A GLIMPSE of the Internet's Fabric

Michael Faath, Rolf Winter
University of Applied Sciences Augsburg
michael.faath@hs-augsburg.de
rolf.winter@hs-augsburg.de

ABSTRACT

Network measurements are essential for network operations and troubleshooting. A number of network measurement projects have developed measurement platforms to not only assess the state of individual networks and services but target certain aspects of the Internet as a whole.

Measuring any aspect of the Internet is challenging for various reasons such as the sheer scale of the Internet or the dynamics of it. Building and operating a platform that has Internet-wide network measurements as a goal is therefore challenging.

In this paper, GLIMPSE—an end user-based network measurement platform—is introduced and its architecture is described in detail. GLIMPSE is an attempt to build a measurement platform based on standard measurement techniques, external logic to trigger measurements and last but not least community support.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Miscellaneous

Keywords

network measurement, measurement platform, end-to-end measurements

1. INTRODUCTION

A growing number of projects attempt to measure certain aspects of the Internet. E.g., there is the RIPE Atlas[1] project, which uses a large number of small hardware devices distributed across the Internet to perform Internet connectivity and reachability measurements. Implementing such measurement platforms, deploying them, operating them, gathering and analyzing the measurement data are all challenging tasks in itself. In this paper, we present one such platform called GLIMPSE, an end user-based, all-software platform for coordinated network measurements. We describe its design goals, its components and describe how it operates and differs from existing platforms for large-scale network measurements.

[1] `https://atlas.ripe.net/`

2. DESIGN GOALS

GLIMPSE has a number of design goals, which makes it occupy a unique place in the design space of existing measurement platforms. First of all, GLIMPSE was always intended as a software-only measurement platform. In other words, it is not primarily intended to run on dedicated hardware devices. This has a number of advantages. The potential user base of GLIMPSE is much larger as a software-only product and there is no up-front investment in hardware necessary. The disadvantage is that there is always a potential that the measurements carried out by GLIMPSE are influenced by other software running on the same device or by device limitations.

Another goal of GLIMPSE was that the probe should be cross-platform by design. Again, this goal was chosen to increase the potential user base but this choice had a severe impact on the measurement techniques the probe can support. E.g. the probe cannot expect root/administrative privileges since e.g., on off-the-shelf smartphones these are not available. Also, operating systems typically offer quite different low-level network access which made the development of consistent measurement methods challenging.

The target audience for GLIMPSE probes was always regular end users and their devices. This was to increase the reach and coverage of the probes. E.g., having probes on regular end user devices such as tablets, smartphones or laptops places probes right at the edge of the Internet from where true end-to-end measurements can be performed. Also, running on typical end devices enables GLIMPSE to measure the actual Internet experience of the user. The problem with this choice is clearly that end user devices are not always on. Also, the probe must have a small resource footprint, i.e. the usage of CPU, memory and other device resources should not be noticeable by the user or impact the performance of other applications on the device. Finally, the privacy of users must be ensured. Not using administrative privileges is actually helpful in this respect.

An important aspect of GLIMPSE was the ability to perform measurement campaigns: large-scale measurement activities involving a larger number of probes. These campaigns are necessary for measurement studies of Internet-wide significance. E.g., estimating the average distance of

VALUETOOLS 2015, December 14-16, Berlin, Germany

DOI 10.4108/eai.14-12-2015.2262670

users to the Google content delivery network infrastructure in terms of hops and latency. This requires that GLIMPSE users donate a small amount of bandwidth to this greater cause. This bandwidth is then used by the GLIMPSE platform to perform measurements. End systems with GLIMPSE can then be instructed to e.g., ping a destination on the Internet. The maximum amount of donated bandwidth per month can be configured by the user. This ability of the platform also means that security of GLIMPSE is of pivotal importance. If compromised, the platform could be misused for distributed denial of service attacks.

