Daisy: Practical Anomaly Detection in large BGP/MPLS and BGP/SRv6 VPN Networks

Alex Huang Feng, Pierre Francois, Stéphane Frenot, Thomas Graf, Wanting Du, Paolo Lucente

Daisy: Practical Anomaly Detection in large BGP/MPLS and BGP/SRv6 VPN Networks

Alex Huang Feng
alex.huang-feng@insa-lyon.fr
Univ Lyon, INSA Lyon, Inria, CITI, EA3720
Villeurbanne, France

Pierre Francois
pierre.francois@insa-lyon.fr
Univ Lyon, INSA Lyon, Inria, CITI, EA3720
Villeurbanne, France

Stéphane Frenot
stephane.frenot@insa-lyon.fr
Univ Lyon, INSA Lyon, Inria, CITI, EA3720
Villeurbanne, France

Thomas Graf
thomas.graf@swisscom.com
Swisscom
Zurich, Switzerland

Wanting Du
wanting.du@swisscom.com
Swisscom
Zurich, Switzerland

Paolo Lucente
paolo@pmacct.net
pmacct.net
Barcelona, Spain

ABSTRACT

We present an architecture aimed at performing Anomaly Detection for BGP/MPLS VPN services, at scale. We describe the challenges associated with real time anomaly detection in modern, large BGP/MPLS VPN and BGP/IPv6 Segment Routing VPN deployments. We describe an architecture required to collect the necessary routing information at scale. We discuss the various dimensions which can be used to detect anomalies, and the caveats of the real world impacting the level of difficulty of such anomaly detection and network modeling. We argue that a rule-based anomaly detection approach, defined for each customer type, is best suited given the current state of the art. Finally, we review the current IETF contributions which are required to benefit from a fully open, standard, architecture.

ACM Reference Format:
Alex Huang Feng, Pierre Francois, Stéphane Frenot, Thomas Graf, Wanting Du, and Paolo Lucente. 2023. Daisy: Practical Anomaly Detection in large BGP/MPLS and BGP/SRv6 VPN Networks. In *Applied Networking Research Workshop (ANRW '23), July 24, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3606464.3606470

ANRW '23, July 24, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0274-7/23/07…$15.00
https://doi.org/10.1145/3606464.3606470

1 INTRODUCTION

Customers subscribing to BGP/MPLS VPN services usually come along with stringent Service Level Agreements. Consequently, Service Providers must be capable of detecting anomalies in their services in a timely fashion, while accommodating for scale. Around 10 thousand L3 VPNs in our Swisscom use case. Long-lasting outages, detected by the customer before the service provider, are detrimental to the perception of service quality, and may dramatically impact the customer business.

The goal of the presented architecture is to provide an anomaly detection solution that scales while being flexible on the following aspects: (i) the dimensions that must be used to detect anomalies are multiple; (ii) VPN customers wear different profiles in terms of normal and abnormal values for such dimensions; (iii) the amount of information collected to produce values for such dimensions is extremely large in such deployments: around 175 thousand messages/second in our use case; (iv) the operating costs for managing an anomaly detection solution must be kept low; and (v) the networking platforms providing the service may come from different vendors and have different monitoring capabilities.

The remainder paper is structured as follows. In section 2, we define what is considered a network anomaly and present the associated challenges behind its detection. In Section 3, we describe the Daisy architecture. In Section 4, we review the ongoing IETF efforts aimed at filling the gaps for a fully open, standard, Anomaly Detection (AD) implementation. And finally, in section 5, we present the first results of Daisy deployment at Swisscom.

2 PROBLEM STATEMENT

We describe some of the challenges associated with customer diversity, and a non-exhaustive list of anomalies targeted by the base recipes from our limited proof of concept deployment setup.

2.1 What is an Anomaly?

An anomaly is defined in this project as follows: *Whatever would let an operator frown and investigate when looking at the collected data-plane, control-plane and management-plane network data relative to a customer*.

That is, we aim at reproducing what human operators do when digging into the data while supporting a customer complaining about the service. We carry out this approach continuously in real time to detect anomalies before the customer does.

In this paper, we suggest a rule based definition of an anomaly per class of customer, assisted by a machine learning based classification of such customers.

2.2 Challenges in customer diversity

Not all VPN customers show the same patterns of service usage and therefore the best approach to detect anomalies may differ from one customer to another. For example, some customers wear clear weekly and day/night traffic patterns, while some others show much flatter network usage. Distinguishing such customers helps in tailoring traffic based anomaly detection that is best suited for each customer profile.

