to not assume responsibility for code that runs on host-provided systems. Any bug or vulnerability in a hardware probe does not affect or compromise another system. If necessary, a probe can easily be disabled by simply unplugging it.

The current generation of probes is well suited to support the functionality of the system. At the moment, the probes are much more constrained by bandwidth limitations, which the hosts can impose themselves, rather than their CPU, memory, or storage capabilities.

Scalability and Resilience

At the time of RIPE Atlas' conception, about 35,000 *Autonomous Systems* (ASs), were active on the Internet. If we wanted to have vantage points in all of them, account for some of them not being active all the time, and aim for some additional redundancy, we needed to plan for three devices in each network—leading to a rough approximation of 100,000 probes. Of course it's naïve to think we can install a device in every single AS, plus bigger and geographically more distributed networks would require even more probes. But this number is very useful as a potential upper limit, and it served as a good base for the infrastructure design.

Having this scalability goal does not mean that we immediately deployed enough infrastructure components to handle the full load. The system is in constant evolution in terms of supporting the hardware infrastructure, the redundancy and scalability of critical components, and the software components we use.

The scalability goal also pointed out early on the need for deploying controls on how the measurements and probes behave. Even with much lower deployed numbers, it's not reasonable to run all measurements on all probes all the time, so there is a need for controlling the amount, distribution, and scheduling of system resources. This need gave rise to numerous principles and components, like the credit system and the scheduler, which we describe in the next section.

The infrastructure is designed to operate in a distributed fashion. Most components have enough local knowledge to fulfill their role but don't necessarily know about the state of the whole system or even other components. This setup makes it possible to operate most functions even in the case of a network split. For example, probes keep on executing measurements even if they are not connected to the infrastructure, and all components (including probes) buffer results in case the "next hop" in the result processing chain is unavailable. The communication protocols are designed such that they can handle temporary disruptions. In this sense the system is self-healing; assuming that infrastructure problems can eventually be resolved, the probes and the associated processing pipeline will eventually converge on a stable state and all buffered data will be delivered. The probes are also "headless" (they don't have a console interface), and it's essential for the system to be able to support this kind of recovery.

Collaborative Approach and the Ecosystem

No global measurement network can grow beyond a trivial size without relying on collaborators and fostering an interested community around it.

Our approach is relatively simple: we ask network operators and users to help us reach new networks and deploy probes in them. By installing a probe in a network that previously didn't contain any, probe hosts increase the number of vantage points and allow RIPE Atlas to execute measurements from this new network. Beyond contributing to the larger network, probe hosts also enjoy tangible benefits:

- By keeping the probe active, the host accumulates "credits," a form of currency they can use to perform their own customised measurements using the RIPE Atlas network.
- As soon as a probe becomes active, it starts a set of "built-in" measurements that can point out potential local issues with the host's network.
- Probe hosts can use the credits they earn to execute measurements using any probe in the network, with numerous rules that regulate the use of these resources, as described later in this article.

The costs for measurements are determined such that they are proportional to the complexity of the measurement, the amount of traffic generated, and a few other factors that depend on the specific measurement type.

We also impose a daily limit on the number of credits that users can spend, the number of measurements each user can run, and the number of probes they can use in these measurements. The purpose of the credit system and these constraints is to prevent abuse and overload of the system itself or of popular measurement targets. However, a lack of credits should not prevent any reasonable use of RIPE Atlas. Credits can be transferred between users, and we provide additional credits for tool developers and specific measurement campaigns.

Security Considerations

Since the very idea of RIPE Atlas is to send and receive network traffic from vantage points across the globe, security and reliability have been very important from the beginning of the project.

In order to reduce the attack surface of the probes, we decided to make them as closed as possible. They don't offer any services that users or programs can connect to (in the TCP/IP sense). Instead, they make only outgoing connections, both towards the RIPE Atlas infrastructure and for measurements. They also don't share any authentication tokens among each other; each probe has its own key that it uses to authenticate itself to the infrastructure. Of course, since the probes are physically in the hands of hosts, it's virtually impossible to make them resilient to tampering or disassembly, but these actions should not affect other probes in the network.

All communication within the infrastructure (probes included) is done via encrypted, mutually authenticated channels. In the case of the probes, this communication is through a *Secure Shell* (SSH) protocol connection that is also used to channel control traffic and result traffic back and forth, whereas we use *Transport Layer Security* (TLS) for communication between the control components.

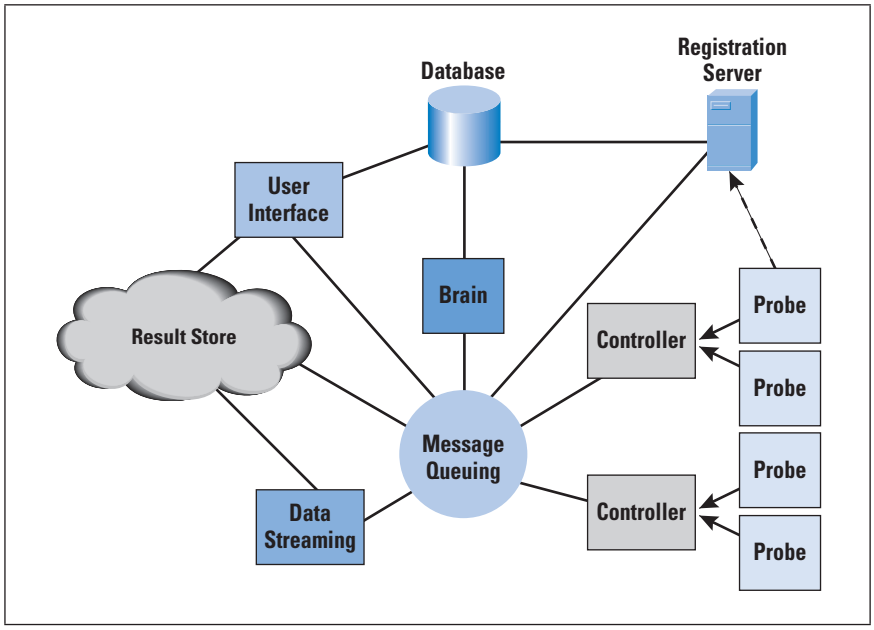
The software running on the probes is field-upgradeable, with the probes being able to verify the authenticity of new firmware via cryptographic signatures. This setup allows us to deploy new functionality as well as security enhancements, so the risk of having a security vulnerability that we are unable to fix is very low.

