2014

The Atlas Continuous Measurement Framework

Individual Project Report

Warwick Louw

General information:

Student:

Name:	Warwick B Louw
Student Number:	514201
Group:	3
Telephone number:	0624984112
Email:	514201@student.Inholland.nl

Company Name:

Address:	Science Park 400 Amsterdam
Postcode and Place:	1098XH
Web-address:	www.nlnetlabs.nl
Contact person:	Willem Toorop
Function:	Software Engineer
Telephone number:	0208884551
Email:	Willem@nlnetlabs.nl

Preface

For their research on Path Maximum Transfer Unit (PMTU) black holes and fragmentation dropping, NLnet Labs utilizes the RIPE Atlas measurement network. To derive meaningful results multiple interdependent measurements need to be performed. This is a process of scheduling and evaluating the results of measurements, based on the outcome of these results more measurements might be scheduled and evaluated. Until now this process has been performed manually.

The labor-intensively of this method makes it a burden to perform frequently and therefore difficult to follow any changes or trends. Also when others wish to reproduce the results, they are burdened with performing this method.

Out of this experience grew the desire to automate this process and capture it in a framework. Not only should a framework continuously monitor the network for the state of PMTU black holes and fragmentation dropping but should too be extensible to monitor other network properties that can only be determined by a multitude of measurements as well.

This report offers an overview of the design goals and decisions to create such a continuous measurement framework.

Contents

Terms

IP	Internet Protocol is the principle communications protocol for information exchange between network devices.
DNS	Domain Name System translates Internet domain and host names to IP addresses.
DNSSEC	Domain Name System Security is a set of protocols that add a layer of security to the DNS.
Ping	A ping is a network administration utility used to test the reachability of a host on an Internet Protocol (IP) network.
Packet	A packet is a portion of a message intended to be sent across a network.
MTU	Maximum Transmission Unit. This refers to the size limit for sending and receiving messages/packets for a particular link on the network. Throughout this report the term PMTU can be used which is the MTU of a particular link.
ICMP	Internet Control Message Protocol is one of the core protocols of the IP suite. It is for various types of signaling.
Fragmentation	Fragmentation occurs when a packet reaches a device on route to the destination where the MTU of the link is too small for the packet. Depending on the IP version or communication protocol the device could fragment the packet into smaller fragments so that it may reach its destination.
Fragmentation dropping	Fragmentation dropping occurs when a fragmented packet is received by a firewall or router and is then dropped as not to let it through into the network. For the most part this is done for security purposes.
IPv6 Network	The Internet Protocol has two versions that are currently deployed in a large scale. IP version 6 was implemented to deal with the depletion of address spaces available to the fourth IP version. IP version 4 allows a maximum of 2^{32} unique addresses, while IP version 6 allows 2^{128} unique addresses.
Atlas	Atlas is a network of probes connected across the internet, distributed and controlled by RIPE NCC.

The company

NLnet Labs is a non-profit organization that develops open source software and open standards for the benefit of the internet. It was founded by Stichting NLnet in 1999 (NLnet Labs History, 2014). Their main concerns are with DNS and routing software, standards, policies and governance of the internet.

NLnet Labs started off with only two members who were working on Domain Name System Security (DNSSEC). The budget of NLnet Labs was initially based on long term investment for development through subsidiaries from many companies including Verisign, Comcast and AFNIC.

The project

Problem description

In May 2011 NLnet Labs was notified that their webpage was irretrievable over Internet Protocol version 6 (IPv6) unless the user lowered the Maximum transfer Unit (MTU) of the network interface on their machine. The reason for this was, the path between the user and NLnet labs contained a smaller link and the methods to deal with Path Maximum transfer Unit Discovery (PMTUD, which ensures continuous communication through the medium) did not work (Toorop, 2013).