For some customers, not receiving or sending traffic to a VPN site is clearly abnormal while other customers have completely silent sites for some periods of time. Customers with never-silent sites can benefit from specific rules whereby a site becoming silent is a major source of concern.

Unsurprisingly, most customers do not suffer from repeated packet drops on the outgoing interface from the Provider Edge router to the Customer Edge router. However, some specific customers rely on very short IPv4 Address Resolution Protocol and IPv6 Neighbor Discovery timers to perform virtual machine mobility. Numerous, yet short periods of packet drops are thus normal and cannot be used to detect an anomaly for such customers. On the contrary, most customers can be covered by observing packets dropped on the outgoing interface of the Provider Edge router.

As a result, from such diversity, there is no one-fits-all anomaly detection approach applicable which would not suffer from too many false positives or false negatives. Our architecture must thus provide the means to support different sets of detection approaches and parameters for different sets of customers.

To avoid researching and configuring the AD behavior to be applied for each customer, clients are grouped according to *Customer profiles*. To achieve this, clustering techniques are used to detect similar behaviors amongst customers.

Still, customers evolve and new approaches to detect anomalies may arise over the course of an AD deployment. As a result, our architecture must support dynamic reconfiguration of a customer profile and support the introduction of plug-ins developed by third parties.

2.3 Examples of typical anomalies

Typical anomalies we detect in our basic AD recipes are: (i) losing connectivity to a VPN site and therefore losing all its associated traffic and BGP routes is a clear sign of an outage; (ii) having significantly less traffic carried for a customer than the week before for customers whose network usage follow a notable weekly pattern is concerning; (iii) a misconfiguration from the operator when orchestrating updates to their network leading customer traffic to a black hole; (iv) silently failing equipment or physical failures such as a fiber cut impacting customers with misconfigured redundant links; or (v) a buggy software upgrade which the rollback itself fails.

This project is focusing only on customer impacted anomalies at this moment. Anomalies to internal transit of traffic are out of scope. Anomaly Detection triggers may lead to a root cause lying in the internal transit but techniques to detect such anomalies are different.

3 GLOBAL DAISY ARCHITECTURE

Daisy architecture is built around three components: data collection (cf. §3.1), anomaly detection (cf. §3.2), and reporting (cf. §3.3).

3.1 Data collection

First, to gain knowledge from the deployed topology, we need to collect network telemetry data [22] from the network devices. We consider the deployed network as the source of truth instead of a model mimicking the deployed infrastructure.

3.1.1 Data dimensions. Different dimensions can be collected from the network allowing the operator to have different points of view on what is happening on their infrastructure. Data-plane counters characterize the customer usage behavior and allow the operator spotting the congestion in their topology. Control-plane events, on the other hand, give an overview on the reachability of the different propagated BGP paths used by the customer; and management-plane information provide the status at a device interfaces level.

These dimensions are collected through IETF standards allowing service providers to rely on industry standards instead of proprietary solutions. IPFIX [4] is used to export forwarded traffic and dropped traffic counters from the Provider Edge routers. We collect BGP events using BMP [20] and are currently developing YANG push [6] to collect the status and counters at the device interface level. We are currently standardizing UDP-Notif [25] for YANG push configured subscription, which allow the streaming of device-related information directly from the line cards of the routers, hence reducing the stress on the router route processor. UDP-Notif is currently supported on some of the routers deployed in the Swisscom network.

The collection of these three dimensions separately gives the operator a picture of how their network is working. A drop of IPFIX counters means either the customer is sending less traffic or the network is dropping packets somewhere along the path. BGP update events indicate a change of topology and withdraw events imply a loss of a prefix and therefore the loss of a forwarding path. When an interface status has gone from UP to DOWN, either the link has an issue or the operator is manually changing its status. These perspectives separately can already suggest a hint of an ongoing anomaly on the network, but if the operator observes various of these events correlated his certainty can be increased.

3.1.2 Heterogeneous telemetry capabilities. Ideally, all the deployed nodes in the network have all the needed network telemetry capabilities and are integrated to the monitoring platform (onboarded), allowing the operator to have all the perspectives when analyzing the collected data. However, this is far from currently deployed architectures. Usually, service providers have topologies with routers that have different capabilities and are partially onboarded to the monitoring platforms.

