TeleNoise: A Network-Noise Module for In-Band Real-Time Telemetry

Vitalii Demianiuk
Ariel University, Israel
vitalii@ariel.ac.il

Sergey Gorinsky
IMDEA Networks Institute, Spain
sergey.gorinsky@imdea.org

Kirill Kogan[†]
Ariel University, Israel
kirill.kogan@gmail.com

***Abstract*—In-band real-time telemetry is a promising direction for management of modern programmable networks. While network noise in the form of packet reordering and loss affects in-band collection of distributed state, there is a need to compute telemetry functions on the collected state correctly despite the network noise. To address this common need, we propose TeleNoise that equips each packet with few sync bits and offers primitives of group affiliation and group completion to support noise-resilient computation of per-group telemetry functions. This paper gives real-world examples of such functions, elaborates on the role of TeleNoise in a modular in-band telemetry architecture, and presents algorithms for the two TeleNoise primitives. We derive analytical guarantees on correctness and performance of the algorithms and report a trace-driven evaluation that corroborates the effective low-overhead profile of TeleNoise, e.g., the assuredly correct operation and at most 1.6 packets of the average measurement lag for 12-packet groups and 3 sync bits.**

I. Introduction

The large scale and programmability of modern computer networks create new challenges and opportunities for network management. On the one hand, the increasing complexity of networks makes it harder to understand their distributed behavior. On the other hand, networks become capable of allocating more communication, storage, and computation resources for collection and analysis of distributed state. Abstractions that enable network telemetry include tiny packet programs [1] and path queries [2]. Telemetry improves network management via packet-drop localization [3], latency analytics [4], and edge-assisted network debugging [5]. Telemetry also provides real-time input to the execution of network protocols [6, 7].

INT [8], IOAM [9], and AM-PM [10] are prominent representatives of in-band real-time telemetry that collect distributed state via the data plane at line rates. The ingress network element embeds instructions into the sent packet, and the traversed elements follow the instructions to record data into the packet. When the packet reaches the egress element, the latter retrieves the collected data. Through the data collection and potential support of atomic functions, such in-band data-plane frameworks enable network-management applications to compute their metrics of interest.

In a number of real-world telemetry applications, correct computation of a metric requires proper handling of *network noise* in the form of packet reordering and loss. The loss rate of a packet stream is an example of the metrics where network noise is the subject of measurement. For computation of many other metrics, network noise is a disruption to overcome because packet reordering and loss distort the data collected through in-band telemetry, e.g., when the egress computes the size that the packet stream had at the ingress. Network noise becomes especially relevant for metrics defined for a group of packets such as a packet stream or its portion.

This paper proposes a *TeleNoise* module to support telemetry functions in their common need of dealing with network noise. TeleNoise operates on an end-to-end stream of packets from source S to destination D, partitions the stream at S into fixed-size groups of consecutive packets, and offers at D two atomic functions of *group affiliation* and *group completion*. Group affiliation determines belonging of the packet to the specific group. Group completion decides whether D has received from the group all the packets that can arrive from this group under the maximum assumed level of network noise.

To implement these two atomic functions, we seek practicable solutions with low communication overhead. In particular, TeleNoise neither expects from nor inserts into the packets any packet sequence numbers. We limit the communication overhead of TeleNoise to few in-band *sync bits* which associate the packet with its group. After discussing how real-world telemetry functions can utilize the TeleNoise primitives of group affiliation and group completion, we design specific algorithms for the two TeleNoise primitives and analytically derive guarantees on their correctness and performance. In these algorithms, the sync bits have the same value in all packets of a group to distinguish this group from adjacent groups. Our work also includes an evaluation on realistic traffic traces.

Our contributions. This paper contributes to in-band real-time telemetry as follows:

- Via the novel primitives of group affiliation and group completion, the proposed TeleNoise module supports robust computation of per-group metrics by telemetry functions in the presence of network noise.

[†] Kirill Kogan died on 13 March 2021 from complications of the coronavirus disease.

[$] The work was supported in part by: Israeli Innovation Authority under the Knowledge Transfer Commercialization Program (MAGNETON), grant 71249; Ariel Cyber Innovation Center in cooperation with the Israel National Cyber Directorate in the Prime Minister's Office; Regional Government of Madrid, grant P2018/TCS-4499, EdgeData-CM; Spanish Ministry of Science and Innovation, grant PID2019-109805RB-I00, ECID.

ISBN 978-3-903176-39-3© 2021 IFIP

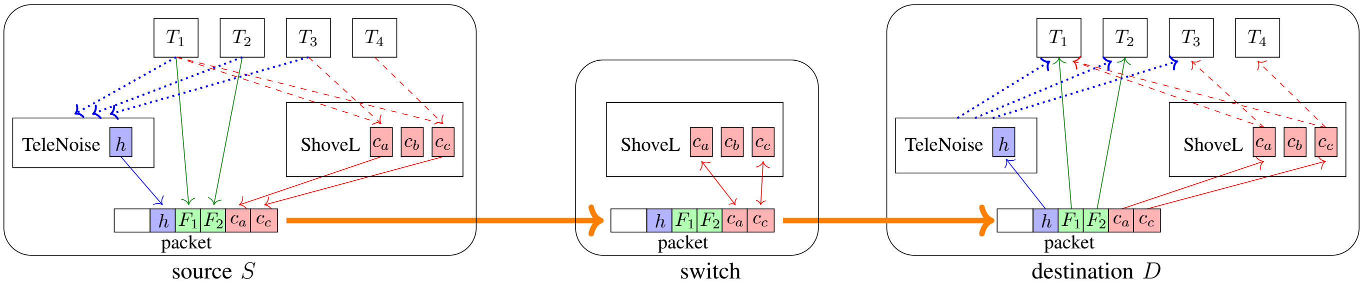

Figure 1. TeleNoise in a modular in-band real-time telemetry architecture.

- We illustrate how real-world telemetry functions can compute their metrics by leveraging TeleNoise.
- We develop algorithms for the TeleNoise primitives and analyze their correctness and performance guarantees.
- Our trace-driven experiments confirm the high effectiveness of TeleNoise despite its low communication overhead, e.g., its assuredly correct operation and at most 1.6 packets of the average measurement lag for 3 sync bits and 12-packet groups.

The rest of the paper has the following structure. Section II presents TeleNoise and its role in network telemetry. Section III provides examples of real-world telemetry functions that benefit from TeleNoise. Section IV designs TeleNoise algorithms for network noise limited to packet loss. Section V extends the work for noise comprising both loss and reordering. Section VI reports the experimental evaluation. Section VII discusses related work. Finally, Section VIII sums up our results.

