SPRINT: Line-Rate In-band Network Telemetry Recovery for Application Optimization

Bingzhen Chen[1], WaiMing Lau[1], Xiaoquan Zhang[1], Fung Po Tso[2], and Lin Cui[1,*]
[1]*Department of Computer Science, Jinan University, Guangzhou, China.*
[2]*Department of Computer Science, Loughborough University, UK.*

***Abstract*—In-band Network Telemetry (INT) has enabled efficient, real-time network monitoring and analysis by embedding telemetry data directly into live traffic. However, INT packet loss is inevitable due to network limitations (e.g., congestion, buffer overflow), which compromises the accuracy and effectiveness of INT-based applications. Existing INT recovery solutions cannot operate on the data plane, making them impractical for latency-sensitive in-network computing (INC) applications (e.g., RHA), which rely on accurate INT for decisions. To address this gap, this paper introduces SPRINT, the first forward-error-correction (FEC) INT recovery mechanism that runs entirely at line-rate within the programmable data plane. SPRINT leverages a novel, hardware-friendly shift powered XOR encoding with nested grouping to minimize redundant state and maximize throughput, thereby providing robust fault tolerance. We have implemented a testbed prototype of SPRINT on P4 hardware (Intel Tofino ASIC) and the BMv2 software switch. Our evaluation shows that SPRINT uses under 10% of on-chip SRAM, achieves a minimum recovery latency of approximately 21 μs, delivers a 99% recovery rate, and maintains strong reliability for INT-based applications, e.g., HPCC and RHA.**

***Index Terms*—P4, In-band Network Telemetry, In-Network Computing, FEC**

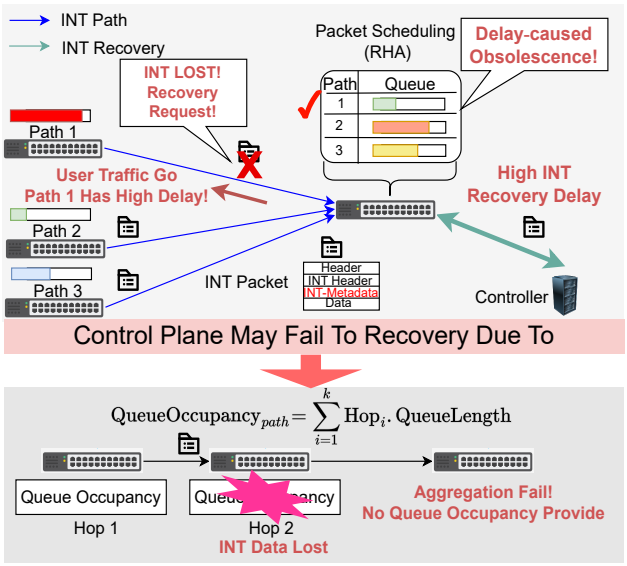


Fig. 1: INT Recovery on Control Plane Suffers in Cases.

I. Introduction

In-band Network Telemetry (INT) is revolutionizing network monitoring by utilizing the programmable data plane (PDP) to gather network states and information at line-rate. It unleashes the power of applications to see inside the network in real time, providing them with the precise statistics needed to execute sophisticated optimizations and automated adjustments on the fly. For the real-time, in-network computing (INC) applications that depend on this data, from congestion control to dynamic load balancing, the loss of even a few INT packets can be catastrophic, leading to incorrect calculations, throughput collapse, and service-level violations.

However, the loss of INT packets is inevitable due to hardware limitations (e.g., buffer overflow) or network congestion, which results in unreliable and discontinuous monitoring data [1]. To address this, previous studies have implemented INT recovery to the control plane or dedicated servers. These approaches leverage redundant packet information to reconstruct INT packets [1], [2], or machine-learning-based approximations to estimate missing telemetry metadata based on INT context collected over extended durations [2], [3], providing network management applications with the recovered data for further analysis and investigation.

Although existing works have demonstrated the feasibility of reconstructing or estimating missing telemetry, they are primarily implemented in the control plane, which introduces several limitations, as illustrated in Fig. 1. Specifically, control-plane-based recovery methods are not directly applicable to the data plane due to fundamental architectural differences. However, many INC applications deployed in the data plane rely heavily on accurate and timely INT data to make decisions, e.g., load balancing [4] and queue scheduling for service-level assurance [5]. Failure to recover lost INT data within the data plane can significantly degrade the accuracy and performance of these applications. Moreover, relying on the control plane for INT recovery introduces additional communication latency (e.g., 30–250 μs intra-pod round trip time [6]), which is unacceptable for many latency-sensitive INC tasks (e.g., RHA, DINT-Based DWRR [7]). In particular, certain types of INT data, such as aggregated telemetry of path-level queue occupancy and congestion status [8], are difficult or even impossible to reconstruct accurately at the controller once lost, requiring per-hop recovery on data plane. These cases

Corresponding author: Lin Cui. Email: tcuilin@jnu.edu.cn
Bingzhen Chen and WaiMing Lau have contributed equally to this paper.

TABLE I: Comparison of existing INT recovery solutions.

Works	Line-Rate	100% Accuracy	Approaches	Data Plane Recovery	Testbed
LossSight [2]	✗	✗	Machine Learning	✗	Control Plane+Tofino
Zoom2Net [3]	✗	✗	Machine Learning	✗	Control Plane
CodedINT [1]	✗	✓	FEC	✗	Open vSwitch
GPINT [11]	✗	✓	Packet Retransmission	✗	BMv2
SPRINT (Our Work)	✓	✓	FEC	✓	Tofino+BMv2

call for hop-by-hop, high-speed recovery mechanisms that operate entirely within the data plane. Therefore, we raise a fundamental question: *Can INT recovery be fully implemented in the data plane while ensuring recovery accuracy, line-rate performance, and reliability?*

Implementing INT recovery on programmable data-plane hardware switches is essential yet inherently challenging: *(i)* Protocol-Independent Switch Architecture (PISA) switches lack native multiplication/division, so the polynomial steps required by RS [9] or LDPC [10] must be emulated with large lookup tables or long shift-XOR chains. Executing both encoding and decoding for every packet would quickly exhaust the limited stage SRAM and logic resources available in each pipeline stage. *(ii)* Each pipeline stage allows only a single *read–modify–write* access to a register. This conflicts with RS/LDPC workflows, which require multiple iterative reads of intermediate symbols. Forcing these iterative operations into the fixed 12-stage pipeline of an Intel Tofino 1 switch, for example, would leave virtually no resources for the recovery process itself. *(iii)* To distinguish true loss from simple packet reordering, the recovery mechanism must wait for a short confirmation window before acting. However, buffering even a small number of INT packets during this interval consumes valuable pipeline resources, potentially starving other applications.