Of course, no system is 100% resilient against attacks. We run a custom firmware, which may have its flaws, on baseline operating systems (*CentOS, OpenWrt,* and *ucLinux*). We believe the current security mechanisms provide adequate protection against casual attackers, and the potential for more serious disruptions to the network is limited to sufficiently resourceful attackers, making it quite low.

Architecture and Infrastructure

The RIPE Atlas architecture is designed with the previously mentioned key principles in mind: scalability, resilience, and so on. The general structure is shown in Figure 2.

Figure 2: The Overall System Structure



A central *Structured Query Language* (SQL) system database contains a lot of the key information: almost all of the information about the probes and their various properties; metadata about measurements; users; credits; etc.

A cluster of *message queue servers* acts as the central nervous system for the architecture. It provides connectivity between the various components, and ensures a one-time, guaranteed message delivery with very small delays. It also eliminates the need for each component to maintain knowledge about which other components are present, whether they are connected, etc. This cluster has enough capacity to both deal with control messages and pass on result data, though at a later point we may split these two tasks or change to a more direct data-reporting channel.

Whenever a probe starts up, it first connects to one of many preconfigured *registration servers*. These servers allow only connections from known probes using their specific public keys. When a probe connects, the servers determine which controller the probe should be directed to based on the location of the probe, the availability of controllers, and other factors. The servers then redirect the probe to the chosen controller and notify this controller about the probe. During this process, they inform the probe and the controller about each other's public keys.

The *controllers* accept probe connections from the list of probes they get from the registration servers; they do not have full knowledge of all probes. Once a probe connects, they keep the channel open to receive results and to notify the probes about measurement tasks. Although theoretically each controller can handle a large number of probes, we're trying not to overuse this capability in order to avoid too much fate-sharing between probes when individual controllers or data centres become disconnected.

The *brains* execute higher-level functions in the system, notably the measurement scheduling. This activity happens based on the requests received via the *User Interface* (UI) and *Application Program Interface* (API), and involves an elaborate set of steps about probe preselection, negotiations with the controllers about probe availability, etc. Brains also act as an interface and filtering service between controllers and the central database in order to propagate probe metadata to the UIs.

The UI takes care of all user interactions. It serves the pages for the web interface and handles the API calls. It also serves data download requests via the API; we will likely separate this activity into a separate component if the number of requests or the amount of data transfers creates a bottleneck.

The data streaming servers provide real-time access to the flow of result data, directly off the result queues, and include a few minutes of "short-term memory" to supply recent results. They can also replay results that have already been stored in the result store.

Finally, the *result store* is a Hadoop/HBase cluster for long-term storage of all results. It also handles periodic calculation of aggregates and various other tasks. The technology used allows for relatively cheap redundant storage of the dataset, as well as a simple way of executing *MapReduce*-type batch operations to extract information from datasets that are otherwise too large to download and analyse.

In addition, there are a number of auxiliary subsystems attached to the infrastructure that perform various functions, such as dealing with DNS ("SOS") requests from probes, managing the administrative tasks, or monitoring and analysing the behaviour of the system in near real time. *RIPEstat*[14] also closely cooperates with the UIs to visualise the stored datasets.

This architecture allows easy expansion if any one component becomes a bottleneck. For example, it's very simple to add more controllers to the system to handle additional probes as needed.

Implementation

Over the years, RIPE Atlas has gone through numerous transitions, shaped both by interactions with the wider community and the challenge of managing so much data. At the start of the project, RIPE Atlas was quite static: result data was restricted to data collected by hosts' own probes, with a handful of visualisations provided by the RIPE Atlas team based on data collected from the (then) small network of a few hundred probes.

Over time, however, we introduced the concept of user-defined measurements, which allowed individual probe hosts to use the growing network to do things we hadn't initially considered. Slowly, and in consultation with the community, the options for these user-defined measurements grew, allowing for more protocols and more options within them, so that hosts can now take advantage of a global network to perform *ping*, *traceroute*, DNS, SSL, and NTP measurements with a plethora of options.

Interfaces

As we expanded the available features, we needed to change the ways in which users interfaced with the system. Where once we had a very data-heavy ExtJS-based website, we built a more user-friendly interface to access a variety of data visualisations. At the same time, we also gave access to our data to those wanting to create their own views by standardising on a *Representational State Transfer* (REST) ful API, streaming API, and result-parsing toolkit.

We also added a few new sections to the website to improve the user experience:

- A *credits* page with a chart to help users understand how many credits they were earning and spending
- A *keys* page to allow users to create API keys that they can use to create new measurements with the RESTful API or share existing result data
- Separate pages for different groups of users, including "ambassadors" (those who help us distribute probes), financial sponsors, and anchor hosts

The largest change was the way in which we display probe and measurement information. The first instances of this change were cluttered, grid-like interfaces, which we replaced with a searchable listing and highly visual renderings of measurement results and probe performance. Many of these new interfaces are powered by the publicly available APIs.

Lastly, we introduced an updated and much more user-friendly interface for creating measurements. Guided by feedback from users of the old ExtJS-based system, measurement creation became a single form with three sections, of which only the first required direct input. Users are left to decide whether they want to accept the defaults for the other sections or be more specific.

This new measurement form was coupled with numerous helper tools, including:

- A credit consumption visualiser that helps users gauge the long-term expense of their proposed measurement
- An elaborate probe selection wizard that allows users to search, filter, and select probes across networks and geographical space: This selector also implements a standard autocomplete that covers ASN, prefix, IP, address family, and current probe status.
- An API example panel that shows the equivalent request as if it were submitted via the API: This panel was designed to serve as a learning tool for those interested in working with the API for future measurement creation.

This new tool, combined with the availability of the RESTful API, made it much easier to create and manage measurements.

