Bagnulo, Marcelo; Burbridge, Trevor; Crawford, Sam; Eardley, Philip; Schoenwaelder, Juergen; Trammell, Brian 2014). Building a standard measurement platform. *IEEE Communications Magazine*, 52(5), pp.: 165-173.

DOI: https://doi.org/10.1109/MCOM.2014.6815908

Building a standard measurement platform

Marcelo Bagnulo
Universidad Carlos III de Madrid
marcelo@it.uc3m.es

Trevor Burbridge
BT
trevor.burbridge@bt.com

Sam Crawford
SamKnows
sam@samknows.com

Philip Eardley
BT
philip.eardley@bt.com

Juergen Schoenwaelder
Jacobs University
j.schoenwaelder@jacobs-university.de

Brian Trammell
ETH Zurich
trammell@tik.ee.ethz.ch

Abstract— **Network management is achieved through a large number of disparate solutions for different technologies and parts of the end-to-end network. Gaining an overall view, and especially predicting the impact on a service user, is difficult. Recently a number of proprietary platforms have emerged to conduct end-to-end testing from user premises, however, these are limited in scale, interoperability and the ability to compare like-for-like results. In this paper we show that these platforms share similar architectures and can benefit from the standardisation of key interfaces, test definitions, information model and protocols. We take the SamKnows platform as a use case and we propose an evolution from its current proprietary protocols to standardized protocols and tests. In particular we propose to use extensions of the IETF's IPFIX and NETCONF/YANG in the platform. Standardisation will allow measurement capabilities to be included on many more network elements and user devices, providing a much more comprehensive view of user experience and enabling problems and performance bottlenecks to be identified and addressed.**

***Index Terms*—Multi-domain measurement platforms, network management, standardization.**

I. INTRODUCTION

Network management has always been the Cinderella child: remembered only reluctantly and after dealing with the more exciting business of adding new applications and link layer technologies. This has led to network management comprising a whole series of more-or-less independent solutions cobbled together to tackle what are seen as more-or-less independent problems. From a technical point of view, this approach becomes harder to sustain with the ever increasing scale, scope and rate of change of the Internet: the tens of thousands of autonomously administered networks; the move from centralized client-server applications to fully distributed cloud computing; the proliferation of super-fast access speeds. From a socio-economic perspective, the problem is even more urgent because the Internet is now vital infrastructure; whilst the technical expert knows that the underlying technology is best effort, from a user's perspective the Internet is a critical utility – in fact, in a UK survey in 2011 of "things you couldn't live without", the Internet was rated second, ahead of water, a cooker and a mobile phone, and behind only sunshine!

In order to gain deeper insight about Internet performance, several large-scale measurement platforms - such as SamKnows[1], RIPE Atlas[2] and Netalyzr[11] - have been deployed over the last few years. They encompass several thousands probes distributed across the Internet. Most of these probes are located in homes and can perform active tests against measurement servers located in the core of the Internet, and they are used to assess the performance of the Internet access from the perspective of an end user. Currently, these measurement platforms are mainly used for benchmarking ISPs.

While these platforms have been proven useful, we believe they have not realized their full potential due to their proprietary and by consequence closed nature. All the current platforms use proprietary protocols and proprietary metrics/tests, which has negative side effects, including:

- The platforms cannot interact with each other. Measurement platforms clearly benefit from the so-called network effect, meaning that a platform is more useful as more probes it has. Because of that, platforms would benefit from some form of interaction that would allow the pooling of the probes from multiple platforms. There are different levels of interaction that can be envisioned, ranging from enabling one platform to use the probes of another platform, to supporting coordinated tests that are executed in parallel on different platforms. This is not possible today as deployed platforms are bespoke systems that cannot interact with other systems through standard interfaces.
- The deployment of platforms is expensive. Since each platform uses proprietary protocols, whenever a new party wants to deploy probes to perform its own measurements it cannot buy off-the-shelf components that natively interact to form a measurement platform. Instead it should be possible to have components from multiple vendors.
- Because the tests and the metrics themselves are proprietary, it is not safe to compare results of

[1] http://www.samknows.com/

[2] https://atlas.ripe.net/

supposedly the same metric, since their definitions may differ. This also implies that the results obtained from different platforms cannot be aggregated to obtain a larger data set, severely reducing the usefulness of the obtained results.

The result is that the community has invested a large amount of resources deploying these platforms, but it is not possible to take the full advantage such infrastructure could provide.

