DOVE: Diagnosis-driven SLO Violation Detection

Yiran Lei*‡, Yu Zhou§, Yunsenxiao Lin*‡, Mingwei Xu*†‡, Yangyang Wang†‡
*Department of Computer Science and Technology, Tsinghua University
†Institute for Network Sciences and Cyberspace, Tsinghua University
‡Beijing National Research Center for Information Science and Technology (BNRist)
§Alibaba Inc.

***Abstract*—Service-level objectives (SLOs), as network performance requirements for delay and packet loss typically, should be guaranteed for increasing high-performance applications, *e.g.,* telesurgery and cloud gaming. However, SLO violations are common and destructive in today's network operation. Detection and diagnosis, meaning monitoring performance to discover anomalies and analyzing causality of SLO violations respectively, are crucial for fast recovery. Unfortunately, existing diagnosis approaches require exhaustive causal information to function. Meanwhile, existing detection tools incur large overhead or are only able to provide limited information for diagnosis. This paper presents DOVE, a diagnosis-driven SLO detection system with high accuracy and low overhead. The key idea is to identify and report the information needed by diagnosis along with SLO violation alerts from the data plane selectively and efficiently. Network segmentation is introduced to balance scalability and accuracy. Novel algorithms to measure packet loss and percentile delay are implemented completely on the data plane without the involvement of the control plane for fine-grained SLO detection. We implement and deploy DOVE on Tofino and P4 software switch (BMv2) and show the effectiveness of DOVE with a use case. The reported SLO violation alerts and diagnosis-needing information are compared with ground truth and show high accuracy (>97%). Our evaluation also shows that DOVE introduces up to two orders of magnitude less traffic overhead than NetSight. In addition, memory utilization and required processing ability are low to be deployable in real network topologies.**

I. Introduction

Internet applications, *e.g.,* telesurgery, cloud gaming, virtual reality streaming, and algorithmic trading, are having more and more strict service level objectives (SLOs) over reachability, delay, packet loss, and bandwidth. SLOs describe the performance expectations, *e.g.,* the end-to-end packet loss rate of cloud gaming traffic should not exceed 5%.

Various reasons lead to SLO violations, *e.g.,* link failure causes packet loss and microburst introduces extra queuing delay. Meanwhile, SLO violations are common in production networks. Jeff Dean's keynote [1] reveals that around 40 to 80 machines suffer from severe packet loss in a DCN per year. Katz-Bassett *et al.* [2] discover reachability problems involving about 10,000 distinct prefixes during 3 weeks. Even in the past 24 hours, tens of Internet outages are reported by ThousandEye [3]. Without timely disposal, SLO violations may have severe consequences, which include degrading user experiences, posing security and function problems, and causing monetary damages to service providers [4]. Google and Bing's report [5] indicates that a 500 ms delay leads to 1.2% revenue reduction. Clay [6] reports that Amazon lost 66,240$ per minute in a sudden 40% drop in web traffic. Therefore, fast recovery from SLO violations is of great significance.

978-1-6654-4131-5/21/$31.00 ©2021 IEEE

To mitigate and resolve SLO violations as soon as possible, it is essential to 1) detect and 2) diagnose SLO violations quickly. Detection denotes discovering SLO violations and reporting alerts. Upon receiving alerts, diagnosis analyzes the causality of the SLO violations. Diagnosis tools Dapper [7], DTaP [8], Provenance [9] and Zeno [10] find the responsible flows and the locations of performance bottlenecks. Detection and diagnosis should work jointly for fast recovery. However, existing detection tools incur large overheads or fail to provide enough information needed by diagnosis.

The detection tools can be classified into two types. The first type is to capture information of all packets. These tools, *e.g.,* NetSight [11] and Planck [12], are able to detect various types of SLO violations (*e.g.,* packet loss, delay and bandwidth) and provide enough information for diagnosis. However, these tools introduce large detection overheads and face scalability issues. The second type is to capture information of a partial set of packets via aggregating and filtering. They can significantly reduce detection overheads. But they fail to provide enough information needed by diagnosis. The types of SLOs they monitored are also limited. For example, LossRadar [13] provides no extra information for diagnosis. INTSight [14] cannot detect SLO violations related to packet loss and percentile delay.

Recognizing the importance and problems of detection and diagnosis, we present DOVE, a diagnosis-driven SLO violation detection system. The key idea is to identify and report the information needed by diagnosis along with SLO violation alerts from the data plane selectively and efficiently. DOVE is able to monitor SLOs of interested flows at the microsecond level. The SLOs include packet loss, max delay, and percentile delay. They are consecutively measured over operator-defined intervals. For example, the SLO of packet loss is that the number of lost packets from switch A to switch B should not exceed 10 every 100ms. Besides, DOVE defines causally related information of SLO violations, collects the information for analysis, and analyzes causality with efficient and automatic techniques. DOVE is distinguished from the related works in two perspectives. First, compared to the state-of-the-art network detection techniques, DOVE yields better scalability and provides good network visibility for SLO troubleshooting. Second, compared to the state-of-the-art network diagnosis techniques, DOVE does not assume exhaustive casual information, which is impractical to acquire in production networks, because the overhead is too large.

In a nutshell, DOVE splits time into epochs and flow paths into segments respectively. For interested flows, DOVE measures packet loss and delay performance per epoch and per segment completely on the data plane by storing and calculating metadata in switches and telemetry headers. If SLO violations are found, alerts are generated and reported to the control plane. Another set of flows are monitored at the same time for any suspicious flow behaviors which might be the causes of SLO violations. Data plane uploads events containing suspicious flow behaviors. Control plane servers passively collect alerts and events and use Provenance-based [8–10] techniques to analyze the causality of alerts. DOVE is designed for data center networks, enterprise networks, and Internet service provider (ISP) networks with the support of partial and incremental deployments.

The contributions of DOVE are the followings:

- We propose a diagnosis-driven SLO violation detection system, which detects SLO violations at fine-grained timescales with low overhead and diagnoses causality of SLO violations automatically and efficiently.
- We propose the Coloring Algorithm for packet loss measurement and an approximate algorithm for percentile delay verification without the involvement of the control plane. DOVE supports partial and incremental deployment with the mechanism of network segmentation.
- We validate the design by implementing DOVE over the commodity programmable switch and P4 software switch. Our evaluation verifies the effectiveness of causality analysis, showing that the accuracy of performance measurement is >97% and the traffic overhead is 70% lower than INTSight, NetSight, and LossRadar.

II. Design Overview

In this section, we present an overview of DOVE's design.

A. Architecture of DOVE

As shown in Figure 1, DOVE has three architectural components. On the data plane, the SLO violation detector first measures flow performance (in terms of percentile delay, max delay, and packet loss) in real-time and then reports alerts to the control plane if the measured performance fails to meet the SLO requirements (*i.e.,* expected flow performance). Alerts are signals showing the existence of SLO incompliance. However, finding the causality of a violation requires more information than the alert itself. Hence we specify suspicious flow behaviors (*e.g.,* heavy hitters), which might be the causes of violations. Suspicious Flow Behavior Monitor monitors flows' behaviors and generates corresponding events if any suspicious behaviors are found. On the control plane, an analyzer collects alerts and events from all DOVE-enabled switches and seeks the cause of a certain alert by retrieving and correlating events.