Visualisations

In addition to downloading results in *JavaScript Object Notation* (JSON) format or accessing them via the streaming API, users can also benefit from the visualisations we have created for some measurement types. These visualisations are usually made with JavaScript and *Scalable Vector Graphics* (SVG). The back end provides only the data, while the visualisation and the data correlation are done completely client-side and are accessible from a web browser without third-party plug-ins. They can be embedded in any HTML page or shared by means of permalinks. These visualisations include:

- Simple, tabulated layouts of key parameters extracted from measurement results: These layouts can give an immediate impression about how the target behaves according to the measurement.
- Maps of all kinds showing success rates, round-trip times, or measurement results, where each involved probe and its latest result are denoted by colour, shape, or size: These maps are constructed to work with practically all types of measurements that the system supports.
- Simple charts of packet loss and round-trip times for *ping* measurements, also featuring a shift and zoom function to explore past results.
- A "seismograph" that shows results of *ping* measurements from multiple probes: This tool allows users to compare results from multiple probes at the same time, making it really powerful to spot common behaviour across multiple ASNs, countries, or regions. The tool also supports zooming and exploring historical data.
- A complete rewrite of the RIPE NCC tool *DNS Monitoring* "(DNSMON)"[15] to visualise long- and short-term behaviour and reachability of root DNS servers and important top-level DNS domains: In many aspects, the RIPE Atlas version of DNSMON goes well beyond the functionality of its predecessor, for example, by incorporating *traceroute* measurements alongside the DNS queries and offering this information to help users diagnose observed performance changes.
- A tool that shows common paths towards particular targets, based on *traceroute* measurements

There is a lot of further potential in providing more visualisations to our users. In particular, we are working on extracting more information from traceroute measurements. Combined with the ability to zoom in or out using ASN, prefix, or even geographical aggregations, this information can give an easy-to-understand, immediate explanation of where network bottlenecks or dysfunctional network paths are located.

Data Storage

A global measurement network as big as RIPE Atlas has unique requirements for storing and accessing the collected data.

As of July 2015, about 8,000 concurrently running measurements produce more than 2,700 results per second. Depending on the measurement type, the size of the result can vary from a few bytes to kilobytes. All measurements ever run since the start of RIPE Atlas have been kept. Hence it is not a surprise that the net size of all measurements until this point has grown to 24 TB. On top of this amount is another 4 TB worth of derived data, resulting in a total of 28 TB of active Internet measurement data. Keeping this data online, indexed, and accessible in real time is a challenge on its own.

Storage Solution

After a period of extensive testing and comparing available storage solutions, we decided on *Apache Hadoop,* because it seemed to be scalable and reliable. In addition, at the time of our decision in 2011, it was one of the few open-source solutions capable of dealing with data on a terabyte level. Related to the open-source nature of Hadoop was a choice of different distributions, and we selected the one created by Cloudera (CDH).

Hadoop, maintained as an Apache project, is a software framework that allows for the distributed processing of large datasets. Its design makes it possible to scale from a single server to thousands of machines, with each machine offering local computation and storage. High availability and data security are built into the library itself, so we didn't have to depend on expensive hardware and we could use inexpensive commodity hardware instead.

The Hadoop distributed file system allows for high-throughput access to the application data. Another key component is *MapReduce,* which is a simple programming model that breaks up large datasets and splits data-processing tasks so they can be run in parallel on multiple nodes in a cluster. On top of this stack we use *HBase,* another part of the Hadoop ecosystem, which provides a structured, table-like storage model for large, distributed datasets.

What we described earlier as the "result store" currently consists of one development cluster and two production clusters, one live and one standby. Each production cluster has 119 machines providing a total of 3.5 TB of memory and 952 CPUs. Of the 119 machines, 110 act as workers on which data is stored and jobs are executed. The other 9 machines schedule jobs and manage cluster resources. Each worker can store approximately 4 TB of data, providing a total of 400 TB of storage capacity (after formatting).

Each production cluster is roughly at two-thirds of its capacity, accounting for 260 TB. The cluster also stores other datasets, such as *Border Gateway Protocol* (BGP) routing data collected by *Routing Information Service* (RIS)[16]. Nevertheless, the storage requirements for RIPE Atlas alone are significant because, in order to achieve reliable storage, Hadoop uses replication, which blows up the requirements for RIPE Atlas data to 75 TB using a replication factor of three.

It might be surprising that we run two production clusters, given that Hadoop has reliability built in, so one cluster might seem to be sufficient. In fact, only by using two clusters were we able to test new features and platform updates without running the risk of long service interruptions in case something went wrong. As with any software, Hadoop is not bug-free, and we especially noticed this reality when we used its first releases just as "big data" was becoming a buzzword. As it moves toward a more stable system, it might be an option in the future to combine the two clusters, resulting in double the current capacity.

Data Flow

From the message queue servers, the measurement data, in the form of *Advanced Message Queuing Protocol* (AMQP) messages, is loaded by daemons running on all worker nodes. The payload, in JSON format, is extracted and stored directly in HBase tables and saved as sequence files. Sequence files are a binary format of the collected data, which act as an efficient input for MapReduce jobs that create tables derived from raw data.

Data derivation is specifically necessary because Hadoop, like many other NoSQL solutions, is a key-value store and hence lacks elaborate concepts of data projection and selection. The key, which is the only available index, becomes very important, and different access use cases require different key layouts, resulting in data duplication.

An often-used derivation is time-based aggregation, which compacts data collected over a long period of time to a fraction of its original size. By extracting statistically relevant values like minimum, maximum, and median, users (via the UI) are still able to get an overview of the measurement result without having to download all of the measurement data.

Initially orchestrated by *cron* jobs, the execution of MapReduce jobs is now done with a software component called *Azkaban*. Azkaban can account for dependencies between datasets and—most importantly—data availability, and the result is much better data quality for the derived datasets.

Next to regularly running jobs, the system also supports ad hoc data processing, which enabled our data scientists to crunch volumes of data that would have never fit on anyone's workstation.

Datasets

RIPE Atlas data consists of multiple tables. From the size value, one can see the difference between raw and aggregated tables, highlighting how aggregation saves a lot of space and makes overviews much faster.

- *Result blobs:* all messages fetched from the message queue servers; this data is the rawest form accessible as a table and acts as the input for most other tables (24 TB)

- *Latest results:* the latest measurement result per probe and measurement (50 GB)
- *Counters:* information about the amount of results delivered per probe and measurement, also used for credit billing (4 GB)
- *DNS details and aggregates:* all DNS measurements and their aggregates, also the ones used for DNSMON (800 + 200 GB)
- *Ping results and aggregates:* contains all *ping* measurements and all its aggregates used for *ping* visualisations (3.5 TB + 140 GB)
- *Traffic:* data on how much traffic a probe produced, upstream and downstream (12 GB)

The front-end servers (RESTful) gain access to the data via Apache *Thrift*. Thrift is a software framework that allowed us to build seamlessly integrated data types and service interfaces between the Hadoop Java environment and the front-end Python environment.

Measurements

RIPE Atlas performs two distinct sets of measurements. The first category is the "built-in measurements," performed by all probes as soon as they are connected; it provides a baseline flow of generally useful results. The second category is the "user-defined measurements" that users specify themselves, which usually run on only a small subset of the probes.