There are some of the rules that are effective when all the routers are onboarded. A simple example would be just comparing, for a customer, the total amount of ingress traffic to the total amount of egress traffic for the same time window. If it does not match, some packets are dropped somewhere along the path. However, to implement these type of rules, an updated inventory of onboarded routers correlated to the customers is needed. Inventories are useful to monitoring platforms but only if they are up to date and accurate. For our use case, only the L3 VPN service inventory has been implemented, to specify which L3 VPNs need to be monitored and for customer metadata, such as the type of the customer (B2B or B2C) and human-readable identifiers. Humans are prone to errors and the last thing a monitoring platform needs is having inaccurate information when detecting anomalies. Therefore, we architectured Daisy relying only on deployed network telemetry data.

3.1.3 Data Aggregation and Data Correlation. Aggregation is essential when monitoring large scale networks since it impacts directly on the scaling of the monitoring platform. We perform aggregation at multiple levels: (1) at the node, by using protocols that allow sampling and aggregation directly from the node such as IPFIX [4] and YANG push [6], (2) at the collector, by aggregating again the collected metrics, and (3) at the database, by requesting accumulated metrics on a customer basis. Most of the ingested network telemetry metrics are IPFIX records that are aggregated by a minute interval by the collector. We are currently aggregating 900k IPFIX messages with customer data-plane per second; 50k IPFIX messages without customer data-plane per second; 100k BMP messages per second in average; and 25k YANG push messages per second using OpenConfig YANG models.

When the different dimensions are collected, we correlate traffic counters and BMP events by leveraging *pmacct* [18]. This is performed by correlating traffic received from or forwarded along a customer adjacency with the BGP routes of the peer. The customers are identified using a extended BGP community that is associated with the Route Target used in the VPN routes of the customer. To identify a site, a second set of BGP communities is used allowing Daisy to know for a traffic flow, the related customers and the source and destination site at the same time.

3.1.4 Data storage. Given the amount of collected metrics, operators cannot store the data for a long term. The retention time limit for Daisy is set to 3 weeks in this project due to operational and cost reasons. The real-time metrics are collected and correlated by *pmacct* and streamed to Apache Kafka. Then, a Kafka connector ingest the data to the time-series database, Apache Druid for our use-case, that can be requested by Daisy and the network operations center (NOC) through a web UI, Imply Pivot.

When an incident is reported, the network telemetry data for that period is cloned for Post-mortem analysis. This allows the deployment team to further analyze the AD behavior and improve the parameters of the different rules without losing the data due to the retention time.

3.2 Anomaly Detection (AD)

With the AD architecture, we aim to reproduce the behavior of a network operator. Based on interviews, we execute rule-based algorithms on the network telemetry data.

3.2.1 Concepts. We organize the Anomaly Detection in Checks, Concern Scores, Pipelines, Strategies and Customer Profiles. *Checks* are specific rules producing a concern score based on a given dimension of the customer data. The *Concern score* is a numerical score representing how alarmed the operator should be from the checked metrics. Checks are organized in *Pipelines* which represent a set of rules to be executed in an ordered manner. Each level is executed only if the previous one has passed a threshold. This allows triggering the computing-expensive checks only if another rule raises its concern score. Of course, in the ideally world of infinite computing power, all rule based checks would be executed at the first level meaning that all rules are executed all the time. A *Strategy* is composed by a set of pipelines to be executed. *Strategies* represent an individual approach to detect if there is something wrong for a customer. And lastly, similarly behaving customers are associated with a *Customer Profile* which is bound to a set of strategies to be applied. At regular, configurable intervals, AD is performed on a customer data

according to its customer profile, evaluating whether an alert should be reported based on the application of its set of strategies. The customer is considered as undergoing an anomaly if one of its associated strategies reports an anomaly.

When all the concern scores are computed for a time window, an aggregation based on a weighted sum of the scores is calculated. We call this aggregation *alert level*. It represents if the operator should be concerned given all the dimensions for the considered period. The weight of each check is configurable as part of the strategy configuration. Additionally, we classify dimensions into several categories, e.g., data plane, control plane, and management plane. An amplification factor is applied to the overall alert level when checks in different categories pass their threshold. The larger the number of checks in different categories are triggered, the more severe the alert level becomes. When the resulting alert level passes a configured threshold, a ticket is issued to the network operations ticketing system.

