Quantifying interference between measurements on the RIPE Atlas platform

Thomas Holterbach
ETH Zürich
thomahol@ethz.ch

Cristel Pelsser
Internet Initiative Japan
cristel@iij.ad.jp

Randy Bush
Internet Initiative Japan
randy@psg.com

Laurent Vanbever
ETH Zürich
lvanbever@ethz.ch

ABSTRACT

Public measurement platforms composed of low-end hardware devices such as RIPE Atlas have gained significant traction in the research community. Such platforms are indeed particularly interesting as they provide Internet-wide measurement capabilities together with an ever growing set of measurement tools. To be scalable though, they allow for concurrent measurements between users. This paper answers a fundamental question for any platform user: Do measurements launched by others impact my results? If so, what can I do about it?

We measured the impact of multiple users running experiments in parallel on the RIPE Atlas platform. We found that overlapping measurements do interfere with each other in at least two ways. First, we show that measurements performed from and towards the platform can significantly increase timings reported by the probe. We found that increasing hardware CPU greatly helped in limiting interference on the measured timings. Second, we show that measurement campaigns can end up completely out-of-synch (by up to one hour), due to concurrent loads. In contrast to precision, we found that better hardware does not help.

1. INTRODUCTION

Public measurement platforms composed of many low-end devices or *probes*, such as RIPE Atlas [1], are increasingly used by researchers and network operators. In addition to measure network performance [2, 3, 4], these platforms are now used to map the Internet [5], detect routing attacks [6], routing anomalies [7] and censorship [8, 6].

To scale and be practical, measurement platforms schedule measurements in parallel, without providing feedback to the user. When put together with the limited hardware and software capabilities, this raises the question of measurement interferences. What is the impact of an increased load on the precision of measurements performed? Do the measurements performed by one participant impact the results obtained by others? If so, by how much? Can this have an impact on previous research results? This paper answers these questions empirically for the RIPE Atlas platform.

By measuring the interference between our own measurements (§3), we show that measurements do indeed interfere with each other, sometimes significantly. More particularly, we found that user-induced interferences can impact two aspects of measurements: *precision* and *synchrony*.

First, the precision of delay measurements (*e.g.*, using ping) performed either from or towards probes can be significantly impacted when other measurements are launched from or toward them (§4).

Second, user-induced interferences can heavily desynchronize experiments performed on multiple probes, even when launched at the same time (§5).

Our key findings are as follows:

- The precision of measurements performed *from* and *towards* the probe are impacted when other measurements use the probe at the same time. On older hardware, delays increase by more than 1 ms in the median case and by more than 7 ms for the 95th percentile (Table 2).
- Measurements are very quickly desynchronized when other measurements are run in parallel. Under heavy load, completion time may be delayed by close to 1 hour (Figure 8).
- Upgrading the probe hardware significantly improves precision levels, but does *not* help ensuring good synchronization levels (§5).
- Previous research results, as well as the RIPE Atlas historic dataset, may have been affected by interfering measurements. We also highlight two techniques to mitigate interferences in the future (§6).

Overall, our results show that measurement interferences should be systematically taken into account when analyzing results from public platforms. To ensure reproducibility, all our measurement and analysis tools are available online [9].

2. THE RIPE ATLAS PLATFORM

We now describe how Atlas works and highlight its increasing popularity among the academic community.

As of April 2015, RIPE Atlas is composed of over 6,700 public probes scattered in 197 countries. Three versions of the probes exist, differing only by their hardware. Version 1 and version 2 are identical except for the amount of RAM they have. Both are Lantronix XPort Pro with a 167MHz CPU, 8MB or 16MB of RAM, respectively and a 16MB flash. The version 3 probe is a revamped TP-Link TL-

IMC'15, October 28–30, 2015, Tokyo, Japan.

DOI: http://dx.doi.org/10.1145/2815675.2815710.

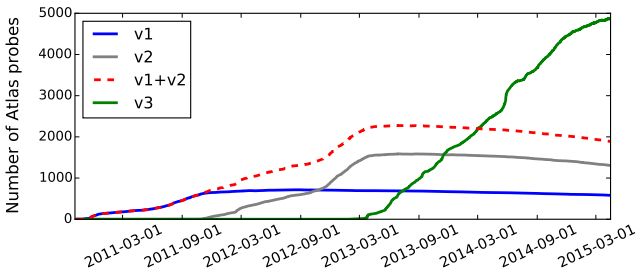

Figure 1: RIPE Atlas is composed of more than 6000 low-end probes which differ by their hardware: v1 and v2 probes are not powerful with respect to v3.