Built-In Measurements

When a probe is plugged in, it initiates connection to the RIPE Atlas infrastructure and is connected to a controller, which sends a list of predefined, built-in measurements. These measurements are designed to provide useful data to the host for monitoring their connectivity and discovering potential local issues, but they also provide a wide array of generally useful results for network visualisations.

Built-in measurements include:

- *Pings* to first and second hops
- *Pings* and *traceroutes* to well-known destinations, such as DNS root servers and RIPE Atlas anchors
- DNS SOA, `version.bind`, `hostname.bind`, `id.server`, and `version.server` monitoring on the DNS root servers
- HTTP and SSL requests to some of the RIPE NCC servers

Built-in measurements are performed via both IPv4 and IPv6 if a probe seems capable of supporting both protocols. The measurements are ongoing, with most of them running every few minutes. The list of running built-in measurements is defined solely by the RIPE Atlas team; probe hosts have no influence over it. Data for the built-in measurements is publicly available.

User-Defined Measurements

The distinct feature that makes RIPE Atlas unique and provides value for all participants is its ability to let users schedule their own measurements using virtually all of the probes throughout the network with great flexibility. Users are free to choose:

- Measurement type (from the supported set)
- Type-specific options such as flags and parameters
- IP version
- Measurement target (for example, hostnames or IP addresses)
- Measurement source (that is, vantage points); this source is the set of probes defined by size and desired properties, such as geography, ASNs, prefixes, and tags. It is also possible to reuse the same sets of probes that were used in previous measurements.
- Measurement start/end time and the frequency of ongoing measurements

Each measurement can be ongoing (with a user-defined frequency) or a one-off measurement. One-off measurements are more expensive to handle, but they have a much faster reaction time, delivering results in a matter of seconds.

User-defined measurements can be created and maintained using the web interface, as well as the RESTful API. Users can stop measurements prematurely if they wish, and can also change the set of involved probes.

User-defined measurements are the most resource-consuming activity within RIPE Atlas, and we therefore need to balance the system capacity with our users' needs. To achieve this balance, we constantly increase the number of probes, throughput of the measurement result delivery, and storage. The credit system also plays a large role. It encourages users to keep their probes connected, so they earn credits to use on user-defined measurements, and to use the system with care. The more resource-intensive a measurement (number of probes, probe CPU, network activity), the more expensive it is.

Current Measurement Types

At the time of writing, RIPE Atlas supports the following measurement types:

- *Ping:* monitors network delays using *Internet Control Message Protocol* (ICMP) IP echo messages
- *Traceroute:* displays the route path and measuring transit delays
- *DNS:* queries Domain Name System servers or resolvers (provides users with similar functionality of well-known tools such as *dig* and *nslookup*)
- *SSL:* queries SSL/TLS certificates of remote servers
- *NTP:* queries Network Time Protocol servers, dumps the reply, and computes some statistics, such as reply time

Every measurement type allows the user to specify parameters for fine-tuning. For example, *ping* allows the user to change the number and size of packets, while DNS has a much wider set of options, allowing users, for example, to set query parameters, recursion, number of retries, and so on.

Future Measurement Types

We constantly work with the RIPE Atlas community to satisfy our users' demands for different measurement functionality. We have started to add new measurement types and continue to add new options for fine-tuning them, as long as they are consistent with the RIPE Atlas mission. The following measurements are likely to be introduced in the near future:

- *HTTP:* queries HTTP servers. The system is technically already capable of doing this task; it is used internally to deliver results, but it's not yet available for public use. Public availability will most likely be restricted to allowing users to perform GET and HEAD requests against a predefined list of targets (initially RIPE Atlas anchors).
- *WiFi:* an intentionally opt-in-only feature to verify the functionality of *Wireless LAN* (WLAN) access points: The intention is to check whether it's possible to connect to a requested, specific *Service Set Identifier* (SSID), and measure the IP connectivity of this interface. We do not intend to use WLAN connection for IP connectivity of the probe itself, and we don't plan to allow passive WiFi scans.

Day-to-Day Development

RIPE Atlas is developed by a team of software engineers with overlapping focuses. Although we don't follow a particular methodology, RIPE Atlas development is agile in the sense of iteratively delivering and improving upon working features, responding to feedback from the community, and continuously focusing on good design and architecture. Development is modular, with each type of component having its own *Git* repository, automated deployment mechanism, and release schedule. The frequency of these release cycles varies according to development intensity. For example, the UI has the highest rate of change and so has a new release almost every week—with larger stretches for fundamental changes that possibly include bug fixes and hot fixes in the interim—while the brain may go weeks between new features and releases.

In addition to the main RIPE Atlas system, two internal environments are fully functional, each made up of the various components necessary to make a working system. The first is a development environment, which is used to test new features, either directly or as a verification stage after modular, test-driven development. Test-driven development is particularly important and powerful when making changes that are cumbersome or time-consuming to test with a real RIPE Atlas network—even a scaled-down development version—where real components have to "talk" to each other at network speeds.

In such cases, it is helpful to write unit tests for each part that represents the desired or correct interaction between components, and to write code until the tests pass. The second environment is a test or preproduction network that runs the next release while it is being prepared for production. Depending on the magnitude of a release, this test can take anywhere from less than a day to a week or more.

Each of these environments, including the production system, has a suite of system tests that verify a range of behaviours, including logging in to the website, creating measurements, and downloading results. This suite of tests acts as an end-to-end sanity check and can catch things that manual testing and unit tests might miss, or issues that may crop up only after unpredictable real-world use. This suite is complemented by a dashboard, which enables interactive visualisation and analysis of patterns and spikes in error rates, and various bespoke statistical graphs.

Probe Hardware Experiences

The first version of RIPE Atlas probes was very limited. The probe was based on a Lantronix XPort Pro module.[17] This module contains a FreeScale MCF5208 ColdFire processor, 8 MB of main memory, and 16 MB of flash storage. It is a 32-bit CPU with no *Memory Management Unit* (MMU). The module was complemented with a power board that takes power from USB (5V) and converts it to the 3.3V the module needs. The power board is then enclosed in a small black case.

Black-Box Model

The limitations of these first-generation probes led to a black-box model. The probes have no buttons; the Ethernet connector has link and activity LEDs and nothing else. Because of the limited amount of main memory, it would be difficult to run a web server on the probes for configuration.