II. TeleNoise module

While INT [8] and IOAM [9] represent general frameworks for in-band real-time telemetry, TeleNoise pursues a more specific purpose of providing support for resilient computation of per-group metrics under network noise. We view TeleNoise as a module that complements the general telemetry frameworks. After introducing TeleNoise and its novel contributions, this section discusses its role in the bigger telemetry picture.

TeleNoise functionality. TeleNoise is an end-to-end solution that operates on stream f of packets from source S to destination D. The module logically partitions the stream at S into fixed-size groups of consecutive packets. For different streams, group size G is configurable to different values between 1 and $|f|$ packets, i.e., a group can vary in size from one packet to the entire stream.

To support per-group telemetry, TeleNoise offers at D two primitives of group affiliation and group completion. Given a packet, the primitive of *group affiliation* identifies the group of the packet. For a specific group, the primitive of *group completion* determines whether D has already received all the packets that it can receive from this group under network noise. TeleNoise guarantees to telemetry functions that the two primitives perform correctly when network noise stays within the maximum assumed level. Hence, the telemetry functions at D can compute their per-group metrics with assured correctness.

One of our design objectives is to keep the communication overhead of TeleNoise low. TeleNoise neither assumes any packet sequence numbers nor inserts such numbers into the packets. The module adds only few *sync bits* per packet to keep the distributed state at S and D consistent. Figure 1 illustrates how TeleNoise at S inserts sync bits h into a packet and how TeleNoise at D extracts the sync bits from the packet. The specific TeleNoise algorithms that we present in Sections IV and V set the sync bits in all packets of a group to the same value that denotes a group type distinguishing the group from adjacent groups. Use of the TeleNoise module is optional for telemetry functions. In Figure 1, TeleNoise uses the same h sync bits in each packet to support telemetry functions T_1, T_2, and T_3 while function T_4 does not utilize TeleNoise.

TeleNoise leverages the sync bits to implement at D the atomic functions of group affiliation and group completion. In particular, TeleNoise defines a group as *completed* when D receives either the last packet from the group or, if the network loses all packets of this group, the first packet from the subsequent groups. When D receives a packet, TeleNoise at D issues a PacketReport() call to the interested telemetry functions, and this call identifies the non-completed group to which this packet belongs. Upon determining that a group has become completed, TeleNoise at D issues a GroupReport() call to notify the interested telemetry functions about the completion of this group.

Unlike AM-PM [10], TeleNoise does not inject any control packets and does not require an explicit notion of time. Instead, TeleNoise communicates distributed state in-band only and deals with time indirectly in regard to the orders of packet departures from S and packet arrivals to D. In particular, we define *measurement lag* $\lambda(J)$ for group J as the number of packets received by D between the moment when J became completed and the moment when TeleNoise determines that this group has become completed.

Architectural and system issues. While TeleNoise provides specialized support for noise-resilient per-group telemetry, we now discuss its role in general in-band telemetry. Telemetry functions do not have to rely solely on TeleNoise in computing their end-to-end metrics. For example, functions T_1 and T_2 in Figure 1 insert fields F_1 and F_2 into the packet header respectively to communicate their function-specific state from S to D.

Similarly, the end-to-end TeleNoise module can work alongside in-network telemetry solutions, e.g., INT and IOAM.

To elaborate on such potential cooperation, we consider a modular in-band telemetry architecture where a *ShoveL* module supports data-plane collection of distributed state at intermediate switches on the path of a packet from S to D. ShoveL provides logic for computation and in-band communication of *characteristics* c_k that capture properties of the distributed state. Examples of the characteristics include the switch identification number and egress buffer occupancy. As with TeleNoise, use of ShoveL by a telemetry function is optional. On the other hand, a function may request ShoveL to collect multiple characteristics. Furthermore, ShoveL efficiently collects the same characteristic for multiple interested functions: the module allocates a single field for the characteristic in the packet header and processes the characteristic only once at each switch. In Figure 1, ShoveL supports characteristics c_a, c_b, and c_c, function T_1 requests c_a and c_c, function T_2 does not request any characteristics, function T_3 requests c_a, function T_4 requests c_c, and the packet header carries c_a and c_c.

Whereas this paper focuses on TeleNoise algorithms and derives analytical guarantees for their correctness and performance, we leave the design of ShoveL to future work and only briefly comment here on system considerations. Because TeleNoise is an event-driven solution that does not involve a concept of time, its practical implementation would benefit from an additional mechanism, such as an explicit signal or timeout, to handle any non-completed groups at the end of the stream, e.g., the last group.

III. Examples of real-world telemetry functions

Before giving our full attention to the design and evaluation of specific TeleNoise algorithms, we utilize this section to provide several examples of real-world telemetry functions that benefit from the TeleNoise support.

Packet-loss function T_l. The *packet loss of a group* refers to the number of packets from this group that are lost on their way from S to D. Upon a packet arrival to D, TeleNoise notifies T_l via a PacketReport call about the group of this packet, and T_l increments its count of packets received from the group. Whenever TeleNoise determines that a group has become completed at D, TeleNoise issues a GroupReport call, and T_l finalizes the number of packets received from the group. By subtracting this number from the group size, T_l calculates the packet loss for the group.

Buffer-occupancy function T_b. A packet on its way from S to D might be queued or dropped at the egress link of a switch because the link buffer contains other packets awaiting transmission. Dividing the number of these buffered packets by the buffer size (i.e., the maximum number of packets that the buffer can accommodate) yields the relative buffer occupancy experienced by the arriving packet at the link. If the arriving packet is dropped because the buffer is full, the relative buffer occupancy experienced by the packet is 100%. With all links considered along the end-to-end path, the maximum observed relative buffer occupancy characterizes the most congested link for the packet. The *buffer occupancy for a group* is the average of the maximum buffer occupancies experienced by the packets of the group.

The buffer-occupancy function exemplifies the telemetry functions that benefit from both end-to-end TeleNoise and hop-by-hop ShoveL modules. As a packet transits from S to D, ShoveL at the switches updates the packet header with the maximum buffer occupancy experienced by this packet so far. Upon a packet arrival to D, TeleNoise informs T_b via a PacketReport call about the group of the packet, and T_b uses ShoveL to obtain the maximum buffer occupancy recorded in the packet. Whenever TeleNoise determines that a group has become completed, a GroupReport call from TeleNoise notifies T_b accordingly, and T_b computes the buffer occupancy for the completed group by averaging the maximum buffer occupancies experienced by all its packets, with each lost packet contributing the maximum buffer occupancy of 100%.

Although function T_b is capable of computing the buffer occupancy without utilizing the TeleNoise support, the computations that average only the maximum buffer occupancies in the received packets might greatly underestimate the correct answer. For instance, if a buffer overflow discards 9 packets, and the 10th packet arrives to D and reports the maximum buffer occupancy of 80%, T_b without TeleNoise computes the buffer occupancy for these packets as 80% while the correct answer computed by T_b with TeleNoise is 98%.