To address these challenges, we introduce SPRINT, a framework that performs reliable in-band telemetry recovery entirely within programmable data plane devices (PDPs). SPRINT consists of two components: shift powered FEC, and the recovery-group module. By using its novel FEC in the data plane, SPRINT protects applications from telemetry loss. Its low-latency pipeline enables multiple in-network computing programs to coexist on a switch without compromising line-rate throughput. Because each switch executes recovery autonomously, SPRINT can be deployed incrementally, traverse congested or lossy links, and deliver timely INT restoration.

In short, the contributions of this paper are as follows:

1) We propose SPRINT. To the best of our knowledge, this is the first fully switch-native, autonomous FEC-based INT recovery system. It can coexist with multiple INT-driven applications on a single switch while sustaining microsecond-scale processing delay.
2) We develop a novel, recovery-group-based protection strategy, supported by formal analysis, that minimizes redundancy while providing robust, per-packet fault tolerance.
3) We implement and evaluate a complete SPRINT prototype on both P4 hardware (Intel Tofino ASIC) and software (BMv2) switches. Our extensive experiments demonstrate line-rate recovery and show significant performance gains for representative transport-layer, control-plane, and in-network computing applications.

II. Background And Related Works

A. Background

Protocol-Independent Switch Architecture (PISA) is a typical programmable data plane architecture that is designed for line-rate packet processing. It consists of a parser, pipeline, and deparser: the parser extracts headers; the pipeline processes them via match-action tables (MAT) on stage-local SRAM, TCAM, and ALUs; and the deparser rebuilds the packet. For instance, the Tofino 1 switch features a 12-stage pipeline with dedicated resources per stage. This architecture is fundamental for INC applications, as it provides the foundation for tools like INT to perform line-rate network-state extraction for instant event detection and response.

In-band Network Telemetry is an approach that enables the programmable data plane (e.g., FPGAs, SmartNICs, and packet processors) to collect and report network state information in real time [12]. Data plane devices can embed INT instructions, enabling data and control plane applications to rapidly monitor, analyze and respond promptly [13]. Leveraging these instructions, machine-learning models can predict service metrics and flag impending service-level objective (SLO) violations, protecting video quality and strict-latency services [5], [14], [15]. For traffic engineering and congestion control, the same per-hop load reports help controllers reroute flowlets [16], [17], push queues towards zero, and fine-tune RDMA transports for rapid convergence in data centers [18], [19]. In anomaly detection, multi-hop INT features feed graph-neural networks and orchestration models that boost detection accuracy and pinpoint silent failures within seconds [20]–[22]. However, per-hop state collection can cause INT telemetry traffic to surge, consume excessive bandwidth, and packet losses only exacerbate this waste; these issues make enhancing INT reliability critical.

B. Related Works

INT Overhead Reduction. Early efforts to reduce INT overhead have focused on in-switch compression and selective reporting—paving the way for schemes. Some schemes transmit only value deltas [8] or send telemetry only when

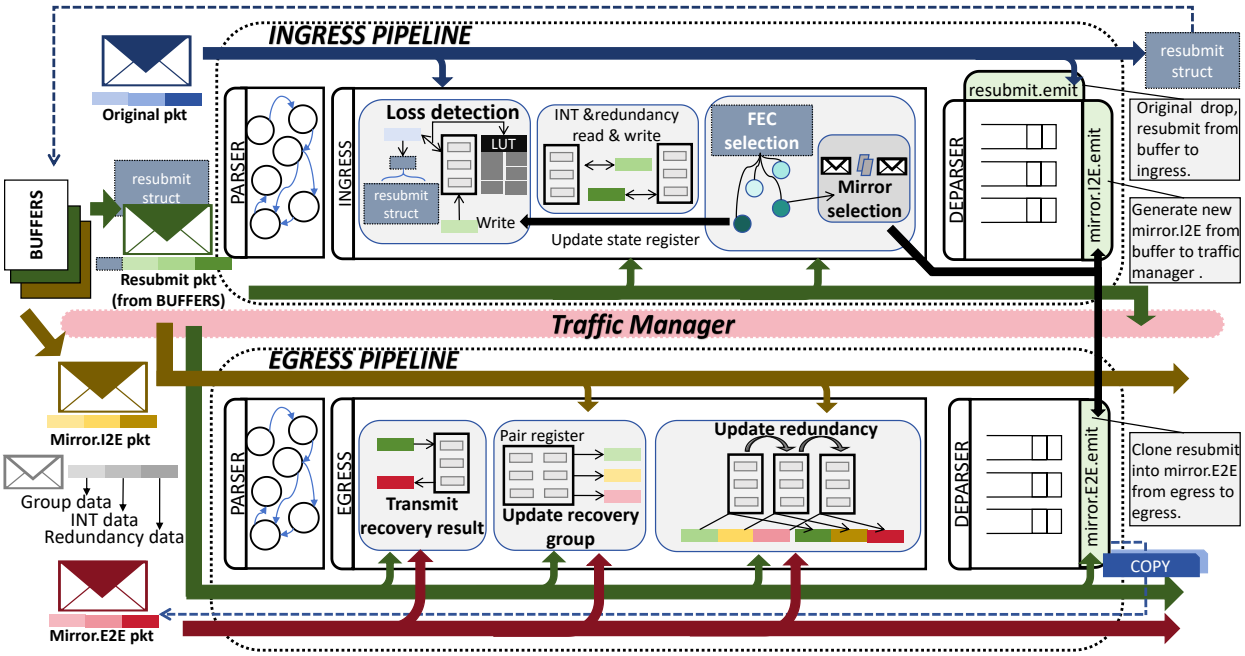


Fig. 2: Overview of the SPRINT workflow across the ingress pipeline, traffic manager, and egress pipeline. Loss detection and recovery are initiated at ingress, coordinated through the traffic manager, and finalized at egress for line-rate operation.

a change exceeds a set threshold [23]. Others drastically reduce bandwidth by sampling and compactly encoding INT, enabling reconstruction at the collector [24], or probe only a sparse subset of links and infer the missing entries with low-rank completion [25]. These mechanisms significantly reduce bandwidth consumption by reassembling the full INT data during downstream processing before it is used as input to INT-based applications. However, when packet loss occurs, not only is the telemetry information itself lost, but the bandwidth consumed for collecting and transmitting that metadata is also wasted. Therefore, INT recovery remains essential to ensure both data availability and overall efficiency.