The attractiveness of this model comes from the fact that a probe host has to connect only the USB connector to a USB port for power and plug in an Ethernet cable for network connectivity. The probe does not run any services, which is also good for security. Because the probes do not cost the hosts anything, we wanted to make them difficult to tamper with to avoid having people re-purpose them for their own uses.

Automatic Firmware Upgrades

Automatic firmware upgrades are another feature of the black-box model. Probe hosts do not have to care about upgrading the probe firmware. At first glance, firmware upgrades are easy enough. The flash memory of the probe is split into two parts; one part contains the firmware, and the other is used to temporarily store measurement results.

During a firmware upgrade, the new firmware is written to the partition that contains the measurement results, and the boot configuration is changed to boot the other partition.

Automatic firmware upgrades mean that almost all bugs can be fixed in later firmware versions, except those that prevent a firmware upgrade in the first place. We also want to use probes for as long as possible, so probes that may have been lying in a drawer for a long period of time should be able to upgrade to the latest firmware. This stipulation places rather tight restrictions on the backward compatibility of the firmware upgrade process.

One question that arises is whether it would be possible for an attacker to upgrade probes with malicious firmware. With the original firmware, this question meant attacking one of a few core servers or the SSH connection between the probe and the registration server. To further reduce the possibility of this attack, probes now verify a digital signature attached to the firmware. The digital signature is made offline and with secret splitting to make sure that multiple people are involved in signing the firmware.

Debugging a Black Box

A question that arises rather quickly is how to debug a black box. What do you do with a probe in a remote location that has no display and no buttons to press? Just about the only interaction possible is to power-cycle the probe. Because probes communicate over the network, we can distinguish two broad classes of problems: ones that prevent the probe from reporting results and ones that do not.

The second class can be solved most of the time by ensuring an appropriate amount of logging and taking advantage of the scale of the system. Over time, logging got more and more refined as a result of lessons learned trying to debug problems. For example, probes regularly log free memory and free disk space, making it possible to spot memory leaks.

For the first class, we have a few techniques and tricks. The first is that probes perform special DNS queries (called "SOS") when they reboot; the queries include the message they want to send as part of the hostname they look up. In some cases, a probe can contact a registration server but nothing else. In that case, we can direct the probe, for example, to an IPv4 address literal if we suspect DNS problems or force the probe to use either IPv4 or IPv6 if we suspect that the problem is with one of the protocols.

All of these tasks can be done without involving the probe host, although they can assist in two ways. Some hosts can run *tcpdump* on their routers, in some cases possibly giving us a clue about what went wrong. The probe also logs details about what goes wrong when it tries to connect, so connecting the probe to a different network allows these logs to be reported. It should be noted that connecting a probe to a home router is more likely to work than most office or data centre setups, because home routers have rather simple dynamic configurations and a lack of firewalls compared to more strictly managed environments.

Static Network Configuration

Most probes obtain their network configuration dynamically, using *Dynamic Host Configuration Protocol* (DHCP) for IPv4 and *Router Advertisements* (RAs) for IPv6. In some networks, however, those services are not available. To support those networks, probes can also receive network configuration from the back-end servers, but this process creates a "Catch 22" because a probe needs network access to be able to obtain network configuration from the back-end servers.

To solve this problem, the probe first has to be connected to a network that dynamically configures the probe. The probe can then fetch the network configuration and store it in flash memory. After a static network configuration successfully communicates to the probe this way, the probe can be moved to the target network.

However, this process introduces two problems. The first is that static network configuration has to be carefully copied across firmware upgrades. The second is that if somehow the static network configuration does not work, the probe has to revert back to dynamic network configuration. This reversion is quite tricky to achieve in practice and, over time, quite a few bugs have caused problems with it. In many cases, they required carefully tweaking the affected probes to get them back on track. Even though this feature adds a lot of complications, it also allows deployment in networks that would otherwise be inaccessible for us, so we expect to support this model in the long run.

It's worth noting that static configuration can also cause unexpected problems. We've seen many cases in which the addresses of DNS resolvers defined for the local network changed over time, but the probe was never informed about it. This change leaves the probe in an inconsistent state that requires manual intervention. For this reason, we do not recommend that hosts use static configuration unless there's no other solution.

Network Problems

One challenge with the black-box model is that debugging problems typically requires the probe to have a working network connection. Unfortunately, misconfigured networks are among the most common problems.

One problem that is relatively easy to work around is misconfigured DNS. Probes have the names and public keys of two registration servers, which are used to bootstrap the connection to the back-end servers. By adding IPv4 and IPv6 addresses to the names, the probes can connect even if DNS resolution fails. The probes try addresses and names in random order until one succeeds. Obviously, DNS measurements are still doomed on such a probe.

It is possible that the probe simply did not get a lease from DHCP, or got the wrong address or default router, etc. In such a case, the infrastructure side can't detect any life signs from the probe, and it is entirely up to the probe host to resolve the problem. Something similar applies to firewalls; sometimes it is possible to spot a firewall if the DNS traffic from the probe is detected, but we see no SSH traffic.

Finally, there are *Path MTU Discovery* (PMTU) problems. IPv4 typically does not have PMTU problems, but IPv6 does. A quick hack to avoid the IPv6 PMTU problems then causes problems with some routers used by probe hosts that make mistakes in TCP *Maximum Segment Size* (MSS) clamping.

All of these issues make troubleshooting a probe that fails to connect a bit of a trial-and-error game. Again, it sometimes helps to ask the probe host to move the probe to a new network so that it can report what it logged about the first network.

Different Probe Versions

One problem caused by the lack of an MMU on the first-generation probes was memory fragmentation. On systems with an MMU, main memory fragmentation is no problem because it does not cause fragmentation of the virtual memory of a process. On systems without an MMU, fragmented main memory cannot be hidden and, essentially, the only way to deal with memory fragmentation is to reboot the probe. Certainly with early firmware versions, the 8 MB of memory would quickly fragment and probes would reboot, sometimes within one day.

Fortunately, Lantronix was responsive to this issue and released a version with 16-MB main memory. These devices became the second-generation probes.

However, the CPU in the version 1 and 2 probes remained slow, with it taking about 30 seconds to set up an SSH connection. The 8 MB that is available for storing measurement results was not a lot if the probes lost their connection for more than a few days. In addition, off-the-shelf travel routers are both more powerful and cheaper than the Lantronix modules, which are designed for industrial applications.