B. Processing Procedure of DOVE

DOVE's processing procedure is composed of two stages. The first stage is configuring network switches. First, we introduce the concept of the epoch, which is the time unit

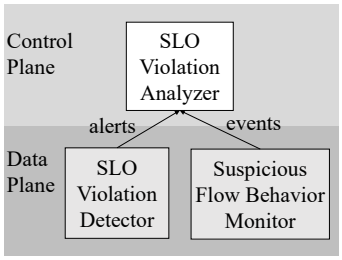


Figure 1. **DOVE architecture.**

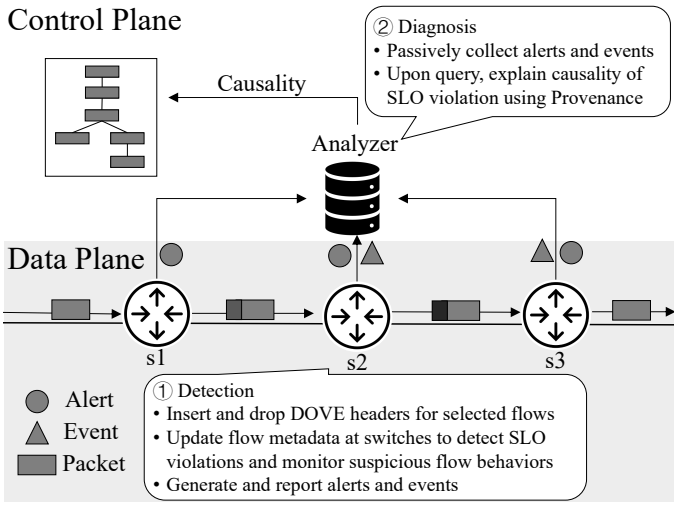


Figure 2. **DOVE processing procedure.**

for measuring and reporting. DOVE splits continuous time into a series of adjacent epochs and sets the epoch length on every switch to be the same. Clocks are synchronized across different switches with IEEE 1588 [15]. Every epoch can be associated with a unique epoch id which represents the same period across switches. Second, DOVE translates SLOs and suspicious flow behaviors into epoch-wise rules. These rules have values functioning as thresholds. (*e.g.,* the threshold for max delay is 20ms = when the measured max delay exceeds the threshold 20ms, an alert should be generated and reported.) Third, DOVE allocates unique IDs for selected and watched flows and populates the specified flows (expressed by the 5-Tuple, source and destination IP addresses, *etc.*), their IDs, and rules (thresholds) to switches (discussed in §II-C).

The second stage is operation. As shown in Figure 2, suppose all switches are DOVE-enabled. Packets of the monitored flow sequentially traverse via switch s1, s2, s3. DOVE uses packet headers to deliver control signals and values for later calculations. On ingress switch s1, DOVE headers are inserted into packets whose SLOs need to be monitored. Packets go through the SLO violation detector and suspicious flow monitor inside each switch along the flow path. On s1, s2, and s3, after the detector and monitor, flows' metadata, which records current performance and behaviors, is updated. On s1 and s2, DOVE headers are updated as well while on egress switch s3, DOVE headers are removed and the packets are restored. Flows' metadata is checked and refreshed per epoch. At the end of each epoch, the measured flow performance and monitored behaviors are compared with installed thresholds.

Once performance fails to meet requirements or suspicious behaviors are found, alerts or events will be generated and reported to the analyzer respectively. Otherwise, no alerts or events will be generated. Then flows' metadata is reset. The analyzer keeps collecting alerts and events passively and analyzes the causality of alerts.

C. Key Ideas of DOVE

1) Specific flows

SLOs of flows with higher priorities need to be monitored and verified. Performance measurements and comparisons are resource-expensive on the data plane. It is impractical and unnecessary to monitor SLOs of the complete set of flows. Therefore, we select a partial set of flows (*i.e.,* V.I.P. flows), called the selected flows, for the SLO violation detector. Three types of SLO, *i.e.,* packet loss, percentile delay, and max delay, are monitored.

It is the network operators' responsibilities to set the selected flows, as the performance of important flows should be guaranteed. Flows are specified by flow IDs, which can be composed of any fields of packets, *e.g.,* source and destination IP addresses, and 5-Tuple. Flow IDs of P4 [16] support exact matching, longest prefix matching, and range matching. All the matched packets of a single flow ID are treated as a single flow for SLO monitoring. In some cases, network operators can precisely assign flows of certain applications as the selected flows. In other cases, with the longest prefix matching and range matching, operators can aggregate distinct flows like 166.111.x.x/16, and set SLOs for the whole. The flexibility in flow IDs enables network operators to monitor SLOs with multiple granularity according to their needs.

In order to diagnose SLO violations with higher accuracy, suspicious flow behaviors of more flows should be monitored. We should cover potential bad flows as much as possible (even the complete set of flows). The set of flows, whose behaviors are monitored by DOVE, is called the watched flows. For watched flows, two types of suspicious flow behaviors, *i.e.,* heavy hitters and heavy changers, are checked by the suspicious flow behavior monitor. Selected flows and watched flows are not necessarily the same. The watched flows cover a wider range of flows in most cases. Note that only SLOs of the selected flows will be verified and only behaviors of the watched flows will be monitored.

A bigger set of watched flows tends to collect more causally related information, leading to higher diagnosis accuracy. There are two ways to assign the set of watched flows. First, network operators know in advance that certain flows are key flows, *e.g.,* applications on certain servers generate large traffic. Such flows can be easily expressed by flow IDs. Second, network operators have no ideas what to monitor. DOVE has the ability to monitor the complete set of flows (only to detect heavy hitters and changers) by using iteratively updating techniques like Sonata [17].

In the DOVE configuration stage, IDs starting from 0 are allocated to all the selected flows as well as the watched flows sequentially. Then the flow IDs of selected and watched flows along with the IDs and flow-specific rules (thresholds) are populated to the data plane by lookup tables (implemented on P4 as match-action tables on each device). On the data plane, resources are allocated and reserved in advance. Any intermediate values used by the detector and monitor are stored in register tables. Each entry is assigned to the flow whose ID is the same as the entry index.

2) Network segmentation

It is expensive and impractical to replace all the forwarding devices in a network all at once. We propose a measuring abstraction to 1) balance scalability and accuracy and 2) support partial and incremental deployment. Most of the previous approaches measure flow performance on an end-to-end or per-hop basis. The former lacks the information of internal nodes while the latter may have severe scalability issues.

Network segmentation splits the flow path into several segments. For each segment, the switch which the flow firstly traverses is called the upstream switch while the other switch is called the downstream switch. Upstream and downstream switches are not necessarily neighbor switches. Except for the ingress and egress switch, the downstream switch of the current segment is also the upstream switch of the following segment. Specifically, the segment starts from the ingress pipeline of the upstream switch and ends at the ingress pipeline of the downstream switch. In DOVE, flow performance (*i.e.,* percentile delay, max delay, packet loss) and corresponding SLOs are measured and set for each segment respectively.

3) Novel algorithms for packet loss and delay measuring on the data plane

The programmable data plane is limited in computation capability. Existing approaches for packet loss measuring have to rely on the computation capability of control plane to synchronize counters (*e.g.,* NetFlow [18], AM-PM [19]) or decode digests (sketch-based techniques, *e.g.,* LossRadar [13]). The latest method QPipe [20] for percentile delay measurement is fully implemented on the data plane, but still unable to provide flow-level information. However, the involvement of the control plane inevitably introduces additional delay and SLOs should be verified for key flows. We overcome the computation limitations of the data plane and design a novel algorithm, called Coloring Algorithm, for packet loss measuring and an approximate algorithm for percentile delay verification without any involvement of the control plane. Hence, packet loss and delay of selected flows can be measured for every segment on a per-epoch basis timely and efficiently.

