Enhancing perfSONAR Measurement Capabilities using P4 Programmable Data Planes

Ali Mazloum
University of South Carolina
Columbia, SC, USA
amazloum@email.sc.edu

Jose Gomez
University of South Carolina
Columbia, SC, USA
gomezgaj@email.sc.edu

Elie F. Kfoury
University of South Carolina
Columbia, SC, USA
ekfoury@email.sc.edu

Jorge Crichigno
University of South Carolina
Columbia, SC, USA
jcrichigno@cec.sc.edu

ABSTRACT

As a key element in the Science Demilitarized Zone (Science DMZ) framework, perfSONAR is a crucial tool for monitoring and troubleshooting network performance. This paper presents a system that leverages the flexibility and granularity of P4 programmable data planes to enhance perfSONAR measurements. P4 is a programming language used to define how network packets are processed in the data plane. The proposed system employs a P4 programmable data plane that operates passively on real traffic. This approach ensures real-time per-flow monitoring and detailed insights, assisting administrators in understanding network traffic patterns. Moreover, the system detects microbursts and seamlessly integrates with a regular perfSONAR node. Experimental evaluations show that the proposed system provides full visibility of the real traffic, reports undetectable problems by a regular perfSONAR node, and notifies administrators if the endpoints of the connection cause low performance.

KEYWORDS

perfSONAR, Science DMZ, P4, Data Transfer Node (DTN), Bandwidth-delay Product (BDP), Router's Buffer Size, Microburst, Millimeter Wave (mmWave).

ACM Reference Format:
Ali Mazloum, Jose Gomez, Elie F. Kfoury, and Jorge Crichigno. 2023. Enhancing perfSONAR Measurement Capabilities using P4 Programmable Data Planes. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3624062.3624596

This work was supported by the U.S. National Science Foundation, Office of Advanced Cyberinfrastructure, Award #2118311.

SC-W 2023, November 12–17, 2023, Denver, CO, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0785-8/23/11.
https://doi.org/10.1145/3624062.3624596

1 INTRODUCTION

In an era where scientific and engineering endeavors are generating unprecedented volumes of data, efficient and reliable network infrastructure has become paramount. From the expansive data produced by cutting-edge facilities such as the Large Hadron Collider [14] to the intricate genetic insights derived from portable DNA sequencers [11], the need for seamless data transfer across networks is more pronounced than ever. While conventional enterprise networks can handle operational data, they face significant challenges when dealing with the influx of terabyte and petabyte-scale data generated by these scientific instruments.

In response to these challenges, the Science DMZ has emerged [9]. Aimed to facilitate large data transfers, the Science DMZ provides a dedicated network segment designed explicitly for the high-speed transmission of large amounts of scientific data. At its core, the Science DMZ is supported by four key pillars: specialized Data Transfer Nodes (DTNs) used for high-rate data movement, high-throughput paths characterized by frictionless data flow, performance measurement mechanisms for real-time network assessment, and security protocols optimized for the unique demands of high-performance scientific environments.

While Science DMZs are essential in facilitating data transfers, a critical dimension consists of advanced monitoring and analytical capabilities to visualize and troubleshoot performance issues. At the core of the Science DMZ, perfSONAR [28] emerges as the tool to effectively implement and execute the crucial tasks of monitoring network performance and troubleshooting. Developed by a collaborative effort among research and education institutions, perfSONAR offers a comprehensive framework for measuring, analyzing, and diagnosing network issues. It enables network administrators, researchers, and operators to gain valuable insights into network performance, ensuring the efficient and reliable operation of their infrastructure. Although perfSONAR is a key tool used to diagnose and troubleshoot network issues, it presents shortcomings such as time accuracy, packet level analysis, and custom network measurements.

This paper proposes a system that enhances a regular perfSONAR deployment by using a P4 programmable data plane as a measurement instrument. The system uses passive optical TAPs to create a copy of the traffic going to a perfSONAR node and sending it to a P4 programmable data plane. The P4 programmable data plane processes the metrics and displays them in a dashboard. P4 is a programming language that defines how network devices process

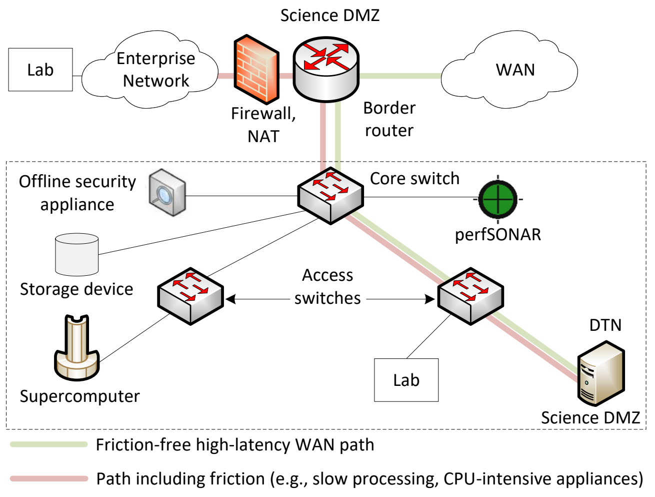


Figure 1: A Science DMZ co-located with an enterprise network.

network data packets [23]. It enables the customization of data plane behavior, allowing network operators to specify how packets are parsed, processed, and forwarded, providing high flexibility and programmability in network operations.

The proposed system minimizes network interference and latency while offering real-time performance insights, making it particularly well-suited for critical and high-performance networks. With this approach, the system implements per-flow monitoring and provides detailed reports for active flows. The reports include individual flow throughput, Round-Trip Time (RTT), packet loss, queue occupancy, and more. This granularity helps network administrators fine-tune network configurations and in troubleshooting problems.

Furthermore, the proposed system introduces microburst detection, a metric critical for network stability. Microbursts are rapid packet influxes that can impair network performance [33]. The system's nanosecond-level reporting of microbursts empowers administrators to pinpoint and address performance bottlenecks. Besides, the system seamlessly integrates with perfSONAR, aligning with the pSConfig command and the default archiver for streamlined reporting and data storage. Administrators can adjust reporting rates and set alert thresholds, enhancing coordination between the system and perfSONAR. These functionalities are implemented as an extension of the pSConfig command.

1.1 Contributions

This paper leverages the flexibility of P4 programmable data planes to enhance the network measurements produced by a perfSONAR node. The main contributions of this paper can be framed as follows:

- Enhancing monitoring without actively engaging with the network. Passive P4 programmable data planes operate over a copy of the network traffic, minimizing interference with regular network operations.
- Introducing real-time performance reporting and per-flow detailed statistics. The system enhances perfSONAR's monitoring abilities on a per-packet basis to improve visibility and accurate network analysis.

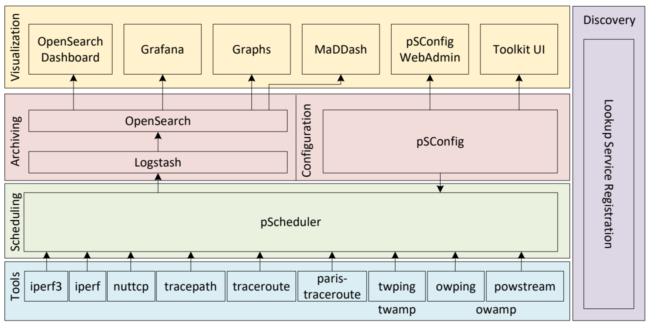


Figure 2: perfSONAR architecture [28].

- Detecting and reporting microbursts. The system offers the capability to pinpoint and address issues arising from microbursts, a challenge often overlooked by conventional monitoring tools.
- Seamless integration with perfSONAR. The system incorporates pSConfig model-based configuration and utilizes the perfSONAR archiver for data storage, facilitating control over reporting rates and alert thresholds throughout a perfSONAR node.