Out-of-order function T_r. The *out-of-order ratio for a group* is the fraction of the packet pairs that D receives out of their original order within the group. At switch S, function T_r inserts its field F_r into the packet header to communicate the sequential number of the packet within its group. When D receives a packet from the group, TeleNoise issues a PacketReport call to inform T_r about the group of the packet, and T_r considers the packet as out-of-order if the sequential number of the packet exceeds the number in the previous packet received from this group. When TeleNoise determines that a group has become completed, T_r receives a GroupReport call from TeleNoise and computes the fraction of the out-of-order packet pairs for the group.

IV. Algorithms for packet loss only

To start designing our algorithms for TeleNoise, we consider the special case where network noise consists of packet loss only, i.e., there is no packet reordering. *Loss parameter L* represents packet loss by imposing an upper limit on the number of consecutively lost packets, i.e., for any packet j of stream f such that j is less than $|f| - L$, destination D receives at least one of packets j through $j + L$. Let $\mathcal{A}$ be the set of all deterministic algorithms that implement TeleNoise for t sync bits and group size G. Also, let h_i refer to the t-bit field used by packet i for the sync bits. The following theorem characterizes the ability of an algorithm in $\mathcal{A}$ to correctly associate a packet with its group.

Theorem 1. *A deterministic algorithm from set $\mathcal{A}$ cannot guarantee correct affiliation of packets with their groups if*

$$L \geq (2^t - 2)G + \max(G, 2) \text{ packets.} \quad (1)$$

Proof. For a stream containing at least 2^tG packets, we construct loss patterns P_1 and P_2 that produce the same packet sequence at D in regard to the sync bits of the received packets even though P_1 and P_2 discard different numbers of packets for at least one group. In this case, any algorithm in $\mathcal{A}$ is unable to assuredly associate every packet with its group. To construct such patterns, we consider two packets i and j in two different groups such that $h_i = h_j$, $i < j$, and the value of $j - i$ is the smallest possible. Let k and m be two packets transmitted by source S between packets i and j, i.e., $i < k < m < j$. Because $j - i$ is the smallest possible, h_k can equal h_m only if packets k and m belong to the same group. Also, both h_k and h_m cannot equal h_i. Hence, $j - i$ is at most $(2^t - 1)G + 1$ packets. Moreover, $j - i$ can equal $(2^t - 1)G + 1$ packets iff i is the last packet in its group I, j is the first packet in its group J, and all packets belonging to groups I and J have the same value in their the sync bits.

For the scenario where $j - i$ equals $(2^t - 1)G + 1$ packets, and G is at least 2 packets, we construct such patterns P_1 and P_2 that both patterns discard packets $i + 1$ through $j - 1$, only P_1 discards the first packet in group I, and only P_2 discards the last packet in group J. In either of the patterns, the other packets of the stream arrive to D in order. Because all packets in groups I and J have the same value in their sync bits, D receives the same packet sequence in regard to the sync bits under either P_1 or P_2. Since only P_1 discards packets in group I, D cannot assuredly decide on affiliation of received packets with group I or J. At the same time, both P_1 and P_2 discard at most $(2^t - 1)G$ consecutive packets. Thus, D cannot guarantee correct affiliation of packets with their groups if condition 1 holds in this scenario.

For the scenarios where $j - i$ is at most $(2^t - 1)G$ packets, or G equals 1 packet, the constructed P_1 and P_2 patterns are as follows: only P_1 discards packet i, only P_2 discards packet j, and both patterns discard packets $i + 1$ through $j - 1$. Under either P_1 or P_2, the consecutive loss is at most $j - i + 1 \leq (2^t - 2)G + \max(G, 2)$ packets, and D receives the same packet sequence. Hence, the theorem holds in these scenarios too. □

Theorem 1 reveals that any TeleNoise algorithm with t sync bits and per-packet resolution, i.e., group size G equal to 1 packet, fails to assure correct group affiliation if the consecutive loss is at least 2^t packets, e.g., only 4 packets for t set to 2 bits. Increasing the group size relaxes the feasibility limit significantly. For example, when G and t are 16 packets and 2 bits respectively, the consecutive loss has to be at least 48 packets to preclude any TeleNoise algorithm from guaranteeing correct affiliation of packets with groups.

We now present Algorithm 1, a TeleNoise algorithm that approaches the feasibility limit in Theorem 1. The algorithm supports per-group telemetry functions $\mathcal{T}$ by implementing the primitives of group affiliation and group completion. In Algorithm 1, all packets of the same group have the same value in their sync bits, and this value distinguishes the group from adjacent groups in the stream. This t-bit value is called a *group type* and varies from 0 to $2^t - 1$. At source S, variable s tracks the group type for the packet that S transmits next, and variable c with values between 0 and $G - 1$ stores the id number of this packet within its group. When sending a packet of the stream, source S copies group type s into sync-bit field $p.h$ of this packet p, increments c modulo G, and – if p is the last packet in its group – increments s modulo 2^t. Destination D detects group completion by tracking the number of the first non-completed group in variable n. Upon receiving packet p, D compares the group type in field $p.h$ of the packet with the t least significant bits (LSBs) of variable n. As long as these t-bit numbers differ, D informs functions $\mathcal{T}$ about the completion of group n and increments n. When t LSBs of n equal sync bits $p.h$ in the packet, D notifies functions $\mathcal{T}$ about receiving packet p from group n.

Algorithm 1 TeleNoise for packet loss only

```
procedure SourceInit()
    $s \leftarrow 0$                          ▷ group type of the next packet
    $c \leftarrow 0$        ▷ id number of the packet within its group
procedure SourceUpdate($p$)
    $p.h \leftarrow s$                     ▷ setting the sync bits in packet $p$
    $c \leftarrow (c + 1) \mathbf{~mod~} G$
    if $c = 0$ then
        $s \leftarrow (s + 1) \mathbf{~mod~} 2^t$
procedure DestInit()
    $n \leftarrow 0$        ▷ number of the first non-completed group
procedure DestUpdate($p$)
    while $p.h \neq n$ mod $2^t$ do
        GroupReport($n$)             ▷ group $n$ is completed
        $n \leftarrow n + 1$
    PacketReport($n$, $p$)      ▷ packet $p$ is from group $n$
```

Theorem 2. *Algorithm 1 guarantees correct group affiliation and group completion to per-group telemetry functions $\mathcal{T}$ if*

$$L \leq (2^t - 1)G - 1 \textit{ packets.} \tag{2}$$

Proof. Consider such packets i and j from groups I and J respectively that j is the next packet received by D after i. Suppose that upon receiving packet i, D has n set to I. Because of the constraint on consecutive packet loss, $J - I$ is at most $\left\lceil \frac{L+1}{G} \right\rceil = 2^t - 1$ groups. If $J \neq I$, D reacts to the arrival of packet j by detecting the completion of groups I through $J - 1$, notifying functions $\mathcal{T}$ accordingly, and setting n to J. If $J = I$, D determines that packet j is from group I of packet i. In both cases, Algorithm 1 provides correct group affiliation and group completion for functions $\mathcal{T}$. □

Theorem 2 shows that the TeleNoise implementation by Algorithm 1 is highly resilient to packet loss. When G and t are 16 packets and 2 bits respectively, Algorithm 1 is robust to consecutive loss of up to 47 packets. The number of sync bits controls a trade-off between the assured resilience level and imposed communication overhead. In particular, the loss resilience of Algorithm 1 more than doubles upon incrementing t while preserving G. For fixed values of t

and L, group size G of $\left\lceil \frac{L+1}{2^t-1} \right\rceil$ or more packets ensures correct operation of Algorithm 1.