This sparked an investigation which then led to the research (Boer & Bosma, 2013) that was focused around two things; they wanted to inventory the different PMTU's on the IPv6 internet and to quantify the PMTUD problems. This research was done with the use of the Ripe Atlas Network which is owned and controlled by RIPE NCC (Ripe Atlas, 2014). These probes execute basic measurements such as pings, Domain Name System (DNS) queries, traceroute and a few more (Atlas probe measurements, 2014).

In January 2013 the same measurements were done by (Bagheri & Boteanu, 2013) as a follow up research to (Boer & Bosma, 2013), the main focus of their research was on PMTUD for DNS.

In June 2013 the same measurements were done once again this time by Willem Toorop with the intention of presenting at the 5th CENR Research and Development workshop (5th CENTR R&D workshop, 2013). The same set of measurements were conducted once again by Willem Toorop in October 2013 to present on the DNS workshop of the RIPE67 (Ripe 67, 2013).

This set of measurements has been done time and time again and each time using the RIPE Atlas web user interface, although later a script for scheduling and reading single measurements was written by Willem Toorop. This is a laborious task to since each measurement has to be scheduled and have the results evaluated manually. Since this task is time consuming it can be discouraging to perform regularly, this means it is hard to monitor for change and to track trends.

Goals

The objective of this project is to design and implement a framework that will continually reproduce certain sets of interdependent measurements in order to monitor certain network properties consistently.

The network properties that are taken into focus are:

IPv6 Capability

The goal is to determine which probes are capable of communicating over an IPv6 network. A number of probes state they are situated on an IPv6 network however are not capable of responding over IPv6. To test for IPv6 capability a simple ping is used and sent to a RIPE NCC anchor over an IPv6 network.

IPv6 DNS capable

The goal is to determine which probes have access to a DNS server. To test which probes have DNS services over an IPv6 network a simple query is sent from the probe to a RIPE NCC anchor. By targeting IPv6 probes the number of probes is limited and this allows the user to track the adoption of IPv6.

DNSSEC

The goal is to determine which probes have access to DNS services over have DNSSEC capability, firstly only probes which have access to DNS are eligible for this test. Secondly a “bogus” domain request is sent to the name server at NLnet Labs and depending on the response we can determine whether the probes’ DNS service has DNSSEC.

MTU

The goal is to determine the MTU of a specific probe the probes will first send a DNS query for a packet with the size of 1500. Once this has completed those who were not successful with transferring the large 1500 sized packet will then query for a packet sized 1480. Those that failed the previous test then query for a 1280 sized packet and so the same for 512 sized packets. By using this elimination method we can save time and credits by quickly eliminating those with smaller MTU’s.

Fragmentation Dropping

The goal is to determine how a probes’ network handles fragments a fragmented DNS query is sent to a custom name server written by Willem Toorop for the monitoring of fragmentation dropping. The targeted probes will query this name server for a fragmented packet using predefined queries will respond with a fragmented answer. If the answer reaches the probe then network the probe is in allows fragmented packets.

Other properties can be interpreted using the results of this project for instance: one could tell which DNS server each probe is using, the most commonly used DNS servers and whether these servers are DNSSEC capable or not.

Approach

After experimentation with the Ripe Atlas API it was decided the best approach would be to create a framework that is built upon the script written by Willem Toorop to communicate with the Ripe Atlas network through their API, as the script is versatile and robust.

The framework would house a series of classes called schedulers which deal with three main principles; gathering the appropriate probes with which to conduct each set of measurements, the set of measurements themselves and then processing the results of this set of measurements. Since each scheduler is tailored for a specific network property the set of measurements to determine the state of that network property are unique.

By running these schedulers over a period of time one could monitor the trend of these network properties for instance in the case of IPv6 DNS capable; the user could see which probes have access to DNS over IPv6 and whether those that did not have access to such services are gaining access to these services.

Design and Implementation

Scheduler

To test various network properties the framework would have to be modular, so that it can be extended by the user to perform measurements that will represent the network property. A base class is created called Scheduler, this class has method named run. Run comprises of measure and process.

Figure 1 illustrates the schema of how these members relate within the scheduler.