1.2 Paper Organization

This paper is organized as follows. Section 2 provides background on Science DMZ, perfSONAR, and P4 programmable data planes. Section 3 describes the proposed system. Section 4 describes the implementation of the system. Section 5 illustrates the performed experiments and use cases. Section 6 presents the related works. Section 7 concludes the paper.

2 BACKGROUND

2.1 Science DMZ

The Science DMZ is a network architecture specifically designed to address the unique data transfer and performance requirements of large-scale scientific research and data-intensive applications [8]. It emerged as a response to the escalating demand for efficient and high-speed data transfer capabilities within research institutions and facilities, where conventional network infrastructures present limitations to accommodate the massive volumes of data generated by modern scientific experiments and simulations.

Figure 1 illustrates a basic architecture of a Science DMZ co-located with an enterprise network. Notice the absence of a firewall or any stateful inline security appliance in the friction-free path. As data sets traverse from the WAN to a DTN, they can either be locally stored within the DTN or written in a storage device. DTNs have a dual-homed configuration featuring a secondary interface linked to the storage device. This strategy enables the DTN to concurrently receive data from the WAN and transfer it to the storage device, avoiding redundant copying. Within the Science DMZ, users in an internal laboratory can have friction-free access to the data residing on the storage device, experiencing uninterrupted availability. In contrast, users located in a laboratory within the enterprise network, protected by a firewall, experience reasonable performance when

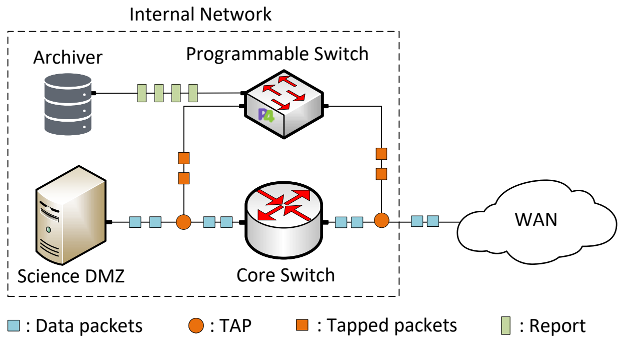


Figure 3: Proposed system overview. A P4 programmable switch calculates measurements from the passive traffic received from a pair of TAPs. The TAPs mirror the traffic entering and existing the core switch.

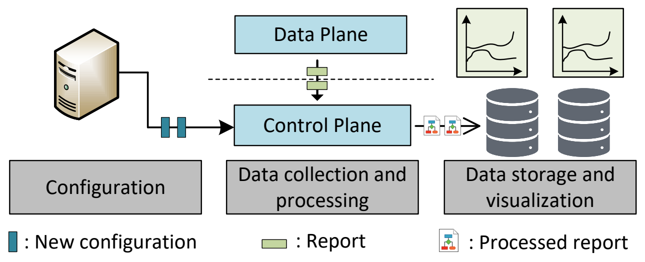


Figure 4: Proposed system architecture.

accessing stored data within the Science DMZ. This differentiation results from the reduced latency between the Science DMZ and enterprise users, which diminishes the performance impact caused by retransmissions resulting from security appliances.

2.2 perfSONAR

perfSONAR is a monitoring framework designed to measure the performance of network infrastructures. It plays a crucial role in enhancing network operations, troubleshooting issues, and optimizing data transfer. Developed through collaborative efforts within the research and education community, perfSONAR addresses the challenges posed by the ever-increasing demand for high-speed and reliable data transfer across distributed networks.

Figure 2 shows the components of perfSONAR architecture. perfSONAR comprises a modular architecture integrating open-source tools (e.g., iPerf3, ping, traceroute) and perfSONAR-specific modules (e.g., pScheduler, pSConfig, MaDDash, and others). The implementation of perfSONAR consists of distributed software agents deployed across network nodes. These nodes collaborate to collect, analyze, and report network performance metrics. The framework is designed to be vendor-neutral and adaptable to different network environments, making it suitable for a wide range of applications. These perfSONAR nodes can coordinate measurement tests using standard protocols to gather performance data, such as throughput, latency, packet loss, and jitter. The collected data is aggregated and presented in a user-friendly format.

2.3 P4 Programmable Data Planes

Traditional network devices operate on fixed, proprietary hardware and software architectures, which limits the type of processing that a switch or router can perform. The surge in diverse applications, from cloud computing to IoT, demanded a network infrastructure that can dynamically adapt and scale. P4 emerged in response to this demand, enabling the implementation of programmable data planes that can be tailored to the unique needs of various use cases [1, 17, 23, 25]

P4 is a programming language that defines how network devices process network data packets. It enables the customization of data plane behavior, allowing network operators to specify how packets are parsed, processed, and forwarded, providing high flexibility and programmability in network operations. Typically, a P4-compatible device has multiple pipelines that can process packets concurrently. Programmers can define different processing behavior for each pipeline. A pipeline contains an Ingress programmable block and an Egress programmable block. The two programmable blocks utilize a programmable parser to fetch the headers of the incoming packets, match-action tables to implement the processing behavior, and a deparser to reassemble the packets.

The proposed system leverages on the flexibility of P4 programmable data planes to enhance perfSONAR visibility and troubleshooting capabilities. Although the latest version of perfSONAR (version 5) [27] aggregates the tests into a single value, the authors are aware that perfSONAR can be as granular as the tool it uses (e.g., iPerf, ping). The latest version of perfSONAR processes the measurements collected by the Tools layer using Logstash [28]. By default, Logstash is programmed to aggregate the result of a perfSONAR test. For throughput tests, Logstash only reports the average value. For RTT tests, Logstash reports the minimum, the maximum, and the mean RTT. Because the default configuration does not reflect the maximum granularity supported by perfSONAR, Logstash's filters can be adjusted. However, this minor granularity addition only applies to the performed active tests and does not allow perfSONAR to have per-flow visibility. Having per-flow visibility is one of the main goals that perfSONAR is aiming to achieve which cannot be done without operating passively on the real traffic [30]

3 PROPOSED SYSTEM

3.1 Overview

The proposed system utilizes a P4 programmable switch to passively monitor real traffic. The system is depicted in Figure 3. A legacy switch (core switch) connects the Science DMZ to the Internet. A TAP duplicates the traffic arriving at and departing from the legacy switch to the programmable switch.

3.2 Architecture

The data plane of the programmable switch collects per-flow measurements where flows are characterized by their 5-tuple (source IP, destination IP, source port, destination port, and protocol). The collected measurements are extracted by the switch's control plane for additional processing before being reported to a perfSONAR archiver. The rate of measurement extraction by the control plane is a variable that a perfSONAR node can configure. The proposed system also allows the administrator to define an alerting threshold