INT Recovery. Prior work on recovering lost telemetry has so far relied on off-path computation, as summarized in Table I. Approximation-based schemes either harness efficient control plane computation and machine learning models, such as generative adversarial networks and transformers, to impute lost INT telemetry metadata, as in LossSight [2] and Zoom2Net [3]. They leverage contextual information such as trends and feature correlations to estimate telemetry values consistent with learned models, while their effectiveness may degrade or recovery may be delayed if related network metadata context is insufficient. In non-approximate, packet-level recovery scenarios, adding forward-error-correction coding [1] or simply retransmitting INT packets [11] can fully restore INT data with negligible computational overhead on the server side, albeit at the cost of extra transmission delay. Although these schemes can restore INT data, recovery is performed in the control plane, preventing line-rate operation and leaving time-sensitive INT-based INC applications unable to regain the performance lost due to telemetry drops.

Overall, INT has made significant contributions to network state monitoring and decision-making for various applications, despite its inherent overhead. Although lost INT data can be recovered at the control plane, the resulting transmission delays can degrade the performance of time-sensitive INT-based applications, especially for INC applications that are latency sensitive. Hence, implementing robust and efficient data-plane recovery mechanisms at line-rate is essential to ensure network reliability, adaptability, and performance.

III. Overview of SPRINT

In this section, we present an overview of SPRINT. It leverages a novel shift powered FEC recovery approach and a nested grouping strategy called the recovery-group to enhance redundancy recovery performance and determine whether INT packets are lost, overcoming multiple limitations and enabling successful deployment in PDPs.

A. SPRINT Workflow

As shown in Fig. 2, SPRINT integrates coordinated logic into both the ingress and egress pipelines to generate, identify, and process different packet types for loss detection and recovery. It maintains state variables such as the expected group and intra-group sequence numbers to match incoming packets and detect losses. Upon arrival, each packet is treated as an original packet and checked against the current state. However, due to the single-access-per-register constraint in each pipeline execution, SPRINT records the updated state, mirror selection, and FEC computation choice in a resubmit structure and reinjects the packet into the ingress. The resulting

resubmit packet updates the state and performs recovery computation and mirroring based on the detected loss. During recovery, SPRINT generates a mirror.I2E packet that bypasses the ingress and carries the original INT data to the egress. At the egress, the recovery-group and redundancy information are updated, while the resubmit packet carries the computed recovery result in its metadata. If two losses occur within the recoverable range, SPRINT further generates a mirror.E2E packet at the egress deparser. As mirror.E2E packets do not traverse the ingress pipeline, recovery is completed by temporarily storing the second recovery result in an egress register via the resubmit packet, which is then applied when the mirror.E2E packet arrives, ensuring correct recovery order. The detailed encoding logic and the design rationale of the recovery-group are presented in detail in Section IV

B. Functionality Overview

Loss detection: SPRINT requires each packet to carry an additional recovery-group field, which serves as its unique identifier. In addition, SPRINT maintains state variables to enable group identification and packet loss detection. When each packet arrives, it is treated as an original packet, and the current packet loss status and missing sequence numbers are determined through a lookup table. Since state variables must be updated for every packet, and due to the hardware limitation that only allows a single access to each register per pipeline stage, SPRINT employs the resubmit mechanism. Specifically, the original packet is re-injected into the ingress pipeline, generating a resubmit packet that carries the updated information to ensure state consistency at line-rate.

FEC and mirror selection: During loss detection, lookup table matching also determines the computation method for data recovery and the mirroring strategy for cloning packets. On the one hand, redundancy calculation involves both shift-powered FEC and XOR, and since shift-powered FEC depends on packet order, the specific loss scenario must be identified for accurate recovery. On the other hand, due to SPRINT's careful design, the selection of mirror operations is well-aligned with the choice of recovery computation, allowing orderly mirroring and proper assignment of different packet types.

Transmit recovery result: The mirror.E2E packet generated in the egress pipeline does not traverse the ingress pipeline, while all recovery computations and results are generated in the ingress pipeline. To address this issue, SPRINT transmits the recovery result by transferring the second recovery data carried in the resubmit packet to the mirror.E2E packet.

Update recovery-group and redundancy: Standalone operation is fundamental to the SPRINT architecture. At the information level, SPRINT relies solely on packet-contained data, enabling each switch to make decisions independently based on the local packet sequence. Logically, the correctness and order of packet transmissions at each switch must be maintained, although recovery from consecutive losses may cause local disorder and redundant ambiguity. Therefore, the module design ensures packet-level independence. The recovery-group uses pair registers for coordinated updates, keeping all outgoing packets ordered. Redundancy is updated in sync with the recovery-group, but recalculating redundancy requires storing significant metadata, which is impractical for hardware switches. Accordingly, SPRINT employs three sets of register arrays to manage four ordered INT packets, as illustrated in Fig. 3. These modules effectively eliminate PDP constraints and preserve the independence of the SPRINT architecture.

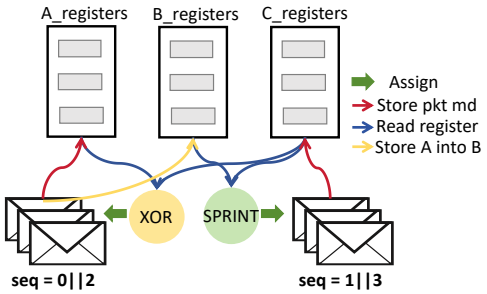


Fig. 3: E2E Register design for mirror.E2E packet in egress.

IV. Detailed Design of SPRINT

This section presents details of the functional architecture of SPRINT, and describes the design improvements and key innovations introduced to address the challenges.

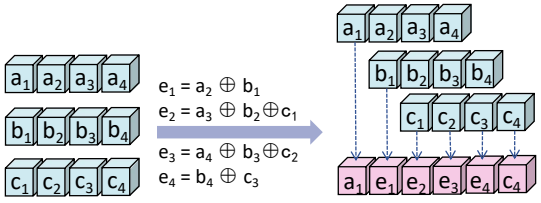
A. Packet Structure

SPRINT requires each packet to carry an additional grouping field, serving as its unique identifier. Each packet is logically divided into two groups: *pre* and *post*, with each group having its own internal sequence number. Therefore, each group is assigned independent values for *pre_gid, pre_seq, post_gid*, and *post_seq*. The configuration of the recovery-group serves as a central theme throughout the entire introduction of SPRINT.

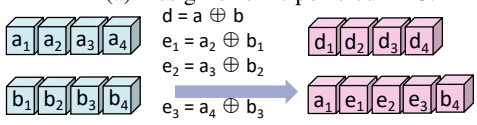
SPRINT requires that each packet carry redundancy. Because our proposed shift powered FEC requires additional bits to prevent information loss, the redundancy length varies across packets. The type of redundancy carried by each packet is determined by its *pre_gid*, packets with odd values carry shift-powered FEC redundancy. In contrast, packets with even values carry redundancy.