In a way, GLIMPSE also attempted to follow the Unix philosophy of "do one thing and do it well" by implementing a number of small, fairly standard measurements. The individual measurements implemented by the probes such as ping or traceroute have all one very specific goal. Composite measurements can be performed by issuing measurement instructions which include an order and settings of the measurements to be performed and a schedule to trigger these measurements at certain times or repetitively. The small set of tools somewhat restricts what can be measured. GLIMPSE however uses a clever combination of these tools, it uses a larger set of probes on the Internet which increases the coverage of these measurements and on top of that it makes use of schedules. This way, a wide range of interesting measurements can be performed.

Cross-cutting all of the above is an issue, which is non-technical: getting users to install the GLIMPSE application and convincing them to donate a little bandwidth. The focus of this paper is on the technical details of the GLIMPSE architecture. Discussions about involving them in network measurements and incentivizing them to install the app by creating a benefit can be found in [3]. For GLIMPSE, this is the possibility to identify potential network issues as well as run own measurements with the provided built-in tools.

3. ARCHITECTURE

GLIMPSE as a whole is a collection of different components serving different purposes. The architecture with all subsystems and how they interact with each other can be seen in figure 1.

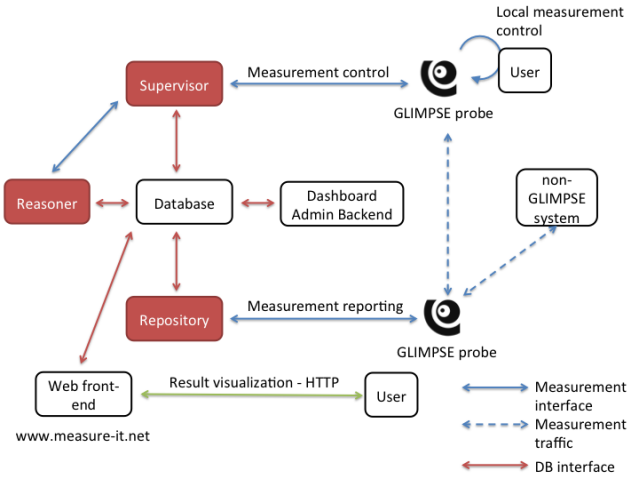


Figure 1: GLIMPSE architecture

It is based on the mPlane[2] project's measurement platform architecture which describes a blueprint for a generic network measurement platform. Besides the probes, the highlighted components (supervisor, reasoner and repository) are the core pieces of it.

The components of the GLIMPSE architecture can be roughly divided into two groups: the ones the user directly interacts with—the probe itself and the web front end—and the servers and services in the background which provide the necessary infrastructure for the probes to receive measurement instructions and send back results. The figure also helps to understand the typical workflow of GLIMPSE: a measurement is defined by an administrator in the admin dashboard. The supervisor then sends instructions to the probes selected for that measurement. After the execution of the measurements, the probes send the results to the repository. Finally, the administrator can observe the results in the dashboard and the reasoner might do automatic and intelligent reasoning to schedule new measurements based on the results. Note that all communication between the probes and other components is secured with HTTPS. Certificates are used to make sure a probe only receives instructions from a legit supervisor.

In the following sections all components are explained in detail.

3.1 Probe

The GLIMPSE probe is the application a user installs on a device. It pulls its configuration and measurement instructions periodically from the supervisor, executes measurements and sends results back to the repository. The app provides the following measurement tools at the moment:

Ping This measurements determines the round-trip-time from the probe to a specified destination (which can be an IPv4 address, an IPv6 address, or hostname). There are multiple parameters available for this measurement, e.g., source- and destination-port, interval-time, or timeout). The protocol used for the measurement can be set by the ping-type parameter: "UDP", "TCP", or "System Ping". The latter uses the systems ping-tool and parses its output. Due to operating system limitations, TCP pings are not available on all supported platforms.

Traceroute determines the path from the probe to a set destination and returns the round-trip-time to each hop on the path. The previously described ping measurement is used to retrieve this information. There are multiple parameters available for this measurement similar to the ping options.