MR3020 router with a 400MHz CPU, 32MB of RAM and a 4MB NAND. The v3 probes are therefore more powerful.

Figure 1 depicts the evolution of the number of public probes per version since the platform inception. The number of v3 probes increased rapidly after they started to be distributed in 2013. While the number and proportion of v1 and v2 probes is decreasing, they remain non-negligible, accounting for 28.2% of the probes in April 2015.

RIPE Atlas uses credits to regulate the platform usage and schedules users' measurements concurrently. As of 2015, RIPE Atlas offers four[1] types of measures to its users: `ping`, `traceroute`, `DNS` and `SSL` [10]. In RIPE Atlas, a measurement is defined by a type, a frequency and set of probes. It can therefore refer to an arbitrary number of individual measurements performed from multiple probes. Users can also provide a start date and an end date. If none is provided, the measurement will start as soon as possible and has to be stopped manually. Measurements can be repeated or run only once (*one-off*). One-off measurements are near real-time if no start time is defined: users should expect results within 10 seconds [10].

RIPE Atlas regulates users load via a credit system. Users earn credits by hosting a probe and use them to perform measurements. RIPE's cost model is based on the resources each measurement needs. `traceroute` is the most expensive measurement, while `ping` is the cheapest. One-off measurements are also more expensive (twice more) than their scheduled counterparts as their arrival is not predictable.

RIPE Atlas uses basic scheduling strategies on each probe to handle concurrent load. The source code of the RIPE Atlas probes is based on `BusyBox` [11]. It has been adapted to improve the event management using the `libevent` library [12]. In addition, probes control the measurements frequency with `eperd`, a cron-like utility that can run measurements at regular intervals. One-off measurements are managed by the utility `eooqd`. Probes receive measurement requests from their controller with a *telnet* daemon. As several users can use a probe at the same time, it is essential to somehow schedule and limit users requests. In 2013, RIPE made the Atlas source code publicly available [13] but not yet the controller's.

Atlas probes are popular sources of measurements and are increasingly used in research. Since its inception, Atlas performed almost 30 million individual measurements[2] (Table 1). V3 probes hosted 2/3 of the measurements, while v1 and v2 probes hosted the rest. In March 2015, the user who used the most credits spent 83.3 million credits [14]. This is enough to perform more than 2,700,000 traceroutes. During the same month, the most used Atlas probe (a v1) provided 608,824 results [14], one every 4 seconds. Finally, the number of concurrent measurements is important. As an illustration, the platform was executing 592,000 concurrent individual measurements when we collected the statistics of Table 1.

	v1	v2	v3	Total
Total	3.1M	7M	19.7M	29.8M
In progress	58K	120K	414K	592K

Table 1: Overall, RIPE Atlas has hosted 29.8 million individual measurements. When we collected those results, 592,000 concurrent individual measurements were running on the platform.

An increasing number of research papers use RIPE Atlas. As an illustration, Machado *et al.* [15] used it to perform more than 3,000 traceroutes between a set of Atlas probes and a destination in Switzerland to see whether traffic stayed in the Schengen space. Fanou *et al.* [16] performed 1,108,709 traceroutes from 214 probes located in Africa to measure the impact of IXPs on interdomain routing in this region. Fiadino *et al.* [17] perfomed DNS requests for `*.whatsapp.net` from 600 Atlas probes to identify IP addresses hosting this service. Cicalese *et al.* [18] performed ping measurements from over 6000 probes located in 350 ASes in order to enumerate and geolocate IP-level anycast replicas.

Atlas probes are becoming popular destinations. Despite being designed for sourcing measurements, Atlas probes are increasingly used as targets by researchers [19, 20, 21, 16]. For example, Aben *et al.* [19] launched 7140 one-off traceroutes between a set of Atlas probes located in Sweden to infer topological properties. As the IP addresses of the Atlas probes are publicly available, users can target them from any possible sources (not necessarily from an Atlas probe). This enables users to perform hybrid measurement campaigns, with powerful machines as sources, and Atlas probes as destinations. Doing so, one can bypass the RIPE Atlas limitations (*e.g.*, frequency, credits cost) while keeping some of its interesting characteristics such as the large number of probes.

Some Atlas probes are more used than others. Due to their geographical position, IPv6 capability or a NAT gateway, some probes are more attractive than others. For instance, the distribution of probes per-country is highly skewed [22, 23]. While there are more than 1,200 Atlas probes in Germany, there are 29 countries with only one Atlas probe. The recent project `sbucket` [24] (supported by RIPE) aims at selecting probes based on spatial distribution rather than uniformly. Doing so would then to increase the load on isolated probes.