To profit from multithreading CPUs, we organize the execution of AD in *Jobs* and *Worker threads*. A *Job* is a task to be executed. They are composed by the check to be executed, the specific check parameters for the customer, the customer identifier, and the time slot within it must be executed. Jobs are generated by a cron-job. *Worker threads* are the instantiated threads executing *jobs*. They get the necessary metrics from the timeseries database and execute the algorithm from the check according to the parameters set by the *job*.

3.2.2 Operational workers. Some of the rule based checks need to establish a correlation between multi-sourced network telemetry data before computing the actual algorithm. This requires extensive computing resources and are usually not needed to be done in real-time. These operational *jobs* are executed in background with less frequency by *operational workers*, which are implemented as a low-priority extension of the aforementioned worker threads architecture. This allows computing costly correlations and network models with a lower frequency than the actual checks. An example of a check needing prior correlation is a rule to detect if an interface status has gone down. In this case, the collected metrics from the router using YANG push [6] only give us information at a interface level but no correlation to the customer flow is provided. Therefore, a correlation to IPFIX [4] data in AD is needed before computing the check.

3.2.3 Pluggable checks and implemented recipies. The workflow for a check is simple, get the network telemetry data, process it and compute a score representing how concerning the operator should be. Scores are set from 0 to 1, being 1 the most concerned the operator should be. Ideally, checks compute a score based on only one dimension or in other words, from one only perspective. This gives the flexibility of combining independent checks using pipelines and strategies.

The operators are the ones checking network telemetry data daily and they are identifying patterns to be checked. Therefore, we implement a check as an extensible class, allowing network operators to implement their own rules and integrating them into AD easily. Currently, a set of initial checks have already been implemented in Python and are being tested in production data:

- *Flow traffic*. The amount of UDP and TCP traffic compared to last week. If there is a big negative difference, we consider that either we are losing packets or the customer has change their behavior and therefore a score should be raised. This check works very well when the customer has a very predictable behavior in traffic usage.
- *Dropped traffic*. An increasing or a burst of drop traffic counters. If the drop counters spikes or remains high, the concern score is raised.
- *Slope on UDP and TCP traffic*. A radical change of the slope of the traffic counters. If the slope change is substantial, the traffic amount sent by customer has radically changed or an ongoing anomaly is happening.
- *BGP withdraw counts*. When a BGP withdraw event occurs, the prefixes present in this event are not accessible anymore in the network and thus, the packets addressed to that prefixes are dropped by the Provider Edge router. If withdraw events spike, we are raising the score of the concern.
- *BGP peer down*. BGP Peer down events spike. If the BGP peering status goes from UP to DOWN, something has happened to the BGP peering and therefore the concern should be raised.
- *Interface state going down*. If the interface status goes from UP to DOWN, something has happened to the link and therefore the concern should be raised.

3.2.4 Client clustering. Customers have different usage behaviors but, at the same time, a set of them can have similar patterns of usage. To ease the deployment and configuration of the AD strategies, we are using a K-Means algorithm to group customers into profiles. K was tuned based on the silhouette score of the obtained clusters.

3.3 Reporting

After detecting an anomaly, an alert to the network operator is triggered so that they can fix the issue. Sections 3.3.1 and 3.3.2 presents how incidents are managed by operators.

3.3.1 Ticketing system. As thresholds are reached, alerting messages are sent to a Kafka topic. The NOC needs then to consume the messages and investigate to fix the issues. The collected information that led to the alert is made accessible to the operator to ease the investigation. While the alert level

remains high, we do not re-issue a new ticket but update the information related to the alert using the same alert identifier. We re-issue a ticket when the alert level went back to normal for a configurable amount of time and goes high up again. A centralization of the generation of alerts identifiers is implemented to allow operators to connect multiple anomaly detection mechanisms to the same ticketing system.

3.3.2 Replaying. To help with post-mortem analysis and improve AD parameters, we provide a mechanism to replay the AD on past data. Jobs issued to the AD mechanism are associated with a time reference and replaying on past data only requires triggering the same AD code execution using past time references.

4 STANDARDIZATION GAPS

This section presents the ongoing IETF efforts required for a fully open and standard implementation of the Daisy architecture.

4.1 From SNMP to YANG push

Over the last decade, management plane has been monitored using SNMP [3]. SNMP limitations are that (1) it is polling based and (2) the exported values are not structured and therefore there is an increased complexity on the collector. The industry has acknowledged these limitations and is pushing at the IETF to develop new protocols solving those issues.