V. Algorithms for general network noise

A. Correctness guarantees

Algorithm 1 does not assure correct group affiliation and group completion if network noise includes packet reordering. For instance, when the network reorders packets i and $i{+}1$ from two consecutive groups, destination D incorrectly associates packet i with the group that follows the group of packet $i+1$. In this section, we generalize the approach of Algorithm 1 to provide correct TeleNoise operation for general network noise that includes both reordering and loss of packets.

We model general network noise via *loss parameter* L as defined in Section IV and *reordering parameter* R that captures the maximal extent of reordering in packets: packet i may reach destination D before packet j only if i is at most $j+R$. Theorem 3 establishes the feasibility limits on assuredly correct group affiliation for any algorithm from the $\mathcal{A}$ set introduced in Section IV.

Theorem 3. *A deterministic algorithm from set $\mathcal{A}$ cannot guarantee correct affiliation of packets with their groups if*

$$L > 0,\ R > 0,\ \text{and } L + 2R \geq (2^t - 1)G \text{ packets.} \qquad (3)$$

Proof. The proof follows the same general logic as the one for Theorem 1: we construct two different loss patterns P_1 and P_2 for the same stream with at least 2^tG packets; because the patterns discard different numbers of packets for at least one group, and either pattern produces the same packet sequence at destination D in regard to the sync bits of the packets, D is unable to assuredly associate packets with their groups. We define packets i and j as in the proof of Theorem 1. If $j-i$ exceeds $2R+1$ packets, we construct such loss patterns P_1 and P_2 that both patterns discard packets $i+R+1$ through $j-R-1$, pattern P_1 discards packet j and delivers packet i to D immediately after packet $i+R$, and pattern P_2 discards packet i and delivers packet j to D directly before packet $j-R$. With either P_1 or P_2, the other packets of the stream arrive to D in order, and D receives the same packet sequence in regard to the sync bits. Because L equals $j-i-2R-1$ packets, which is at most $(2^t-1)G-2R$ packets, D cannot assuredly guarantee correct association of packets with their groups if conditions 3 hold. For $j-i$ at most $2R+1$ packets, we use a similar construction to reach the same conclusion. □

In accordance with Theorem 3, no algorithm in set $\mathcal{A}$ can assure resilience to consecutive loss of 3 packets and reordering by 2 packets when G and t equal 1 packet and 3 bits respectively. As in the case with packet loss only, an increase in the group size relaxes the feasibility limits substantially. If we increase G to 16 packets and keep t at 3 bits, the value of $L+2R$ has to be at least 112 packets – e.g., consecutive loss of 60 packets and reordering by 31 packets – to prevent any algorithm in set $\mathcal{A}$ from ensuring correct affiliation of packets with groups.

Algorithm 2 TeleNoise for general network noise

```
procedure DestInit()
    $n \leftarrow 0$        ▷ number of the first non-completed group
procedure DestUpdate($p$)
    d ← $(p.h - n$ mod $2^t + 2^t)$ mod $2^t$
    while $d \geq \delta$ do
        GroupReport($n$)        ▷ group $n$ is completed
        $n \leftarrow n + 1$
        $d \leftarrow d - 1$
    PacketReport($n + d$, $p$)    ▷ $p$ is from group $n + d$
```

Algorithm 2 generalizes Algorithm 1 to robustly implement the TeleNoise primitives of group affiliation and group completion when network noise is general and close to the feasibility limits in Theorem 3. At source S, Algorithms 1 and 2 operate identically. The following assumptions underlie the operation of Algorithm 2 at destination D: if D receives a packet from group J, then each group that precedes J by δ or more groups is completed; in this case, there always exists such group I that every group preceding group I is completed and that D has not yet received any packet from group $I+\delta$ or subsequent groups. In Algorithm 2, destination D uses variable n to track the number of this group I. Upon receiving a packet, D computes d as the modulo-2^t difference between sync bits $p.h$ of this packet p and t LSBs of n. Variable d expresses the number of groups by which the group of packet p overshoots group n. As long as d is at least δ groups, D informs functions $\mathcal{T}$ about the completion of group n, increments n, and decrements d. When d is less than δ groups, D notifies functions $\mathcal{T}$ about receiving packet p from group $n+d$.

Figure 2 illustrates the operation of Algorithm 2 for t, G, and δ equal to 3 bits, 32 packets, and 3 groups respectively. The first non-completed group at D is group 5. If D receives packet p from group J where J is between 8 and 12, then overshoot d of group J is between 3 and 7 groups respectively, and D notifies about the completion of groups 5 through $2+d$. If J is between 5 and 7, then the overshoot is between 0 and 2 groups respectively, and D does not issue a completion notification for any group.

Theorem 4. *Algorithm 2 guarantees correct group affiliation and group completion to per-group telemetry functions $\mathcal{T}$ if*

$$R \leq (\delta - 1)G \text{ and } L + R \leq (2^t - \delta)G - 1 \text{ packets.} \qquad (4)$$

Proof. The proof uses induction to establish the invariants that every group preceding group n is completed and that D has not yet received any packet from group $n+\delta$ or subsequent groups. These invariants ensure that Algorithm 2 correctly associates each packet with its group and announces group completion for completed groups only.