In this paper, we propose to extend a set of standard protocols to build a large-scale measurement platform. As a consequence, it would be possible to build a measurement platform using standard off-the-shelf components and build federations of measurement platforms. The context of this work is the efforts being carried out in several standards development organizations (SDOs): at the IETF's LMAP and IPPM working groups; at the Broadband Forum's WT-304 activity; and at the IEEE's project P802.16.3.

In this paper we follow an evolutionary approach. We examine the state of the art in measurement platforms, focusing on the SamKnows platform as a case study and we propose an evolution of the platform towards open standards. We start by describing the current architecture of existing platforms in section II and then we describe how to evolve to a standard architecture in section III, the information model in section IV, standard metrics in section V and standard protocols in sections VI and VII.

II. State of the Art in Measurement Platforms

Here we examine three current active network measurement platforms, RIPE Atlas, perfSONAR, and SamKnows, as background for identifying the requirements for a standard measurement platform.

A. *RIPE Atlas*

Atlas (http://atlas.ripe.net) is an active Internet measurement network developed and run by the RIPE NCC. Small hardware probes, produced by RIPE, perform basic active measurements: ICMP ping, traceroute, and DNS, HTTP, and SSL certificate retrieval queries. It is aimed at engineers with some networking expertise, for whom its primary advantage is that it provides multiple vantage points from which these measurements can be run. Data collected from these probes are also available for research purposes, and provide a data source for RIPE Labs research studies. Users build up credit by hosting RIPE probes, and can use this credit to run their own measurements, as well.

Currently, slightly fewer than 4,000 probes are connected to the network at any given time; basic tests include ping and traceroute to a selection of DNS servers, as well as ping measurements to the first two hops to the Internet, via IPv4 and IPv6.

Control is distributed among a set of RIPE Atlas controllers, running a proprietary control and reporting protocols designed by RIPE running over SSH making use of multiple SSH channels to logically separate different traffic types. Results are centralized at the controllers, and are available via a web interface as well as a RESTful API.

B. *perfSONAR*

perfSONAR [10] is a service-based platform for distributed passive and active measurements. It features a three-layer architecture, including a user interface layer and a services layer backed up by a measurement point layer. The controllers and collectors reside in the services layer.

The basic services provided include:

- a measurement point (MP) service, which performs measurements and is a service wrapper around an entity in the measurement point layer;
- a measurement archive (MA) service, providing access to historical monitoring data and/or storing the results produced by an MP;
- a transformation service (TS), which provides transformation of data (e.g. aggregation, correlation, filtering) provided by other services;
- a lookup service (LS), which provides discovery of other services;
- an authentication service (AS), which works with the lookup service to provide access control restrictions to measurement services; and
- a resource protector service (RP), which avoids overload of shared measurement resources.

The measurements supported include ping and traceroute, as with RIPE Atlas, as well as one-way delay via OWAMP [12] and achievable bandwidth testing. perfSONAR can also integrate passive observation of network flows and SNMP counters. Multiple user interfaces and data analysis tools run atop the perfSONAR service, which is itself concerned primarily with providing distributed access to measurement data and capabilities.

The protocols used between the different components of perfSONAR use NMWG[3] messages, with the XML schema defined by the Open Grid Forum. These messages are carried in a SOAP message, carried over HTTP.

The network is aimed at larger network operators, primarily those associated with research networks connected to Internet2 and/or GEANT, to support performance troubleshooting by their network engineering staffs.

C. *SamKnows*

Currently the SamKnows platform consists of three distinct components: measurement probes, measurement servers and

[3] http://nmwg.internet2.edu/

management infrastructure as depicted in Figure 1.a.

The measurement probes are small Linux-based hardware devices. The SamKnows platform currently accounts for over 40.000 probes. These are deployed into volunteers' homes, inline with their existing home network (the probes can operate as Ethernet bridges or routers). The probes run measurements against measurement servers and real endpoints on the Internet when the end user is not actively using their connection. Cross-traffic is identified on both the wired and wireless interfaces. Probes pull their configuration from the management infrastructure upon start-up and check for updates periodically thereafter. The testing configuration, frequency and other associated parameters are all configured remotely.

Measurement servers are Linux servers running a set of custom server applications to support the tests discussed below. These applications on the measurement servers are relatively trivial; almost all of the measurement logic is performed on the client-side (at the probe). Measurement servers can be deployed anywhere, and probes can be configured to test against them in any desired combination and frequency.