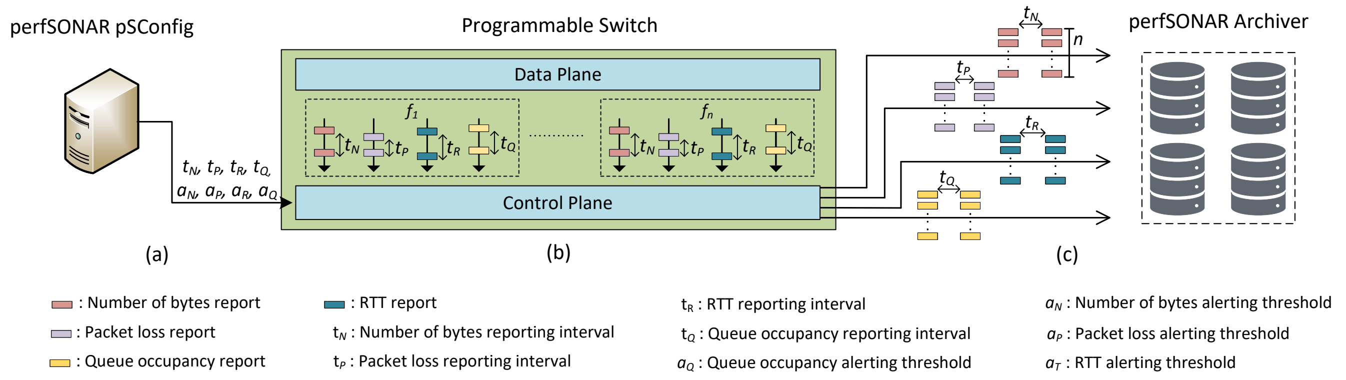

perfSONAR pSConfig
Programmable Switch
Data Plane
Control Plane
perfSONAR Archiver
$t_N, t_P, t_R, t_Q,$
a_N, a_P, a_R, a_Q
f_1
f_n
t_N
t_P
t_R
t_Q
n
(a)
(b)
(c)
: Number of bytes report
: Packet loss report
: Queue occupancy report
: RTT report
t_N : Number of bytes reporting interval
t_P : Packet loss reporting interval
t_R : RTT reporting interval
t_Q : Queue occupancy reporting interval
a_Q : Queue occupancy alerting threshold
a_N : Number of bytes alerting threshold
a_P : Packet loss alerting threshold
a_T : RTT alerting threshold

Figure 5: The interaction between the different components of the proposed system. (a) The perfSONAR pSConfig module is responsible for configuring the control plane of the programmable switch by updating the reporting speed and the alerting thresholds of the four monitored metrics. (b) The data plane is responsible for collecting the measurements, and the control plane is responsible for extracting and processing the measurements from the data plane before forwarding them to the archiver. (c) The perfSONAR archiver stores the measurements received from the control plane.

for each monitored metric. If the threshold is exceeded, the switch alerts the administrator and increases the rate of measurement collection in order to get higher visibility. The number of reports per second, the alerting threshold, and the increase in the reporting rate when the threshold is exceeded are all customizable and can be configured from the configuration layer of the perfSONAR node (i.e., using pSConfig).

As depicted in Figure 4, the proposed system has four main components. The first component is the data plane of the programmable switch. The data plane is responsible for monitoring the traffic. The second component is the control plane of the switch. The control plane is responsible for extracting and processing the measurements from the data plane before sending them to the perfSONAR archiver, which represents the third component in the proposed system. Besides storing the collected data, the archiver allows the real-time visualization of the metrics extracted from the data plane. The last component is the configuration daemon. The daemon runs on a perfSONAR node and configures the control plane of the programmable switch.

As indicated by Figure 5(a), the configuration layer of perfSONAR is responsible for configuring the time intervals during run time. The four time intervals, t_N, t_P, t_R, and t_Q represent the rate at which the control plane extracts the number of bytes, packet losses, RTT, and queue occupancy measurements, respectively from the data plane. This layer can also configure alerting thresholds (a_N, a_P, a_R, and a_Q) for the four metrics. If one of the alerting thresholds is exceeded, the control plane notifies the administrator and increases the collection rate to a value defined by the administrator.

The interaction between the data plane and the control plane is shown in Figure 5(b). The data plane receives packets from *n* different flows representing the real traffic. For each of the *n* flows, the data plane continuously monitors four metrics: the number of bytes, the RTT, the retransmission rate (number of packet loss), and the queuing delay (queue occupancy). The control plane utilizes the APIs provided by the switch manufacturer to access the measurements maintained by the data plane at run-time. The control plane extracts and reports the measurements of all the flows simultaneously, as shown in Figure 5(c).

Table 1: Comparison between a regular perfSONAR deployment and the enhanced perfSONAR integrated with P4.

	Regular perfSONAR	**P4-perfSONAR**	**Comment**
Measurements type	Active measurements	Active and passive measurements	Unlike active measurements, passive measurements induce no overhead on the network
Measurements source	Injected traffic	Real traffic	Measurements not collected from the real traffic might not be representative
Granularity	Limited granularity	Per-flow and per-packet granularity	Tools utilized by the rgular perfSONAR are not designed to support sub-millisecond granularity
Visibility	Limited by the active tests	Real-time visibility over all data transfers	P4-perfSONAR provides visibility as long as there is a data transfer
Microburst detection	Not supported	Detects and reports microbursts with nanosecond granularity	Microbusrt detection requires monitoring the queue on a per-packet basis
Detecting the source of the connection limitation	Not supported	Detects and reports limitations caused by the endpoints	The regular perfSONAR does not detect problems at the endpoints because it does not operate on their traffic

3.3 Features

Table 1 compares the regular perfSONAR and perfSONAR integrated with P4 (refered to by P4-perfSONAR in the table). The features of the proposed system and their added value are explained in the remainder of this section.

3.3.1 Passive Measurement. Unlike active measurement, passive measurement does not add overhead or consume network resources. Through tapping, the programmable switch operates on a copy of the actual network traffic. The switch does not actively participate in the network, and consequently, there is no risk of increasing network latency, bottlenecks, or affecting the overall performance. This makes passive monitoring suitable for critical and high-performance networks where minimal interference is crucial [13].

3.3.2 Per-flow Monitoring. The proposed system significantly enhances the monitoring capabilities of perfSONAR. Typically, perfSONAR runs periodic active measurements to monitor the network. This provides a periodic estimation of the state of the network (e.g., available bandwidth); however, it lacks the ability to report the performance of the real traffic. The proposed system enhances the visibility of perfSONAR by reporting the network performance in real-time. Besides, the system provides per-flow detailed reports. The data plane updates the statistics of each flow on a per-packet basis. The statistics are continuously updated and maintained by dedicated stateful registers where the data plane can track 2048 active flows simultaneously. Because providing a per-packet report overwhelms both the control plane and the data collector (i.e., perfSONAR archiver), the administrator has the flexibility to configure the reporting interval, which can be in the order of milliseconds.

The proposed system also provides a detailed report for each terminated long flow. The report uses the nanosecond granularity of the switch to provide the flow's start and end times. The report also includes the total number of packets sent, the total number of bytes, the average throughput, and the number and percentage of retransmissions. This report helps administrators understand the type of traffic they are interacting with and consequently helps them tune their network.

3.3.3 Microburst Detection. In addition to enhancing the visibility of perfSONAR, the proposed system reports microbursts, a metric not supported by any tool used by perfSONAR. Microbursts occur when a massive amount of packets arrives in a very short duration. The duration of a microburst can be lower than a hundred microseconds [33], making it undetectable without per-packet visibility [5]. Yet, microbursts can significantly reduce the performance of networks. The burst of packets fills a router's buffer, increasing the flows' RTT. If the burst is big enough, it might bloat the buffer, leading the router to drop all the packets arriving during the burst.

```
1.  psconfig config-P4 --metric throughput --samples_per_second 1

2.  psconfig config-P4 --metric RTT --samples_per_second 2

3.  psconfig config-P4 --metric queue_occupancy --alert --threshold 30
        --samples_per_second 10
```

Figure 6: Configuration example.

Because of the nature of some TCP congestion control algorithms, packet losses can lead to significant performance degradation. The proposed system is able to report the starting time and the duration for any microburst with nanosecond granularity. To do so, the switch monitors the queue occupancy on a per-packet basis. The sudden increase in queue occupancy is reported as a microburst.