III. Design

In this section, we describe the design of the detector, monitor, and analyzer. Table I displays the fields in the DOVE telemetry header. Note that DOVE header is only inserted into packets of the selected flows. The detailed usage of the fields will be discussed in the following subsections.

A. SLO Violation Detector

The detector works at the ingress pipeline. For each selected flow, during each epoch, at each downstream switch, SLO

Table I
Fields in DOVE telemetry header.

Field	Description	Bit
up_switch_id	upstream switch id	8
egress_port	port to which the packet is forwarded at upstream switch	9
packet_num	number of packets sent from upstream switch	32
control_bit	indicating the operation of downstream switch	1
color_bit	indicating color of the packet	1
pad	keep DOVE header byte-aligned	5
timestamp	timestamp at ingress pipeline	48

Violation Detector keeps answering the question: for the segment, does the real performance satisfy the expectation? If the answer is no, an alert is generated and reported. This procedure hides the measuring details inside the detector itself and only exposes the result to the control plane, which greatly reduces traffic overhead. The detector measures and compares two types of SLOs: 1) packet loss, 2) delay.

1) Packet Loss

SLO for packet loss follows the pattern that the number of lost packets or packet loss rate in the current epoch should not exceed n. We first calculate packet loss using the Coloring Algorithm and then compare it with the threshold. The key idea of the Coloring Algorithm is that: 1) set counters at the upstream and downstream switch and 2) synchronize counter values via telemetry headers and calculate packet loss at the downstream switch periodically.

Specifically, upstream switch dyes all packets red or green by setting the color_bit field to 0 or 1 respectively in the DOVE header. We say red and green are opposite of each other for clear description in later parts. Packets' color shifts when a new epoch starts and keeps unchanged for the rest of the epoch. Red and green counters are set to count the number of red and green packets sent from the upstream switch. In the first half of the current epoch, the upstream switch sets control_bit field to 0 and packet_num field to the value of the opposite color counter (*e.g.,* if the upstream switch is sending red packets in the current epoch, copy the green counter value to the packet_num field). When the second half of the current epoch starts, the opposite color counter is reset to 0 to be prepared for the next epoch (as packets' color in the next epoch will be the opposite color). During the second half of the current epoch, set control_bit field to 1 and packet_num field to 0. Algorithm 1 is the pseudo-code at the upstream switch.

At the downstream switch, red and green counters are set to count the number of received red and green packets. When the control_bit field in the header is 0, copy and store packet_num field in the register, which is the number of sent packets of the previous epoch. The number of received packets of the previous epoch is stored in the opposite color counter at the downstream switch. When seeing the control_bit is 1 for the first time, calculate packet loss with opposite color counter and packet_num and store the result in the register. Due to the inability to operate division on the data plane, we calculate the number of lost packets instead of packet loss rate (see Algorithm 2).

Algorithm 1: Coloring Algorithm - upstream switch

```
if new_epoch is True then
    shift_packet_color()
if first_half_of_epoch is True then
    control_bit = 0
    if packet_color is red then
        packet_num = green_counter
    else
        packet_num = red_counter
else
    control_bit = 1
    packet_num = 0
    if packet_color is red then
        green_counter = 0
    else
        red_counter = 0
if packet_color is red then
    red_counter ++
    color_bit = 0
else
    green_counter ++
    color_bit = 1
```

Algorithm 2: Coloring Algorithm - downstream switch

```
if color_bit is 0 then
    red_counter ++
else
    green_counter ++
if control_bit is 0 then
    register_pn = packet_num
    trigger = 1
else
    if trigger is 1 then
        if color_bit is 0 then
            packet_loss = register_pn - green_counter
            green_counter = 0
        else
            packet_loss = register_pn - red_counter
            red_counter = 0
        trigger = 0
```

The reason to calculate packet loss on the second half of the epoch is that we need to wait for a certain time until the number of the previous epoch's packets is stabilized, as packets near epoch borders may arrive out of order. Control bits of all packets are set so that downstream switches can still receive control signals even when some packets are lost. Downstream switches only calculate packet loss once per epoch to save computation resources and increase processing speed. Note that for the Coloring Algorithm, the clocks of the upstream and downstream switches are not necessarily synchronized. Storing packet numbers sent from senders and calculating packet loss are both controlled by clocks at upstream switches. Clocks at downstream switches are only used to retrieve packet loss results. In addition, a network device may function as the upstream and downstream switch for adjacent segments at the same time, meaning both Algorithm 1 and 2 should be implemented at the ingress pipeline and there should be 4 counters in total.

2) Delay

Percentile delay and max delay are measured. SLO for percentile delay asks the question of whether ηth-percentile delay in the current epoch exceeds d. It is difficult to accurately measure ηth-percentile delay directly on the data plane due to limited computation resources and capability. We use an approximate algorithm to answer this question.

Suppose the bandwidth of the selected flow is B bps, epoch length is E seconds and packet size is S bits. Then the estimated number of packets per epoch is $E \cdot B/S$. We use N to denote $E \cdot B/S$. Next, we discuss two situations. Firstly, if $(N-1) \cdot \eta\%$ is an integer, the η-th percentile value of N values, which are sorted in ascending order, is the $(1+(N-1) \cdot \eta\%)$-th one. In this case, whether η-th percentile delay exceeds d can be converted to whether the number of delay exceeding d is larger than $N-(N-1) \cdot \eta\%-1$. Secondly, if $(N-1) \cdot \eta\%$ is not an integer, the η-th percentile value of N sorted values is some number between the $(1+\lfloor (N-1) \cdot \eta\% \rfloor)$-th value and the $(1+\lceil (N-1) \cdot \eta\% \rceil)$-th value. In this case, if the number of delay exceeding d is larger than $N-\lfloor N \cdot (1-\eta\%) \rfloor - 1$, the SLO requirement is violated. If the number is smaller than $N-\lfloor N \cdot (1-\eta\%) \rfloor - 1$, the SLO requirement is satisfied. However, it is unclear if the number equals to $N-\lfloor N \cdot (1-\eta\%) \rfloor - 1$. With inaccuracy (*i.e.,* ignoring possible violations when two numbers are equal in the second situation), we can combine two situations and simply judging whether the number of the delay exceeding d is larger than $N-\lfloor N \cdot (1-\eta\%) \rfloor - 1$. If the answer is yes, an alert is generated and reported. When processing packets, subtract the timestamp carried in the header from the timestamp at the ingress pipeline to get the packet segment delay. If the segment delay is larger than delay threshold d, increase the counter, which records exceeding times, by 1. Before forwarding the packet, copy the timestamp at the ingress pipeline to the header's timestamp field. Note that this algorithm is only an approximate way to verify SLO. It is not suitable for flows whose bandwidth is changing dramatically.

The SLO for max delay is that the max delay of the current epoch should not exceed m. Similar to percentile delay, packet segment delay is calculated. The register, which holds the max delay so far, is compared with the segment delay and properly updated. At the end of the epoch, the register value is compared with threshold m and reset to 0.