Suppose that the invariants are true immediately before the arrival of packet p to D. Upon receiving the packet, D computes d to capture the number of groups by which group J of packet p overshoots group n. By induction, J is at least n. If J is less than $n+\delta$, then d is less than δ, and D does not announce

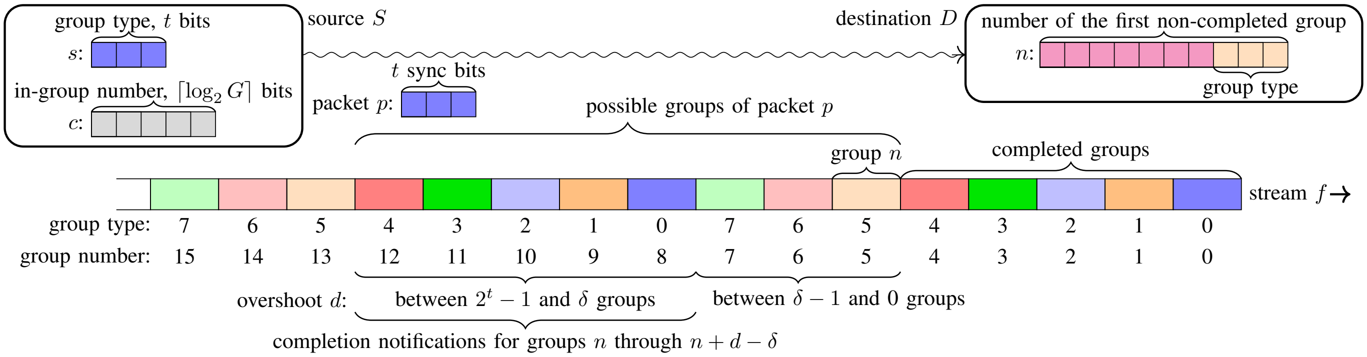
group type, t bits
s:
in-group number, $\lceil \log_2 G \rceil$ bits
c:
source S
destination D
number of the first non-completed group
n:
group type
t sync bits
packet p:
possible groups of packet p
group n
completed groups
stream f
group type: 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
group number: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
overshoot d:
between $2^t - 1$ and δ groups
between $\delta - 1$ and 0 groups
completion notifications for groups n through $n + d - \delta$

Figure 2. Operation of Algorithm 2 with $t = 3$ sync bits, group size $G = 32$ packets, and $\delta = 3$ groups. Group 5 is the first non-completed group at D.

any group completion and notifies functions $\mathcal{T}$ only about the arrival of packet p from group $n+d$, which is group J of this packet p. If J is at least $n+\delta$, then p is the first packet to reach D among all the packets from group $n+\delta$ and subsequent groups. Hence, at most $L+R$ packets separate group $n+\delta-1$ and packet p at source S (in the worst case, the network loses the first L of these packets and delivers the next R packets to D after packet p), J is between $n+\delta$ and $n+2^t-1$ due to the $L+R$ constraint in conditions 4, and the overshoot is at least δ groups. In this case, D notifies functions $\mathcal{T}$ about the completion of groups n through $J-\delta$ and sets n to $J-\delta$. Since R is less than $(\delta-1)G$ packets, all groups that precede group $J-\delta$ at S are completed. Therefore, both invariants hold after D processes packet p. □

Theorem 4 reveals that even small communication overhead of several sync bits provides high assured resilience to network noise. For example, when G is equal to 32 packets, t equals 3 bits, and δ is 2 groups, Algorithm 2 is assured to operate correctly if R and $L+R$ are at least 32 and 159 packets respectively.

Parameter t, which denotes the number of sync bits, regulates a trade-off between the communication overhead and assured network-noise resilience of Algorithm 2. For instance, incrementing t by 1 bit without changing δ and G relaxes the $L+R$ constraint in conditions 4 by at least twice. Also, changing t and δ to $t+1$ and $2\delta-1$ respectively enables halving the group size without tightening conditions 4.

For fixed values of G and t, parameter δ controls a trade-off between loss resilience and reordering resilience of Algorithm 2. Note that network noise satisfies conditions 4 for δ exceeding 2^{t-1} groups only if it satisfies these conditions for δ equal to 2^{t-1} groups. Hence, we here consider the δ values between 1 and 2^{t-1} groups. When δ equals 1 group, Algorithm 2 exhibits the highest resilience to packet loss and is equivalent to Algorithm 1, which does not guarantee correct operation with reordering by only one packet. For δ equal to 2^{t-1} groups, Algorithm 2 is guaranteed to operate correctly if R and $L+R$ are at least $(2^{t-1}-1)G$ and $2^{t-1}G-1$ packets respectively.

For fixed values of L, R, and t, we define G_{min} as the minimal group size that satisfies conditions 4 for at least one δ value. G_{min} is the smallest among the G values that satisfy the following inequality:

$$\left\lceil \frac{R}{G} \right\rceil + \left\lceil \frac{L+R+1}{G} \right\rceil \leq 2^t - 1. \qquad (5)$$

For instance, when L equals 110 packets, R equals 40 packets, and t is 3 bits, we have G_{min} of 31 packets. According to the guarantees provided by Algorithm 2 to telemetry functions $\mathcal{T}$, the number of consecutive non-completed groups that have delivered a packet to D is always at most δ. If a telemetry function maintains state at D for each such group, the memory requirements at D increase linearly with δ. Thus, when group size G is fixed, parameter δ in Algorithm 2 presents a trade-off between assured reordering resilience and memory requirements of $\mathcal{T}$ at D. Among those values of δ that equip Algorithm 2 with at least some assured robustness to both reordering and loss, δ equal to 2 groups minimizes the memory requirements of $\mathcal{T}$ at D. For fixed values of L, R, t, and G satisfying inequality 5, the minimal δ_{min} value of δ that satisfies conditions 4 equals $\left\lceil \frac{R}{G} \right\rceil + 1$ groups.

B. Performance guarantees

Since the network can lose or reorder packets, no online algorithm at destination D can always detect the completion of a group at the actual moment of the group completion because of the uncertainty whether the missing packets of the group are indeed lost or will still arrive. This section analyzes the measurement lag of Algorithm 2. As defined in Section II, the measurement lag for a group is the number of packets received by D between the actual group completion and group-completion detection by the algorithm. Below, we establish upper bounds on the measurement lag of Algorithm 2 in relation to $\mathcal{B}(\delta, G)$ which denotes the total number of packets in $\delta-1$ groups, i.e., $\mathcal{B}(\delta, G)$ equals $(\delta-1)G$ packets.

Theorem 5. *If conditions 4 hold, the measurement lag of Algorithm 2 is at most $2\mathcal{B}(\delta, G) + 1$ packets for each group and at most $\mathcal{B}(\delta, G) + 1$ packets on average.*

Proof. Consider all packets that arrive to D between the completion of group J and the detection of the group completion by Algorithm 2. Because R is at most $(\delta-1)G$ packets, these packets belong to either $\delta-1$ groups preceding group J or

$\delta-1$ groups following group J. Accounting also for the packet that triggers the group-completion detection by its arrival to D, we bound measurement lag $\lambda(J)$ for group J to be at most $2\mathcal{B}(\delta,G)+1$ packets.