HTTP download speed measures the currently available bandwidth by downloading a file from a public HTTP-server (speed test). It is possible to specify a number of settings such as the number of threads used to download the file in parallel and get results for each thread separately.

Bulk transfer capacity This measurement determines the so-called bulk transfer capacity[6]. It is measured from one probe to another GLIMPSE probe by sending bulk data over a single TCP connection.

[2] `http://www.ict-mplane.eu`

DNS lookup This returns all available DNS records for a given hostname. The DNS server used for the measurement can be specified by a parameter.

Reverse DNS lookup Determines the hostname for a given IPv4 or IPv6 by doing a reverse DNS lookup.

Trains of packet pairs Determines the sending and receiving speed for the probe by using trains of packet pairs[6]. This measurement is considered experimental.

UPnP discovery This measurement performs a UPnP discovery using the MiniUPnP library[3]. The result is a dump of the raw data returned by this library.

For multiple reasons only active measurements are supported by GLIMPSE. As described before, passive measurements would require root privileges on some platforms, but we also do not want to analyze user generated data on end user devices for privacy reasons.

The measurements described above are all written as plugins. It is therefore possible to write a measurement for the probe without in-depth knowledge of the app itself. At the moment, these plugins are compiled directly into the probe, but it might be possible in the future to load them dynamically to allow users to include own measurements into the software.

We have developed two front ends to support different use cases: a graphical user interface and a console-only interface. The latter is intended to run on servers or embedded devices in the background. The user can only view the log output of the probe and see the results on the GLIMPSE website, but can not interact directly with the probe. The GUI version of the probe is intended to run on a desktop computer or a mobile device. The user can see the scheduled measurements and the available results. It is also possible to execute measurement by the user with the help of a toolbox (figures 2a and 2b). The results of these user-scheduled measurements can be viewed in the app but are not sent back to the GLIMPSE repository. This makes the GUI version of the GLIMPSE probe a general purpose network measurement and debugging tool for its users.

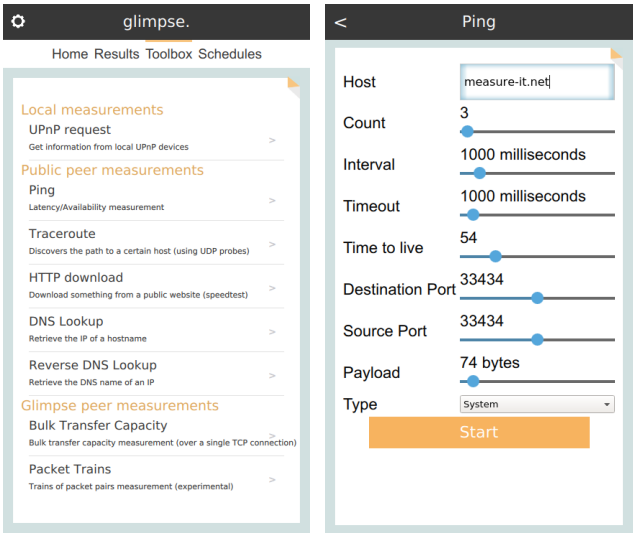
The user can cap the monthly traffic volume the app is allowed to cause for externally triggered measurements. Additionally, users with a mobile connection like UMTS or LTE can specify if they want to allow measurements on those interfaces at all and can set the maximum traffic volume for those interfaces separately.

The GLIMPSE probe is written using Qt[4], a cross-platform application framework for C++. Therefore, the probe can be compiled and executed on every platform supporting Qt5, this includes Windows, Linux, Mac OS X, Android and iOS. We provide binary packages for some of those systems for an easy installation. The complete probe code is open source and can be found on GitHub[5].