Table II shows the fields of the alert header. An alert is for a segment. The upstream switch and egress port is expressed in up_switch_id and egress_port fields, which are copied from DOVE header (Table I). The downstream switch is the current switch and ingress_port field is filled by querying the standard metadata. When an alert should be generated, the original packet is cloned. Then the alert header is inserted into the cloned packet. The clone packet is truncated to remove the original payload and finally uploaded to the analyzer. If the current switch is the upstream switch of the adjacent segment, update the up_switch_id and egress_port fields in the DOVE header.

Table II
Fields in alert header.

Field	Description	Bit
index	ID of the selected flow	32
up_switch_id	upstream switch id	8
egress_port	port to which the packet is forwarded at upstream switch	9
switch_id	downstream switch id	8
ingress_port	port from which the packet is received at downstream switch	9
max_delay	indicating SLO violation of max delay	1
percentile_delay	indicating SLO violation of percentile delay	1
packet_loss	indicating SLO violation of packet loss	1
pad	keep alert header byte-aligned	3
epoch	epoch id	32

B. Suspicious Flow Behavior Monitor

The monitor works at the egress pipeline. For each watched flow, during each epoch, the suspicious flow behavior monitor keeps looking for any flow behaviors which may be the causes of SLO violations and reports events like the detector. The causes of delay violations include high queue occupancy, inter- and intra-switch loop, *etc.* The causes of packet loss violations include queue overflow, link corruption and failure, software bugs, *etc.* Among these causes, the queue-related cause is the most common one, because the hardware, software, and configurations are relatively fixed compared with dynamic network conditions. Therefore, the monitor focuses on two types of suspicious flow behaviors: 1) heavy hitter, 2) heavy changer. Note that other causes, *e.g.,* hardware and software failure, and misconfigurations, are not directly detected by DOVE. DOVE is unable to diagnose upon these factors. However, in cases where SLO violations are detected without any monitored suspicious flow behaviors, network operators can infer the existence of errors in hardware, software, or configurations.

1) Heavy Hitter

An event is uploaded if the traffic of the watched flows on the current epoch exceeds the threshold l. Heavy hitters contribute much to the queuing time in the switch. Since packet queuing is processed after ingress pipeline, long queues significantly increase segment delay and queue overflow causes packet loss, which makes heavy hitters responsible for SLO violations. Threshold l can be adjusted according to network traffic and queue capacity. When processing packets, the length of the current packet is added to a register. At the end of the epoch, the register value is read and compared with l.

2) Heavy Changer

If the traffic increase from the previous epoch to the current epoch exceeds threshold i, an event is uploaded. Besides, treat newly established flows as heavy changers. We do not consider a fast decrease in traffic here since it does not deteriorate queue occupancy. Although the absolute flow rate may not be large, several heavy changers may break the balance and cause congestion at switches. At the egress pipeline, two registers take turns recording the traffic size of the current epoch. At the end of the epoch, the traffic size of the current one is compared with the traffic size of the previous one. The register

Table III
Fields in event header.

Field	Description	Bit
index	ID of the watched flow	32
switch_id	switch id	8
ingress_port	port from which the packet is received	9
egress_port	port to which the packet is forwarded	9
heavy_hitter	indicating the flow is a heavy hitter	1
heavy_changer	indicating the flow is a heavy changer	1
pad	keep event header byte-aligned	4
epoch	epoch id	32

of the previous epoch is then reset to record at the next epoch. Table III shows the fields of the event header. Similar to alerts, event packets are cloned and truncated. However, events are observed behaviors at each DOVE-enabled switch. The concept of segments does not apply to events.

Both the detector and monitor need to set up thresholds. The thresholds are empirical and should be set by network operators. They are the performance objectives of various flows. Under different network conditions, for multi-prioritized flows, with different network segments, the objectives are different. Still, some methods can be applied when operators initialize thresholds. With the objectives of end-to-end delay or packet loss, the segment SLOs should be a fraction of the end-to-end SLOs, which can be estimated by the hop numbers in the segment.

C. SLO Violation Analyzer

We first introduce a diagnostic tool, provenance [9], which can get causal explanations of an event (the event discussed here is different from §III-B) in a distributed system. Provenance maintains each event in the system and records the direct causes of events. When network operators query the explanation of a certain event, provenance can recursively trace back the causal events, which removes unrelated ones and saves time. Provenance can be represented as a DAG. The vertices of the DAG are events and the edges show the causal relations (*e.g.,* the edge $e1 \rightarrow e2$ in the DAG means that event $e1$ is the cause of event $e2$). Like Network datalog (NDlog) [21], we use $A(@X, p_a)$:- $B(@Y, p_b), condition(p_a, p_b)$ as rules to represent the derivation from one event (*i.e.,* tuple in the NDlog context) to another. In the above rule, p_a and p_b are parameters of event A, B. Event A at node X is derived from event B at node Y if $condition(p_a, p_b)$ is satisfied. In turn, if event $A(@X, p_a)$ is observed, event $B(@Y, p_b)$, whose p_b satisfies $condition(p_a, p_b)$, should be the cause.

In DOVE, the analyzer collects alerts whose causality should be explained. However, finding the direct causes of alerts is difficult and impractical. The accurate and direct causes of an alert (if we only consider switch queue as the major reason) is a series of events (not the concept in §III-B) which are certain flows occupy switching queues to certain extents at certain times. With these series of events, we can explain: 1) which flows contribute to SLO violations, 2) when these flows contribute to SLO violations and 3) to what extent these flows contribute to SLO violations. How to acquire the series of events? A brute-force approach is to directly snapshot switch queues, which includes flows' occupancy of the queues, whenever a packet of the selected flow is received at the switch. However, even with compression techniques for the snapshots, considering the huge number of flow packets, there are not enough resources to implement the brute-force snapshot technique on the data plane. Other queue snapshot techniques, *e.g.,* ConQuest [22], and BurstRadar [23], can only snapshot partial queues at the partial time, which fail to capture the exhaustive queue information as well.

Table IV
Provenance rules.

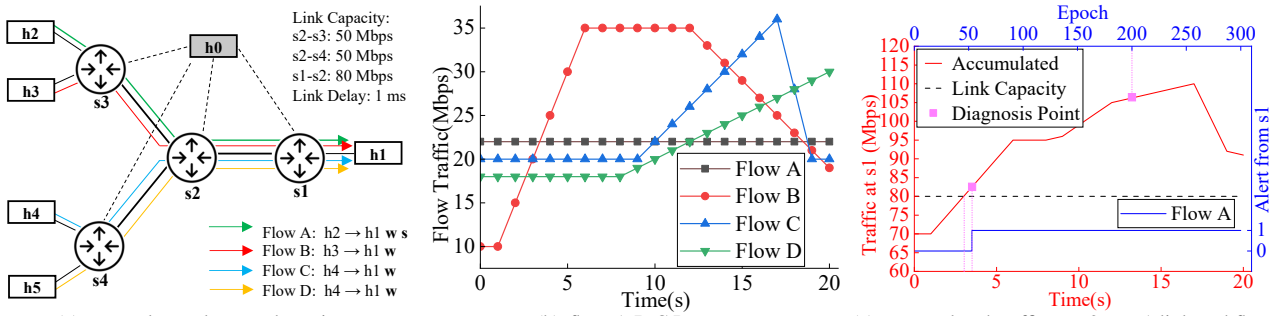
ID	Rules
1	$A(@Y, p_i, Z, p_e, f, t_1)$:- $L(@Z, *, p_e, *, t_2)$, $t_1 - t_2 \leq \epsilon_1$
2	$A(@Y, p_i, Z, p_e, f, t_1)$:- $LI(@Z, *, p_e, *, t_2)$, $t_1 - t_2 \leq \epsilon_2$
3	$A(@Y, p_i, Z, p_e, f, t_1)$:- $L(@Y, p_i, *, *, t_2)$, $t_1 - t_2 \leq \epsilon_3$
4	$A(@Y, p_i, Z, p_e, f, t_1)$:- $LI(@Y, p_i, *, *, t_2)$, $t_1 - t_2 \leq \epsilon_4$

Therefore, instead of collecting and linking direct and accurate causes, we treat the events discussed in §III-B as the causes of alerts. This is a trade-off between the resources and the accuracy of diagnosis. We believe this is reasonable because as discussed in §III-B, heavy hitters and heavy changers can be the causes of SLO violations. Next, we introduce our approach to construct the provenance for causality analysis.