B. Shift Powered FEC

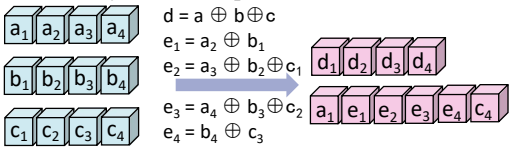
XOR operations are inherently simple, and their operand symmetry enables fast mapping. However, symmetry leads to irreversible information loss. Improving fault tolerance requires more redundancy, which increases network overhead. Define α as fault tolerance and $R(\alpha)$ as the redundancy-to-metadata ratio, with $R(\alpha) = \frac{\text{redundancy}}{\text{metadata}}$. Under conventional XOR-based recovery, protecting two metadata units against arbitrary losses requires three redundant elements to uniquely resolve the missing values. Thus, for two metadata, $R(100\%) = 150\%$; for three, $R(66\%) = 100\%$.



(a) Design of shift powered FEC.



(b) Analysis of shift powered FEC in R (100%).



(c) Analysis of shift powered FEC in R (66%).

Fig. 4: Design of shift powered FEC.

Therefore, we propose *shift powered FEC*, a shift-based misaligned XOR method that preserves data order and compensates for information loss. This approach requires only one additional byte for data discrimination and represents metadata order through chained encoding. The extra byte carries order information, effectively remedying the irreversible loss caused by XOR compressing two bytes into one. Given 4-byte metadata $A = (a_1, a_2, a_3, a_4)$ and $B = (b_1, b_2, b_3, b_4)$, their shift powered result R is defined as $R = (A0 \oplus 0B) = (a_1 \oplus 0, a_2 \oplus b_1, a_3 \oplus b_2, a_4 \oplus b_3, 0 \oplus b_4)$.

The shift powered FEC does not limit the number of metadata units involved in calculations, which is similar to XOR. Fig. 4a illustrates the specific process and results when the count of metadata is three. By simultaneously employing Shift powered FEC and XOR redundancy, the total amount of redundancy is reduced, leading to a significant improvement in link throughput. Let x denote the number of metadata units and y denote the byte length of each unit. When two redundancies are used, the redundancy ratio is calculated as $R(\frac{2}{x}) = \frac{2y-1+x}{x \times y}$. Figs 4b and 4c provide detailed descriptions, indicating a noticeable reduction in redundancy ratios when α equals 100% and 66%, respectively. Additional results are presented in Table II. The results demonstrate that shift powered FEC can significantly reduce redundancy while maintaining the same error recovery capability.

C. Design of Recovery-group

As shown in Fig. 5, the nested recovery-group design in SPRINT includes two group types: *pre* and *post*. Each group contains four packets, indexed from 0 to 3 to represent their order within the group, and this number can be adjusted according to loss tolerance requirements, with bounded packet losses being deterministically recoverable within each recovery-group. *Pre* and *post* groups share a single, incrementing group ID sequence, avoiding collisions across categories. Except for the first two packets, each packet includes both pre and post group numbers along with its group index. Consequently, each packet carries both INT and redundant data, thereby enabling the implementation of a nested grouping mechanism within the programmable data plane.

TABLE II: Efficiency comparison between shift powered FEC and Normal XOR.

Bytes	x	Shift powered FEC	XOR
4 bytes	2	R(100%)=112.5%	R(100%)=150%
	3	R(66%)=83%	R(66%)=100%
	4	R(50%)=68%	R(50%)=75%
5 bytes	2	R(100%)=110%	R(100%)=150%
	3	R(66%)=80%	R(66%)=100%
	4	R(50%)=65%	R(50%)=75%

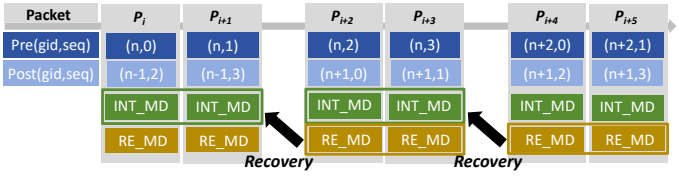


Fig. 5: Recovery-group design of SPRINT: blue shows grouping, green INT metadata, and yellow redundancy metadata.

This grouping scheme ensures that lost packets can be recovered within a relatively short period. To recover a packet with sequence number 0, at most, it is only necessary to wait until the arrival of the packet with sequence number 3, thereby meeting the stringent latency requirements of INC. Since each packet carries independent recovery information, SPRINT enables dual protection. Even if one redundant packet is lost, a high probability of successful recovery can still be maintained. By eliminating the inherent header duplication found in traditional independent metadata redundancy schemes, the nested grouping approach not only reduces bandwidth usage but also enhances overall network topology performance. In summary, the nested recovery-grouping scheme offers higher efficiency, provides finer-grained recovery capability, and maintains robust performance even under high packet loss rates by interleaving recovery across adjacent groups.

V. Discussion and Analysis

A. Effectiveness of Shift Powered FEC

As discussed in §IV-B, XOR recovery for any two INTs typically requires three redundancies [26], while shift powered FEC needs only two. On the one hand, two redundancies prevent the reuse of redundant computation types. On the other hand, the overall recovery rate can be flexibly adjusted based on the protected data. Furthermore, the rationality of using two redundancies is demonstrated in §V-B.

Theorem 1. *For any group consisting of n INTs and two redundancies, the shift powered FEC can accurately recover any two losses within the group.*

Proof. Denote n INTs as $P_1, P_2, \ldots, P_n$, where P_i is represented as $P_i = (p_{i,1}, p_{i,2}, \ldots, P_{i,j}, \ldots, p_{i,m})$, with $i \in 1, 2, \ldots, n$ and $j \in 1, 2, \ldots, m$. Additionally, two redundancies, R_1 and R_2, are constructed. Specifically, R_1 is generated by performing a XOR on all INTs, while R_2 is produced using shift powered FEC applied to all INTs.