Now, consider a pair of groups J and I such that $|J-I|$ is at most $\delta-1$ groups. The packets of group J can be accounted towards measurement lag $\lambda(I)$ of group I only if group I becomes completed before group J. Otherwise, the packets of group I can be accounted towards measurement lag $\lambda(J)$ of group J. Among all m groups in stream f, there are at most $m(\delta-1)$ such pairs of groups. Therefore, the total measurement lag for all m groups is at most $m(\delta-1)G+m$ packets, and the average measurement lag is at most $\mathcal{B}(\delta,G)+1$ packets. □

In Algorithm 2, group size G controls a trade-off between the measurement lag and assured resilience to network noise. In particular, $\mathcal{B}(\delta,G)$ scales linearly with G. For fixed values of L, R, and t, we define G_λ and δ_λ as the group size and δ value that minimize $\mathcal{B}(\delta,G)$ while assuring the correct operation of Algorithm 2.

Theorem 6. *With reordering by at least 1 packet, G_λ and δ_λ equal* $\max\left(\left\lceil\frac{L+R+1}{2^t-2}\right\rceil, R\right)$ *packets and 2 groups respectively.*

Proof. When there is reordering by at least 1 packet, δ in conditions 4 is at least 2 groups, and these conditions hold iff G is at least $\max\left(\left\lceil\frac{L+R+1}{2^t-\delta}\right\rceil, \left\lceil\frac{R}{\delta-1}\right\rceil\right)$ packets. Thus, $(\delta-1)\max\left(\left\lceil\frac{L+R+1}{2^t-\delta}\right\rceil, \left\lceil\frac{R}{\delta-1}\right\rceil\right)$ represents the minimum value of $\mathcal{B}(\delta,G)$ for Algorithm 2 to operate correctly. Note that with δ being at least 2 groups, the values of $(\delta-1)\left\lceil\frac{L+R+1}{2^t-\delta}\right\rceil$ and $(\delta-1)\left\lceil\frac{R}{\delta-1}\right\rceil$ are the smallest when δ equals 2 groups. Hence, we obtain δ_λ and G_λ as 2 groups and $\max\left(\left\lceil\frac{L+R+1}{2^t-2}\right\rceil, R\right)$ packets respectively. □

To illustrate Theorem 6, we consider L, R, and t equal to 80 packets, 5 packets, and 3 bits respectively. In these settings, G_λ of 15 packets and δ_λ of 2 groups minimize the upper bounds on the measurement lag of Algorithm 2 at 31 packets for each group and 16 packets on average. When there is no packet reordering, the G_λ and δ_λ values that minimize the upper bounds on the measurement lag are $\left\lceil\frac{L+1}{2^t-1}\right\rceil$ packets and 1 group respectively.

Let us now contrast G_λ and δ_λ, which minimize the measurement-lag upper bounds, with the G and δ settings that minimize the group size while assuring the correct operation of Algorithm 2 as discussed in Section V-A. The following theorem considers G_{min} and $\delta_{min}=\left\lceil\frac{R}{G_{min}}\right\rceil+1$ groups and demonstrates that the measurement-lag upper bounds remain close to minimal.

Theorem 7. *$\mathcal{B}(\delta_{min},G_{min})$ exceeds $\mathcal{B}(\delta_\lambda,G_\lambda)$ by less than G_{min} packets.*

Proof. Since δ_{min} equals $\left\lceil\frac{R}{G_{min}}\right\rceil+1$ groups, $\mathcal{B}(\delta_{min},G_{min})$ exceeds R by less than G_{min} packets. Because $\mathcal{B}(\delta_\lambda,G_\lambda)$ is at least R packets, the result follows. □

We conclude this section by presenting Algorithm 3 that modifies Algorithm 2 at destination D to decrease the measurement lag. The modification targets the scenario where D receives all packets of a group. To announce the completion of such group as soon as the last of its pending packets arrives to D, Algorithm 3 at D maintains for each of groups n through $n+\delta-1$ a counter of packets that have arrived from this group. When a packet arrival from a group brings to G the count of packets received from the group, D announces the completion of this group without any measurement lag. While requiring additional $\delta\lceil\log_2 G\rceil$ bits to store the packet counters at D, Algorithm 3 can substantially reduce the average measurement lag.

VI. Experimental Evaluation

Methodology. We adopt the method of the evaluations in VL2 [11], pFabric [12], pHost [13] and RoDiC [14] and drive simulations with realistic traffic traces generated from the data-mining distribution of stream sizes [11]. We generate a total of 10^6 streams, with at most $\frac{2}{3}\cdot 10^6$ packets in a stream, and use YAPS [15] simulator in its unreliable transport configuration. The considered leaf-spine network topology contains 4 spine switches and 9 ToR leaf switches with 16 servers connected to each ToR switch, i.e., 144 servers altogether. 90% of all streams traverse a spine switch. Our simulation code is public [16].

To simulate different congestion levels, we vary congestion parameter β that controls the expected time between the transmissions of two consecutive packets from the same source. While congestion becomes smaller as β increases, we consider β values between 0.625 and 1.25. Buffer size Δ at the intermediate switches varies from 9 to 59 packets. In our *standard experiment* settings, β and Δ are 1 and 24 packets respectively. Whereas t is either 2 or 3 sync bits in the experiments, δ is between 2 and 2^{t-1} groups. For all considered values of t and δ, Algorithms 2 and 3 are assured to operate correctly on all streams with the group size of at least 33 packets in the standard experiment. Hence, we vary G from 5 up to 33 packets and evaluate only relatively large streams where the number of packets is at least $8\cdot 33+1$, i.e., 265 packets.

Impact of group size G on algorithm correctness. Since Algorithms 2 and 3 have identical correctness properties, our correctness-related experiments assess only Algorithm 2. For given t and δ, we evaluate all the considered streams in the standard experiment in regard to two metrics: $Z(t,\delta)$ measures the percentage of the streams for which Theorem 4 does not assure correctness of Algorithm 2, and $E(t,\delta)$ records the percentage of the streams for which Algorithm 2 actually makes at least one error.

Figure 3a manifests that the assured correctness of Algorithm 2 grows steeply with the group size. With $t=2$ sync bits and $\delta=2$ groups, more than 23% of the streams violate conditions 4 for G equal to 5 packets, and $Z(2,2)$ declines dramatically to 0.5% and 0.003% (the latter corresponds to 3 streams only) for the group size of 15 and 25 packets respectively. The algorithm correctness is assured when G is at least 31 packets. The $E(2,2)$ line shows that Algorithm 2

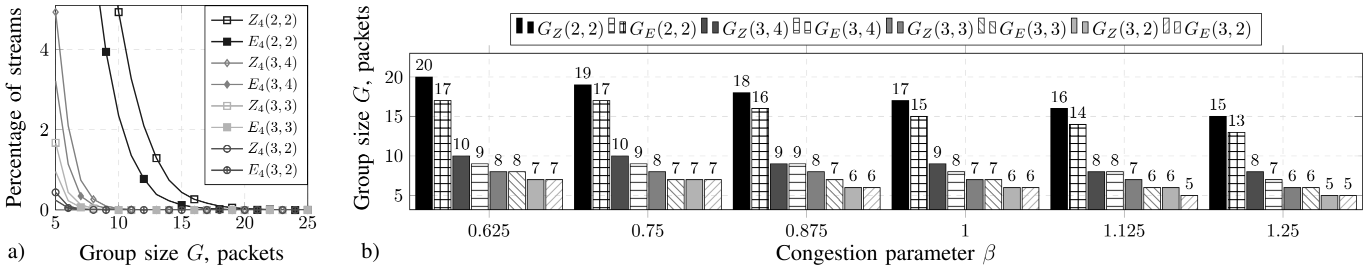

Figure 3. Assured correctness vs. actual correctness of Algorithm 2, dependencies on: a) group size G and b) congestion parameter β.