[1] `HTTP` measurements are possible but are restricted to researchers and other interested users on a case-by-case basis.

[2] As a measurement may involve a large number of probes, the number of individual measurements is more representative of the load of the platform.

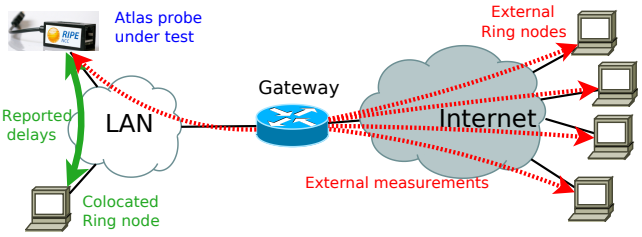

Figure 2: As opposed to traditional measurements which pass through the Internet (red arrows), the packets between the tested Atlas probe and its colocated Ring node (green arrow) always stay in the local network, thus preventing our measurements from being polluted by Internet variations.

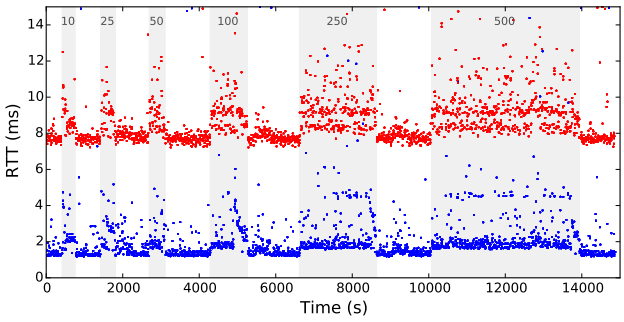

Figure 3: Delays measured *from* a v2 probe systematically increase when concurrent one-off traceroutes are launched on this probe.

3. QUANTIFYING INTERFERENCE

We describe how we quantify interference between measurements performed on a RIPE Atlas probe. We take the perspective of one user λ and one probe ρ and measure the effects on the results reported by ρ to λ when: *i)* ρ originates; or *ii)* is the target of concurrent measurements. In particular, we look at changes in the delay reported by ρ when concurrent one-off traceroutes are originated or when ρ is being used as ping destination. We use NL Ring nodes [25] as destinations (resp. as sources) of the pings sourced on (resp. destined to) ρ. We also look at changes in the completion time of one-off traceroute experiments performed on ρ.

We measure the delay reported by a probe using ping Delay-based measurements are indeed the most sensitive to concurrent load. In contrast, `traceroute`, `SSL`, and `DNS` output is less impacted by extra delay.

We also study the decrease in synchrony by measuring the completion time of one-off traceroutes performed on the probe.

. . . when increasing the number of concurrent measurements sourced from a probe To generate load on a probe, we launch an increasing number of one-off traceroutes from it using the REST API [26]. We use traceroute because it uses the most resources, as indicated by the higher cost. It is also one of the tools mostly used by researchers.

. . . when increasing the number of concurrent measurements targeting a probe The second technique we use to load a probe consists in gradually increasing the number of ICMP echo requests (800 bytes) targeted to it. We use a set of NL Ring nodes as sources. Each source sends 16 echo requests per second. We start with a single source. Every 2 minutes, we add a new source. We stop when there are 115 sources ($115 * 16 = 1840$ ping/s). While such frequencies are not common, experiments that use Atlas probes in a mesh-like fashion [16, 19, 20, 21] or that ping them from machines not limited in ping frequency may generate such a load. We use several Ring nodes as sources to mimic real experiments. To perform remote pings on multiple Ring nodes and collect the results, we built a tool [9] atop Scamper [27].

. . . and while preventing the effects from external factors We want to focus on the behavior of the probe and avoid network interferences. For each experiment, we measure the delay between the tested Atlas probe and a colocated Ring node in the same LAN (*i.e.* there is no IP hop between them). Because packets between the Atlas probe and its colocated Ring node always stay in the same LAN, we prevent our measurements from being polluted by Internet variations (Figure 2). We obtained these pairs of colocated Atlas probe and Ring node by a traceroute campaign between each Ring node and Atlas probes in the same AS. The results depicted in Table 2 all come from measurements done between an Atlas probe and its colocated NL Ring node.

4. DECREASED PRECISION

We now use our methodology (§3) to measure: *i)* the decrease in precision of delay-based measurements (this section); and *(ii)* the decrease in synchrony produced by concurrent measurements (§5). We performed all our measurements on multiple probes (at least two per version) to ensure conformity. As their number is not negligible and their decrease in precision and synchrony is serious, the next figures only focus on v2 probes.