The probes support a wide range of measurements, including:

- UDP round-trip latency and packet loss (long-lived test, sampled periodically)
- UDP one-way jitter (singleton)
- DNS resolution time and failure rate
- Web browsing (measures transaction time to download a page)
- ICMP round-trip latency and packet loss
- Video streaming (measuring buffer under-runs using a fixed streaming rate)
- Latency and packet loss under load
- TCP throughput, downstream and upstream, using one or multiple concurrent TCP connections.

The management infrastructure consists of three key components:

1) The *Data Collection Server*: This handles all management interaction with the probes. All communications are conducted over HTTPS (TLS). The communications protocol is a simple but proprietary one. Requests for configuration updates use GET requests, with the current configuration version in the query string, and the response is the tuple <latest_version, package_url>. If the client determines an upgrade is required, it fetches the configuration package (a tar.gz) from the package_url. Measurement results are uploaded by the probes using the POST method, with a comma-delimited body containing the measurement results (one row per result). Each row contains at a minimum the tuple <probe_id, metric_id, timestamp_utc>. Additional fields are determined by the metric type. Measurement results are written locally to disk on the data collection server and queued for import into the database.

2) The *database*: This stores all probe metadata, recent raw measurement data, and summarised historical data. Raw measurement data is imported in bulk from the Data Collection Server frequently (once per minute). This raw data is kept in the database for approximately 3 months, after which time it is archived to flat files. During this time, summaries of the raw data are generated (at reduced resolution) for presentation to end-users. Multiple database servers are typically deployed (some operating with master-master replication, others as read-only slaves) for redundancy and throughput.

3) *Web reporting portal* and *web services*: This presents the summarised results (stored in the database) to end-users and client-side applications.

Note that only the probe initiates all communications. No results are collected or stored on the measurement servers; that is all handled by the probes.

III. A Standard Architecture for Measurement Platforms

A reference architecture provides a common framework and helps identify the protocols that are needed between the different elements of the architecture. In order to hint how the SamKnows platform could evolve towards our reference architecture, Figure 1.b shows them overlaid. The reference architecture contains the following elements:

- *Measurement agents* perform network measurements. They are pieces of code that can be executed in specialized hardware (hardware probe, like the case of SamKnows) or on a general-purpose device (like a PC or mobile phone). Measurements may be active (the agents generate test traffic), passive (agents observe user traffic), or some hybrid form of the two. A measurement agent can perform two distinct roles: either a *Measurement Client* (hereafter MC) or a *Measurement Server* (hereafter MS). They correspond to the SamKnows measurement probes and measurement servers respectively. Note that the MC initiates a test whilst the MS simply responds to the MC.
- A *Controller* manages MCs by informing them which tests they should perform and when, and also where to report the measurement results and when. We refer to them as the Test and Report Schedules. This is a fundamental component since it is in charge of scheduling measurement activities performed by the MCs.
- A *Collector* accepts measurement results from the MCs, once their tests are complete. The Controller and Collector functions are both performed by the SamKnows Data Collection Server.

We believe these are the main components that it is critical to standardise, although a measurement platform may include

other components such as: a results database, which receives results from Collector(s) and processes and stores them; and data analysis tools, which use the data to isolate faults, present results (similar to the SamKnows Web portal) and interact with an operator's OAM systems.

Having identified the components of the reference architecture, we can easily identify the protocols involved:

- Protocols between the MCs and the MSs. These are the actual tests performed by the platforms and will be covered in section V.
- The protocol between the MC and the Controller. We will call this the Control protocol and we will cover it in section VI.
- The protocol between the MC and the Collector. We will call this the Report protocol and we will cover it in section VII.

An additional component that is relevant and useful to be standardized is an API to retrieve measurement results data from the platform. This would enable a number of applications. For example in the case of an ISP that is using the measurement platform to monitor its network, it would allow it to export the measurement data into its Operation and Management systems in a standard way. While we acknowledge that is a key component it is out of the scope of any standardization work at this point in time and this is why we do not cover this in any detail in this paper.

IV. Information model

Before defining the Control and Report protocols, it is sensible first to define the information model: an abstract, protocol-neutral definition of the data to be transferred. Later we present proposals for protocols and associated data models that implement the information model: NETCONF-YANG for the Control protocol and IPFIX for the Report protocol (Section VI and VII respectively).