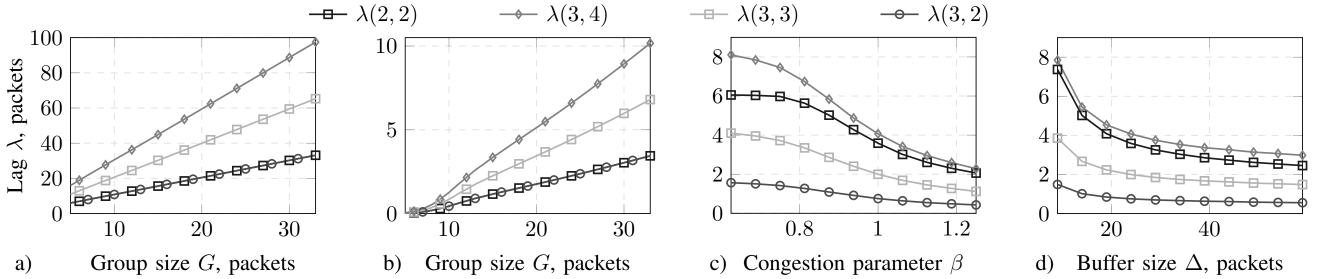

Figure 4. Average measurement lag $\lambda(t, \delta)$ of: a) Algorithm 2 and b-d) Algorithm 3.

operates incorrectly for 19%, 0.13%, and 0.001% (1 stream only) of the streams for the group size of 5, 15, and 25 packets respectively. When G is at least 26 packets, Algorithm 2 operates correctly for all the streams. $E(2,2)$ is consistently smaller than $Z(2,2)$ because conditions 4 are sufficient but not necessary for Algorithm 2 to operate correctly.

Figure 3a also unveils that an increase in t significantly expands the range of the group sizes for which the correct operation of Algorithm 2 is guaranteed. Even for G equal to 5 packets, the $Z(3,4)$, $E(3,4)$, $Z(3,2)$, and $E(3,2)$ values are below 5%, 4%, 0.5%, and 0.3% respectively. When t and δ equal 3 bits and 2 groups respectively, the group size of 11 or more packets is sufficient to assure correct operation of Algorithm 2 for all streams. As δ decreases, $Z(t,\delta)$ and $E(t,\delta)$ drop significantly because loss parameter L becomes more dominant over reordering parameter R in the tolerated network noise.

Impact of β and buffer size Δ on correctness. In assessing the impact by β and Δ, we ignore few outliers, which constitute 0.2% of the streams examined before, and report the results for the other 99.8% of the streams in respect to the following two metrics: $G_Z(t,\delta)$ refers to the minimum group size such that Theorem 4 assures correctness of Algorithm 2 for these streams, and $G_E(t,\delta)$ captures the minimum group size such that the actual operation of Algorithm 2 on these streams is correct.

Figure 3b exhibits the dependencies of $G_Z(t,\delta)$ and $G_E(t,\delta)$ on β when buffer size Δ equals 24 packets. As β increases from 0.625 to 1.25, the $G_Z(2,2)$, $G_E(2,2)$, $G_Z(3,2)$, and $G_E(3,2)$ values decrease from 20, 17, 7, and 7 packets to 15, 13, 5, and 5 packets respectively. For both kinds of metrics, the growth of β from 0.625 to 1.25 triggers the value decline by at most 5 and 2 packets when t is 2 and 3 sync bits respectively. Thus, Algorithm 2 can operate correctly with small groups even under heavy congestion. The correctness of the algorithm operation is largely indifferent to the buffer size. As Δ increases from 19 to 59 packets with β equal 1, either $G_Z(t,\delta)$ or $G_E(t,\delta)$ changes by at most 1 packet for all the examined t and δ values.

Impact of group size G on measurement lag. To evaluate average measurement lag $\lambda(t,\delta)$ of Algorithms 2 and 3, we consider only those streams on which the algorithms operate correctly. Figures 4a and 4b show the dependence of $\lambda(t,\delta)$ on the group size in the standard experiment for Algorithms 2 and 3 respectively. While the upper bound on the average lag in Theorem 5 is linear in G, the actual average lag of Algorithm 2 exhibits a similar dependence and numerically stays close to this theoretical bound. The $\lambda(2,2)$ and $\lambda(3,2)$ lines fully overlap because they depict the same setting where δ equals 2 groups. Figures 4a and 4b reveal that Algorithm 3 reduces lag $\lambda(t,\delta)$ by an order of magnitude compared to Algorithm 2. For example, lag $\lambda(3,2)$ is 11.7 and 0.6 packets for Algorithms 2 and 3 respectively when the groups are sized to 11 packets to assure that both algorithms operate correctly on all the examined streams. The decrease in the average measurement lag is dramatic because the arrival of all packets from a group, i.e, the scenario targeted by Algorithm 3, is a common occurrence, which becomes more frequent with smaller groups.

Impact of β and buffer size Δ on measurement lag. Finally, we evaluate how β and Δ affect lag $\lambda(t,\delta)$ of Algorithms 2 and 3. For fixed values of t and δ, we select the smallest G value such that Theorem 4 assures correctness of the two algorithms on all streams examined in the experiments. For Algorithm 2, lag $\lambda(t,\delta)$ remains close to the upper bound in Theorem 5 regardless of the β and Δ values. Hence, Figures 4c and 4d focus on Algorithm 3, for which the lag is highly sensitive to β and Δ. As β increases from 0.625 to 1.25, Figure 4c demonstrates that the $\lambda(2,2)$, $\lambda(3,2)$, $\lambda(3,3)$, and

$\lambda(3,4)$ values significantly decline by a factor of 3.6, 2.9, 3.7 and 3.6 respectively because a larger fraction of the groups have all their packets delivered to D when congestion decreases. While we show above that buffer size Δ makes only a small impact on the minimum group size needed by the algorithm to operate correctly, Figure 4d unveils that lag $\lambda(t,\delta)$ of Algorithm 3 decreases sharply when the buffer size grows, again due to the increasing prevalence of the scenario targeted by the algorithm. For instance, with 3 sync bits and δ equal to 2 groups, $\lambda(3,2)$ is 1.6 and 0.5 packets when the buffer accommodates 9 and 59 packets respectively.