Delays measured *from* the probe increase when concurrent measurements are launched *on* it. We launched ping measurements from the Atlas probe and towards eight random Ring nodes plus the colocated Ring node. The ping rate towards each destination is 9 ping/min, averaging 1.4 ping/s over all destinations. We increase the load on the probe by launching successively 10, 25, 50, 100, 250, and 500 one-off traceroutes.

Figure 3 shows the impact of the concurrent one-off traceroutes on the delay measured from a v2 probe. The blue points are RTTs between the Atlas probe and its colocated Ring node, while the red points are RTTs between the Atlas probe and another Ring node. The gray areas are the periods when one-off traceroutes are running. The number above each gray area is the number of one-off traceroutes executed. To quantify the impact, we compare the median, 95th percentile, and standard deviation of the ping measurements before the one-off traceroutes (the white area preceding the gray area) and during the one-off traceroutes execution time (gray area). The difference is reported in Table 2.

Delays measured from the probe systematically increase when one-off traceroutes are performed. Starting 100 one-off traceroutes increases the median delay of the concurrent pings by more than 1 ms. For v1 and v2 Atlas probes, the standard deviation is seriously impacted: +16.3 ms (v1) and +7.4 ms (v2). Atlas probes v3 show less effect, the median is only increased by 0.06 ms while the standard deviation is not impacted; this is due to v3 probes having more power. Sur-

impact on ping delay ...	sourced on probe			destined to probe		
when increasing load ...						
on probe		50^{th}	95^{th}	stdev	50^{th}	95^{th}	stdev
	(on : 100 traceroutes + 1.4 ping/s)			(on : 100 traceroutes, to : 9 ping/s)		
	v1	1.10 ms	7.30 ms	16.3 ms	0.61 ms	0.72 ms	0.04 ms
	v2	1.20 ms	7.70 ms	7.40 ms	0.50 ms	0.62 ms	0.02 ms
	v3	0.06 ms	0.10 ms	0.00 ms	0.06 ms	0.05 ms	0.00 ms
towards probe		50^{th}	95^{th}	stdev	50^{th}	95^{th}	stdev
	(on : 9 ping/min, to : 400 ping/s)			(on* : 9 ping/min, to : 1000 ping/s)		
	v1	0.11 ms	1.90 ms	15.2 ms	0.20 ms	5.40 ms	33.0 ms
	v2	0.22 ms	2.90 ms	3.90 ms	0.45 ms	2.60 ms	1.10 ms
	v3	0.00 ms	0.04 ms	0.00 ms	0.00 ms	0.00 ms	0.00 ms

Table 2: Quantification of interferences for v1, v2 and v3 probes. At the top, the probe is loaded by sourcing 100 one-off traceroutes. At the bottom, the load comes from incoming pings. Columns represent benchmarking measurements. On the left, we look at the impact of a load on the ping delay reported by the probe. On the right, pings are destined to the probe. With more powerful hardware, v3 probes are less sensitive to load than v1 and v2. *We used these pings to quantify the impact a load towards the probe produces on ping delay sourced on the probe (bottom-left).

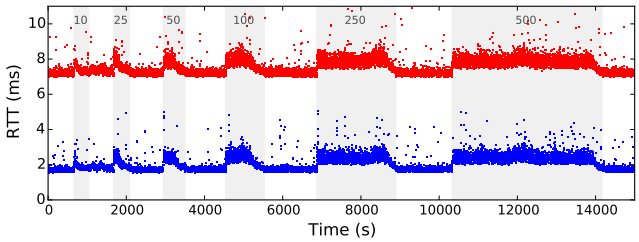

Figure 4: Delays measured *towards* a v2 probe systematically increase when concurrent one-off traceroutes are launched on the probe.

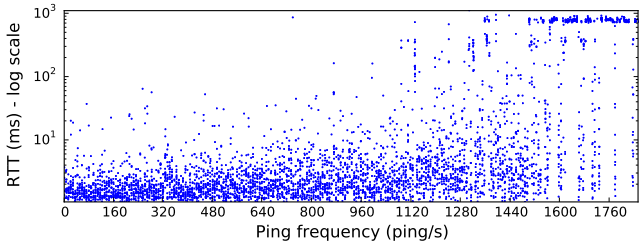

Figure 5: Delays measured *from* a v2 probe increase as the ping frequency targeting the probe increases.

prisingly, the number of one-off traceroutes does not change the magnitude of the impact but increases its duration: 10 one-off traceroutes impact as severely as 100 the concurrent ping measurements. As soon as the one-off traceroutes are done, RTTs go back to normal almost immediately.