Suppose two INTs, say P_x and P_y, where $1 \leq x < y \leq n$, are lost randomly. The receiver subsequently receives the remaining data, i.e., $P_1, \ldots, P_{x-1}, P_{x+1}, \ldots, P_{y-1}, P_{y+1}, \ldots, P_n$, together with the R_1 and R_2, and identifies the indices of the lost INTs. The redundant information R_1 and R_2 are then each XOR with all the received INTs, resulting in $R1'$ and $R2'$:

$$R1' = R_1 \oplus P_1 \oplus \cdots \oplus P_{x-1} \oplus P_{x+1} \oplus \cdots \oplus P_{y-1} \oplus P_{y+1} \oplus \cdots \oplus P_n = P_x \oplus P_y. \quad (1)$$

$$R2' = R_2 \oplus P_1 \underbrace{00\cdots0}_{\text{n-1}} \oplus \cdots \oplus \underbrace{00\cdots0}_{\text{x-2}} P_{x-1} \underbrace{00\cdots0}_{\text{n-x+1}} \oplus \underbrace{00\cdots0}_{\text{x}} P_{x+1} \underbrace{00\cdots0}_{\text{n-x-1}} \oplus \cdots \oplus \underbrace{00\cdots0}_{\text{y-2}} P_{y-1} \underbrace{00\cdots0}_{\text{n-y+1}} \oplus \underbrace{00\cdots0}_{\text{y}} P_{y+1} \underbrace{00\cdots0}_{\text{n-y-1}} \oplus \cdots \oplus P_n \underbrace{00\cdots0}_{\text{n-1}}$$
$$= \underbrace{00\cdots0}_{\text{x-1}} P_x \underbrace{00\cdots0}_{\text{n-x}} \oplus \underbrace{00\cdots0}_{\text{y-1}} P_y \underbrace{00\cdots0}_{\text{n-y}}. \quad (2)$$

By detecting the loss order, the values of x and y can be directly determined. When $y > x + m$, the values of P_x and P_y can be derived directly from their positions in $R_2{}'$.

When $y < x + m$, without loss of generality, we may assume $y = x + 1$, since this boundary case is representative of all such scenarios. Therefore, we obtain: $R1' = (p_{x,1} \oplus p_{y,1}, p_{x,2} \oplus p_{y,2}, \ldots, p_{x,m} \oplus p_{y,m})$, and $R2' = (p_{x,1}, p_{x,2} \oplus p_{y,1}, \ldots, p_{x,m} \oplus p_{y,m-1}, p_{y,m})$.

After representing the system as a matrix of unknowns, the matrix attains a rank of $2 \times m$, indicating that it is full-rank. Under full-rank conditions, each unknown has a unique solution. Therefore, with the redundancy offered by shift powered FEC, all losses can be uniquely determined. □

B. Analysis of Recovery-group

The primary objective of nested grouping is to maximize the utilization of each packet, thereby enhancing transmission efficiency and reducing protocol overhead.

Theorem 2. *The recovery-group design of SPRINT can achieve Pareto optimal trade-off between redundancy overhead and recovery capability.*

Proof. Given various strategies of grouping value, the quantity of redundant data y, the number of INT x, and the INT length z collectively influence both loss tolerance and redundancy rate. Consider the multi-objective optimization problem for maximizing the packet loss tolerance F_1 and minimizing the redundancy ratio F_2, where:

- $F_1 = \frac{y}{x}$: the packet loss tolerance, where $x \geq 2$ and $2 \leq y \leq x$.
- $F_2 = \frac{z+(z+x-1)\times(y-1)}{x\times z}$: the redundancy ratio, where $z > 0$; as INT metadata, z is commonly 2, 4, or 8.

The objective functions F_1 and F_2 are inherently conflicting. Increasing F_1 typically requires introducing more redundancy, leading to a higher F_2, and vice versa. Therefore, the problem naturally fits the paradigm of multi-objective optimization, where F_1 is to be maximized and F_2 minimized.

Due to the conflict between F_1 and F_2, improving one objective usually leads to a deterioration of the other, and it is generally impossible to optimize all objectives simultaneously. Moreover, both redundancy and tolerance are essential, making it infeasible to rank all solutions using a single metric to obtain a unique optimum. Therefore, this problem fits the definition of Pareto optimality, and the most suitable solution can be selected from the Pareto front according to specific requirements. □

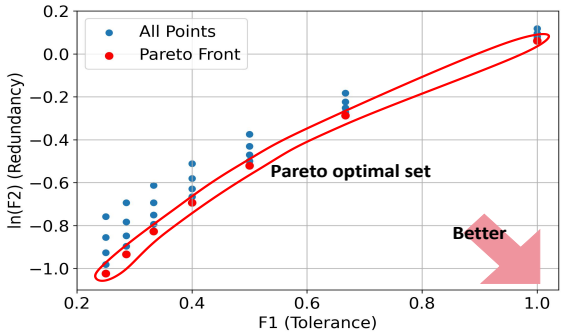


Fig. 6: F_1 and F_2 values for different x and z combinations.

SPRINT selects parameter values from the Pareto-optimal solution set. We set $F_3 = F_1 - \alpha F_2$, where α is a weight balancing loss-tolerance and redundancy ratio. The partial derivative of F_3 with respect to y is: $\frac{\partial F_3}{\partial y} = \frac{1}{x} - \alpha \cdot \left(\frac{z+x-1}{xz}\right)$. This indicates that the impact of y is relatively minor, so y can be fixed while focusing on x and z. Since INT metadata size is protocol-defined and hard to change, but loss-tolerance can be adjusted by grouping, we prioritize optimizing F_2 and set $\alpha \geq 1$. Therefore, $\frac{\partial F_3}{\partial y} < 0$ is optimal, resulting in $y = 2$. Fig. 6 enumerates all feasible x and z combinations, and shows the Pareto front in the optimal set. Results indicate efficiency improves with larger INT fields, while group size can be set based on required tolerance.

VI. Implementation

SPRINT is implemented on a programmable switch prototype platform and targeted to run on the Intel Tofino ASIC switch. SPRINT is developed using the $P4_{16}$ language, fully leveraging the high-throughput and low-latency characteristics of the Tofino ASIC. To clearly illustrate the overall data plane processing flow of SPRINT, the core pipeline logic is summarized in the form of pseudo-code, as shown in Algorithm 1, which encapsulates all the design features described above.

Given that pipelines cannot persistently buffer packets or support complex if-else logic, we replaced the approximately

Algorithm 1: SPRINT Pipeline Algorithm

Input: Packet Set P; pre-configured lookup tables $LUTs$;

for ***each*** $P_i \in P$ *in Ingress* **do**
  if $P_i.Resub\_Flag == 0$ **then**
    $State\_Vals \leftarrow state\_Vals\_Register$;
    $Resub\_Vals \leftarrow LUTs$;
    $Resubmit(Resub\_Vals)$;
  if $P_i.Resub\_Flag == 1$ **then**
    $State\_Vals\_Register \leftarrow Resub\_Vals$;
    $P_i.INT\_md \leftarrow Computing \leftarrow Resub\_Vals$;
    $Mirror.I2E(P_i) \leftarrow Resub\_Vals$;
for ***each*** $P_i \in P$ *in Egress* **do**
  $P_i.Grp\&Red\_md \leftrightarrow Grp\&Red\_Registers$;
  $Mirror.E2E(P_i) \leftarrow P_i.Bridge\_md$;
`Send(`P_i`);`

800 lines of branching code in the original P4 implementation with multiple parallel lookup tables, achieving efficient and scalable packet loss detection and localization. Furthermore, to address the fixed-length limitation of hardware registers, SPRINT employs slicing to effectively handle redundant information of varying lengths, thereby minimizing both computational and storage overhead. During the packet recovery process, SPRINT requires mirroring operations; however, Tofino allows only one mirror action per pipeline, and the resubmit and mirror primitives cannot be executed simultaneously. To flexibly adapt to these hardware constraints, we embed the recovery and resubmit at different processing cycles.