3.3.4 Identifying if the Connection is Limited by the Network or by the Sender/Receiver. In a network, connections can encounter various levels of restrictions, stemming from different elements of the communication process: the sender, the receiver, or the network infrastructure itself [15]. The proposed system is capable of determining whether these limitations arise from the network or from the sender/receiver.

Specifically, the system can identify whether a connection's performance is restricted by the network or by the sender/receiver. If the network is the source of the limitation and the underlying cause isn't related to queuing issues (e.g., small buffer size, microbursts), it might be necessary to conduct active measurements to precisely pinpoint the root of the problem.

However, engaging in active measurements is not advisable when a connection's restriction originates from the sender or receiver. This is due to two key reasons. Firstly, introducing active measurements in such cases would introduce additional overhead to the network, potentially exacerbating the issue. Secondly, these measurements likely fail to accurately identify the underlying problem when it's sender/receiver-related. This is because PerfSONAR nodes do not process the traffic generated by the DTNs. perfSONAR collects measurements mainly from the active tests performed between two perfSONAR nodes.

In essence, the system's ability to differentiate between network-related and sender/receiver-related limitations offers valuable insights into the appropriate course of action. Active measurements can be selectively applied based on the nature of the limitation, ensuring efficient troubleshooting.

3.3.5 Seamless Integration with perfSONAR. To support seamless integration with perfSONAR, the proposed system had to satisfy two conditions: 1) the programmable switch should be configurable through the configuration layer of perfSONAR (i.e., through pSConfig module), and 2) the switch should use perfSONAR archiver to store the collected data. Regarding the first requirement, new functionality has been added to the pSConfig module, which is responsible for configuring the switch's control plane at run time. The added functionality allows a perfSONAR node to control the number of reports per second sent by the control plane to the archiver. Consider Figure 6. *config-P4* is the added command to pSConfig through which the node configures the control plane. The *--metric* parameter specifies to which metric the configuration is to be applied. The *--samples_per_second* specifies the number of samples to be reported by the control plane to the archiver. The first line sets the rate of throughput reports to one per second, and the second line sets the number of RTT reports to two per second. The configuration will be applied to all metrics if the administrator does not use the *--metric* parameter. The *config-P4* command also allows the administrator to define a threshold for each metric that will trigger an alert if its value is exceeded. In the third line, the rate of queue occupancy

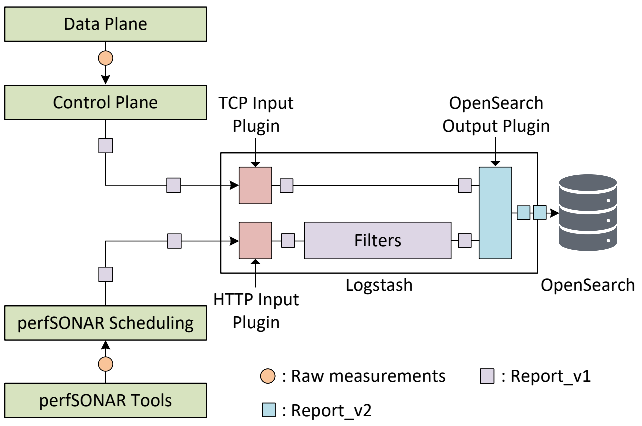


Figure 7: Connecting the proposed system with perfSONAR's archiver. The control plane formats the raw measurements extracted from the data plane to create structured reports (Report_v1). Logstash receives the reports via the TCP input plugin, adds the metadata required by the OpenSearch database, and forwards the new reports (Report_v2) to OpenSearch.

reports will be set to 10 reports per second if the queue occupancy exceeds 30%.

Regarding the second requirement, the control plane of the programmable switch is connected to Logstash through the TCP input plugin [12]. Logstash is the data processing pipeline used by perfSONAR to process the measurements reported by the Tools layer. Logstash ingests the data through the input plugins, transforms and processes it through the filters, and ships it to the database through the OpenSearch output plugin. As depicted in Figure 7, the proposed system leverages the current output plugin used by the standard perfSONAR implementation to store the data. The switch's control plane restructures the raw measurements extracted from the data plane to allow Logstash to operate on them. Logstash then adds the metadata required by the OpenSearch database by utilizing the OpenSearch output plugin. After that, Logstash ships the final version of the reports to the archive (i.e., the OpenSearch database).

4 IMPLEMENTATION

To support per-flow monitoring, the data plane of the switch is programmed to group packets into flows using the hash of the 5-tuple. After that, the data plane detects long flows using count-min sketches (CMS) [7]. After detecting a long flow, the data plane reports the ID of the flow (i.e., the hash of the 5-tuple), its source and destination IP, and its reversed ID. The reversed ID is calculated by reversing the hashing order of the source and destination fields of the 5-tuple (the source and destination IP and the source and destination ports). The reversed ID is used to identify the flow to which an acknowledgment packet (ACK) belongs, allowing the data plane to calculate the RTT of TCP flows.

4.1 Throughput Monitoring

For each flow, the programmable switch calculates the number of bytes, the queuing delay, the packet losses, and the RTT. The calculated measurements are stored inside stateful registers which are

Algorithm 1: RTT and packet loss calculation

```
Data plane():
    $hdr \leftarrow pkt.extract(headers)$;
    $flow\_ID \leftarrow hash(5\_tuple)$;
    $pkt\_type \leftarrow get\_type(hdr.tcp.flag, hdr.ip.total\_len)$;
    if $pkt\_type == Seq$ then
        $prev\_seq\_no \leftarrow prev\_seq\_register[flow\_ID]$;
        if $hdr.tcp.seq\_no < prev\_seq\_no$ then
            $pkt\_loss\_register[flow\_ID]$ += 1;
        else
            $prev\_seq\_register[flow\_ID] \leftarrow hdr.tcp.seq\_no$;
        $rev\_flow\_ID \leftarrow hash(rev\_5\_tuple)$;
        $exp\_ack \leftarrow hdr.tcp.seq\_no + (hdr.ip.total\_len - 4 * hdr.ip.ihl - 4 * hdr.tcp.data\_offset)$;
        $pkt\_sig \leftarrow (rev\_flow\_ID, exp\_ack)$;
        $eack\_register[pkt\_sig] \leftarrow time.now()$;
    else if $pkt\_type == ACK$ then
        $pkt\_sig \leftarrow (flow\_ID, hdr.tcp.ack\_no)$;
        $arrival\_time \leftarrow eack\_register[pkt\_sig]$;
        if $arrival\_time != 0$ then
            $rtt \leftarrow time.now() - arrival\_time$;
            $rtt\_register[flow\_ID] \leftarrow rtt$;
```

indexed by the flow ID. The most straightforward metric to calculate in the data plane is the number of bytes. The programmable switch utilizes the total length field of the IPv4 header to get the packet's length in bytes. After that, the switch updates the register cell that maintains the number of bytes of the current flow. The control plane uses the APIs provided by the manufacturer of the switch to read the register values. The throughput is then calculated in the control plane by dividing the number of bits over the reporting duration.

4.2 Queue Occupancy Monitoring

Although monitoring queue occupancy on a fine-grain resolution is not supported by perfSONAR due to software limitations, it can be easily performed by a programmable switch. The programmable switch calculates the time packets spent inside the core switch to calculate its queue occupancy. The TAPs duplicate each packet twice. The first duplication occurs at the ingress port of the core switch, and the second duplication occurs at the egress port of the switch. The programmable switch calculates the time difference between the arrival of the two copies to calculate the queuing delay. The queuing delay of the flow to which the packet belongs is stored in a stateful register. The control plane samples the queuing delay and calculates the queue occupancy. Queue occupancy equals the queuing delay divided by the buffer size of the core switch.