Run

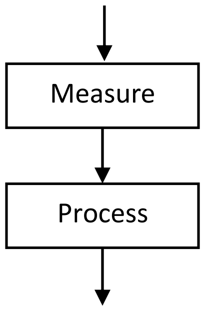

Figure 1 Run

For each network property a scheduler is created, this scheduler inherits from the main Scheduler class thus inheriting several methods. Below is an explanation of the process each scheduler follows. When this Scheduler is called to run (which is inherits the base scheduler) it will first call the local *measure* method. The program flow is illustrated in figure 2:

Measure

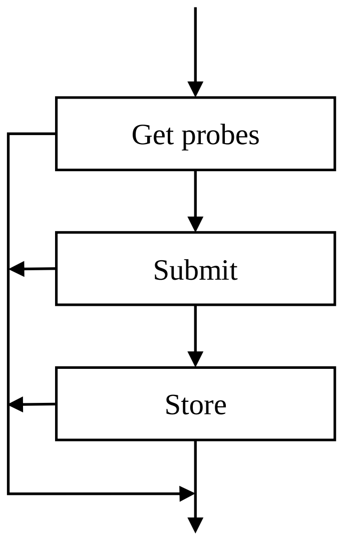

Figure 2 Measure

Measure will first attempt to retrieve suitable probes using the *get probes* method (illustrated in figure 3), if no suitable probes are found the program will return to the *run* method. If suitable probes are found then the method will proceed to submit the measurement to RIPE Atlas using the RIPE Atlas API accessed through the atlas program. If the submission is unsuccessful then the program will return to the *run* method. If the submission is successful then the program will continue and store the measurements details into the database. These measurement details include; when the measurement was created, which network property this measurement is associated with, which probes were targeted for this measurement and other details. The program will then return to the *run* method.

Get Probes

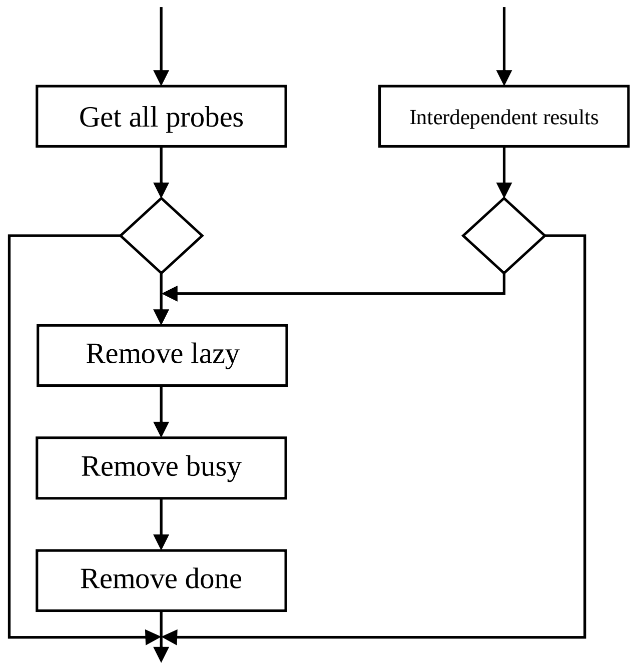

Figure 3 Get probes

Get probes will either request all probes that are available or request all probes that qualify for this measurement based on another schedulers' results, this depends on the scheduler in question. For instance when testing the MTU of a probes link, the scheduler will query those that failed the previous MTU test, therefore are eligible to test for a lower MTU. Once these probes are received, *Get probes* will continue to filter out those probes that have been proven to be lazy. Meaning probes that have been targeted for that measurement but do not have at least 5 results within the last week. Then busy probes will be filtered out, these are probes that have been targeted for the same scheduler but have not yet finished with the previous measurement. By filtering out the busy probes it ensures that there are never unnecessary results within the database. Then finally remove probes that have done this measurement at least 5 times within the last week and have results for each time, they are in fact done for this network property.