VII. Evalution

We have implemented and evaluated SPRINT on both hardware P4 switches (equipped with Intel Tofino ASIC) and software switches of BMv2.

A. Experiment Setup

The testbed P4 hardware switch (Flnet S9180-32X with Tofino ASIC) is connected to servers equipped with two Intel Xeon Platinum 8259CL CPUs (2.50 GHz, 24 cores), 256GB RAM, and ConnectX-3 Pro 40Gbps NICs, via 40Gbps fiber links. SPRINT was comprehensively deployed on hardware switches. We controlled packet loss by randomly dropping packets at the source host and measured arrivals at the destination host to evaluate SPRINT's recovery efficiency and resource usage.

In addition to hardware experiments, we further validated SPRINT on a P4 software switch (BMv2) in Mininet under different network topologies. BMv2 experiments examined SPRINT's recovery across various packet loss conditions. Since no prior work has achieved data-plane in-network recovery, we also simulated CodedINT [1] on BMv2 as a baseline for comparison with XOR recovery.

B. Performance in Recovery Rate

We evaluate recovery effectiveness from two key dimensions. The first is the recovery rate, defined as the ratio of recovered to lost, directly indicating the recovery capability of SPRINT. The second is the receiver packet success arrival rate, calculated as the fraction of packets successfully received relative to those sent. This metric reflects the overall impact of the recovery mechanism on end-to-end transmission under varying packet loss conditions.

We configured packet loss rates between 1% and 50%, measuring the proportion of recoverable INT metadata across these conditions; notably, higher loss rates further highlight SPRINT's recovery capability. As shown in Fig. 7a, grouping four packets yields a recovery success rate near 99% at a 10% loss rate, and around 94% arrival rate even with 30% loss, demonstrating robust recovery performance. Grouping six packets slightly degrades performance, yet retains high recovery under low loss.

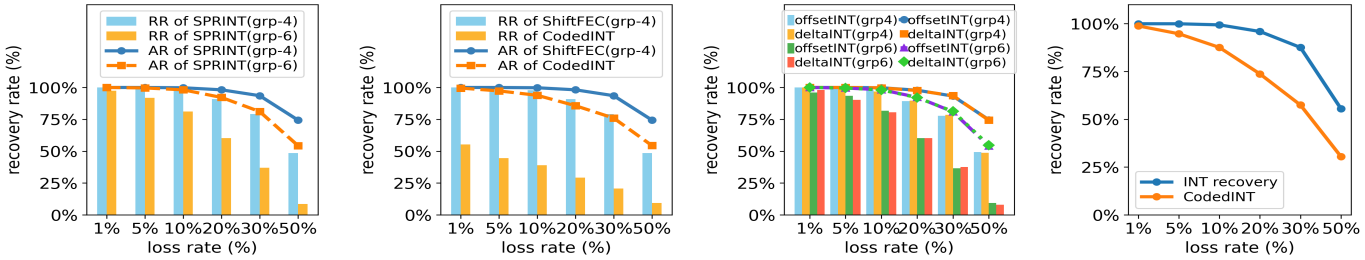
Fig. 7b compares SPRINT and CodedINT, with the latter employing only XOR. The results demonstrate that SPRINT outperforms CodedINT in packet recovery effectiveness, with improvements ranging from twofold to fourfold. Moreover, CodedINT is not implemented on PDP and thus cannot support the INC program effectively.

SPRINT is orthogonal to INT aggregation schemes such as OffsetINT and DeltaINT. It operates as a standalone, data-plane recovery mechanism that protects INT packets after aggregation, independent of any specific INT-based task.Fig. 7c indicates that SPRINT consistently maintains a high recovery capacity in different INT sizes and types, compatible with different PDP-based measurement works. Fig.7d shows that in a multi-node link with uniform packet loss, SPRINT's independent recovery and regrouping at each switch ensure efficient transmission even with multi-node losses, confirming SPRINT's standalone capability. In contrast, simpler schemes like CodedINT lack these mechanisms, resulting in lower efficiency in such scenarios.

C. Delay and Throughput

To assess SPRINT's throughput, we deployed forwarding applications for INT packets of different sizes and measured the resulting rates. As shown in Fig. 8a, SPRINT consistently achieves average throughput over 39.9 Gbps for all INT sizes, suggesting its potential extends beyond these results.

Pipeline processing time is the total time from when a packet enters the switch ingress parser to its final forwarding. We measured this for normal traffic, with cumulative distribution functions (CDFs) at various loss rates shown in Fig. 8b. The results indicate that SPRINT introduces minimal processing time under low-loss conditions, as few packets are recovered. Even at high loss rates, if consecutive losses exceed recovery-group capacity and recovery cannot be triggered, packet processing time remains low. Overall, the pipeline shows an average processing delay of about 20 µs, meeting INC's low-latency requirements and SPRINT's design goals.



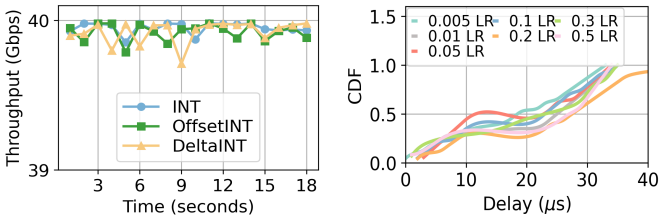
(a) Comparison of Recovery under Different Groups in SPRINT.

(b) Comparison between SPRINT and CodedINT.

(c) The collaborative work of offsetINT, DeltaINT, with SPRINT.

(d) Comparison of SPRINT and CodedINT in multi node links.

Fig. 7: The recovery rate (RR) and arrival rate (AR) performance of SPRINT in different loss environments.



(a) Throughput of different methods.

(b) Delay of different loss rate (LR).

Fig. 8: Performance of throughput and latency of SPRINT.

TABLE III: Hardware Resource Consumption on Tofino ASIC.