However, because the duration of microbursts can be in the order of tens of microseconds, the sampling approach might not detect them. For this, microburst detection should be fully implemented in the data plane. The data plane measures the queue occupancy on a per-packet basis. Any sudden increase can be instantaneously detected by the data plane. The data plane can then report the event to the control plane.

4.3 RTT and Packet Loss Monitoring

The last two metrics calculated by the data plane are the RTT and the packet losses. The proposed system adopts RTT and packet loss calculation algorithm from [6]. RTT represents the time between a packet being sent and its corresponding acknowledgment received. As the sender transmits data packets to the receiver, the receiver acknowledges the receipt of these packets by sending acknowledgment packets back to the sender. The acknowledgment number in these packets represents the next expected sequence number, indicating that all bytes up to that number have been successfully received. If the sender receives an acknowledgment packet with an acknowledgment number that is less than the last transmitted sequence number, it signifies that the receiver has not yet confirmed the reception of the data. This inconsistency indicates a requirement for retransmission, which, in turn, implies a packet loss scenario.

Any packet with a non-zero payload might be acknowledged by a future ACK number. The sequence number of the packets is used to calculate the expected future ACK number (eACK). When an ACK packet is received, the ACK number is extracted and compared against the previously stored eACKs. If a match occurs, the switch decrements the current timestamp from the timestamp of the matched eACK. This time difference is the RTT of the flow.

The pseudocode of the RTT and packet loss calculation program is summarized in Algorithm 1. The data plane first parses the packets' headers and obtains the *flow_ID*. It then identifies the packet type (i.e., whether it is a data packet or an ACK packet). The packet type is obtained by inspecting its TCP flag and total length. For data packets (referred to by Seq in the pseudocode), the data plane compares the current packet's sequence number with the previous packet's sequence number (*prev_seq_no*). If the current sequence number is smaller, the switch increments the count of packet losses because retransmission has occurred. Otherwise, the switch updates the value of the previous sequence number to the current sequence number. After that, the data plane calculates the reversed *flow_ID* by hashing the 5-tuple in a reversed order. Then, the data plane calculates the eACK number of the current packet. The *flow_ID* and the eACK are used to create the packet's signature. The arrival time of the packet is then stored in a register indexed by its signature (*eack_register* in the pseudocode).

For ACK packets, the data plane creates a signature using the *flow_ID* and the ACK number. After that, the created signature is used to index *eack_register*. The extracted value represents the timestamp of the data packet that corresponds to the current ACK packet. The RTT is then calculated by subtracting the retrieved timestamp from the current timestamp. The final step is to store the RTT in the *rtt_register* at the index *flow_ID*. The control plane obtains the per-flow packet losses and per-flow RTT by reading the *pkt_loss_regsiter* and the *rtt_register*. The control plane can also report the percentage of packet losses by dividing the number of packet losses of a flow by the number of packets sent by the same flow.

4.4 Detecting Flows not Constrained by the Network

Besides calculating different metrics, the data plane can identify if a flow is limited by the network or by the sender/receiver. To determine whether a flow's limitations stem from network factors or sender/receiver constraints, the programmable switch employs a method that involves monitoring the fluctuation of the flight size (the count of transmitted bytes awaiting acknowledgment) in relation to observed packet losses. If the flight size remains stable despite the absence of packet losses (implying a lack of increment), the limiting factor is likely the sender or the receiver. Conversely, when the flight size expands in conjunction with detected packet losses, the network is likely the influencing factor behind the limitations. This approach is drawn from insights presented in the work by Ghasemi et al. [15].

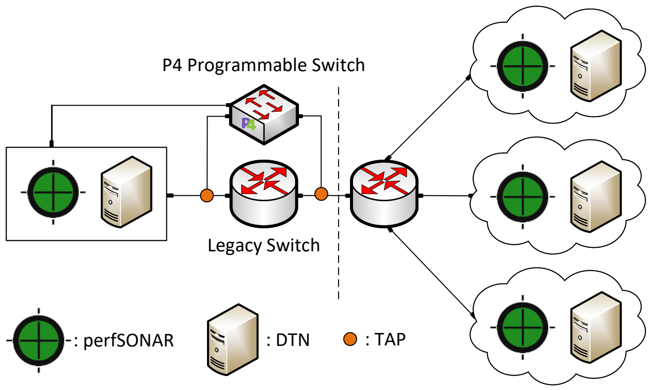


Figure 8: Topology used for the experiments.

5 EXPERIMENTATION

It's important to emphasize that this paper does not seek to replace perfSONAR; instead, its objective is to complement and enhance its capabilities. The system's intention is to augment perfSONAR by providing finer detail to certain metrics (such as throughput, RTT, and packet loss) and by introducing support for novel measurements (including queue occupancy and microbursts). This section will delve into an evaluation of the integration of a P4 programmable switch with perfSONAR. First, the section will highlight the improved granularity that supplements the existing perfSONAR metrics (throughput, RTT, and packet loss). Following that, the section will show the significance of the newly added metrics in enhancing the functionality of perfSONAR.

5.1 Topology Setup

In reference to Figure 8, the experimental topology encompasses an internal network and three external networks. Each of these networks is equipped with a DTN for data exchange and a perfSONAR node dedicated to monitoring network performance. The interconnections between networks involve two legacy switches. Notably, the link interconnecting these switches acts as a performance bottleneck, operating at a throughput of 10 Gigabits per second (Gbps).

To perform passive measurements, two passive optical TAPs capture packets arriving at the ingress and egress ports of the legacy switch directly linked to the internal network. The Optical Taps are special devices made by Fiber Instrument Sales. They work with single-mode fiber and can handle speeds up to 100Gbps [29]. The captured packets are then directed to a P4 programmable switch. The programmable switch used is Edgecore Wedge100BF-32X switch

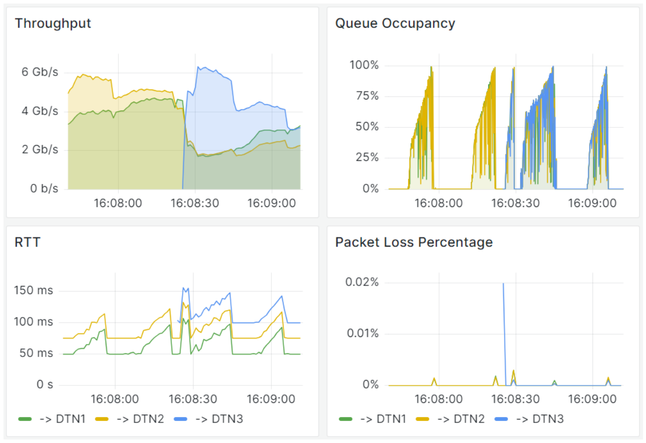


Figure 9: Per-flow measurements.

(Intel Tofino) [21]. The programmable switch's control plane is connected to the local perfSONAR node, leveraging its archiving capabilities to store collected measurements. The data generation process is facilitated by the iPerf3 tool [10], which generates controlled traffic patterns for testing and evaluation purposes. Additionally, to visually present the collected data in real-time, the Grafana visualization platform [18] is employed.

In all tests, there will be an exchange of traffic from the internal DTN to all external DTNs. Unless specified otherwise, the interval used by the programmable switch to report the measurements is one second, and Grafana will group the reported measurements by their destination IP address (i.e., destination DTN). The RTTs between the local DTN and the three external DTNs are 50, 75, and 100 milliseconds (ms), respectively. All perfSONAR tests are initiated from the internal DTN.