A new environment has been standardized to manage the networks in the last years: NETCONF [7]. NETCONF messages are XML-encoded allowing to get and push device configurations in a structured way. In parallel, the development of the YANG modeling language [1] has allowed operators and vendors to develop models for these configuration messages.

On the monitoring side, YANG has been integrated to NETCONF with YANG push [6]. Subscription to YANG notifications allow pushing datastore updates on a periodical basis or on an event basis. The network telemetry data sent from the node is structured based on a YANG model allowing both flexibility from the vendor and simplicity on the collection.

Despite the standardization of this push-based telemetry method, important gaps still need to be fixed. First, the NETCONF notification header is still not defined with a YANG model. The impact of this gap is not having a standard way to encode messages with encodings other than XML. We are proposing this model in [16] to allow the node to encode messages in JSON [17] or CBOR [23]. Second, versioning of the pushed YANG messages is still not supported. YANG versioning is still being standardised at NETMOD working group with [24] and [5]. We are proposing an extension to NETCONF notifications in [13] to support versioning in the network telemetry environment. Third, with the UDP-based solution for YANG push [25], packet loss is possible and therefore a way to monitor packet drops is needed. A sequence number in the YANG notification header is proposed in [14]. And fourth, to monitor the stress at each point on the collection chain when pushing these notifications, a timestamp in the header is proposed in [12]. Using this time reference and the timestamp at the collector, congestion can be supervised.

4.2 Monitoring IPv6 Segment Routing

New technologies are emerging and first deployments of Segment Routing over IPv6 [8] are already taking place. The main concern when deploying such new technologies is impacting the customer traffic during migrations. Thus, monitoring during migrations is essential for operators. Yet, a standard way to monitor SRv6 is missing. To solve this and continue monitoring customer flows using the same mechanisms, we are extending IPFIX. We propose exposing the Segment Routing Header [9] with new IPFIX Information Elements in [10]. This visibility both adds Segment Routing information to IPFIX and allows having the correlation to customer flows.

4.3 On-path delay

At this moment, we can monitor (1) the customer traffic behavior using IPFIX metrics, (2) a BGP topology through BMP and (3) a device related status with YANG push. New metrics, such as the delay, can also be very useful when monitoring the network. Though, no standard way exists to monitor it directly from the in-flight packet. Two philosophies are proposed at the IETF. Passport mode: the metrics are gathered from the on-path node and exported only at the last node; and Postcard mode, all the metrics are exported at each node along the path. To monitor the onboarding and facilitate the correlation through IPFIX, postcard mode is preferred. IOAM [2] with Direct Export Option [21] solve this issue by encapsulating the in-flight packet with a header triggering on-path nodes to exports statistics of the flow. Though, a timestamp is missing in the header preventing the computation of the delay. We are proposing an extension to carry the timestamp in the IOAM header in [15] and proposing the IPFIX Information Elements in [11] to have the delay metrics directly from the nodes already correlated to the customer flows.

5 FIRST RESULTS

Daisy has been tested in Swisscom lab and is currently being deployed in production. 11000 nodes are onboarded to the monitoring platform with different monitoring capabilities and AD is at the time of writing this paper deployed for evaluation on a subset of customers of Swisscom BGP/MPLS VPN services. The deployed AD is currently using IPFIX without customer data-plane, BMP messages and YANG push notifications. 6 first service outages, 3 in real-time and 3 in replay mode, has already been detected by our platform.