3.2 Supervisor

The supervisor is the single point of control for the probes. As the probes might not be directly reachable from the Internet (e.g., when they are behind a home gateway), the probes pull configurations and measurement instructions via

[3] http://miniupnp.free.fr/
[4] https://www.qt.io/
[5] https://github.com/HSAnet/glimpse_client



(a) Toolbox (b) Ping

Figure 2: Probe screenshots

an HTTP GET request from the supervisor. The configurations include the address of the repository and different timings for when to pull configurations, measurement instructions and when to send the results. We have defined five timings which are also used to determine when to execute a measurement:

Immediate As soon as possible.

OneOff Once at a specific time.

Periodic Repeated based on a given period, start and end timestamps can be set.

Calendar A cron-like timing to specify the months, days, hours, minutes and seconds something should be done.

OnDemand This gives the probe the control to do something when and as often as it likes.

The first four timings follow the information model of the IETF working group LMAP[2]. The OneOff, Periodic, and Calendar timing allow also to set a randomness factor. This is helpful if e.g., multiple probes get the same measurement and timing instruction but should not execute all at the exact same time (to not DDoS a service).

An excerpt of a measurement instruction in JSON[6] can be seen in listing 1.

```
"id": 43,
"task": {
    "id": 14,
    "method": "ping",
    "options": {
        "count": 4,
        "destination_port": 33434,
        "host": "measure-it.net",
        [...]
    }
},
```

[6] http://www.json.org/

```
"timing": {
    "periodic": {
        "interval": 3600000,
        "randomSpread": 30000,
        "start": "2015-07-13T00:58:00"
    }
}
```

Listing 1: Measurement Instruction

It shows a ping measurement to `measure-it.net` which is carried out every 60 minutes including a random offset of 30 seconds at max, sending four packets to the given destination port.

3.3 Repository

The repository is the interface between the probes and the central GLIMPSE database. The probes send their results with an HTTP POST request to the repository for storage into a PostgreSQL database. If the probe has problems while sending results—e.g. because it has connection problems—they will be cached locally until a connection to the repository can be established. All server-side components of the architecture have direct access to the database, only the probes need to go through the repository for security reasons.

Not only the direct results from a measurement are sent to the repository but also some information about the system to judge whether or not the results can be trusted. This auxiliary information includes the processor load, memory usage, connection mode (e.g., wireless LAN) and local traffic counters. Abnormal values of system state (such as high CPU load) or connection details (such as low signal strength on wireless) can be used to filter results. For an analysis, one should only use those measurement results with a high confidence regarding their correctness.

An example of a result gained from executing the measurement instruction seen before is shown in listing 2.

```
{
  "probe_result": {
    "round_trip_ms": [
      9.823,
      9.917,
      9.805,
      9.846
    ],
    "round_trip_stdev": 0.04261215669779576,
    "destination_ip": "141.82.57.241",
    "round_trip_max": 9.917,
    "round_trip_received": 4,
    "round_trip_avg": 9.84775,
    "round_trip_min": 9.805,
    "round_trip_sent": 4,
    "round_trip_loss": "0"
  },
  "start_time": "2015-09-10T11:58:16Z",
  "end_time": "2015-09-10T11:58:17Z",
  "duration": 848,
  "error": "",
  "pre_info": {
    "available_disk_space": 2481373184,
    "available_traffic": 31431168,
    "connection_mode": 5,
    "cpu_usage": 0.0136986300349236,
    "signal_strength": 100,
    "used_traffic": 530688,
    "battery_level": -1,
    [...]
  },
  "post_info": {...}
}
```

Listing 2: Measurement Instruction

As the instruction requested four packets per ping, the result shows four round trip times in `round_trip_ms`. Multiple statistics are directly calculated in the app (average, min, max, ...) to reduce the processing load while analyzing results. This result shows an average round trip time for a ping to `measure-it.net` of approximately 9.8 milliseconds.

3.4 Reasoner

A reasoner is a purpose-built piece of software part of the GLIMPSE back end. It tries to do intelligent reasoning about measurement results and it will automatically schedule additional measurements if needed. This has a number of advantages. E.g., new measurements can be created not by updating the probes themselves, but by adding intelligence to the GLIMPSE back end. Also, unsupervised measurements can be performed, where additional measurements are only performed in case they are really needed.