We believe this is a powerful approach that will prove useful. While in this paper we argue for a standardized measurement platform, we also believe that some deployments will use other protocols due to environmental constraints. For example, some of the already deployed DSL access networks may decide to use a transport based on the Broadband Forum protocol TR069 [1] based transport. Defining a protocol-independent information model allows these platforms to use different protocols while still exchanging the same information with the same control and reporting capabilities (for example, the Controller could specify the same calendar-based schedule of the same test with the same configuration parameters). Thus a single information model ensures a very high level interoperability between different control and reporting protocols.

The information model encompasses the elements described next.

The Control protocol carries information about the Test and Report Schedules.

The Test Schedule defines which tests a MC has to perform and with what test parameters (including the MSs to test against). It also covers how to reschedule tests in case connectivity is temporarily lost (e.g., a device turned off) or in situations where there is too much cross traffic to execute a test. Finally, it contains information about when the tests should be performed.

The Report Schedule defines when test results are reported (how often), where to (the Collector's address) and in what format, and what to do if reporting fails.

The structure for the information model of the Test and Report Schedule is presented in UML in Figure 2. In this proposal the schedule specifies which pre-configured tests to conduct and which reports the results should be included in. It allows for the test and report configuration to be done irregularly and the test schedule to be updated separately. Schedule timing options include periodic, calendar-based and one-off scheduled or instantaneous tests. The multiple timing options allow for different measuring purposes, for example, calendar based timing allows to target a specific time (for example perform measurement while night-time that are unlikely to disturb the users or peak time, to observe the highest load on the network) while instantaneous tests allows to schedule a tests as soon as possible to troubleshoot an ongoing event.

The information model for the Report protocol includes the MC's identifier; the time of the report; a description of the test (essentially an 'echo' of the Test Schedule, which may be done by reference to a Template, see Section VII); and the actual measurement results (which are highly test-specific).

V. IPPM based Tests

The test protocols are executed between a MC and a MS. The SamKnows platform performs proprietary tests based on standard Internet protocols (e.g. TCP, UDP, ICMP, DNS, etc). For example, in order to measure UDP latency, the SamKnows probes send UDP packets, of a certain length, from a set of ports and use a periodic schedule with a certain rate. When reporting the results, they exclude the outliers using a specific statistical method (e.g. providing the 95th percentile mean or interval). However, another platform measuring UDP latency would most likely make different choices for the above, and so it would not be safe to compare the results obtained from the different platforms.

The IPPM working group at the IETF has defined a large set of metrics for delay, packet loss, jitter and many others. A natural approach would be to use these metrics as the tests in a standard measurement platform. However, if we try to map the tests that are actually performed by the SamKnows platform to the defined IPPM metrics, we find that the IPPM

metrics are not well-defined enough to be useful as test descriptors. The problem is that the IPPM metrics leave too many degrees of freedom to the actual implementation. For example, all the IPPM metrics leave the packet type as an open parameter. This means that referencing a particular IPPM metric does not define whether the packets are TCP, UDP, ICMP or something else. There seems to be a clear gap where further standards could help.

In order to close that gap, we propose to complement the IPPM metric definition by specifying the open parameters that fundamentally affect the test (like the packet type) and leaving open only a few parameters that do not change the nature of the test (like the source or destination address).

We can map some of the tests available in the SamKnows platform to existing IPPM metrics plus additional specification. As a few examples to illustrate this operation:

- SamKnows defines a UDP latency test. IPPM defines a Round-Trip Delay metric in [2]. In order to bridge the gap between the two of them, we need to specify the packet type (UDP packet, payload length and content), the scheduling types (Periodic in this case), the output type (raw or 95th percentile mean). The source and destination ports and addresses are parameters as well as the time of execution.
- SamKnows defines an ICMP packet loss test. IPPM defines a Round Trip Loss metric in [3]. To use the IPPM specification we need to specify the packet type (ICMP echo request and reply messages), the scheduling and the output type, as above. The input parameters are the source and destination addresses as well as the time of execution are open parameters.

By extending the IPPM specifications we create standardized metrics, so that different implementations and platforms produce measurement results that are comparable. We have specified several other tests in [4].

Once there are well defined standard tests then they can be referred to in the Control and Report protocols, so that a MC can unambiguously understand from the Controller what test to perform (and later the Collector knows what test the MC has done).