Delays measured *towards* the probe increase when concurrent measurements are launched *on* it. We chose eight random Ring nodes plus the colocated Ring node and ping from them towards the Atlas probe with a frequency of 1 ping/s, summing up to a load of 9 ping/s. We then perform successively 10, 25, 50, 100, 250, and 500 one-off traceroutes from the Atlas probe.

Figure 4 shows the impact of the one-off traceroutes on the delay measured towards the Atlas probe. The blue points are the delays reported between the colocated Ring node and the Atlas probe while the red points are the delays reported between another Ring node and the Atlas probe. Again, gray indicates periods when one-off traceroutes are running.

The impact on pings targeting the probe is relatively lower (Table 2). When 100 one-off traceroutes are executed, the median of RTTs targeting a v2 Atlas probe increases by 0.5 ms. Despite the lower impact, we can easily see RTT shifts.

Delays measured *from* and *towards* a probe increase when it is used as a destination by concurrent measurements. We first launch pings from the Atlas probe towards its colocated Ring node with a frequency of 9 ping/min. We then use an increasing set of Ring nodes to target the probe with 800 bytes pings, each of them sending 16 ping/s (§3).

Figure 5 shows the impact on delay measured *from* the probe. Unlike with one-off traceroute measurements, the impact now increases with the number of pings directed towards the probe. When the frequency reaches 400 ping/s, the median delay reported by the probe increases by 0.22 ms, while the 95th percentile increases by 2.90 ms and the standard deviation by 3.90 ms. The probe becomes completely overloaded when the frequency reaches 1000 ping/s. This leads to very high delays (~1000 ms). Also, 10% of the pings are lost when the frequency becomes higher than 1280 ping/s. Here, the probe is the target of the load. Traffic is just sent to the probe, without involving the RIPE Atlas controller. We believe the inaccuracy increases progressively because the load per unit of time also increases. The controller cannot smooth the load by spreading it in time.

Figure 6 illustrates similar effects on the delays measured *towards* the probe. At the bottom of the figure, each box shows the inter-quartile range of RTTs between the colocated Ring node and the Atlas probe. The line in the box depicts the median value; the whiskers show the 1st and the 99th percentile, respectively. The top figure indicates the packet loss percentage. When reaching 1000 ping/s, the median RTT increases by 0.45 ms and the 95th percentile increases by 2.60 ms. As in Figure 5, when the frequency

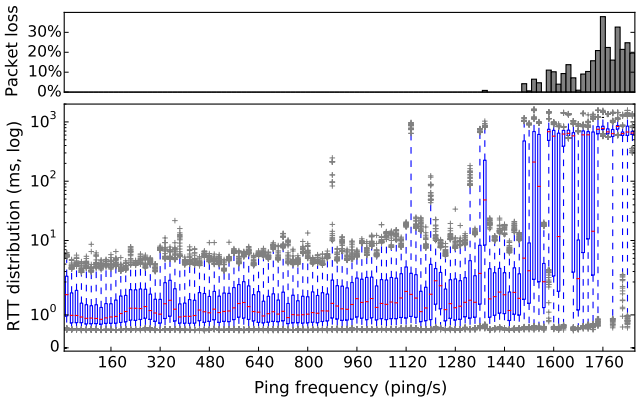

Figure 6: Delays measured *towards* a v2 probe increase as the pings frequency targeting this probe increases. Packet losses may appear if the ping frequency *towards* the probe becomes too high.

becomes even higher, the probe becomes completely overloaded. Reported delays skyrocket (~1000 ms) and some requests are lost.

Interference effects are compounded when combining source and destination load. So far, we have quantified separately the impact of using a probe as source or as destination. In reality, a probe may be used both as source and as destination at the same time. We could expect these interference effects to be additive, but our experiments show that *these effects are compounded*.

To quantify, we first start pings between an Atlas probe and its colocated Ring node (9 ping/min). We then start to flood the probe using the set of Ring nodes as described before. Finally, we start series of 25 one-off traceroutes. Figure 7 shows the results. The blue points are the measured delays between the probe and the colocated Ring node. The red vertical line indicates when we start to flood the probe with pings. The gray areas are the periods when one-off traceroutes are running. Before starting to flood the probe, we performed 25 one-off traceroutes in order to be able to compare the interference effects produced by these traceroutes with and without the ping flood. Each green point on the top indicates a traceroute success. The success rate of each one-off traceroute series is also mentioned.

When compounding source and destination load, delays measured from the probe increase even further. During the second series of one-off traceroutes, the standard deviation of the delay reported by the probe to the colocated Ring node is 30.8 ms and the 95th percentile is 23.9 ms. These values are far higher than the addition of the interference effects produced by a non-combined load on source and destination (Table 2). Success rate is also effected. 99% of the pings are lost during the last one-off traceroute series.