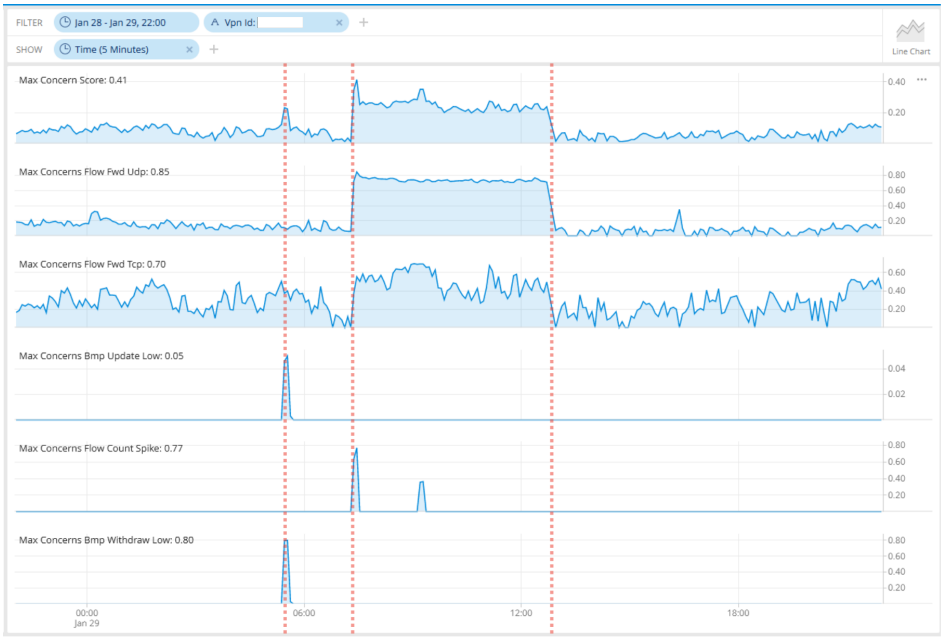

Figure 1: Result of applying rule-based Anomaly Detection to network telemetry data coming from an incident.

To continue testing and improving AD, we are executing AD on past incidents using the replay mode. Figure 1 shows an execution of AD on real customer network telemetry data. The incident is due to a change on the IPSEC configuration on the client side impacting the traffic of this customer. Figure 1 shows 6 rows of concern scores computed by AD. The first row is the alert level, the aggregation of all the computed concern score based on a weighted sum. The following rows are the concern score of the different computed checks. The second and third row are rules raising the score when the forwarded traffic on UDP and TCP are much lower compared to last week. The fourth row is a rule triggered when there is a burst of BGP Update events. The fifth row is checking for spikes on the flow count and the last row is a rule triggered when there is a burst of BGP Withdraw events. The incident started on the first vertical line, on the second line, the incident was reported and on the third, it was resolved. Even if this incident was not due to an internal configuration change, we can see that at the time the incident was reported, the concern score for UDP traffic has gone up to 0.8 and is impacting the alert level. This shows that with the right rules and the right thresholds, alerting based on deterministic rule-based checks can be effective.

6 CONCLUSION AND FUTURE WORKS

In this paper, we present the reasons why Anomaly Detection is important for VPN network operators. As a first step, we argue that to be deployable, we can only resort to a rule-based approach. We review the challenges coming along with real world deployments and describe our AD architecture aimed at fitting with such an environment. We examine the first implemented rule-based checks and review the current gaps at the IETF standardization preventing an optimized deployment of Daisy. Then, we present first results proving that a rule-based approach applying network operators knowledge to the network telemetry data can be effective.

While existing customers can be onboarded in Daisy with a profile based on their previously monitored data, new customers for which no data was ever collected leave us empty-handed. We currently lack a better approach than waiting for data to be produced for a few weeks.

We also plan to study the applicability of the currently defined parameters to other BGP/MPLS VPN networks. As the parameters for the implemented checks increase, we hope to be able to use machine learning techniques to find the optimal parameters for each customer.

The Daisy architecture has been found to be suited to perform trending analysis aimed at detecting upcoming capacity issues and planning for upgrades. We plan to define specific strategies dedicated to this purpose.

7 ACKNOWLEDGEMENTS

We would like to thank David Fernández Blanco, Ahmed Elhassany, Marco Tollini, Severin Dellsperger and Vivekananda Boudia for their invaluable support and contributions to this work.

REFERENCES

[1] BJORKLUND, M. The yang 1.1 data modeling language. RFC 7950, RFC Editor, August 2016.

[2] BROCKNERS, F., BHANDARI, S., AND MIZRAHI, T. Data fields for in situ operations, administration, and maintenance (ioam). RFC 9197, RFC Editor, May 2022.

[3] CASE, J. D., FEDOR, M., SCHOFFSTALL, M. L., AND DAVIN, J. R. Simple network management protocol (snmp). STD 15, RFC Editor, May 1990. http://www.rfc-editor.org/rfc/rfc1157.txt.

[4] CLAISE, B., TRAMMELL, B., AND AITKEN, P. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, Sept. 2013. http://www.rfc-editor.org/rfc/rfc7011.txt.

[5] CLARKE, J., WILTON, R., RAHMAN, R., LENGYEL, B., STERNE, J., AND CLAISE, B. Yang semantic versioning. Internet-Draft draft-ietf-netmod-yang-semver-11, IETF Secretariat, April 2023.