VI. A NETCONF/YANG BASED CONTROL PROTOCOL

The Control protocol is executed between the Controller and a MC to configure test and report schedules, which was defined in a protocol-neutral fashion in section IV. In this section we propose to use the IETF standard protocol NETCONF and the IETF standard data modelling language YANG. This would be an alternative to the proprietary protocol currently used in the SamKnows platform (section II).

During the last 10 years, the IETF has developed a generic protocol to support device configuration called NETCONF and an associated data modelling language called YANG [5]. The NETCONF protocol provides a remote procedure call mechanism running over a secure transport (SSH or TLS). On top of the generic RPC layer, a number of specific operations are defined to retrieve and edit a device's configuration (e.g., get-config, edit-config, copy-config, delete-config). In addition, there are standard operations to support coarse and fine-grained locking or to implement configuration change transactions over a number of devices. The configuration data manipulated by NETCONF is a structured document conceptually stored in a configuration datastore and serialized using XML. The structure and semantics of the configuration data manipulated by NETCONF is defined using the YANG data modelling language. YANG in addition allows a data modeller to define (i) new operations that extend the core set of generic configuration management operations provided by NETCONF and (ii) notifications that can be emitted by a device when certain events occur.

The NETCONF protocol was originally targeted at devices such as routers or switches in provider backbone networks and large enterprise networks. In these environments, configuration changes are usually pushed to the devices by a management application that can initiate NETCONF sessions as needed. In large-scale measurement platforms, however, the MCs are behind a network address translator and so must establish the communication session and (periodically) pull their configuration from the Controller. The usage of NETCONF, therefore, requires the provision of a 'call home' mechanism allowing devices to initiate the establishment of a NETCONF session. While this is currently not supported by the standardized NETCONF transports, it seems relatively easy to add this feature to the NETCONF over TLS transport: the NETCONF client is configured with a schedule indicating when to establish a TCP connection to a NETCONF configuration server. The NETCONF server then acts as a TCP client establishing the TCP connection and as a TLS client establishing a TLS session. At this point, client and server roles are swapped, such that the MC takes the role of a NETCONF server and the Controller takes the role of a NETCONF client, pushing any pending configuration changes to the device.

Consider the following example. Suppose a Controller wants to request a MC with the IP address 192.0.2.1 (say) to perform a UDP latency test to a destination IP address 203.0.113.1, using source port 23677 and destination port 34567. The test is a singleton test performed at 08:00 UTC. The test is to be performed without cross-traffic and the output type is raw. The use of NETCONF/YANG to send the information in this example is depicted in Figure 3.

While NETCONF can, in principle, also be used to push measurement results to a Collector, it seems that IPFIX is a much better fit for this task as described below. The configuration parameters needed by an IPFIX exporter can easily be configured via NETCONF since there is already a

standard IPFIX configuration data model [6], as described in the next section.

VII. An IPFIX based Report Protocol

As mentioned earlier, the SamKnows platform uses HTTP with a proprietary protocol on top of it to convey test results data from the MCs to the Collector. In this section, we propose the use of the IETF standard IPFIX protocol for that purpose.

IPFIX [7] is a unidirectional, transport-independent export protocol for binary data records, with a focus on network measurement and operations applications. The structure of the data records is described in-band by Templates, which refer to Information Elements (IEs) from a common data model managed by the Internet Assigned Numbers Authority. The basic IEs cover most Layer 3 and Layer 4 measurement needs, and the information elements can be extended [8].

IPFIX organizes data records into Messages. A Message is a sequence of Sets preceded by a Message Header which, among other things, includes an Observation Domain ID (identifying where the records in the Message were measured) and an Export Time (when the Message was originally sent).

A Set contains Records preceded by a Set Header, which contains a Set ID identifying the type of the records the Set contains. Template Sets, identified by a special Set ID, contain Templates, which are sequences of IE identifiers and lengths; these define the fields of the records they describe. A Template's ID matches the Set ID of the Sets containing records described by the Template. Since many records may be described by a single Template, IPFIX's data representation is more efficient than those based on inline record structures (e.g. XML, JSON).

In IPFIX terminology [9], the MC encompasses both the Metering Process (MP) and the Exporting Process (EP), while the Collector is the Collecting Process (CP). IPFIX is used between the EP/MC and the Collector/CP.

We next explore how to use IPFIX to report measurement results by defining a Template.

Part of the information can be conveyed using the fields in the IPFIX header, namely:

- Information about the MA: The MA identifier can be sent in the Observation Domain field of the IPFIX header.
- Information about the time of the report: The Export Time field that can be used to convey this information.

