VeriStat: Verifiable QoS Compliance with Statistical Guarantees in Programmable Data Planes

Pankaj Aswal
Dept. of Computer Science & Engg
Visvesvaraya National Institute of Technology
Nagpur, India
pankajaswal789@gmail.com

Prashanth PVN
Dept. of Computer Science & Engg
Visvesvaraya National Institute of Technology
Nagpur, India
prashanthpvn@gmail.com

***Abstract*—The growing reliance on mission-critical network services demands a shift from traditional, empirical monitoring toward verifiable Quality of Service (QoS) compliance. Existing telemetry and measurement solutions provide useful statistical summaries of network behavior but lack formal, auditable accuracy guarantees essential for enforcing stringent Service Level Agreements (SLAs). To address this gap, we propose VeriStat, Verifiable QoS compliance with Statistical Guarantees in programmable data planes. VeriStat employs a source-controlled sampling mechanism implemented on a P4-programmable edge device, enabling scalable and low-overhead data collection. Sampled QoS outcomes are modeled as Bernoulli random variables, and Hoeffding's inequality serves as the statistical core to compute the minimum sample size required to bound measurement uncertainty within a specified accuracy and confidence level. A prototype implementation on P4 BMv2 validates the framework under varied network conditions and sampling configurations. Results demonstrate that VeriStat can deliver continuous, sample-efficient, and mathematically verifiable QoS compliance reports with negligible performance overhead.**

***Index Terms*—Verifiable QoS, Statistics, Programmable Data Plane, P4**

I. Introduction

The increasing demand for reliable and predictable network performance, driven by applications in high-frequency trading, industrial IoT, and telemedicine, has elevated Quality of Service (QoS) compliance from a desirable feature to a stringent, auditable requirement. Guaranteeing that network metrics like latency and packet loss adhere to rigid Service Level Agreements (SLAs) is non-negotiable, as even marginal degradation can lead to significant operational or financial consequences. However, achieving robust verifiable QoS remains a fundamental challenge. The dynamic nature and massive scale of modern networks makes it computationally infeasible to monitor every packet, forcing operators to trade accuracy for scalability when designing practical verification systems.

Traditional QoS monitoring systems, such as perfSONAR [1], use active probes to estimate delay and loss; however, probe traffic often diverges from real user traffic. Flow-based export mechanisms such as NetFlow, sFlow, and IPFIX [3], [4] aggregate statistics over intervals but sacrifice accuracy at finer timescales, offering limited suitability for SLA verification. P4 (Programming Protocol Independent Packet Processors) based In-band Network Telemetry (INT) [5] such as DLINT, PINT [7], enhances network visibility by embedding metadata within real time packets but require coordination across devices in the network. Further, these works inherently prioritize minimizing network overhead by collecting only a statistical representation of the network state. Other P4-based frameworks such as P4DM [9], P4-perfSONAR [10] and Lean [11] extend these capabilities for specific monitoring tasks, however, they still rely on empirically driven counters. While the existing works provide valuable statistical insights into network health, they are fundamentally incapable of providing the verifiable guarantees necessary for verifiable QoS compliance.

This paper presents VeriStat: Verifiable QoS Compliance with Statistical Guarantees in Programmable Data Planes, a framework that enables mathematical verification of QoS without extensive per-flow monitoring. The approach shifts sampling responsibility from intermediate switches to ingress, where packets are deterministically selected and tagged for monitoring. Instead of maintaining per-flow state, traffic is grouped by service class such as application type allowing scalable verification with minimal data-plane overhead. Downstream P4-programmed switches simply recognize tagged packets and export compact telemetry digests, enabling lightweight but accurate QoS visibility across the network.

VeriStat employs Hoeffding's inequality and Bernoulli-based modeling as the statistical foundation for QoS compliance verification. By precisely modeling sampled outcomes as Bernoulli random variables, the system derives a provable limit on the deviation between the measured empirical estimate and the true QoS value. This integration allows operators to compute the precise minimum sample size required to achieve the desired accuracy with guaranteed confidence, fundamentally transforming raw flow data into verifiable and auditable compliance statements. The complete proposed system is implemented and validated using P4 enabled BMv2 software switches emulated using Mininet, demonstrating continuous QoS verification under both compliant and non-compliant network conditions with minimal performance overhead.

II. Related Work

A. Traditional QoS Monitoring and Measurement

Early efforts in network performance monitoring predominantly relied on active measurement systems, including perf-

SONAR [1] and Two-Way Active Measurement Protocol (TWAMP) [2], which inject synthetic probes to estimate delay and loss along end-to-end paths. While suitable for long-term trend analysis, probe-based techniques often deviate from the real experience of user traffic and introduce additional control overhead. Passive export mechanisms such as NetFlow, sFlow, and IPFIX [3] aggregate per-flow statistics from routers and switches, enabling efficient traffic characterization but offering limited temporal granularity and accuracy. More recent streaming telemetry systems [4] enhance freshness but remain device-centric and lack statistically verifiable accuracy. Collectively, these approaches provide valuable operational observability but are insufficient for auditable QoS verification.