We have implemented one reasoner as an example: the traceroute-reasoner. It performs measurements similar in nature to Paris traceroute using the simple traceroute implementation on the probes[1]. It tries to determine whether multiple paths between a probe and a specified endpoint exist by executing multiple UDP traceroute measurements with different source and destination port combinations. At first, n traceroute measurement instructions with the same target but with different destination- and source-ports are sent to a probe. All paths found in the results of the measurements are stored into the database. If these contain multiple paths, n new traceroute instructions are sent to the probe with the same target but again with different destination- and source-port pairs. This is repeated as long as new paths are discovered.

3.5 Dashboard

To make the administration of the platform as easy as possible, a dashboard was developed. It provides an overview of all available probes and allows one to schedule new measurements with the help of a graphical interface. Results for already executed measurements are shown as graphs or in a table view. An example of such a result graph can be seen in figure 3.

The figure shows a scatter plot for two HTTP download measurements, i.e. downloads from two different servers, on one probe. As can be seen, the dashboard makes initial result evaluation quite easy. The administrator of the system gets a good overview of measurement results and can then use additional tools to e.g., explain phenomena in the data, filter potentially erroneous results or correlate results with other data.

These graphs will be also provided to the user (the ones running the probes) in the web front end and the app at a later point. This makes it easy for the user to see what the probe is doing and what it has found out.

4. FUTURE WORK

At the moment, we can schedule measurements to be performed regularly on the devices. The user can only execute one-shot measurements without a schedule in the graphical version of the app. We want to add the possibility for the

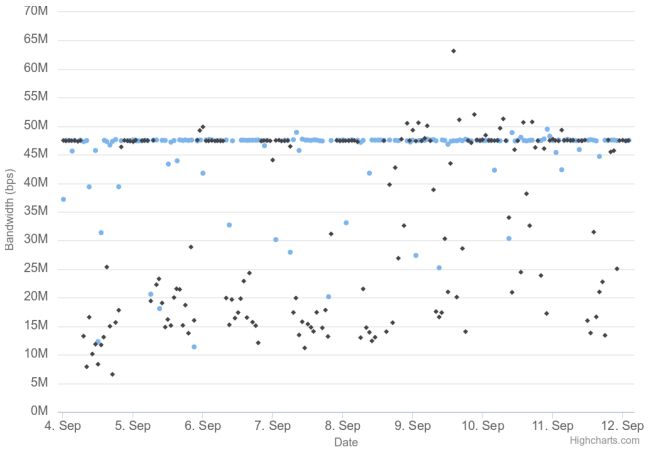


Figure 3: HTTP Download Scatterplot

users to define own schedules in the web front end to be executed on their devices. This would make the app more interesting to system administrators who can use this to monitor their own infrastructure.

As described before, we want to enable users to write own measurements and include them as a dynamically loaded plugin into the probe. This would make it necessary to have a capability discovery mechanism, as not all probes would have the same set of measurements anymore. GLIMPSE is already prepared for that as it can announce the available measurements via so-called specifications as defined by the mPlane project[12].

Another feature of the graphical version of the GLIMPSE probe which is in preparation at the time of writing is an easy to use network troubleshooting interface. It will basically consist of three buttons on the main page of the app, which will offer the user three basic choices, in case he or she experiences network issues. The first choice is a quick speed test, to see whether the problem is bandwidth related. This basically mimics current user behaviour, when using services such as speedtest.net. The second choice is for users that suspect that the network is not working at all. This will trigger a number of tests in the local network and beyond. The third choice is for users that experience performance problems but the Internet does not appear to be down completely. Here, the measurements focus on potential bandwidth and latency problems.

Besides all the technical aspects of the GLIMPSE probes, the human user is another good source of information regarding aspects such as the human perceived quality of the Internet connection. Only the user can provide interesting auxiliary information such as the details about the broadband plan it has at home (e.g., the maximum download rate). The graphical version of the GLIMPSE probe will therefore get a survey interface, where the user will be asked to participate in voluntary surveys.