This situation led to the third-generation probes, which are based on the TP-Link TL-MR3020 travel routers (see Figure 3). They have a much faster CPU and 32-MB main memory, but only 4-MB flash memory. Fortunately, a USB connection was available where we could plug in a USB flash device, although this need required a redesign of the firmware. During normal operation, the probe runs from the external USB flash, which is encrypted to make the probe more tamper-resistant. On the built-in flash is a small amount of code that can switch to the external USB and can also fetch firmware over the network and write it to the USB flash device.

This approach has a disadvantage in that it is a lot more complex than what runs on the version 1 and 2 probes. However, an advantage is that, if the USB flash becomes corrupt or nonfunctional because of a bug, the built-in flash version can simply overwrite it with a fresh copy. Upgrading the built-in flash is also possible, but ensuring that all combinations work proves to be quite tricky.

Finally, a need for high-capacity probes that would be rack-mounted in data centres resulted in the anchors. Anchors are rack-mounted PCs running CentOS Linux. From a firmware point of view, the main differences are that the probe code runs as a regular application and all network management is left to normal CentOS configuration. To the host, an anchor is a black box just like the regular probes, but remote-management cards make it possible to configure the anchors directly. In contrast to normal probes, anchors also run various basic services, allowing them to act as measurement targets as well.

Figure 3: Atlas Probe Version 3

Hardware Problems

With the version 1 and 2 probes, hardware problems were, and remain, rare. A very small fraction of the devices have developed hardware failures, and those failures fall into three groups: power failures, Ethernet failures, and flash-memory failures. We have not looked into it deeply, but power failures are likely a failure of the power regulation board. A few probes developed problems with the Ethernet interface; typically, they can send packets but cannot receive them. Finally, with some probes, the flash memory has broken.

The situation is quite different with the version 3 probes. One of the first things that became apparent is that some USB ports used to power the probes did not supply enough power. The curious effect is that only the external USB flash device refuses to start, leaving the probe with a configuration as if no USB flash device is plugged in. We solved this problem by having the probe perform special DNS queries to indicate that it cannot find a USB flash device.

Some time later, the USB sticks once again proved to be a weak link, although we should note that the number of failures is a few dozen out of thousands of probes. The USB sticks sometimes fail to work in various ways: some become read-only, while others reset their configuration to a default state. Typically, they then show a capacity of only 64 MB and have simple patterns as serial numbers. Finally, actual data corruption has occurred in a few cases.

Far less frequently than broken USB flash devices, sometimes the TP-Link devices develop power failures or problems with the Ethernet interface.

IPv4 and IPv6

Probes support both IPv4 and IPv6; however, IPv4 and IPv6 differ in many subtle ways. One feature that does not occur in IPv6 (or at least we are not aware of any probe in the network that deploys it) is *Network Address Translation* (NAT)[20]. When a probe reports a measurement result, the local IPv4 address is, in many cases, a local address defined by RFC 1918[21]. The RIPE Atlas system tries to keep track of what the corresponding public IPv4 address is, and the back end inserts that address in the measurement results when they come in. For IPv6, that complexity is not implemented.

At first, all probes with a global IPv6 address were assumed to be IPv6-capable. However, it turns out that a significant fraction of those addresses have a *Unique Local Address* (ULA)[22] and have no actual Internet connectivity. A second issue with IPv6 is that a probe may have multiple IPv6 addresses. The problem here is that, without special measures, it is not clear which address a probe will use in a measurement. A probe may use different addresses depending on the measurement target.

It is attractive to think of a probe being in a certain AS. However, each address can be in a different AS, meaning a probe can have different ASs for its IPv4 and IPv6 addresses, and a probe with multiple IPv6 addresses may, at least theoretically, also have different ASs for each address.

RIPE Atlas Community

From the beginning we knew that, in order to succeed, RIPE Atlas would need a strong community around it, both to help us grow the network by hosting probes and anchors and to help spread the word about the usefulness of the project for network operators, researchers, and interested users. We also rely on the Internet community and anyone using RIPE Atlas to give us feedback about new features we develop, useful functionality they would like to see, and the future direction of the project.

We regularly update the community about the development of RIPE Atlas via articles on RIPE Labs[18] that explain new features and functionality and ask for feedback.

We also publish different use cases and analyses that employ RIPE Atlas data, and have a special collection on RIPE Labs[19] that includes articles and blog posts written by external RIPE Atlas users about their own experiences, as well as scientific papers based on RIPE Atlas data, and presentations about RIPE Atlas given by members of the community at various conferences.

The RIPE Atlas network is essentially a volunteer-based project; it relies almost entirely on a community of probe and anchor hosts to install the hardware devices in their own networks. Most probe hosts initially heard about RIPE Atlas from the RIPE NCC directly, either via our website, mailing lists, or at various conferences, and applied online for their probe.

Initially, we distributed probes via post, free of charge, to anyone who applied, with the goal of reaching critical mass. As we've come closer to reaching our goal of 10,000 probes in the past year, we've started being more selective in distributing probes and have employed checks to ensure that probes are distributed to ASNs that don't already have a probe connected within them (although RIPE NCC members can receive a probe regardless of this stipulation). Anchor hosts also play an important role in strengthening the RIPE Atlas network and boosting its capacity, and, unlike regular probe hosts, they contribute to the project financially by purchasing the required hardware.

We also rely heavily on our ambassadors—currently more than 240 volunteers—who help us distribute probes and give presentations about RIPE Atlas at conferences all over the world. The RIPE NCC also has partnerships with the other *Regional Internet Registries* (RIRs) to help distribute probes in their service regions.

Each year, numerous organisations support RIPE Atlas by contributing to the project financially, and we are grateful to them for their support.

The RIPE Atlas website contains numerous "Community" pages that highlight new probe hosts, those hosts with the largest number of measurements and credits spent, anchor hosts, sponsors, and the different conferences where ambassadors will be available to answer questions and hand out probes.

RIPE NCC members enjoy several special benefits, including receiving extra credits for their own use and having access to a recently developed webinar on advanced use of RIPE Atlas measurements. During the webinar they have the opportunity to learn from and interact directly with RIPE Atlas developers.

In March 2015, we also hosted the first *RIPE Atlas Hackathon,* a three-day event in which developers, designers, computer science students, and open data enthusiasts were invited to use RIPE Atlas data to develop useful, creative, and stunning visualisations for the benefit of the entire Internet community.

The hackathon produced some very promising results, and we hope to host more such events in the future.

The past five years have been very exciting for the RIPE Atlas development team. It has been interesting to see the system grow from an idea into more fully realised concepts, then a prototype, and finally into a full service. RIPE Atlas is certainly still evolving, and its continued development is very much based on a steady stream of ideas and suggestions from the community.