With all probes methods, all probes are tested at a week's interval meaning we can minimize the cost of the measurement by cutting out the probes whose results are known for that network property also meaning that these probes can be tested again a week later to see whether their results have changed. This is how trends for the specific network property are monitored.

Submit

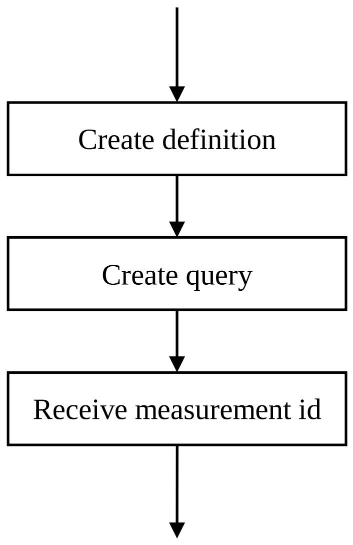

Figure 4 Submit

Submit deals with creating the query then passing it on in JavaScript Object Notation () form to the RIPE Atlas server so that the test can be performed. An appropriate query is created using a definition, for instance in IPv6 Capable will construct the query as a ping request to the specific RIPE NCC anchor then couple that with the appropriate probes determined using the *Get probes* method. Once the definition and then query has been created it will be submitted and the measurement id will be received.

Store

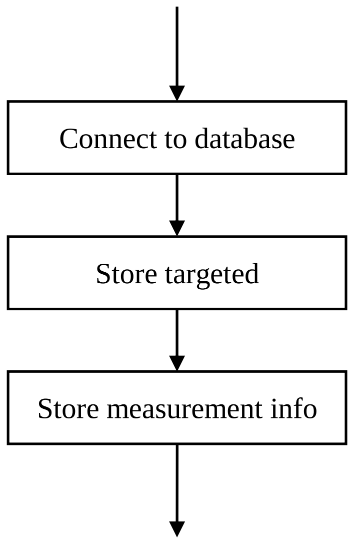

Figure 5 Store

Store will first attempt to connect to the database, if the database cannot be located the database will be created using a predefined structure then connect to it. Once a connection with the database has been established *Store* will then store all the necessary information; first starting with which probes are targeted then the information of the measurement such as type of measurement.

Process

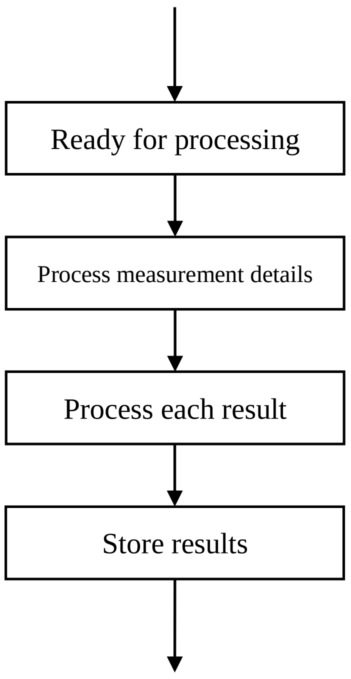

Figure 6 Process

Process is concerned with processing the results of a measurement. It will first determine which of the measurements pending processing are done and are ready to be processed. It will begin processing all those that are ready, first by filling the data into the measurement table then process each result within the measurement; first determining whether the result was successful for that probe or not then storing all the data within the results table.

Storage

The data is split between 3 tables in the database: measurements, targeted and results. The data shares a relationship with the measurement id, so that all data can be linked.