The above evaluation confirms that TeleNoise offers effective support to per-group telemetry functions. In particular, with t, G, and δ equal to 3 bits, 12 packets, and 2 groups respectively, Algorithm 3 is guaranteed to operate correctly on all examined streams and has the average measurement lag of at most 1.6 packets in all evaluated scenarios.

VII. Related work

RoDiC [14, 17] proposes packet grouping and state overlap as the bases for robust distributed monitoring of the stream size. TeleNoise adopts these principles to support noise-resilient computation of general per-group telemetry functions.

TeleNoise complements prior work on data-plane telemetry by offering the novel primitives of group affiliation and group completion that facilitate robust computation of per-group metrics. While INT [8] is an extensive monitoring framework based on per-packet data collection, INT does not offer primitives for noise-resilient per-group telemetry. Whereas IOAM [9] defines formats and procedures for communication of telemetry information via data packets, TeleNoise can leverage IOAM for communication of its sync bits from the source to the destination. With TeleNoise supporting robust telemetry for end-to-end packet streams, OmniMon [18] deals with network-wide telemetry. Both TeleNoise and AM-PM [10, 19–21] target per-group telemetry. While AM-PM fixes a time interval to delineate packet groups, TeleNoise fixes the group size and thereby removes a need for additional packets as an out-of-band control channel. Also, unlike our work, AM-PM does not provide analytical guarantees for correct metric computation.

In addition to the above proposals that represent conceptual differences of TeleNoise from related work, a much larger body of previous research on in-band telemetry extends and applies the representative approaches [22]. For example, PINT [6] uses approximation methods to reduce the communication overhead of INT and studies the impact on HPCC congestion control [7].

VIII. Conclusion

This paper presented TeleNoise that adds few sync bits per packet and offers the novel primitives of group affiliation and group completion to support per-group telemetry functions in their common need of dealing with network noise. We provided examples of such functions, discussed the TeleNoise role in a modular in-band telemetry architecture, designed specific TeleNoise algorithms, and analyzed their correctness and performance properties. Our trace-driven evaluation of the algorithms confirmed their high effectiveness with low communication overhead, such as the assuredly correct operation on all the examined streams and average measurement lag of at most 1.6 packets when TeleNoise uses 3 sync bits with groups sized to 12 packets. In future work, we will expand the evaluation scope, e.g., experiment on other traffic traces.

References

[1] V. Jeyakumar, M. Alizadeh, Y. Geng, C. Kim, and D. Mazières, "Millions of Little Minions: Using Packets for Low Latency Network Programming and Visibility," in *SIGCOMM*, 2014, pp. 3–14.

[2] S. Narayana, M. Tahmasbi, J. Rexford, and D. Walker, "Compiling Path Queries," in *NSDI*, 2016, pp. 207–222.

[3] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. Liu, J. Padhye, B. T. Loo, and G. Outhred, "007: Democratically Finding the Cause of Packet Drops," in *NSDI*, 2018, pp. 419–435.

[4] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. A. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z. Lin, and V. Kurien, "Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis," in *SIGCOMM*, 2015, pp. 139–152.

[5] P. Tammana, R. Agarwal, and M. Lee, "Simplifying Datacenter Network Debugging with PathDump," in *OSDI*, 2016, pp. 233–248.

[6] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic In-Band Network Telemetry," in *SIGCOMM*, 2020, p. 662–680.

[7] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "HPCC: High Precision Congestion Control," in *SIGCOMM*, 2019, pp. 44–58.

[8] The P4.org Applications Working Group, "Telemetry Report Format Specification, Version 2.0," 2020, https://github.com/p4lang/p4-applications/blob/master/docs/telemetry_report_v2_0.pdf.

[9] F. Brockners, S. Bhandari, and T. Mizrahi, "Data Fields for In-situ OAM," IETF, Internet-Draft draft-ietf-ippm-ioam-data-12, 2021.

[10] T. Mizrahi, G. Navon, G. Fioccola, M. Cociglio, M. G. Chen, and G. Mirsky, "AM-PM: Efficient Network Telemetry using Alternate Marking," *IEEE Network*, vol. 33, no. 4, pp. 155–161, 2019.

[11] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in *SIGCOMM*, 2009, pp. 51–62.

[12] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal Near-Optimal Datacenter Transport," in *SIGCOMM*, 2013, pp. 435–446.

[13] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, "pHost: Distributed Near-Optimal Datacenter Transport over Commodity Network Fabric," in *CoNEXT*, 2015, pp. 1:1–1:12.

[14] V. Demianiuk, S. Gorinsky, S. Nikolenko, and K. Kogan, "Robust Distributed Monitoring of Traffic Flows," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 275–288, February 2021.

[15] G. Kumar, A. Narayan, and P. Gao, "YAPS = Yet Another Packet Simulator," 2021, https://github.com/NetSys/simulator.

[16] V. Demianiuk, S. Gorinsky, and K. Kogan, "TeleNoise: A Network-Noise Layer for Real-Time In-Band Telemetry (Code for Evaluation)," 2021, https://github.com/chavit/TrafficMonitoring.

[17] V. Demianiuk, S. Gorinsky, S. Nikolenko, and K. Kogan, "Distributed Counting along Lossy Paths without Feedback," in *SIROCCO*, 2018, pp. 1–4.

[18] Q. Huang, H. Sun, P. P. C. Lee, W. Bai, F. Zhu, and Y. Bao, "OmniMon: Re-Architecting Network Telemetry with Resource Efficiency and Full Accuracy," in *SIGCOMM*, 2020, p. 404–421.

[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," IETF, RFC 8321, 2018.

[20] T. Mizrahi, C. Arad, G. Fioccola, M. Cociglio, M. Chen, L. Zheng, and G. Mirsky, "Compact Alternate Marking Methods for Passive and Hybrid Performance Monitoring," IETF, Internet-Draft draft-mizrahi-ippm-compact-alternate-marking-05, 2019.

[21] A. Riesenberg, Y. Kirzon, M. Bunin, E. Galili, G. Navon, and T. Mizrahi, "Time-Multiplexed Parsing in Marking-based Network Telemetry," in *SYSTOR*, 2019, pp. 80–85.

[22] L. Tan, W. Su, W. Zhang, J. Lv, Z. Zhang, J. Miao, X. Liu, and N. Li, "In-band Network Telemetry: A Survey," *Computer Networks*, vol. 186, 107763, February 2021.