It has also been nice to see how more and more people—network operators and researchers, but also regular Internet users—have started using RIPE Atlas to measure to their heart's content. We are happy to see people build tools for their specific uses and share them. We are very grateful to all our users, probe and anchor hosts, sponsors, ambassadors, contributors and everyone who has helped build this network—and hope we can count on your continued involvement for the benefit of the entire Internet community. You can get involved with RIPE Atlas by visiting this website:
https://atlas.ripe.net/get-involved/

Conclusion

RIPE Atlas is a globally deployed tool to actively measure the IP layer of the Internet. It is a collaboration of many, led by the development team at the RIPE NCC. It is useful for both ad hoc observations and long-term data collection. The number of its permanently operating vantage worldwide points stands at more than 8,000 and is constantly increasing. Everyone can contribute, and anyone who does contribute can use the entire system worldwide. Measurement results are stored for many years and thus everything based on RIPE Atlas results can be independently verified.

References

[1] "Test Traffic Measurement Service (TTM),"
http://ttm.ripe.net/

[2] RIPE Home Page, **https://www.ripe.net/**

[3] RIPE Analyse Start Page, **https://www.ripe.net/analyse**

[4] Roderic Fanou, "On the Diversity of Interdomain Routing in Africa,"
https://labs.ripe.net/Members/fanou_roderick/on-the-diversity-of-interdomain-routing-in-africa

[5] Andreas Strikos, "Amsterdam Power Outage as Seen by RIPE Atlas,"
https://labs.ripe.net/Members/andreas_strikos/amsterdam-power-outage-as-seen-by-ripe-atlas

[6] Robert Kisteleki, "The AMS-IX Outage as Seen with RIPE Atlas,"
https://labs.ripe.net/Members/kistel/the-ams-ix-outage-as-seen-with-ripe-atlas

[7] Emile Aben, "How RIPE Atlas Helped Wikipedia Users," **https://labs.ripe.net/Members/emileaben/how-ripe-atlas-helped-wikipedia-users**

[8] Emile Aben, "Measuring Countries and IXPs with RIPE Atlas," **https://labs.ripe.net/Members/emileaben/measuring-ixps-with-ripe-atlas**

[9] Collin Anderson and Philipp Winter, "Global Network Interference Detection Over the RIPE Atlas Network," **https://www.usenix.org/conference/foci14/workshop-program/presentation/anderson**

[10] Salim Gasmi, "Visualising RIPE Atlas Anchor Measurements," **https://labs.ripe.net/Members/salim_gasmi/visualising-ripe-atlas-anchor-measurements**

[11] Emile Aben, "Visualising Network Outages With RIPE Atlas," **https://labs.ripe.net/Members/emileaben/visualising-network-outages-with-ripe-atlas**

[12] Emile Aben, "Infrastructure Geolocation – Plan of Action," **https://labs.ripe.net/Members/emileaben/infrastructure-geolocation-plan-of-action**

[13] Vesna Manojlovic, "RIPE Atlas Hackathon Results," **https://labs.ripe.net/Members/becha/ripe-atlas-hackathon-results**

[14] RIPEstat, **https://stat.ripe.net/**

[15] RIPE DNSMON, **https://atlas.ripe.net/dnsmon/**

[16] RIPE Routing Information Service (RIS), **http://ris.ripe.net/**

[17] Lantronix XPort Pro, **http://www.lantronix.com/device-networking/embedded-device-servers/xport-pro.html**

[18] RIPE Labs, **http://labs.ripe.net**

[19] RIPE Atlas User Experiences, **http://labs.ripe.net/atlas/user-experiences**

[20] Geoff Huston, "Anatomy: A Look Inside Network Address Translators," *The Internet Protocol Journal*, Volume 7, No. 3, September 2004.

[21] Daniel Karrenberg, Yakov Rekhter, Eliot Lear, and Geert Jan de Groot, "Address Allocation for Private Internets," RFC 1918, February 1996.

[22] Brian Haberman and Robert M. Hinden, "Unique Local IPv6 Unicast Addresses," RFC 4193, October 2005.

[23] The map on page 5 shows the world using a *Mollweide Projection,* **https://en.wikipedia.org/wiki/Mollweide_projection**

THE RIPE NCC is the Regional Internet Registry for Europe, the Middle East, and parts of Central Asia. As such, it allocates and registers blocks of Internet number resources to its membership in this region, which consists mainly of Internet service providers, telecommunication organisations, and large corporations. It is a not-for-profit organisation that works to support the RIPE (*Réseaux IP Européens*) community and the wider Internet community. The following RIPE NCC staff have contributed to this article: Daniel Karrenberg (overall vision and probe firmware); Robert Kisteleki (architecture and team leader); Antony Antony and Philip Homburg (probe firmware); Christopher Amin, Massimo Candela, Viktor Naumov, Daniel Quinn, Andreas Strikos, Johan ter Beest and Christian Teuschel (infrastructure); Emile Aben and René Wilhelm (research); Suzanne Taylor Muzzin (communications); Mirjam Kühne and Vesna Manojlovic (community building). The authors can be contacted at: **atlas-ipj@ripe.net**

Rob Blokzijl Receives 2015 Postel Service Award

The Internet Society recently announced that its prestigious *Jonathan B. Postel Service Award* was presented to Rob Blokzijl for his pioneering work, 25 years of leadership at *Réseaux IP Européens* (RIPE), and for enabling countless others to spread the Internet across Europe and beyond. Dr. Blokzijl was selected by an international award committee, comprised of former Jonathan B. Postel award winners, which placed particular emphasis on candidates who have supported and enabled others in addition to their own specific actions.

During the 1980s, Dr. Blokzijl was active in building networks for the particle physics community in Europe. Through his experience at the *National Institute for Nuclear and High Energy Physics* (NIKHEF) and *The European Organization for Nuclear Research* (CERN), he recognized the power of collaborating with others building networks for research and travelled worldwide to promote cooperation across networkers. In the 1990s, Dr. Blokzijl was influential in the creation of the Amsterdam Internet Exchange, one of the first in Europe. His most widely recognized contribution is as founding member and 25-year chairman of RIPE, the European open forum for IP networking. Dr. Blokzijl was also instrumental in the creation of the *RIPE Network Coordination Centre* (RIPE NCC) in 1992, the first Regional Internet Registry in the world.