The information describing the test is included in a Template set that contains multiple IEs for each of the different pieces of information we need to convey. This includes:

- An identifier of the metric used for the test. In order to convey that we need to define a new IE, let's call it metricIdentifier.
- An identifier of the scheduling strategy used to perform the test. Again, this will be a new IE, called testSchedule.
- An identifier of the output format. A new IE outputType is needed.
- An identifier of the environment, notably, whether cross traffic was present during the execution of the test. A new IE is needed for this testEnvironment.
- The input parameters for the test. Most of these can be expressed using existing IEs, such as sourceIPv4Address, destinationIPv4Address, etc.

The information describing the test results widely varies with each test, but can include the time each packet was sent and received, the number of sent and lost packets and other information. Again most of these can be expressed using existing IEs, and some new ones can be defined if needed for a particular test.

As an example, suppose a MC wants to report the result from the UDP latency test requested by the Controller in the previous section using IPFIX. The IPFIX report message for this test is depicted in figure 4.

VIII. Conclusions

Network operators have many disparate technology and network management tools, but few that provide an overall assessment of what user experience might be like. Growing interest has led to a number of over-the-top measurement platforms. While these platforms all differ, we have shown that it would be possible to abstract common architectural components and could share some common standard interfaces and data models that would enable a degree of interoperability.

Standardisation discussions have already commenced in the IETF and Broadband Forum. In this paper we have taken the SamKnows platform as a case study and we have proposed an evolution of the platform towards standard protocols. In particular, we have shown how this can be done by using NETCONF and IPFIX for the control and report protocols respectively. We presented other two platforms, RIPE Atlas and perfSONAR to illustrate that several deployed platforms share a similar architecture, which hints that it would be feasible for the different platforms to adopt the proposed standard solution.

While traditional network management tools are applied across the breadth of the network, the emerging end-user premise tools are currently limited to selective deployments on user devices and dedicated measurement boxes. Standardisation will allow these approaches to break through into mainstream network and service management. We can imagine that network operators can embed measurement capabilities in a wide range of network and CPE devices (such as Home Gateways) as well as on internal network elements. All of these devices can be controlled by a single

framework and the measurement results can be collected together to provide a comprehensive end-to-end view, as well as between known network locations. Tests can be operated continuously across all lines or on a randomly selected subset of lines for purposes such as capacity planning and network design. Problems can be investigated through adapting the test configurations, schedules and selection of lines tested. This can help identify problems in the network and with equipment or suppliers, and to isolate whether the issue is in the shared part of the network, a vendor hardware problem affecting many users, unique to a single user line, in the home network or an over-the-top service. Comparable data can also be shared between horizontal or vertically arranged network operators, with service providers and with other parties such as regulators.

IX. References

[1] BBF TR-069, “CPE WAN Management Protocol v1.1”, Issue 1, Amendment 2, Broadband Forum, December 2007.

[2] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", IETF RFC 2681, September 1999.

[3] [RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, August 2012.

[4] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A registry for commonly used metrics. Independent registries", draft-bagnulo-ippm-new-registry-independent-00 (work in progress), January 2013.

[5] Schönwälder, J., Björklund, M. and Shafer, P., “Network Configuration Management Using NETCONF and YANG.” IEEE Communications Magazine 48(9), September 2010.

[6] Muenz, G., Claise, B., and Aitken, P., “Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols”, IETF RFC 6728, October 2012.

[7] Trammell, B., and Boschi, E., “An Introduction to IP Flow Information Export (IPFIX)”, IEEE Communications Magazine 49(4), April 2011.

[8] Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements", draft-ietf-ipfix-ie-doctors-07 (work in progress), October 2012.

[9] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", IETF RFC 5470, March 2009.

[10] Hanemann, A., Boote, J. W., Boyd, E. L., Durand, J., Kudarimoti, L., Lapacz, R., Swany, D. M., Zurawski, J., Trocha, S., "PerfSONAR: A Service Oriented Architecture for MultiDomain Network Monitoring", Proceedings of the Third International Conference on Service Oriented Computing, Springer Verlag, LNCS 3826, December, 2005.

[11] Kreibich, C., Weaver, N., Nechaev, B., and Paxson, V., “Netalyzr: Illuminating The Edge Network” Internet Measurement Conference (IMC), Melbourne, Australia, 2010.