Currently, GLIMPSE is in beta state. The probe has been open sourced, the console version is regarded as stable and a number of embedded probes are deployed, busily collecting data. The GUI version is still under development but expected to be stable soon. What remains to be done is to add the GLIMPSE probe to app stores for various platforms.

5. RELATED WORK

As briefly mentioned in the introduction, there are a large number of measurement tools and platforms available for various purposes, by different organizations using different approaches. In this section, we only compare GLIMPSE to those platforms which are closest in terms of either goals, approach or tool set. We also briefly discuss emerging standards in this general space.

Of all measurement platforms available, Dasu[7] is the by far most closely related to GLIMPSE. This is because Dasu is also an end user-based probe. In contrast to GLIMPSE, Dasu originally came as a plugin to a popular BitTorrent client, whereas at the time of writing, a beta version of a standalone Dasu client is also available. Both versions offer multiple small measurement tools similar to GLIMPSE, but the BitTorrent client plugin can also measure the user-generated BitTorrent traffic passively. This is not possible with GLIMPSE as it is a standalone software and was not developed to monitor user-generated traffic. In fact, user-generated traffic is never touched at all for privacy reasons. One of the big differences between Dasu and GLIMPSE is the platforms the software can run on: the Dasu plugin can run anywhere where the BitTorrent client is available, the standalone application is available for Windows and Mac OS X. GLIMPSE on the other hand can also be installed on Linux and mobile operating systems like Android and iOS.

Project Bismark[11] was originally developed to measure the performance of home networks with the help of a custom home gateway. Today, it is also available as Android App, Raspberry Pi image and OpenWrt package. It can perform active and passive measurements depending on the deployment and is intended for researchers to schedule and analyse measurements.

Another measurement project is NETI@Home, a software installed on end-hosts which relies purely on passive measurements. It "collects various statistics about Internet performance"[10] by sniffing packets sent and received on the host it is installed on. There are no active components included and—while also being an end host-based system–it is an example for a measurement project with complete different goals than GLIMPSE.

Netalyzr[4] is a network measurement tool designed to run as a Java applet in the webbrowser of a user[7]. It offers multiple measurements, e.g., to measure the bandwidth and the reachability of different ports. It is limited to the capabilities of a Java applet and can not run continuously in the background of an end users device. It does not offer measurement campaigns but always runs the same suite of tests. There is also an Android app available offering the same measurements as the website.

Various organizations such as Ripe NCC (Atlas) and research institutes such as CAIDA (Scamper)[8] have built and deployed quite successful measurement platforms. Obviously, network measurements are important for our understanding of the operational aspects of the Internet. Foremost however, network operators should be interested in network measurements. Therefore, it should come as no surprise that Standards Developing Organizations (SDOs) such as the IETF and the Broadband Forum have started to work on standards and protocols to manage and control measure-

[7] `http://netalyzr.icsi.berkeley.edu/`
[8] `https://www.caida.org/tools/measurement/scamper/`

ment platforms. Most prominently, the LMAP (Large-Scale Measurement of Broadband Performance) working group of the IETF is busily working on a standard based on YANG and Netconf for this exact purpose[9][8]. The scope of the work however has been restricted to devices under the control of a single administrative entity (the network operator)[5]. This—for an SDO—understandable restriction explains the protocol choice, but it is questionable whether Netconf is a sensible choice for more open platforms such as GLIMPSE, which is based on JSON over HTTPS. This has a number of advantages e.g., it is more likely to pass firewalls. The mPlane project in which GLIMPSE was developed has has also published its protocol as an Internet draft[12] and made a reference implementation available[9].

6. CONCLUSIONS

Implementing a measurement platform is a daunting task for various reasons. The implementation effort is high because in order to involve a representative set of devices, the probe part needs to be cross-platform. Also, server components need to be written and web-interfaces play a significant part for administration, visualization and analysis. A broad skillset is required in the development process and for the operational aspects of the platform.