[6] CLEMM, A., AND VOIT, E. Subscription to yang notifications for datastore updates. RFC 8641, September 2019. https://www.rfc-editor.org/info/rfc8641.

[7] ENNS, R., BJORKLUND, M., SCHOENWAELDER, J., AND BIERMAN, A. Network configuration protocol (netconf). RFC 6241, RFC Editor, June 2011. http://www.rfc-editor.org/rfc/rfc6241.txt.

[8] FILSFILS, C., CAMARILLO, P., LEDDY, J., VOYER, D., MATSUSHIMA, S., AND LI, Z. Segment routing over ipv6 (srv6) network programming. RFC 8986, RFC Editor, February 2021.

[9] FILSFILS, C., DUKES, D., PREVIDI, S., LEDDY, J., MATSUSHIMA, S., AND VOYER, D. Ipv6 segment routing header (srh). RFC 8754, RFC Editor, March 2020.

[10] GRAF, T., CLAISE, B., AND FRANCOIS, P. Export of segment routing over ipv6 information in ip flow information export (ipfix). Internet-Draft draft-ietf-opsawg-ipfix-srv6-srh-08, IETF Secretariat, March 2023.

[11] GRAF, T., CLAISE, B., AND HUANG FENG, A. Export of on-path delay in ipfix. Internet-Draft draft-ietf-opsawg-ipfix-on-path-telemetry-02, IETF Secretariat, March 2023.

[12] GRAF, T., CLAISE, B., AND HUANG FENG, A. Support of network observation timestamping in yang notifications. Internet-Draft draft-tgraf-yang-push-observation-time-00, IETF Secretariat, March 2023.

[13] GRAF, T., CLAISE, B., AND HUANG FENG, A. Support of versioning in yang notifications subscription. Internet-Draft draft-tgraf-netconf-yang-notifications-versioning-03, IETF Secretariat, January 2023.

[14] GRAF, T., QUILBEUF, J., AND HUANG FENG, A. Support of hostname and sequencing in yang notifications. Internet-Draft draft-tgraf-netconf-notif-sequencing-01, IETF Secretariat, March 2023.

[15] HUANG FENG, A., FRANCOIS, P., CLAISE, B., AND GRAF, T. On-path delay data field for in situ operations, administration, and maintenance (ioam). Internet-Draft draft-ahuang-ioam-on-path-delay-00, IETF Secretariat, March 2023.

[16] HUANG FENG, A., FRANCOIS, P., GRAF, T., AND CLAISE, B. Yang model for netconf event notifications. Internet-Draft draft-ahuang-netconf-notif-yang-01, IETF Secretariat, March 2023.

[17] LHOTKA, L. Json encoding of data modeled with yang. RFC 7951, RFC Editor, August 2016.

[18] LUCENTE, P. pmacct. http://pmacct.net, 2003 - 2021.

[19] PRESUHN, R. Management information base (mib) for the simple network management protocol (snmp). STD 62, RFC Editor, December 2002. http://www.rfc-editor.org/rfc/rfc3418.txt.

[20] SCUDDER, J., FERNANDO, R., AND STUART, S. BGP Monitoring Protocol. RFC 7854, RFC Editor, June 2016.

[21] SONG, H., GAFNI, B., BROCKNERS, F., BHANDARI, S., AND MIZRAHI, T. In situ operations, administration, and maintenance (ioam) direct exporting. RFC 9326, RFC Editor, November 2022.

[22] SONG, H., QIN, F., MARTINEZ-JULIA, P., CIAVAGLIA, L., AND WANG, A. Network telemetry framework. RFC 9232, RFC Editor, May 2022.

[23] VEILLETTE, M., PETROV, I., PELOV, A., BORMANN, C., AND RICHARDSON, M. Encoding of data modeled with yang in the concise binary object representation (cbor). RFC 9254, RFC Editor, July 2022.

[24] WILTON, R., RAHMAN, R., LENGYEL, B., CLARKE, J., AND STERNE, J. Updated yang module revision handling. Internet-Draft draft-ietf-netmod-yang-module-versioning-09, IETF Secretariat, April 2023.

[25] ZHENG, G., ZHOU, T., GRAF, T., FRANCOIS, P., HUANG FENG, A., AND LUCENTE, P. Udp-based transport for configured subscriptions. Internet-Draft draft-ietf-netconf-udp-notif-09, IETF Secretariat, March 2023.