5.2 Per-flow Monitoring

This section is dedicated to demonstrating the real-time reporting capabilities of the proposed system, showcasing its ability to provide instantaneous updates on key metrics such as throughput, queue occupancy, RTT, and packet losses for individual flows. To illustrate this functionality, an experiment was conducted, involving the introduction of a third data transfer alongside two pre-existing data transfers. The visual representation of this experiment can be observed in Figure 9.

Within the depicted figure, the upper left graph presents per-flow throughput, while the bottom left graph presents per-flow RTT. Moving to the upper right graph, queue occupancy is presented, while the bottom right graph presents per-flow packet losses. Through these four graphs, TCP behavior can be clearly illustrated when a new flow joins the network. These visualizations provide valuable insights into how the proposed system responds to changing conditions, enabling swift analysis of network dynamics as new flows are incorporated.

Initially, TCP endeavors to distribute the available bandwidth equitably among all active connections. This equitable distribution becomes evident through the observations in the first graph, where

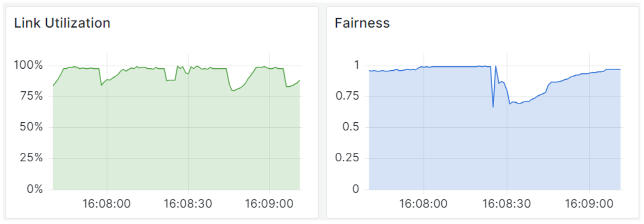


Figure 10: Additional traffic statistics calculated by the control plane.

the throughput of the two flows converges to an approximate parity (around 5 Gbps for each flow).

Additionally, when a new TCP connection is established or is followed by a period of inactivity, the sender adjusts the network's existing bandwidth allocation and congestion status. To achieve this, the sender initiates a burst of packet transmission. This burst aids the sender in promptly identifying the feasible data transmission volume without inducing congestion.

This behavior is illustrated through the analysis of queue occupancy and the count of packet losses. As a new flow becomes part of the network, the queue rapidly fills up, manifesting as a sudden surge in the queue occupancy graph. Notably, during the burst phase, the queue's capacity is often exceeded, leading to significant packet losses, as depicted in the packet loss percentage graph.

5.3 Additional Traffic Statistics

In addition to the measurements collected by the data plane, the control plane can provide invaluable insights into the network's traffic dynamics. This is achieved by leveraging access to the measurements of all active flows. Consequently, the control plane gains the capability to extract numerous metrics such as total link utilization, the count of active flows, aggregate packets sent, aggregate bytes transmitted, and more.

Moreover, the control plane performs computations that surpass the data plane's computational and resource constraints. For instance, the control plane can calculate the fairness among TCP flows. Notably, this involves determining the Jain's fairness index, a quantifiable fairness metric, by employing the following equation [22]:

$$F = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \cdot \sum_{i=1}^{N} x_i^2} \tag{1}$$

The components of the equation are as follows:

- F is the Jain's fairness index.
- N represents the number of flows.
- x_i corresponds to the resource allocation (e.g., bandwidth) for the ith flow.

By applying this index, the control plane effectively assesses the equity among different flows, allowing for a comprehensive evaluation of traffic distribution within the network. This index quantifies the fairness of resource allocation among multiple flows, with a value closer to 1 indicating a more equitable distribution

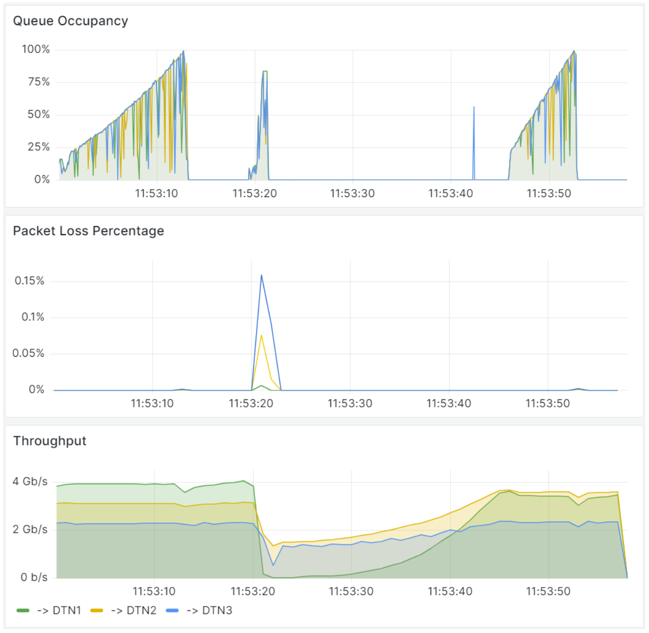
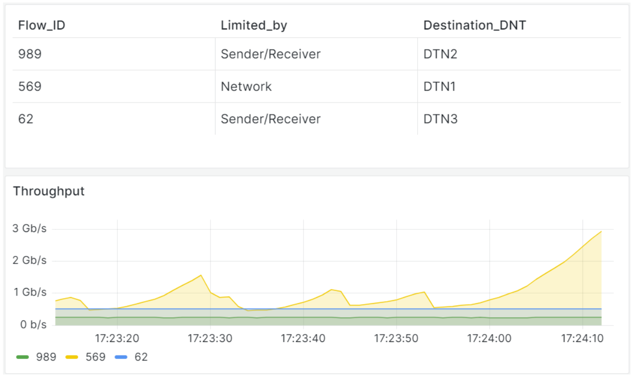


Figure 11: Microburst detection. The microburst bloats the queue, resulting in a significant increase in the packet loss percentage, and severe degradation in the throughput.



Flow_ID	Limited_by	Destination_DNT
989	Sender/Receiver	DTN2
569	Network	DTN1
62	Sender/Receiver	DTN3

Figure 12: Reporting the flows limited by network and those limited by the sender/receiver.

of resources. There are situations when one host is bottlenecked by a bandwidth smaller than the others, which reduces fairness. Monitoring the fairness index enhances the overall monitoring and management of network resources, resulting in more optimized and balanced performance.

Illustrated in Figure 10, the link utilization and fairness of traffic are depicted within the time interval showcased in Figure 9. Despite the link being fully utilized, the fairness index reveals a notable departure from equitable resource distribution for approximately 20 seconds. This departure implies that although the link was saturated, the available bandwidth wasn't uniformly shared among the flows during this duration.

To identify the underlying cause of this observed disparity, the fairness graph in Figure 10 should be cross-referenced with the throughput graph in Figure 9. This analysis shows that the period of unfairness aligns with the time taken by the three TCP flows to converge subsequent to the introduction of the third flow.

5.4 Use Cases

5.4.1 Detecting Small-sized Buffers. In this experiment, the average RTT for the flows is set to 100 ms. As per the established guideline, the buffer size should be one Bandwidth-delay Product (BDP) [8]. The BDP is calculated as the product of the bandwidth and the RTT, resulting in $10Gbps \times 100ms = 1000$ Megabits, which translates to 125 Megabytes. Accordingly, the buffer size was configured to be $BDP/4$ or precisely 31.25 Megabytes, representing a small-sized buffer. The proposed system detects microbursts and shows their impact on queue occupancy, percentage of packet losses, and throughput. If the queue fails to absorb the microburst, leading to significant packet losses and degradation in the throughput, then the buffer size should be reevaluated. The experimental results are illustrated in Figure 11. Across the three flows, the percentage of lost packets notably escalated for two flows, surpassing 0.05% for one of them and exceeding 0.15% for the other. It took approximately 25 seconds for the throughput of these flows to recover. The programmable switch's data demonstrates that the buffer could not efficiently absorb the microburst, signaling a requirement for buffer size adjustment. By tracking the count and magnitude of microbursts, administrators can set an appropriate buffer size that effectively mitigates microburst impact without incurring undue queuing delays.