Making things worse, this relatively high up-front investment comes with a very uncertain success in terms of user adoption. It is unfortunately questionable, whether the platform itself will be a success and whether users are willing to participate. There are platforms that are successful with activating users mostly based on altruism such as BOINC. Whether network measurements as an altruistic motive is sufficient remains to be seen.

This paper however has not focussed on these non-technical issues but solely on the technical aspects of one measurement platform: GLIMPSE. In particular, we have described its design goals, its architecture and its available measurement tools. At the time of writing, only very few functions are missing in the GUI version of the probe before it can be released. The back end, the console version and the web front end are all operational. A probe base of around 15 probes are currently deployed for testing purposes and have been running for months executing various measurements in different environments.

We hope that the measurement and troubleshooting tools that are freely available to users of the app are helping to make GLIMPSE sufficiently attractive for tech-savvy users to install the probe. The troubleshooting interface is being designed for the less technical audience as a value proposition for the app. More information regarding GLIMPSE is available on the project website at `https://www.measure-it.net`. Currently, the console version and a beta of the graphical version can be downloaded from there.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project “mPlane”).

8. REFERENCES

[1] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC ’06, pages 153–158, New York, NY, USA, 2006. ACM.

[2] T. Burbridge, P. Eardley, M. Bagnulo, and J. Schoenwaelder. Information Model for Large-Scale Measurement Platforms (LMAP). Internet-Draft draft-ietf-lmap-information-model-06, IETF Secretariat, July 2015. `http://www.ietf.org/internet-drafts/draft-ietf-lmap-information-model-06.txt`.

[3] M. Faath and R. Winter. Measurements with the masses. In *IRTF & ISOC Research and Applications of Internet Measurements (RAIM) Workshop*, Yokohama, Japan, 2015.

[4] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the Edge Network. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC ’10, pages 246–259, New York, NY, USA, 2010. ACM.

[5] M. Linsner, P. Eardley, T. Burbridge, and F. Sorensen. Large-Scale Broadband Measurement Use Cases. RFC 7536, RFC Editor, May 2015.

[6] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *Network, IEEE*, 17(6):27–35, Nov 2003.

[7] M. A. Sanchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamente, B. Krishnamurthy, and W. Willinger. Dasu: Pushing Experiments to the Internet’s Edge. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[8] J. Schoenwaelder and V. Bajpai. A YANG Data Model for LMAP Measurement Agents. Internet-Draft draft-ietf-lmap-yang-01, IETF Secretariat, July 2015. `http://www.ietf.org/internet-drafts/draft-ietf-lmap-yang-01.txt`.

[9] J. Schoenwaelder and V. Bajpai. Using RESTCONF with LMAP Measurement Agents. Internet-Draft draft-ietf-lmap-restconf-00, IETF Secretariat, July 2015. `http://www.ietf.org/internet-drafts/draft-ietf-lmap-restconf-00.txt`.

[10] J. Simpson, CharlesRobert and G. Riley. Neti@home: A distributed approach to collecting end-to-end network performance measurements. In C. Barakat and I. Pratt, editors, *Passive and Active Network Measurement*, volume 3015 of *Lecture Notes in Computer Science*, pages 168–174. Springer Berlin Heidelberg, 2004.

[11] S. Sundaresan, S. Burnett, N. Feamster, and W. de Donato. BISmark: A Testbed for Deploying Measurements and Applications in Broadband Access Networks. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 383–394, Philadelphia, PA, June 2014. USENIX Association.

[12] B. Trammell and M. Kuehlewind. mPlane Protocol Specification. Internet-Draft draft-trammell-mplane-protocol-00, IETF Secretariat, August 2015. `http://www.ietf.org/internet-drafts/draft-trammell-mplane-protocol-00.txt`.

[9] `https://github.com/fp7mplane/protocol-ri`