Key points. We observed significant interferences on delay measurements for v1 and v2 probes. These probes compose 28% percent of the platform. An important portion (34%) of the public experiments available result from experiments on v1 and v2 probes.

5. INCREASED ASYNCHRONY

We now study the impact of concurrent load on completion time. Atlas measurements are indeed scheduled over

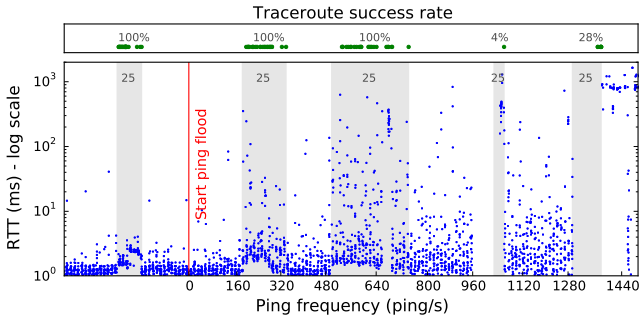

Figure 7: Increasing the source and destination load at the same time greatly increases the interference between measurements. One-off traceroutes take more time to execute, and worse, may fail. RTTs measured from the probe become higher.

time to limit the instantaneous load on a probe. When load increases, so does the completion time. We measure the time delta between when we request traceroute measurements and the time they finish.

Completion time significantly increases with the number of traceroutes. Figure 8 shows that the completion time may be 6.7 minutes (resp. 4.5 minutes) when requesting 50 one-off traceroutes on a v2 (resp. v3) probe. It takes up to 41 minutes with 500 one-off traceroutes on a v3 probes. All probe versions, including v3, are subject to a significant increase in completion time. Further experiments have shown completion times greater than one hour, even for v3 probes.

Completion time increases with the load *towards* the probe. In Figure 7, while the completion time for the first 25 one-off traceroutes takes up to 6.2 minutes, it takes up to 11.3 minutes for the second series of one-off traceroutes and up to 20.2 minutes for the third series. Sending 500 ping/s to a probe may then multiply the one-off traceroutes completion time by more than 3. When the ping frequency becomes too high, most of the traceroutes fail.

Key points Under load, requested measurements may be delayed, rendering the platform unsuitable to synchronized measurements. One could not ensure that pings or traceroutes start simultaneously on multiple probes. This is especially a problem when one wants to measure the effect of a single event from multiple vantage points, or an exogenous event. *This problem applies to all probe hardware—including the most powerful v3.*

6. DISCUSSIONS

We now describe the impact for researchers working with the platform (§6.1) as well as two solutions on how to mitigate interference in practice (§6.2).

6.1 Impact for researchers

On previous works. As described earlier, many research papers have used RIPE Atlas. Some of them relied on delay-based measurements [3, 4, 18, 28, 6, 16] which can be impacted by interferences. For instance, Rimondini *et al.* [3] used PELT [29], a changepoint detection algorithm, to detect shifts in RTTs and correlate them with routing changes. We mimicked such experiments on the RTTs of Figures 3 and 4 and detected a changepoint each time a one-off tracer-

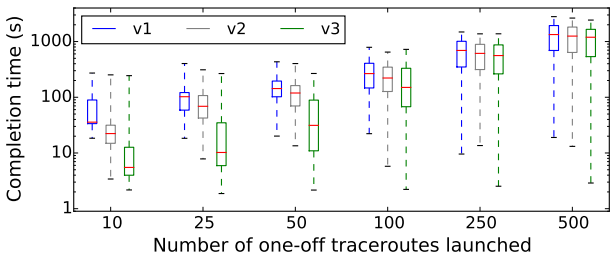

Figure 8: One-off traceroutes completion time is also impacted by concurrent measurements, independent of the hardware used. Results can be delayed by more than half an hour—making it impossible to perform synchronized experiments.

oute series starts and stops. Cicalese *et al.* [18] used the minimum value of ten successive RTTs to enumerate and geolocate IP-level anycast replicas. As a 1ms difference in latency measurements corresponds to a 100 km radius in geodesic distance, such studies may also be polluted by interference effects. An operator can use Atlas probes to measure the performance of her network. In this case, interference effects highlighted on the Figure 5 and 6 could wrongly trigger congestion alarms. Based on our results, any delay-based measurement obtained from v1 and v2 probes should be avoided if a precision below 15 ms is required.