5.4.2 Determining if a Connection is Limited by Sender/Receiver or by the Network. In this experiment, three tests are performed. In the test involving DTN1, the network is set to be the bottleneck by introducing 0.01% packet losses. In the test involving DTN2, the receiver is set to be the bottleneck by reducing its TCP buffer size. In the test involving DTN3, the sender is set to be the bottleneck by setting its sending rate to 500 Mbps which is less than the bottleneck link (10 Gbps).

The results of the experiment are depicted in Figure 12. The measurements are grouped into flows using the hash of the 5-tuple. The throughput of Flow1, or the flow from the internal DTN to DTN1, is fluctuating because of the induced packet losses. The programmable switch reported that the flow is limited by the network. For Flow2 and Flow3, the throughput is steady at around 250 Mbps and 500 Mbps, respectively. The programmable switch reported that both flows are limited by either the sender or the receiver but not by the network.

5.4.3 Detecting Throughput Degradation Caused by mmWave Blockages in Data Centers. A Conventional Hierarchical Tree-based Data Center Network (DCN) is a type of network architecture used in data centers to manage and distribute data traffic within the facility. Conventional Tree-based DCNs may suffer from limited scalability, high cost (cables might contribute by 8% of the overall infrastructure cost [32]), high energy consumption, and low cross-section bandwidth [20].

To mitigate these problems, researchers have been investigating the capability of mmWave wireless networks as a possible solution

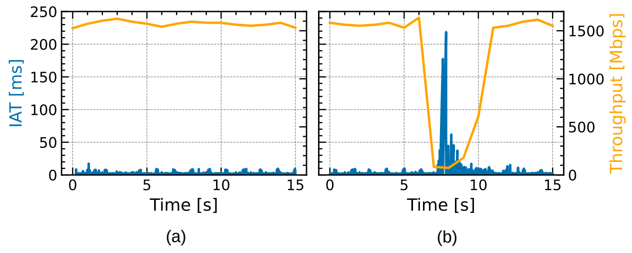


Figure 13: Blockage can be identified by the significant increase in the IAT values during the blockage: (a) no blockage; (b) blockage at t=7s [26].

[19]. This technology is promising because it supports multi-Gbps point-to-point communication besides providing high spatial reuse. However, a common issue in mmWave networks is the line of sight (LOS) blockage. Devices communicating through the mmWave technology steer their beams in a process known as beamforming to maximize the signal strength in the direction of the other communicating end (i.e., LOS). Because the signal is focused in a narrow beam, a gray failure occurs when the LOS is blocked. Blocking the LOS (which can happen if data center employees walk through it) significantly degrades the connection quality.

Previous work utilized P4 programmable data planes to detect degradation caused by the blockage problem in mmWave networks [26]. Consider Figure 13. Upon blockage, the inter-arrival time (IAT) of the packets increases by multiple orders of magnitude. By monitoring the IAT of the packets, programmable data planes can report the degradation caused by a blockage. Figure 14 shows the time required by a P4-based system to detect and react to LOS blockage compared to throughput-based and RSSI-based systems. The P4-based system utilizes a programmable data plane device to measure the IAT. The Throughput-based system utilizes a controller to monitor throughput degradation. RSSI-based system is employed by off-the-shelf devices. It monitors the received signal strength indicator (RSSI) to detect and react to blockages. The gray rectangle indicates the time of blockage. As shown by the figure, P4-based system detects the blockage before the throughput degrades. The system outperforms the throughput-based system and the RSSI-based system.

6 RELATED WORK

NetSage [31] is a data analysis and visualization platform that combines data from various sources into a single unified view. It is open to the public and accessible via https://portal.netsage.global/grafana, and its software is open source. NetSage uses active and passive measurements to provide performance visualizations. It can collect data from routers, switches, active testing sites, and science data archives. NetSage analysis longitudinal data to understand longer-term trends and behaviors.

OSG Network Monitoring Platform [2] is a comprehensive network monitoring platform that supports data collection, processing, storage, and visualization. This platform collects measurements from the tests performed by perfSONAR. This platform stores data in a short-term store location (last six months), in a long-term store location, and in a backup location. Further, the platform has a centralized configuration system that configures the tools to be used in the tests,

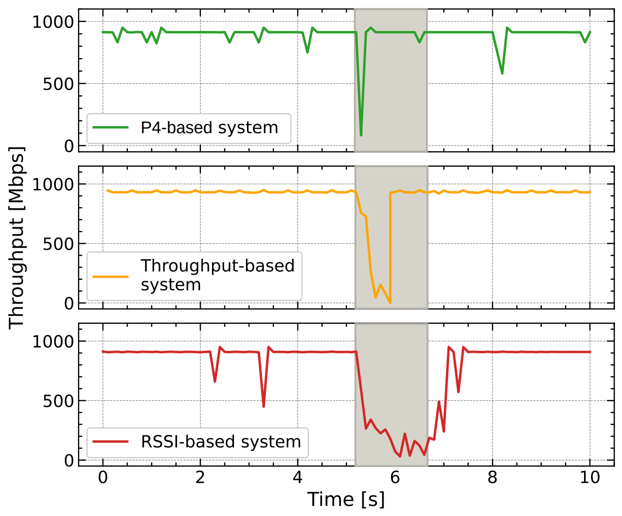


Figure 14: Recovery speed of the P4-based system, throughput-based system, and the RSSI-based system. The gray rectangle represents the 2 seconds blockage [26].

the participating nodes in the tests, and the test schedule for the entire infrastructure.

Bezerra et al. [3] integrates In-band Network Telemetry (INT) [23] with AmLight to provide sub-second network monitoring and performance evaluation metrics. Programmable switches were utilized to collect measurements at the data plane level. The collected measurement is then directed to AmLight Collector, which is responsible for receiving, parsing, processing, and generating operational reports. Through INT, AmLight was capable of providing instantaneous bandwidth utilization, monitoring packet drop probability and jitter, and tracing packets.

Kfoury *et al.* [24] addressed the challenge of identifying congestion control algorithms (CCAs) using P4-programmable data planes. CCAs regulate sending rates to prevent network congestion. When multiple CCAs coexist on the same link, the performance tends to favor CCAs that exhibit more aggressive behavior. Therefore, it becomes crucial to accurately identify and separate them to isolate their dynamics. However, this identification is not straightforward as CCAs are not advertised in packet headers. To address this issue, the authors proposed P4CCI, a system that detects the CCA of a flow at line rate by utilizing P4-programmable data planes (PDPs). The PDP calculates and extracts the flow's bytes-in-flight and forwards this information to a deep learning model for classification. Once classified, flows are allocated into dedicated queues based on their CCA type. The system was implemented and tested on real hardware using Intel's Tofino ASIC. The results show that P4CCI's ability to accurately detect and separate CCAs leads to an enhancement in network performance.

Gomez *et al.* [16] evaluated the performance of the Bottleneck and Round-trip Time version (BBRv2) CCA through Mininet emulation. The study highlighted improved coexistence of BBRv2 and CUBIC compared to BBRv1 and CUBIC. The authors showed the effectiveness of BBRv2 in reducing RTT unfairness and achieving a better bandwidth distribution. Additionally, it was reported that BBRv2 demonstrates improved adaptability to RTT and bandwidth variations compared to BBRv1. Results also show that BBRv2 reduces the average flow completion time (FCT) of concurrent flows.