Events (in §III-B) are denoted as $L(@X, p_i, p_e, f, t)$ and $LI(@X, p_i, p_e, f, t)$, meaning the watched flow f is observed as a heavy hitter or heavy changer, whose ingress port is p_i and egress port is p_e, in switch X at epoch t. Alerts are denoted as $A(@Y, p_i, Z, p_e, f, t)$, meaning 1) SLO violations of the selected flow f are observed at epoch t, 2) the SLO violations happen on the segment $Z \rightarrow Y$, and 3) flow f leaves upstream switch Z from port p_e while entering downstream switch Y from port p_i. Table IV shows the correlation rules for alerts and events. Rule 1 and 2 correlate SLO violations with heavy hitters and heavy changers which share the same egress ports with the selected flow f in upstream switch Z. Rule 3 and 4 link alerts to heavy hitters and heavy changers whose ingress ports are the same with f in downstream switch Y. The rules pick out the flows which share the same queues with the selected flows and hence the picked flows should be the cause of SLO violations. $*$ in the rules means wildcard while ϵ is an adjustable parameter controlling the time adjacency of linked alerts and events. With the above rules and collected alerts and events, we can construct a provenance graph (a DAG) in the central analyzer and find the causes of SLO violations.

IV. Evaluation

We implement DOVE on P4 software switch (BMv2) [24] (822 lines of code) and Barefoot Tofino [25] (1940 lines of code) (The analyzer includes 438 lines of code). We run Mininet-emulated networks on a Linux server with 2× 2.40 GHz Xeon E5-2620 v3 CPU and 64 GB RAM. We show the effectiveness of the Provenance-based technique by a case study. By comparing the received alerts and events with the ground truth, we show the coverage rates of ground truth alerts and events reach above 97%. Then, we compare DOVE's bandwidth overhead with LossRadar [13], INTSight [14] and



(a) network topology and routing (b) flow A,B,C,D rate (c) accumulated traffic on s2 → s1 link and flow A alerts received from s1

Figure 3. **Provenance case study. Selected flow A suffers performance degradation from flow B,C,D competition on s2 → s1 link.**

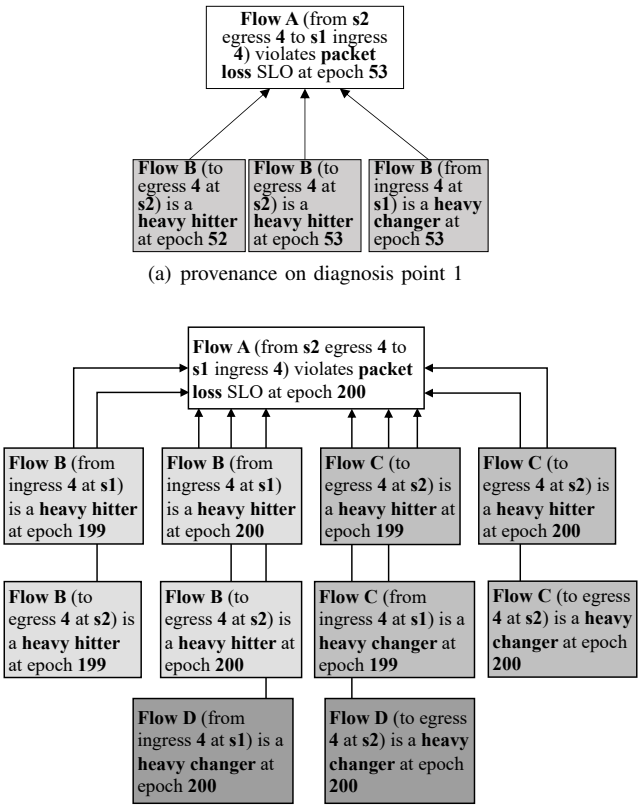
NetSight [11], showing DOVE generates the least traffic overhead. Finally, we compare DOVE's resource utilization (*i.e.,* TCAM, SRAM) with INTSight and adapted Switch-Pointer [26] in 6 real WAN topologies.

Despite all the processing logic to measure SLOs on the data plane, the additional logic into the Tofino ASIC pipeline has little impact on the processing throughputs, as long as the logic can fit into the resource constraints. The packet processing with DOVE is at the line rate.

A. Case Study: Provenance

We set up a simulation using Mininet with the topology shown in Figure 3(a). There are 4 switches, all of which are DOVE-enabled. Link capacities of $s3 \rightarrow s2$ and $s4 \rightarrow s2$ are 50 Mbps while link capacity of $s2 \rightarrow s1$ is 80 Mbps. Link delay is 1 ms. 2 hosts are attached to $s3$ and $s4$ respectively and one is attached to $s1$. 4 flows traversing from h_2, h_3, h_4, h_5 to h_1 are named flow A, B, C, D. Host h_0 serves as the analyzer and collects alerts and events via out-of-band links. The rates of flow A, B, C, D are shown in 3(b). The flows are first created and written into pcap files using the Python Scapy library. Then we use Tcpreplay to replay the traffic. The epoch length of DOVE is set to be $2^{16} = 65536$ microseconds. The threshold for packet loss and max delay is set to be 10 and 1.5 ms respectively. Percentile delay SLO is that 90-percentile delay should not exceed 1.2 ms. These SLOs are monitored on segments $s3 \rightarrow s2, s4 \rightarrow s2$ and $s2 \rightarrow s1$. Flows that are larger than 20 Mbps and flows that increase more than 2 Mb every second are considered heavy hitters and heavy changers. Such suspicious behaviors are reported from $s1$, $s2$, $s3$, $s4$. Flow A is the only selected flow while flow A, B, C, D are all watched flows.

We focus on the segment from $s2$ to $s1$. Figure 3(c) shows the accumulated flow rate (the left axis) with the running time (the bottom axis). It also shows whether an alert is received (1 for received) in each epoch (the top axis). Note that the epochs and running time are synchronized. The accumulated traffic fills link capacity at 3 s. Only after a short period (0.437 s), the first SLO violation alert of flow A is received. This is reasonable because although the queue in $s1$ starts piling at 3 s, the performance of flow A is not degraded much at the beginning. As the accumulated traffic continues to grow, flow A's SLOs can no longer be satisfied and alerts are reported in all of the following epochs.