[12] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., Zekauskas, M., “A One-way Active Measurement Protocol (OWAMP)”, IETF RFC 4656, September 2006.

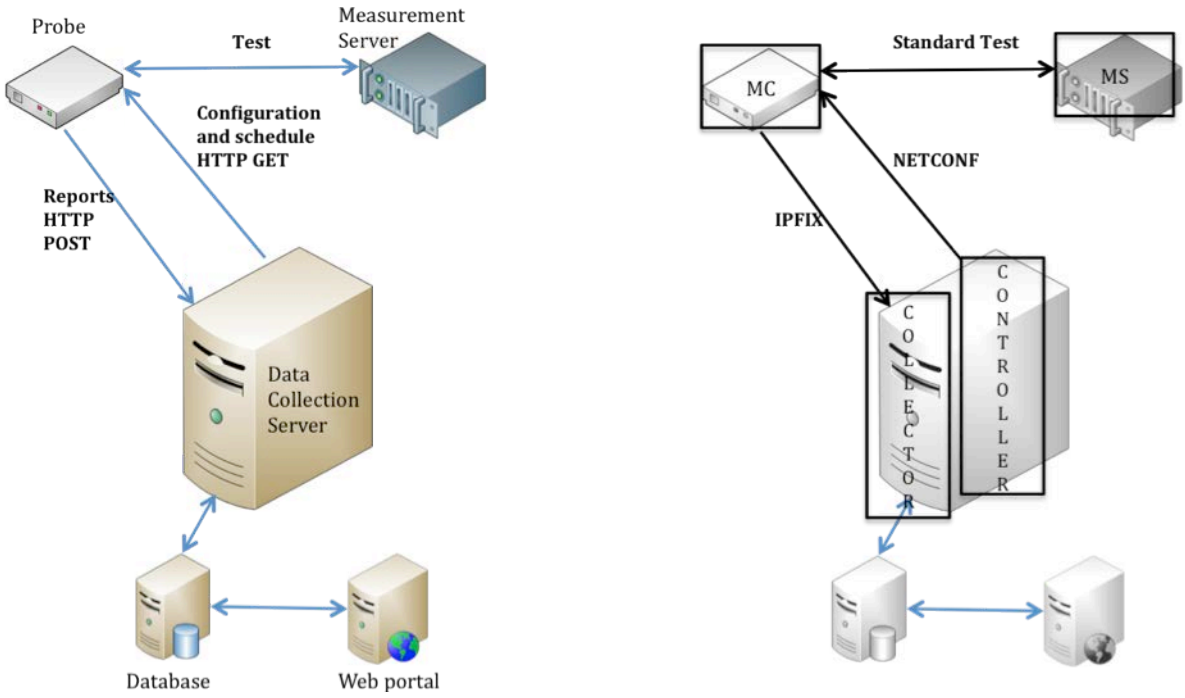

Figure 1: Measurement platform architecture. Figure 1.a) shows the current SamKnows architecture, whilst in Figure 1.b) the proposed reference architecture is overlaid on top of the SamKnows architecture.