The proposed system in this paper utilizes programmable switches to enhance the monitoring and troubleshooting functionalities of perfSONAR. The traces produced by perfSONAR are used to understand longitudinal traffic behavior (NetSage and OSG Network Monitoring Platform), detect anomalies in the traffic [4], and train network-wide anomaly detection systems [34]. By providing higher visibility and more detailed reports, the proposed system can enhance the performance of all the platforms that utilize perfSONAR traces. Furthermore, the proposed system allows perfSONAR to detect degradation problems not caused by the network.

7 CONCLUSION

The integration of P4 with perfSONAR yields substantial benefits by passively monitoring real traffic in real-time and on a per-flow basis. The real-time passive analysis executed through a P4 programmable switch significantly augments perfSONAR's troubleshooting capabilities. This integration equips perfSONAR with the ability to discern flows constrained by sender/receiver limitations, facilitating informed decisions on the need for active measurements. This synergy not only elevates monitoring precision but also streamlines the diagnostic process for optimized network performance.

REFERENCES

[1] Ali AlSabeh, Joseph Khoury, Elie Kfoury, Jorge Crichigno, and Elias Bou-Harb. 2022. A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment. *Computer Networks* (2022).

[2] Marian Babik, Shawn McKee, Brian Bockelman, Edgar Hernandez, Edoardo Martelli, Ilija Vukotic, Derek Weitzel, and Marian Zvada. 2019. Improving WLCG networks through monitoring and analytics. In *EPJ Web of Conferences*, Vol. 214. EDP Sciences, 08006.

[3] Jeronimo Bezerra, Italo Brito, Arturo Quintana, Julio Ibarra, Vasilka Chergarova, Renata Frez, Heidi Morgan, Marc LeClerc, and Arun Paneri. 2021. Deploying per-packet telemetry in a long-haul network: the AmLight use case. In *2021 IEEE Workshop on Innovating the Network for Data-Intensive Science (INDIS)*. IEEE, 44–49.

[4] Prasad Calyam, Jialu Pu, Weiping Mandrawa, and Ashok Krishnamurthy. 2010. Ontimedetect: Dynamic network anomaly notification in perfSONAR deployments. In *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 328–337.

[5] Xiaoqi Chen, Shir Landau Feibish, Yaron Koral, Jennifer Rexford, Ori Rottenstreich, Steven Monetti, and Tzuu Wang. 2019. Fine-grained queue measurement in the data plane. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*. 15–29.

[6] Xiaoqi Chen, Hyojoon Kim, Javed M Aman, Willie Chang, Mack Lee, and Jennifer Rexford. 2020. Measuring TCP round-trip time in the data plane. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*. 35–41.

[7] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[8] Jorge Crichigno, Elias Bou-Harb, and Nasir Ghani. 2018. A comprehensive tutorial on science DMZ. *IEEE Communications Surveys & Tutorials* (2018).

[9] Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. 2013. The science DMZ: A network design pattern for data-intensive science. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.*

[10] Jon Dugan, Seth Elliott, Bruce Mah, Jeff Poskanzer, and Kaustubh Prabhu. 2014. iPerf3, tool for active measurements of the maximum achievable bandwidth on IP networks. *URL: https://github. com/esnet/iperf* (2014).

[11] E. Waltz. 2016. Portable DNA Sequencer MinION Helps Build the Internet of Living Things . https://tinyurl.com/fysezkef

[12] Elastic. 2023. TCP input plugin. https://www.elastic.co/guide/en/logstash/current/plugins-inputs-tcp.html

[13] Brian Eriksson, Paul Barford, and Robert Nowak. 2008. Network discovery from passive measurements. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. 291–302.

[14] European Organization for Nuclear Research. 2023. Towards the future: tackling upcoming data challenges. https://home.cern/about/computing

[15] Mojgan Ghasemi, Theophilus Benson, and Jennifer Rexford. 2017. Dapper: Data plane performance diagnosis of TCP. In *Proceedings of the Symposium on SDN Research*. 61–74.

[16] Jose Gomez, Elie Kfoury, Jorge Crichigno, Elias Bou-Harb, and Gautam Srivastava. 2020. A performance evaluation of TCP BBRv2 alpha. In *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*.

[17] Jose Gomez, Elie F Kfoury, Jorge Crichigno, and Gautam Srivastava. 2022. A survey on TCP enhancements using P4-programmable devices. *Computer Networks* (2022).

[18] Grafana Labs. 2023. The Open Observability Platform. https://grafana.com/

[19] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. 2011. Augmenting data center networks with multi-gigabit wireless links. In *Proceedings of the ACM SIGCOMM 2011 conference*. 38–49.

[20] Abdelbaset Hamza, Jitender Deogun, and Dennis Alexander. 2016. Wireless communication in data centers: A survey. *IEEE communications surveys & tutorials* 18, 3 (2016), 1572–1595.

[21] Intel. 2023. Wedge 100BF-32X, 100GbE data center switch, Barefoot Networks, an Intel® company. https://www.edge-core.com/productsInfo.php?cls=1&cls2=180&cls3=181&id=335

[22] Raj Jain, Arjan Durresi, and Gojko Babic. 1999. Throughput fairness index: An explanation. In *ATM Forum contribution*, Vol. 99.

[23] Elie Kfoury, Jorge Crichigno, and Elias Bou-Harb. 2021. An exhaustive survey on P4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. *IEEE access* (2021).

[24] Elie Kfoury, Jorge Crichigno, and Elias Bou-Harb. 2023. P4CCI: P4-based online TCP congestion control algorithm identification for traffic separation. In *IEEE International Conference on Communications (ICC), Rome, Italy.*

[25] Ali Mazloum, Elie Kfoury, Jose Gomez, and Jorge Crichigno. 2023. A Survey on Rerouting Techniques with P4 Programmable Data Plane Switches. *Computer Networks* (2023).

[26] Ali Mazloum, Elie Kfoury, Sanjib Sur, Jorge Crichigno, and Nasir Ghani. 2023. Enhancing Blockage Detection and Handover on 60 GHz Networks with P4 Programmable Data Planes. (2023).

[27] perfSONAR Project. 2023. perfSONAR 5.0.3 Release Notes. https://www.perfsonar.net/releasenotes-2023-06-16-5-0-3.html

[28] perfSONAR Project. 2023. What is perfSONAR? https://tinyurl.com/4xef95h9

[29] Fiber Instrument Sale. 2022. Optical TAP. https://www.fiberinstrumentsales.com

[30] Brian Tierney, Joe Metzger, Jeff Boote, Eric Boyd, Aaron Brown, Rich Carlson, Matt Zekauskas, Jason Zurawski, Martin Swany, and Maxim Grigoriev. 2009. perfSONAR: Instantiating a global network measurement framework. *SOSP Wksp. Real Overlays and Distrib. Sys* 28 (2009).

[31] Katrina Turner, Mahesh Khanal, Tyson Seto-Mook, Alberto Gonzalez, Jason Leigh, Andrew Lake, Sartaj Baveha, Samir Faci, Brian Tierney, and Daniel Doyle. 2020. The NetSage Measurement Framework: Design, Development, and Discoveries. In *2020 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*. IEEE, 45–56.

[32] Hars Vardhan, Navine Thomas, Seong Ryu, Bhaskar Banerjee, and Ravi Prakash. 2010. Wireless data center with millimeter wave network. In *2010 IEEE global telecommunications conference GLOBECOM 2010*. IEEE, 1–6.

[33] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. 2017. High-resolution measurement of data center microbursts. In *Proceedings of the 2017 Internet Measurement Conference*. 78–85.

[34] Yuanxun Zhang, Saptarshi Debroy, and Prasad Calyam. 2016. Network-wide anomaly event detection and diagnosis with perfSONAR. *IEEE Transactions on Network and Service Management* 13, 3 (2016), 666–680.