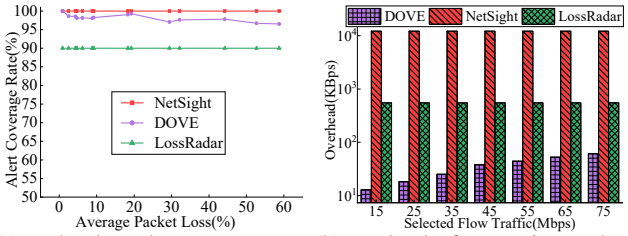
(a) provenance on diagnosis point 1

(b) provenance on diagnosis point 2

Figure 4. **Provenance result.**

Next, we choose two alerts shown as diagnosis points on Figure 3(c) to verify the effectiveness of provenance. The analysis results are shown in DAG as Figure 5. The first alert is mainly caused by flow B in Figure 4(a). This corresponds to the real traffic because flow B is a heavy changer (5Mbps) and heavy hitter (22.5 Mbps) near 3.43s. For the second alert, at 13.1s, the rate of flow B starts to drop but the absolute rate is still large (33 Mbps) at the moment. Flow C is a heavy hitter and changer. Flow D is a heavy changer but its rate is still less than 20 Mbps. The diagnosis result shown in Figure 4(b) corresponds to the real traffic change.

The number of watched flows is strongly connected with diagnosis accuracy. If the set of watched flows can cover all the heavy hitters and heavy changers like Figure 3, the



(a) packet loss alert coverage rate with 20 incast flows

(b) overhead of generating packet loss alerts on 8Gbps link with 1% packet loss

Figure 5. **Coverage rate and overhead of packet loss alert.**

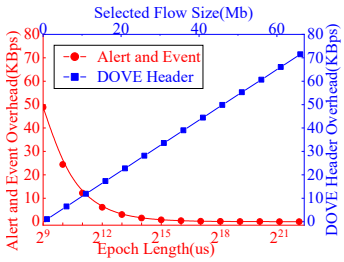


Figure 6. **Overhead of alert, event, and DOVE header.**

diagnosis will have promising results. As for the selected flows, their SLOs are measured and compared individually. Important flows are set to be selected flows by the network operators on a need-to basis. The number of selected flows has no relation to the diagnosis accuracy.

B. Coverage Rate and Bandwidth Overhead

We evaluate the coverage rate of ground truth alerts and events, showing that most (>97%) of the ground truth alerts and events are correctly reported. We use Mininet to build a 2-switch topology in this section.

1) Packet Loss

We run a fix-sized flow at the 50 Mbps rate on the link. Fix packet loss threshold. By gradually decreasing the link capacity from 50 Mbps, the average packet loss rate on the link increases. The number of received alerts showing violations of packet loss SLO increases as well. We evaluate the coverage by seeing whether there are any epochs during which packet loss SLO is not satisfied but the corresponding alert is never generated and reported.

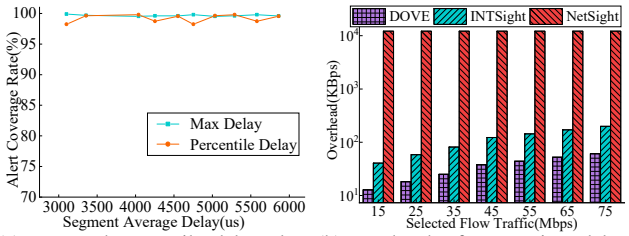
First, we need to find the ground truth of packet loss alerts. When the flow traverses the P4 software switch, each switch writes the traffic trace into pcap files and each packet of the flow is tagged with the timestamp when the packet enters or leaves the switch interfaces. We split the time when packets enter the upstream switch into epochs. Each packet can be classified into its epochs using the entering timestamp. Then each packet entering the upstream switch can be checked whether it is received at the downstream switch using downstream pcap files. Therefore, we can calculate the packet loss for each epoch, which is the packet loss ground truth. Finally, compare the ground truth packet loss with the threshold to get ground truth alerts. By comparing epoch id between received alerts and ground truth alerts, we calculate the coverage rate of ground truth alerts shown in Figure 5. As the average packet loss rate increases, the coverage of alerts is slowly dropping. This is because all of the packets which belong to the second half of the epoch are lost. The control bit of these packets is 1, which should have triggered a downstream switch to calculate packet loss. Due to the loss of upstream packets, packet loss is not calculated at the downstream switch, and alerts are not reported. However, this phenomenon is acceptable because when the average packet loss rate is big, the number of alerts is large enough to alert network operators.

We compare the coverage rate with NetSight [11] and LossRadar [13]. Since NetSight has the history of every packet on the control plane, it can easily calculate packet loss in each epoch on the control plane and generate alerts correctly. The coverage can reach 100%. LossRadar uses sketch-based techniques to store counter values on the data plane. Then decode and calculate packet loss on the control plane. However, with 20 incast flows, LossRadar can only decode 90% lost packets of the selected flow with 2KB digests (DOVE only needs 0.72KB for a selected flow) and therefore can only cover 90% alerts. The coverage comparison is shown in Figure 5(a). As shown in Figure 5(b), we compare bandwidth overheads of DOVE, NetSight and LossRadar. The total traffic running on the link is 8Gbps with a 1% average packet loss. DOVE generates around two orders of magnitude less overhead than Netsight, one order of magnitude less overhead than LossRadar.

The bandwidth overhead of DOVE consists of two parts, which are 1) DOVE header, and 2) alerts and events. Because alerts and events are reported periodically from a limited number of switches, there exist upper limits for the alert and event overhead, where every DOVE-enabled switches upload alerts and events on each epoch. The upper limits for alert and event overhead are proportional to the number of DOVE-enabled switches and have a negative relation to the epoch length. Considering one switch, as shown in the left and bottom axis in Figure 6, a finer epoch length has deeper visibility into network status at the cost of generating more traffic overhead. Network operators should set proper epoch lengths according to the network size and their needs. For the DOVE header, since the header is inserted into every packet of select flows, the overhead is proportional to the size of selected flows, as shown in the right and top axis in Figure 6. Besides, when the size of the selected flow and the epoch length are both small, the alert and event overhead can be the main contributor to the total overhead. As the size of the selected flow continually increases, the header overhead becomes the dominant one.

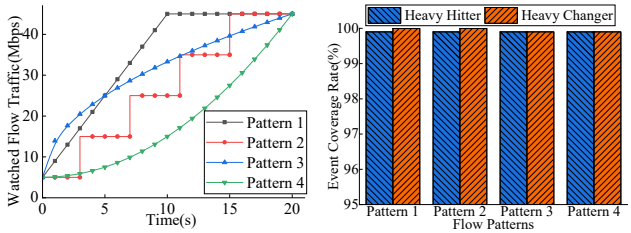
2) Delay

We run a fix-sized flow at a 50 Mbps rate on the link. Fix max and percentile thresholds. By gradually increasing the link delay from 3000 μs to 6000 μs, we test the alert coverage. By analyzing the pcap files written by P4 software switch like §IV-B1, we can accurately get the max delay and 90-percentile delay of each epoch and therefore get ground truth alerts. The coverage for max delay alert reaches near



(a) max and percentile delay alert coverage (b) overhead of generating delay alerts on 8Gbps link

Figure 7. **Coverage rate and overhead of delay alert.**



(a) traffic for testing events coverage rate (b) event coverage rate of heavy hitters and heavy changers

Figure 8. **Test traffic patterns and event coverage rate of heavy hitters and heavy changers**

100%. For percentile delay, though the average rate of the flow is fixed, the traffic size on different epochs fluctuates, making the estimated packet number per epoch not accurate. The inaccuracy leads to false negatives of alerts.