On publicly available data. RIPE Atlas makes publicly available all the results collected with the platform since its inception in 2010 [30]. Researchers using these data should consider the impact of interferences. Especially for data collected before 2013—prior to v3 probes. We suggest researchers to be very careful when using publicly available delay measurements.

6.2 Solutions

Provide feedback to users with a measurement confidence index. A fundamental problem with Atlas is that the user has no visibility on the concurrent load of the platform. For that, we argue that RIPE can return a "confidence index" along with each result. The index would be function of the platform concurrent load. High (resp. low) load would lead to low (resp. high) confidence. Obviously, computing this metric should be done based on passive measurements to not stress the platform even more. We are currently working on calibrating such a metric using our measurements.

Enforce synchronization. While real-time is not a reasonable objective on shared platforms, more precise scheduling is achievable by maintaining a lower load on the probes and delaying upcoming measurements in favor of already scheduled events. Upon a measurement request, the user could then be informed of the exact timing of her experiment. Such an approach is however not possible if users do not all have the same privileges and some experiments can be preempted.

7. RELATED WORK

Other researchers have observed measurement interference and its impact on RTT. As an example, the effects virtualization can produce on measured delays have already been pointed out [31, 32]. As a number of large-scale platforms use VMs [33, 34, 25], tools such as [35] for PlanetLab, provide users with information about the state of these platforms and their nodes. In contrast, RIPE Atlas does not use virtualization and relies on a scheduler to share resources among users.

Gangam *et al.* [36] introduced heuristics to schedule a set of measurements between a set of nodes in order to avoid interference effects. However, this does not work when external measurements use Atlas probes as destinations.

Sanchez [37] *et al.* proposed a technique to coordinate experiments of a large-scale measurements platform in order to avoid undesired load on a network or a device. This solution is based on contracts, that give their holder specified rights over a set of resources for a limited period of time. In contrast, RIPE Atlas applies static rate limits for users and measurements. An user cannot run more than 100 simultaneous measurements, and the ping frequency is limited to one ping per minute.

Dasu [38] is a software-based measurement platform hosted by voluntary nodes located at the edge of the network. To enable finer-grained synchronization between a set of measurements (on the order of milliseconds), Dasu adopts a remote triggering execution model. In contrast, the one-off measurements provided by RIPE Atlas are launched as soon as possible (best-effort).

In [23], Bajpai *et al.* showed that RTTs from v1 and v2 probes to the first hop router are consistently higher than for v3 probes. They do not however study the relation between the measured delays and the load of the probes.

Mok *et al.* [39] proposed a technique to reduce packet sending time on low-end devices such as Atlas probes. This technique may be useful to counteract some of the interference effects we expose in this paper.

8. CONCLUSION

We presented the first measurement study of user-induced interferences on the RIPE Atlas platform. We found that measurements do interfere with each other. Delays reported from the probe increase and vary more when they compete with concurrent measurements. Measurement campaigns can further be arbitrary delayed, making it hard to perform simultaneous experiments from multiple probes.

Our findings also bring up new, non-trivial research questions: how can we design measurement platforms that provide more isolation between users, while still being efficient (*i.e.*, not requiring a global lock). We plan to explore this direction in the future.

9. ACKNOWLEDGMENTS

We wish to thank both the RIPE Atlas and the NLNOG RING support teams for accommodating our measurements and promptly replying to our questions.

10. REFERENCES

[1] RIPE NCC. RIPE Atlas. [Online]. Available: https://atlas.ripe.net

[2] S. Roy and N. Feamster, "Characterizing correlated latency anomalies in broadband access networks," in *ACM SIGCOMM 2013 (Poster Session)*, 2013.

[3] M. Rimondini, C. Squarcella, and G. Di Battista, "Towards an Automated Investigation of the Impact of BGP Routing Changes on Network Delay Variations," in *PAM*, 2014.

[4] G. Da Lozzo, G. Di Battista, and C. Squarcella, "Visual discovery of the correlation between bgp routing and round-trip delay active measurements," *Computing*, vol. 96, no. 1, pp. 67–77, 2014.

[5] A. Faggiani, E. Gregori, A. Improta, L. Lenzini, V. Luconi, and L. Sani, "A study on traceroute potentiality in revealing the internet as-level topology," in *IFIP Networking 2014*, 2014.

[6] A RIPE Atlas View of Internet Meddling in Turkey . [Online]. Available: https://labs.ripe.net/Members/emileaben/a-ripe-atlas-view-of-internet-meddling-in-turkey

[7] T. Yakimov, "Detecting routing anomalies with ripe atlas," april 2014.