Computational	eMatch xBar	tMatch xBar	Gateway	VLIW	Hash bits
Usage	9.11%	0.00%	20.31%	11.46%	7.77%

Memory	SRAM	TCAM	Map RAM	TableID	Stash
Usage	9.06%	0.00%	10.07%	51.56%	3.65%

D. Switch Resource Usage

Table III shows SPRINT's hardware resource utilization on a Tofino switch. Regarding memory, SPRINT's MAT uses only exact-match operations, consumes no TCAM, and occupies less than 10% of SRAM. Around 50% of the table ID space in the Tofino ASIC remains available for future expansion. The low utilization of VLIW slots, hash bits, and stash memory minimizes contention risk, allowing SPRINT to coexist efficiently with most data-plane tasks.

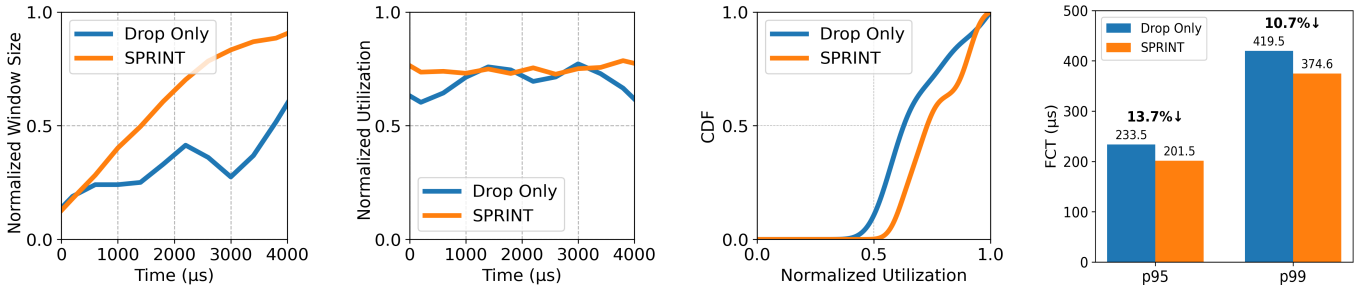
E. Application Performance

We conducted three experiments to evaluate how SPRINT enables INT recovery and improves INT-based applications. To this end, we simulate real-link scenarios across three representative use cases: (1) HPCC (High Precision Congestion Control) [18], (2) a control plane routing strategy based on INT latency prediction using Convolutional Neural Networks (CNN), and (3) RHA (Remaining-Hop-Aware) [5] packet scheduling on data plane.

1) HPCC: HPCC leverages INT to collect per-hop telemetry, including egress timestamp, cumulative transmitted bytes, and instantaneous queue length. The receiver echoes these fields in each ACK, which enables the sender to accurately estimate link utilization and adjust its congestion window precisely, maintaining near-empty queues while approaching full link utilization. In the experiment, SPRINT protection is applied to both INT fields and ACK sequence number. Fig. 9 reports the results under transient congestion and INT packet loss. To emulate bursty traffic, 100–300KB flows are injected at a rate of approximately 20000 flows/s following a Poisson process. The performance of HPCC is then evaluated by comparing its behavior with and without INT-based recovery (Drop Only). The setup sustains approximately 85% link utilization on 40Gbps links, with a round-trip time (RTT) of about 50 μs. This performance is achieved in a multi-hop leaf-spine topology where INT recovery is enabled on a single switch.

Fig. 9a and 9b zoom in on switch queue build-up and an INT loss around 3000 μs, showing that HPCC collapses its window near 2000 μs after misreading a transient utilization spike, which triggers roughly 10% utilization jitter and a pronounced window drop. Around 4000 μs, as the INT loss fades and congestion is relieved through reduced RTT and shorter queues, HPCC with Drop Only re-evaluates utilization and increases the sending window, improving throughput. In contrast, SPRINT restores the INT state promptly, allowing HPCC to maintain a steady window and consistently high bandwidth. The full traces presented in Fig.9c and 9d show that, under congestion, the CDF of normalized utilization for SPRINT consistently shifts to the right of that of the Drop-Only baseline, indicating improved link utilization across the distribution. Specifically, 90% of samples achieve a normalized utilization of approximately 0.92 with SPRINT, compared to 0.86 under Drop-Only. In terms of flow completion time (FCT), SPRINT reduces the 99th percentile from 419.5 μs to 374.6 μs, and the 95th percentile from 233.5 μs to 201.5 μs, yielding tail latency improvements of 10.7% and 13.7%, respectively. These results demonstrate SPRINT's effectiveness in enhancing both link efficiency and latency under load.

2) Routing strategy based on INT delay prediction: Evaluation of a control-plane routing strategy that uses a CNN to predict queuing latency from roughly hundreds of thousands of INT records collected in a k=4 fat-tree topology emulated in Mininet with BMv2 switches, where transient congestion



(a) Normalized flow window size comparison under INT loss.

(b) Normalized link utilization over time under INT loss.

(c) CDF of Normalized Utilization for Complete Traces.

(d) p99 and p95 Flow Completion Time (FCT) for Complete Traces.

Fig. 9: Performance comparison of HPCC under INT packet loss and completed traffic.

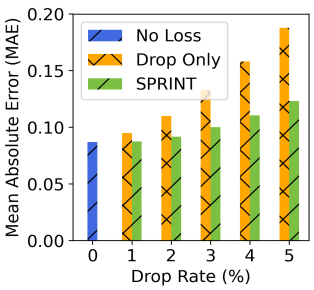
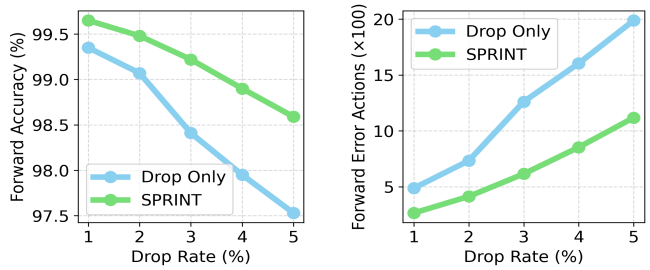


Fig. 10: Evaluation of a CNN-based routing strategy for queuing latency prediction under varying packet drop rates.



(a) Forward accuracy under varying INT packet loss rates.

(b) Forward Error Actions under varying INT packet loss rates.

Fig. 11: Evaluation of RHA scheduling under varying INT packet-loss rates relative to a zero INT loss baseline.

events were introduced to induce queue variations, shows that missing INT packets degrade inference accuracy. A simple CNN with two convolutional layers and a fully connected regression head is used for prediction, trained on 70% of the data and tested on the remaining 30%.