We compare the bandwidth overhead of DOVE with some other works able to measure delay. INTSight [14] is an SLO detection tool, which measures end-to-end delay and sends alerts for SLO violations. We believe our work outweighs INTSight by providing more fine-grained measurements and alerts based on network segmentation. As shown in Figure 7(b), DOVE's overhead is 70% lower than INTSight due to smaller telemetry headers.

3) Event

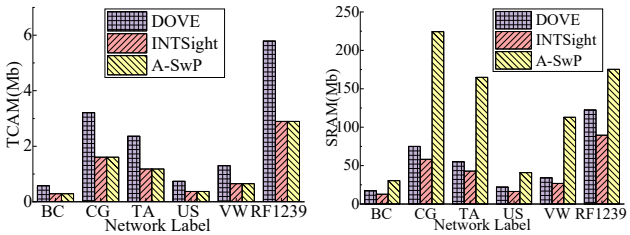
We run flows of 4 different patterns shown in Figure 8(a) on the link. Linear growth, step-growth, concave growth, and convex growth are the common types we see in daily traffic. We set 15 Mbps as the threshold of heavy hitters and 3 Mbps as the threshold of heavy changers. Similar to §IV-B1, we split the time into epochs according to the timestamp recorded in pcap files, count the traffic of every epoch and judge whether events should be reported. After comparison, we find that ground truth coverage of 4 patterns reaches above 99.9% as Figure 8(b).

C. Resource Utilization

We use REPETITA [27], which contains 260 network topologies and demand matrices, to estimate resource utilization in real WAN. First, we adapt SwitchPointer [26] by uploading packet telemetry headers to the control plane at the egress switch to get rid of end hosts for a fair comparison. Denote adapted SwitchPointer as A-SwP. We evaluate the resource utilization of DOVE, INTSight, and A-SwP on 6 network topologies. The 6 networks and their metadata are shown in Table V. Each network has a different number of nodes, links, and average path length. All the nodes in

Table V
Metadata of network topologies.

Network	Label	Nodes	Links	Average Path Length
Bell Canada	BC	48	130	5.3
US Signal	US	61	158	6.0
VTLWavenet	VW	92	192	13.1
TATA	TA	145	388	9.9
Cogent	CG	197	490	10.5
RF1239	RF1239	315	1944	4.0



(a) TCAM utilization (Mb) (b) SRAM utilization (Mb)

Figure 9. **Device resource utilization.**

the network are DOVE-enabled. For each pair of source and destination nodes, there are 512 selected flows and 512 watched flows. Hence, networks with more nodes have a larger number of monitored flows. The SRAM and TCAM utilization has linear relations with the number of flows. So larger networks require more resources. Note that resource utilization has no direct relations with the network topologies but only the number of flows. Figure 9 shows DOVE's, INTSight's and A-SwP's usage of TCAM and SRAM. The TCAM usage of DOVE is twice as much as that of INTSight. This is because DOVE has two lookup tables for selected flows and watched flows respectively. The total number of flows DOVE monitors is twice that of INTSight. A-SwP's usage of SRAM is the largest because it stores lists of switch ids on the data plane. DOVE does not record path information while INTsight stores fix-sized pathID to save spaces. DOVE's SRAM usage is more than INTSight since packet loss measuring in DOVE requires many registers used as triggers and counters. DOVE's TCAM and SRAM requirements are within the resources of Tofino, which has 44 Mb TCAM and 370 Mb SRAM. Note that the above resource utilization is based on the situation where DOVE is fully deployed. The resource utilization can be further reduced by partial and incremental deployment.

V. Discussion

In this section, we discuss clock synchronization, epoch implementation, and comparisons with sketch-based solutions.

Clock synchronization across different switches, which is required by all one-way delay measuring mechanisms, is essential in DOVE's delay measurement. IEEE 1588 [15] can achieve microsecond-level time synchronization, which is supported by increasing switch vendors. Packet loss measurement is not affected by clock asynchronization. Storing the number of sent packets and calculating the previous epoch's packet loss are both controlled by the control bits in the telemetry header. Besides, when clocks are asynchronized, Provenance rules should search in larger ranges. The same time period may have different epoch ids reported by different switches.

Larger searching ranges should cover the responsible events while linking more false-positive events.

Epoch implementations are different on Tofino [25] and BMv2 [24]. Tofino has the packet generator function, which can generate a certain number of packets at certain ports at certain intervals (*e.g.,* every 200 nanoseconds). This function can be used as timers. The generated packets can be used as pointers to traverse through the register tables. BMv2 does not have the packet generator function. For each selected and watched flow, a register is used to record epoch id, which is calculated by right shifting the timestamp (*e.g.,* by 16 bits). For every newly arrived packet of the selected and watched flows, its epoch is calculated and then compared with the one stored in the register. Timer triggers timeout signals if two are different and the register is updated with the latest epoch id.

Sketch-based solutions and DOVE are fundamentally different in design choices. Considering the resource constraints on the data plane, sketch-based solutions can store wider ranges of data, but at the risk of simple processing logic and inaccuracy, e.g., hash collisions and bucket collisions. For example, ElasticSketch [28] uses 9 stages of 12 on the Tofino pipeline to cover the full set of flows but is only designed to detect heavy hitters, changers, *etc.* On the contrary, the SLO measuring of DOVE has complicated data plane logic to verify SLOs, but only focuses on a limited number of selected flows. SLO measuring of DOVE uses 11 stages of 12 on the Tofino pipeline to implement the logic. Besides, SLO alerts should be accurate and timely as they remind network operators of the abnormal network status. Sketch-based solutions are naturally with inaccuracy and errors, and their process of decoding introduces extra delay. These features are not applicable to SLO measuring.

VI. Related Work

DOVE is a **debugging system** composed of **measuring** and **diagnosis**. In this section, we discuss and compare related works in these areas.

A. Performance Measuring

Traditional measuring tools, including Ping, traceroute, OWAMP [29], can actively probe flow path and measure the performance of delay and packet loss. But the probe packets may not be treated the same as data packets by the network. Traditional passive measuring tools, *e.g.,* SNMP [30] and NetFlow [18], can retrieve traffic counter values in switches to get flow information. However, traditional tools work at coarse timescales and fail to measure short-lived flows such as micro-burst flows.

Recent works, *e.g.,* LDA [31], RLI [32], MAPLE [33], develop data structures and algorithms to measure average delay at fine-grained timescale in different aggregation levels. QPipe [20] measures percentile delay of aggregated traffic on the data plane, failing to provide flow-specific information. Pingmesh [4] deploys an always-on delay detection system in a large-scale DCN. DOVE leverages the programmable data plane as the clean-slate solution to detecting per-packet delay more straightforwardly. For packet loss measuring, sketch-based approach LossRadar [13] compresses the values of flow packet counters in bloom filter structure, then decodes and calculates packet loss on the control plane. AM-PM [19] counts packets of the same color and synchronizes counter values on the control plane. DOVE develops a novel Coloring Algorithm running completely on a data plane.