[8] C. Anderson, P. Winter, and Roya, "Global network interference detection over the ripe atlas network," in *4th USENIX Workshop on Free and Open Communications on the Internet*, August 2014.

[9] [Online]. Available: https://github.com/nsg-ethz/atlas_interference

[10] RIPE Atlas - User-Defined Measurements. [Online]. Available: https://atlas.ripe.net/docs/udm/

[11] N. Wells, "Busybox: A swiss army knife for linux," *Linux J.*, vol. 2000, no. 78es, Oct. 2000.

[12] libevent - an event notification library. [Online]. Available: http://libevent.org/

[13] Releasing RIPE Atlas Measurements Source Code . [Online]. Available: https://labs.ripe.net/Members/philip_homburg/ripe-atlas-measurements-source-code

[14] Community Information, contributions, and hosts that stand out. [Online]. Available: https://atlas.ripe.net/get-involved/community/

[15] G. Machado, C. Tsiaras, and B. Stiller, "Schengen Routing: A Compliance Analysis," in *AIMS*, 2015.

[16] R. Fanou, F. Pierre, and E. Aben, "On the Diversity of Interdomain Routing in Africa," in *PAM*, 2015.

[17] P. Fiadino, M. Schiavone, and P. Casas, "Vivisecting whatsapp in cellular networks: Servers, flows, and quality of experience," in *TMA*, 2015.

[18] D. Cicalese, D. Joumblatt, D. Rossi, M.-O. Buob, J. Auge, and T. Friedman, "A fistful of pings: Accurate and lightweight anycast enumeration and geolocation," in *IEEE INFOCOM*, 04/2015 2015.

[19] Measuring Countries and IXPs with RIPE Atlas. [Online]. Available: https://labs.ripe.net/Members/emileaben/measuring-ixps-with-ripe-atlas

[20] How does the MENOG Region Measure up? [Online]. Available: https://labs.ripe.net/Members/mirjam/how-does-the-menog-region-measure-up

[21] Measuring Countries and IXPs in the SEE Region. [Online]. Available: https://labs.ripe.net/Members/emileaben/measuring-countries-and-ixps-in-the-see-region

[22] Percentage of connected probes per country. [Online]. Available: https://atlas.ripe.net/results/maps/density/

[23] V. Bajpai, S. J. Eravuchira, and J. Schönwälder, "Lessons learned from using the ripe atlas platform for measurement research," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, pp. 35–42, Jul. 2015.

[24] RIPE-Atlas-sbucket. [Online]. Available: https://github.com/cod3monk/RIPE-Atlas-sbucket

[25] NLNOG Ring. [Online]. Available: https://ring.nlnog.net

[26] Creating Measurements with the RIPE Atlas Restful API. [Online]. Available: https://atlas.ripe.net/docs/measurement-creation-api/

[27] M. J. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the internet," in *IMC*, 2010.

[28] RIPE Atlas - Superstorm Sandy. [Online]. Available: https://labs.ripe.net/Members/emileaben/ripe-atlas-superstorm-sandy

[29] R. Killick and I. A. Eckley, "changepoint: An r package for changepoint analysis," *Journal of Statistical Software*, vol. 58, no. 3, pp. ??–??, 6 2014.

[30] RIPE Atlas - Public measurements. [Online]. Available: https://atlas.ripe.net/measurements/#!public

[31] J. Whiteaker, F. Schneider, and R. Teixeira, "Explaining packet delays under virtualization," *SIGCOMM Comput. Commun. Rev.*, vol. 41, 2009.

[32] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using planetlab for network research: Myths, realities, and best practices," *SIGOPS Oper. Syst. Rev.*, 2006.

[33] Planetlab, "Planetlab: An open platform for developing, deploying and accessing planetary-scale services," http://planet-lab.org.

[34] M-Lab, "Measurement lab," http://www.measurementlab.net.

[35] K. Park and V. S. Pai, "Comon: A mostly-scalable monitoring system for planetlab," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, Jan. 2006.

[36] S. Gangam and S. Fahmy, "Mitigating interference in a network measurement service," ser. IWQoS '11, 2011.

[37] M. A. Sánchez, F. E. Bustamante, B. Krishnamurthy, and W. Willinger, "Experiment coordination for large-scale measurement platforms," in *ACM SIGCOMM Workshop on C2B(I)D*, 2015.

[38] M. A. Sánchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamante, B. Krishnamurthy, and W. Willinger, "Dasu: Pushing experiments to the internet's edge," in *NSDI*, 2013.

[39] R. K. P. Mok, W. Li, and R. K. C. Chang, "Improving the packet send-time accuracy in embedded devices," in *PAM 2015*.