"Rob's technical expertise and tireless work had a profound impact on the development of the Internet as we know it today," said Kathy Brown, President and Chief Executive Officer of the Internet Society. "Beyond the breadth of his technical contributions, Rob is known across the Internet community for his strong leadership and unwavering commitment to collaboration and cooperation, exemplifying the spirit of this award."

Rob Blokzijl and his wife Lynn at the IETF 93 Plenary

The Postel Award was established by the Internet Society to honor individuals or organizations that, like Jon Postel, have made outstanding contributions in service to the data communications community. The award is focused on sustained and substantial technical contributions, service to the community, and leadership.

The Internet Society presented the award to Dr. Blokzijl, including a US$20,000 honorarium and a crystal engraved globe, during the 93rd meeting of the *Internet Engineering Task Force* (IETF) held in Prague, Czech Republic, July 19–24, 2015.

The Internet Society (**`www.internetsociety.org`**) is the trusted independent source for Internet information and thought leadership around the world. It is also the organizational home for the IETF. With its principled vision, substantial technological foundation and its global presence, the Internet Society promotes open dialogue on Internet policy, technology, and future development among users, companies, governments, and other organizations. Working with its members and Chapters around the world, the Internet Society enables the continued evolution and growth of the Internet for everyone.

IESG Statement on Maximizing Encrypted Access To IETF Information

The *Internet Engineering Task Force* (IETF) has recognised that the act of accessing public information required for routine tasks can be privacy sensitive and can benefit from using a confidentiality service, such as is provided by *Transport Layer Security* (TLS).[1] The IETF in its normal operation publishes a significant volume of public data (such as Internet-drafts), to which this argument applies. The IETF also handles non-public data (such as comments to *NomCom*, the IETF's nominating committee) that requires confidentiality due to the nature of the data concerned.

The *Internet Engineering Steering Group* (IESG) and the broader community[3] have further concluded that there can be other harmful effects in continuing to access public data as clear-text. Recent massive-scale man-on-the-side intermediary attackers have been seen to take advantage of the absence of security to mount active attacks that would be more difficult had a transport security mechanism such as TLS been used.[2, 4].

The IESG has therefore agreed that all IETF information must, by default, be made available in a privacy-friendly form that matches relevant best current practices. Further, all future embedded interactions with the IETF (such as **`<a>`** tags in HTML) should default to causing access via that privacy-friendly form. For content currently accessed using the HTTP protocol, using HTTPS URIs and appropriate TLS cipher-suites[5] will be the preferred access mechanism, however this direction encompasses more than HTTP traffic alone.

However, as there may be tools affected by this, and recognising that there are a number of IETF participants who prefer to continue to access materials via cleartext, or who have issues with using standard confidentiality services, the IESG are also requiring that public information continue to be made available in clear, for example via HTTP without TLS.

The changes caused by this statement should only need operational systems work and should be transparent to almost all consumers of IETF information. There are a small number of cases where these changes might cause some issues, for example, the current Internet-Draft boilerplate text, which uses the **`http:`** URI scheme. The IESG will work with the broader community, tools teams, and IETF Secretariat to make these adjustments while minimising disruption to the community.

Note that the "secure/privacy-friendly as the default according to best practices" principle set out in this statement applies to all IETF information, regardless of the protocol used to access that information.

References

[1] Stephen Farrell and Hannes Tschofenig, "Pervasive Monitoring Is an Attack," RFC 7258, BCP 188, May 2014.

[2] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ronald Deibert, and Vern Paxson, "China's Great Cannon," **`https://citizenlab.org/2015/04/chinas-great-cannon/`**

[3] Richard Barnes, "Deprecating Non-Secure HTTP," Mozilla Security Blog, **`https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/`**

[4] Nicholas Weaver, "A Close Look at the NSA's Most Powerful Internet Attack Tool," *WIRED,* March 13, 2014. **`https://www.wired.com/2014/03/quantum/`**

[5] Yaron Sheffer, Peter Saint-Andre, and Ralph Holz, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," RFC 7525, BCP 195, May 2015.

IANA Transition Update

As announced in our March issue (Volume 18, No. 1), the United States *National Telecommunications and Information Administration* (NTIA) announced its intent to "...transition Key Internet Domain Name Functions to the global multistakeholder community" in March 2014. For the latest information on this process, please visit **`https://www.icann.org/stewardship`**

Call for Papers

The *Internet Protocol Journal* (IPJ) is a quarterly technical publication containing tutorial articles ("What is...?") as well as implementation/operation articles ("How to..."). The journal provides articles about all aspects of Internet technology. IPJ is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. In addition to feature-length articles, IPJ contains technical updates, book reviews, announcements, opinion columns, and letters to the Editor. Topics include but are not limited to:

- Access and infrastructure technologies such as: Wi-Fi, Gigabit Ethernet, SONET, xDSL, cable, fiber optics, satellite, and mobile wireless.
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance.
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, troubleshooting, and mapping.
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, cloud computing, and quality of service.
- Application and end-user issues such as: E-mail, Web authoring, server technologies and systems, electronic commerce, and application management.
- Legal, policy, regulatory and governance topics such as: copyright, content control, content liability, settlement charges, resource allocation, and trademark disputes in the context of internetworking.

IPJ will pay a stipend of US$1000 for published, feature-length articles. For further information regarding article submissions, please contact Ole J. Jacobsen, Editor and Publisher. Ole can be reached at **ole@protocoljournal.org** or **olejacobsen@me.com**

Supporters and Sponsors

Publication of this journal is made possible by:

Supporters

Diamond Sponsors

Ruby Sponsor

Sapphire Sponsors

Emerald Sponsors

Corporate Subscriptions

Individual Sponsors

Lyman Chapin, Steve Corbató, Dave Crocker, Jay Etchings, Martin Hannigan, Hagen Hultzsch, Dennis Jennings, Jim Johnston, Merike Kaeo, Bobby Krupczak, Richard Lamb, Tracy LaQuey Parker, Bill Manning, Andrea Montefusco, Tariq Mustafa, Mike O'Connor, Tim Pozar, George Sadowsky, Scott Seifel, Helge Skrivervik, Rob Thomas, Tom Vest, Rick Wesson.

For more information about sponsorship, please contact `sponsor@protocoljournal.org`

The Internet Protocol Journal
NMS
535 Brennan Street
San Jose, CA 95131

ADDRESS SERVICE REQUESTED

The Internet Protocol Journal is published quarterly and supported by the Internet Society and other organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol.

Email: ipj@protocoljournal.org
Web: www.protocoljournal.org