B. Debugging System

SLO debugging techniques include INTSight [14]. INTSight can only attribute SLO violations to a small set of flows whose performance is just measured. It cannot monitor SLOs of percentile delay and packet loss.

Mirroring based techniques, *e.g.,* NetSeer [34], NetSight [11], Planck [12] and EverFlow [35], upload fine-grained information of every packet to control plane for measuring, debugging, logging, *etc.* However, the overhead of these methods is large, and they face severe scalability problems. Filters are used to selectively upload featured packets at the risk of losing key information.

End-host based techniques include SwitchPointer [26], TPP [36], *etc.* They rely on end hosts for retrieving and storing telemetry headers, which is usually designed for DCN. DOVE applies to a wider range of networks such as enterprise networks and WAN, where network operators are less likely to modify network stacks or do not have control over end hosts.

C. Diagnosis Techniques

Diagnosis techniques, including Dapper [7], DTaP [8], Zeno [10],Scout [37], *etc.*, are used to give detailed explanations of event causes in the distributed system. But all these methods require detailed queue information at an arbitrary time to diagnose, which is hard to acquire from network switches directly. DOVE makes Provenance applicable to SLO violation analysis by defining alerts, events and sacrificing certain accuracy.

VII. Conclusion

Detection and diagnosis are indispensable parts of fast recovery from SLO violations. To provide enough information needed by diagnosis and keep the overhead low, we propose DOVE, a diagnosis-driven SLO violation detection system. DOVE uploads alerts and events from the data plane selectively and efficiently. The design of DOVE includes the Coloring Algorithm, an approximate algorithm, and network segmentation. The evaluation testifies DOVE's effectiveness, the high accuracy (>97%) of alerts and events, and the low overhead. Memory utilization and required processing ability are low to be deployable in real network topologies.

Acknowledgement

We thank all the reviewers and our shepherd Marco Canini for their valuable comments and advice. This research is supported by the National Key R&D Program of China (2019YFB1802504) and the National Science Foundation of China (61625203, 61832013, and 61872426). Prof. Mingwei Xu and Dr. Yangyang Wang are the corresponding authors.

REFERENCES

[1] J. Dean, "Designs, lessons and advice from building large distributed systems," LADIS.

[2] E. Katz-Bassett, H. V. Madhyastha, J. P. John, D. Wetherall, and homas Anderson, "Studying black holes in the internet with hubble," in *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI 08)*. San Francisco, CA: USENIX Association, Apr. 2008.

[3] ThousandEyes, "Internet outages map on thousandeyes," https://www.thousandeyes.com/outages/.

[4] C. Guo, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *SIGCOMM*. ACM, August 2015.

[5] E. Schurman and J. Brutlag, "Performance related changes and their user impact," https://nanopdf.com/download/performance-related-changes-and-their-user-impact_pdf.

[6] K. Clay, "Amazon.com goes down, loses $66,240 per minute," https://www.forbes.com/sites/kellyclay/2013/08/19/amazon-com-goes-down-loses-66240-per-minute/.

[7] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a large-scale distributed systems tracing infrastructure," Google, Inc., Tech. Rep., 2010.

[8] W. Zhou, S. Mapara, Y. Ren, Y. Li, A. Haeberlen, Z. Ives, B. T. Loo, and M. Sherr, "Distributed time-aware provenance," *Proc. VLDB Endow.*, vol. 6, no. 2, p. 49–60, Dec. 2012.

[9] P. Buneman, S. Khanna, and W. C. Tan, "Why and where: A characterization of data provenance," in *Proceedings of the 8th International Conference on Database Theory*, ser. ICDT '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 316–330.

[10] Y. Wu, A. Chen, and L. T. X. Phan, "Zeno: Diagnosing performance problems with temporal provenance," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 395–420.

[11] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: Using packet histories to troubleshoot networks," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, Apr. 2014, pp. 71–85.

[12] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca, "Planck: Millisecond-scale monitoring and control for commodity networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, p. 407–418, Aug. 2014.

[13] Y. Li, R. Miao, C. Kim, and M. Yu, "Lossradar: Fast detection of lost packets in data center networks," ser. CoNEXT '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 481–495.

[14] J. Marques, K. Levchenko, and L. Gaspary, "Intsight: Diagnosing slo violations with in-band network telemetry," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 421–434.

[15] "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, 2008.

[16] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," vol. 44, no. 3, p. 87–95, Jul. 2014.

[17] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 357–371.

[18] "Cisco ios netflow," https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html, Jul 2017.

[19] G. Fioccola, A. Capello, M. Cociglio, L. Castaldelli, M. Chen, L. Zheng, G. Mirsky, and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring," RFC 8321, Jan. 2018.

[20] N. Ivkin, Z. Yu, V. Braverman, and X. Jin, "Qpipe: Quantiles sketch fully in the data plane," ser. CoNEXT '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 285–291.

[21] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica, "Declarative networking," *Commun. ACM*, vol. 52, no. 11, p. 87–95, Nov. 2009.

[22] X. Chen, S. L. Feibish, Y. Koral, J. Rexford, O. Rottenstreich, S. A. Monetti, and T.-Y. Wang, "Fine-grained queue measurement in the data plane," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 15–29.

[23] R. Joshi, T. Qu, M. C. Chan, B. Leong, and B. T. Loo, "Burstradar: Practical real-time microburst monitoring for datacenter networks," in *Proceedings of the 9th Asia-Pacific Workshop on Systems*, ser. APSys '18. New York, NY, USA: Association for Computing Machinery, 2018.

[24] Barefoot Networks, "P4-bmv2," https://github.com/p4lang/behavioral-model.

[25] ——, "Barefoot tofino switch," https://barefootnetworks.com/technology/.

[26] P. Tammana, R. Agarwal, and M. Lee, "Distributed network monitoring and debugging with switchpointer," ser. NSDI'18. USA: USENIX Association, 2018, p. 453–466.

[27] S. Gay, P. Schaus, and S. Vissicchio, "Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms," 2017.

[28] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 561–575.

[29] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)," IETF, RFC 4656, 9 2006.

[30] J. Case, M. Fedor, and M. Schoffstall, "A Simple Network Management Protocol (SNMP)," IETF, RFC 1098, 4 1989.

[31] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese, "Every microsecond counts: Tracking fine-grain latencies with a lossy difference aggregator," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 255–266.

[32] M. Lee, N. Duffield, and R. R. Kompella, "Not all microseconds are equal: Fine-grained per-flow measurements with reference latency interpolation," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, p. 27–38, Aug. 2010.

[33] ——, "Maple: A scalable architecture for maintaining packet latency measurements," in *Proceedings of the 2012 Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 101–114.

[34] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi, P. Zhang, D. Cai, M. Zhang, and M. Xu, "Flow event telemetry on programmable data plane," ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 76–89.

[35] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng, "Packet-level telemetry in large datacenter networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 479–491.

[36] V. Jeyakumar, M. Alizadeh, Y. Geng, C. Kim, and D. Mazières, "Millions of little minions: Using packets for low latency network programming and visibility," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, p. 3–14, Aug. 2014.

[37] J. Gao, N. Yaseen, R. MacDavid, F. Vieira Frujeri, V. Liu, R. Bianchini, R. Aditya, X. Wang, H. L. , D. Maltz, M. Y. , and B. Arzani, "Scouts: Improving the diagnosis process through domain-customized incident routing," in *SIGCOMM*. ACM, August 2020.