Fig. 10 shows that SPRINT restores lost telemetry and, with INT loss between 1% and 5%, lowers the CNN mean absolute error (MAE) by 8% to 34% relative to Drop Only, and at 1% loss attains an MAE of 0.087, essentially matching the loss free baseline of 0.086. Consequently, routing decisions result in nearly no INT loss quality, even with 2% loss, providing the necessary fault tolerance for INT-driven INC.

3) RHA: The RHA scheduler selects forwarding paths using each INT packet's remaining hop count and per-hop queuing time to meet SLOs. INT loss produces inaccurate queue estimates, leading to SLO violations and weaker transport robustness. This experiment evaluates RHA under INT loss by comparing SPRINT and a Drop Only baseline against an ideal zero-loss baseline, using two metrics: (i) Forwarding Accuracy: the fraction of packets whose chosen queues match the zero-loss schedule. (ii) Forwarding Error Actions: the number of forwarding decisions that deviate from that schedule. Experiments used 40 Gbps links and observed a round-trip time of roughly 50 μs, matching the topology in Fig. 1.

Fig. 11 presents the forwarding performance of RHA as the INT packet loss rate increases, using the zero loss configuration as the baseline. At 1% INT loss, RHA with SPRINT attains 99.7% accuracy with only a handful of forwarding error actions. When loss grows from 1% to 5%, the Drop Only variant sees accuracy decline from 99.4% to 97.5% and its forwarding error actions climb from roughly 500 to more than 2,000. With SPRINT recovery enabled, accuracy stays above 98.5% and error actions stay below 1,000. These results confirm that SPRINT restores missing INT reports, sustains in-network congestion control, and significantly improves both SLO compliance and forwarding efficiency.

VIII. Conclusion

In this paper, we propose SPRINT, an innovative solution deployed on a programmable hardware data plane to achieve line-rate processing. By introducing a shift powered FEC and a nested recovery-group structure, SPRINT addresses key challenges in fault tolerance and packet utilization, significantly enhancing throughput and recovery accuracy. Our experiments show that SPRINT delivers microsecond-level delay at line-rate, improving network performance. Looking ahead, SPRINT has the potential to enable ultra-reliable network applications, setting the stage for high performance networks.

Acknowledgement

This work is partially supported by National Natural Science Foundation of China (NSFC) under Grant 62172189; the Innovate UK grants 10098627, 10186103 and 600648.

REFERENCES

[1] Z. Ma, S. Tang, W. Tao, Y. Xue, and Z. Zhu, "Codedint: Leveraging network coding to improve the visibility of in-band network telemetry (int)," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 3654–3659.

[2] L. Tan, W. Su, W. Zhang, H. Shi, J. Miao, and P. Manzanares-Lopez, "A packet loss monitoring system for in-band network telemetry: detection, localization, diagnosis and recovery," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4151–4168, 2021.

[3] F. Gong, D. Raghunathan, A. Gupta, and M. Apostolaki, "Zoom2net: Constrained network telemetry imputation," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 764–777.

[4] C. Zheng, B. Rienecker, and N. Zilberman, "Qcmp: Load balancing via in-network reinforcement learning," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing*, 2023, pp. 35–40.

[5] Z. Xu, X. Chen, and Z. Zhu, "Int-assisted adaptive packet scheduling in pdp switches for end-to-end latency control," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[6] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 139–152.

[7] Y. Liu, F. Zhou, L. Feng, W. Li, and J. Gao, "Dint-based dwrr: Decentralised int-based packet scheduling method for multipath communication," *IEEE Transactions on Network and Service Management*, 2025.

[8] M. Qian, L. Cui, F. P. Tso, Y. Deng, and W. Jia, "Offsetint: Achieving high accuracy and low bandwidth for in-band network telemetry," *IEEE Transactions on Services Computing*, vol. 17, no. 3, pp. 1072–1083, 2023.

[9] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[10] W. E. Ryan *et al.*, "An introduction to ldpc codes," *CRC Handbook for Coding and Signal Processing for Recording Systems*, vol. 5, no. 2, pp. 1–23, 2004.

[11] G. Simsek, D. Ergenç, and E. Onur, "Reliable and distributed network monitoring via in-band network telemetry," *arXiv preprint arXiv:2212.14876*, 2022.

[12] "In-band Network Telemetry (INT) Dataplane Specification," https://p4.org/p4-spec/docs/INT_v2_1.pdf, accessed on: July 22, 2023.

[13] 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, p. 107763, 2021.

[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*, 2020, pp. 421–434.

[15] L. C. de Almeida, R. Pasquini, and F. L. Verdi, "Using machine learning and in-band network telemetry for service metrics estimation," in *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*. IEEE, 2021, pp. 33–39.

[16] N. Katta, A. Ghag, M. Hira, I. Keslassy, A. Bergman, C. Kim, and J. Rexford, "Clove: Congestion-aware load balancing at the virtual edge," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, 2017, pp. 323–335.

[17] J. Hyun and J. W.-K. Hong, "Knowledge-defined networking using in-band network telemetry," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2017, pp. 54–57.

[18] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "Hpcc: High precision congestion control," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 44–58.

[19] Y. Zhao, S. Wang, S. Han, D. Zhou, G. Peng, and T. Huang, "Racc: Rapid and accurate int-based congestion control in rdma network," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 3811–3816.

[20] Y. Duan, C. Li, G. Bai, G. Chen, F. Zhou, J. Chen, Z. Gao, and C. Zhang, "Mfgad-int: in-band network telemetry data-driven anomaly detection using multi-feature fusion graph deep learning," *Journal of Cloud Computing*, vol. 12, no. 1, p. 126, 2023.

[21] R. Hohemberger, A. G. Castro, F. G. Vogt, R. B. Mansilha, A. F. Lorenzon, F. D. Rossi, and M. C. Luizelli, "Orchestrating in-band data plane telemetry with machine learning," *IEEE Communications Letters*, vol. 23, no. 12, pp. 2247–2251, 2019.

[22] C. Jia, T. Pan, Z. Bian, X. Lin, E. Song, C. Xu, T. Huang, and Y. Liu, "Rapid detection and localization of gray failures in data centers via in-band network telemetry," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[23] S. Sheng, Q. Huang, and P. P. Lee, "Deltaint: Toward general in-band network telemetry with extremely low bandwidth overhead," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–11.

[24] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "Pint: Probabilistic in-band network telemetry," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 662–680.

[25] X. Zeng, K. Xie, Y. Li, K. Xu, G. Xie, J. Wen, Y. Cong, and W. Liang, "Int-mc: Low-overhead in-band network-wide telemetry based on matrix completion," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 8, no. 3, pp. 1–30, 2024.

[26] R. E. Ziemer and W. H. Tranter, *Principles of communications*. John Wiley & Sons, 2014.