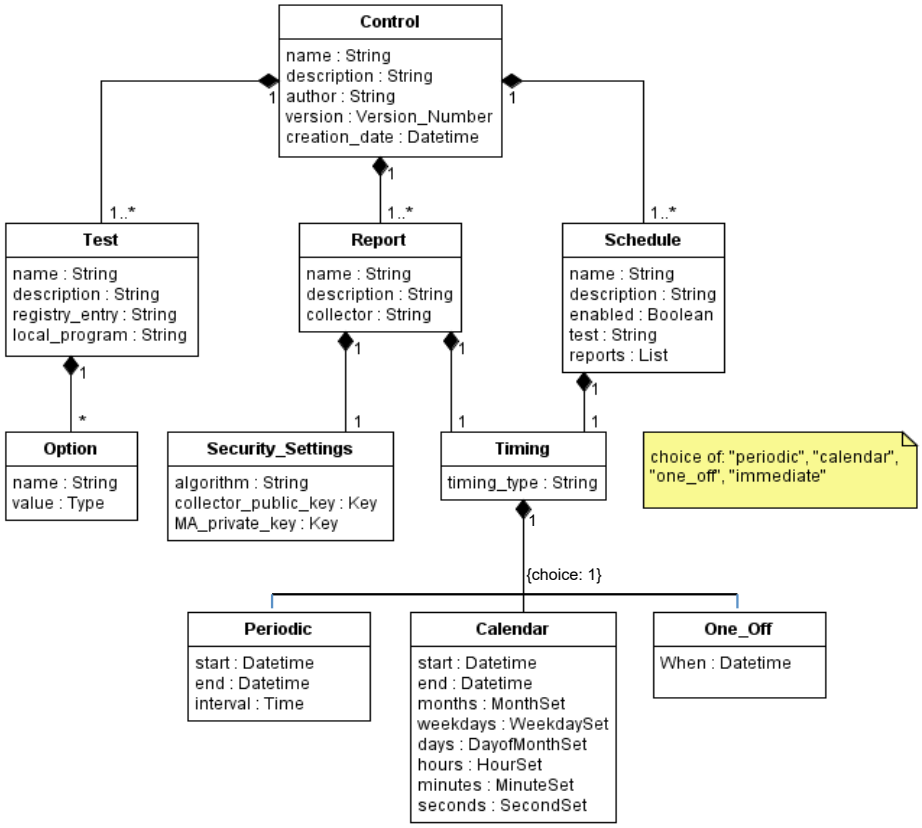

Figure 2: Proposed Information model for the control protocol, in UML

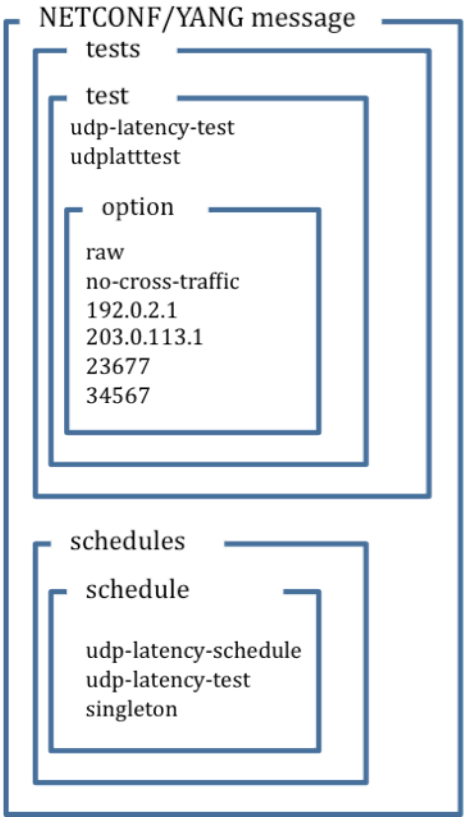

Figure 3: Example of a NETCONF/YANG control message for a UDP Latency test

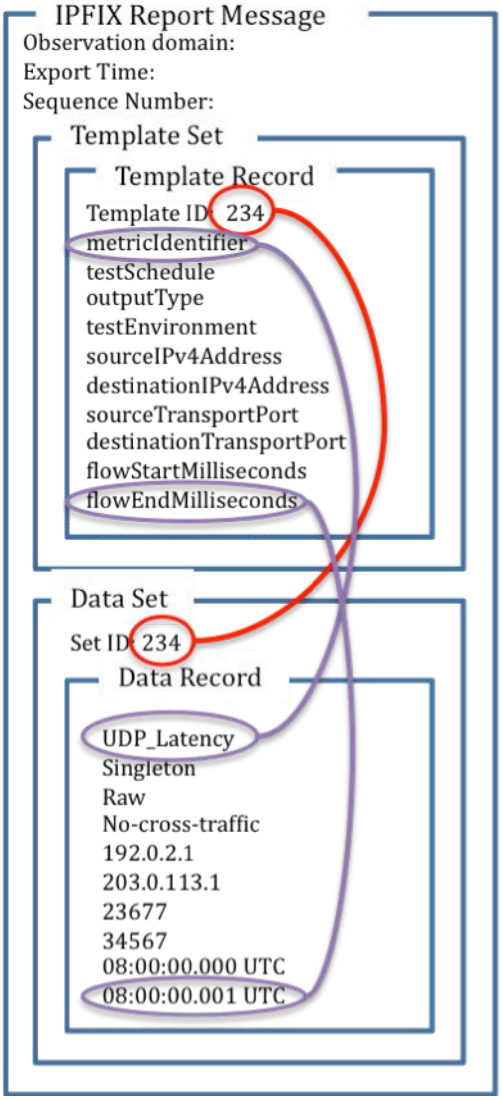

Figure 4: Example of an IPFIX report message for a UDP Latency test