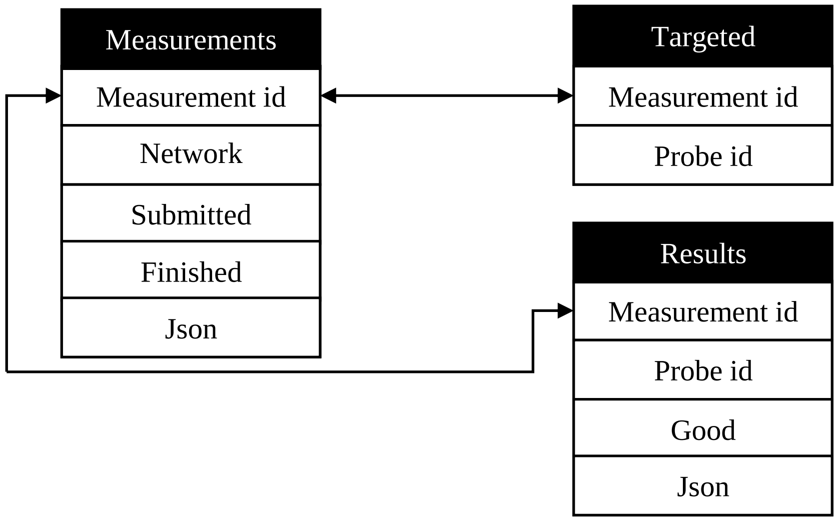

Figure 7 Database Scheme

The data has a relationship based around the measurement id, this id is linked to each data set for that measurement. The measurements table contains all the meta data of the measurement; this is compiled from information like the time the measurement was created, what type of measurement is it, which network property is being tested with this measurement and the raw data of the measurement. The target table contains the id of all probes targeted for the measurement and the id of the measurement itself. The results table contains the results for each probe. The columns are listed as: measurement id, probe id, whether the result was successful and the raw data for that result.

Conclusion

The atlas continuous measurement framework was created as a tool for measuring network properties using the Ripe Atlas network. Through its design it has become an extensible meaning anyone could use the framework to monitor the state of a network property such as those discussed but should also allow the user to create their own tests. Anyone using this framework is able to reproduce the set of measurements that another has done with relative ease meaning these test can be done from various locations and compare results. Most importantly this framework allows for continual measurement to allow the user to monitor these as trends.

Appendix

Requirements

To run the Scheduler the user will need either Linux or Windows operating system. Python installed ether version 2.7 or 3.x. Depending on your choice of database engine you could either use MySQL or SQLite, if you will are using SQLite you would need SQLite and the SQLite module for the appropriate python version found on the user’s machine and same from MySQL.

An additional requirement is that the user would need to be a RIPE NCC member with credits to perform the measurements within the Scheduler.

Usage

For Linux:

Ensure the file is executable i.e. run:

chmod +x “path”

Using CRONTAB include:

0 10 * * * “path” – to run the script every day at 10:00

E.g. 0 10 * * * /home/user/Atlas/Scheduler.py

For Windows:

You can use schtasks:

schtasks /Create /SC HOURLY /TN Scheduler /TR “path” – to run the Scheduler every hour.

References

5th CENTR R&D workshop. (2013, June). Retrieved from Centr: https://centr.org/index.php?q=node/2619

Atlas probe measurements. (2014). Retrieved from Ripe Atlas: https://atlas.ripe.net/docs/udm/

Bagheri, H., & Boteanu, V. (2013). *Making do with what we've got: Using PMTUD for a higher DNS responsiveness.* Retrieved from http://www.nlnetlabs.nl/downloads/publications/report-pmtud-bagheri-boteanu.pdf

Boer, M. d., & Bosma, J. (2013). *Discovering Path MTU Black Holes on the Internet Using the RIPE Atlas.* Retrieved from http://www.nlnetlabs.nl/downloads/publications/pmtu-black-holes-msc-thesis.pdf

NLnet Labs History. (2014). Retrieved from NLnetLabs: http://www.nlnetlabs.nl/labs/about/

Ripe 67. (2013, October). Retrieved from Ripe 67: https://ripe67.ripe.net/

Ripe Atlas. (2014). Retrieved from Ripe Atlas: https://atlas.ripe.net/about/

Toorop, W. (2013, June). *Using PMTUD for a higher DNS responsiveness*. Retrieved from NLnet Labs: http://www.nlnetlabs.nl/blog/2013/06/04/pmtud4dns/