B. Data-Plane Telemetry and Network Health Monitoring

The advent of P4-based programmable data planes revolutionized network telemetry by enabling In-band Network Telemetry (INT) [5], where switches can directly embed performance data into packets as they traverse the network. INT has become foundational tool for fine-grained visibility within the data plane. By embedding hop-by-hop metadata into live packets, INT allows operators to measure queueing delay, path variation, and device state directly within the forwarding pipeline. Existing in-band telemetry schemes such as DLINT, PLINT, and PINT have primarily targeted operator-centric network visibility. DLINT [7] provides fine-grained per-packet monitoring but incurs significant per-switch overhead. PLINT and PINT [8] reduce this cost through probabilistic or periodic sampling, yet still require each switch to independently perform sampling decisions, which complicates coordination and scalability in large networks. More recently, INT-Source [13] shifts some decision-making to the source by selecting which flows or packets should carry INT headers, thereby reducing redundant instrumentation. However, INT-Source continues to focus on maximizing network coverage rather than verifying compliance with specific performance objectives.

C. Programmable Data-Plane QoS Monitoring

The programmability of P4 switches has inspired numerous performance-monitoring systems with high-speed and low-latency capabilities. P4DM [9] employs data-plane timestamping to measure link delay; Lean [11] and NetVision [12] use compact sketches to detect high-loss or high-latency flows; and P4-perfSONAR [10] integrates programmable switches into existing measurement infrastructures for real-time QoS tracking. While these frameworks demonstrate the expressive power of P4 for in-network measurements, they remain fundamentally network-driven focused on data collection and visibility rather than formally bounded verification accuracy. Their results are typically validated through empirical experiments, lacking statistical guarantees that relate sample size, error bounds, and confidence levels.

In Summary, active probing, telemetry-based visibility, and programmable data-plane monitoring, the dominant goal has been network observability, not verifiable assurance. Existing methods either rely on synthetic probes, maintaining exhaustive per-flow state, or lack formal mechanisms that quantify estimation confidence. The challenge of achieving statistically provable QoS verification where operators can configure and validate accuracy with guaranteed error bounds—remains largely unaddressed. This work addresses that gap by introducing a source controlled, class-based sampling framework that integrates Hoeffding-based concentration bounds within a programmable data plane. By shifting the sampling logic to the ingress and applying non-asymptotic statistical analysis, the proposed approach bridges the gap between theoretical assurance and practical, scalable verification.

III. The Statistical Foundation: Hoeffding's Inequality for Verifiable QoS

Hoeffding's inequality is a powerful concentration inequality in probability theory. It provides a non-asymptotic upper bound on the probability (δ) that the average ($\hat{\mu}$) of a set of bounded, independent random variables deviates from its true expected value (μ) by more than an error margin (ϵ). The proposed approach leverages this inequality to achieve provable statistical guarantees for QoS compliance. The inequality requires that the random variables are bounded and independent. For independent random variables $X_1, \ldots, X_n$ bounded in the range $[a_i, b_i]$, the general expression is:

$$P(|\hat{\mu} - \mu| > \epsilon) \leq 2 \exp\left(- \frac{2\varepsilon^2 n^2}{\sum_{i=1}^{n} (b_i - a_i)^2} \right) \quad (1)$$

A. Modeling Sampled Packet Outcomes as Bernoulli Random Variables.

In order to apply Hoeffding's inequality for QoS verification, we model the outcome of monitoring a single sampled packet as a Bernoulli Random Variable (X_i), that has only two possible, mutually exclusive outcomes: 1 and 0 corresponding to the labels success and failure. The probability of success is given by $P(X = 1) = p$, while the probability of the failure is $1 - p$, i.e., $P(X = 0) = 1 - p$.

For example, when measuring packet loss: $X = 1$ (Success) is assigned to the event that the packet is lost. $X = 0$ (Failure) is assigned to the event that the packet is not lost. The parameter p is the true, unknown probability of packet loss. This modeling particularly for metrics like packet loss rate, satisfies the bounding requirement for Hoeffding's inequality, as the variable can only take values in the fixed interval $[0, 1]$. Further, by employing per-class Pseudorandom Number Generation (PRNG) for sampling, the individual packet outcomes $(X_1, X_2, \ldots, X_n)$ are treated as a set of independent variables.

B. The True Mean and Measured Mean

Hoeffding's inequality bounds the difference between the actual and observed metric:

1) The True Mean (μ): This represents the unknown, real QoS metric for the entire traffic class (e.g., the true

packet loss rate). It is the expected value of a single packet outcome given by,

$$\mu = E[X_i] = P(\text{Packet Lost}) \tag{2}$$

2) The Measured Mean ($\hat{\mu}$): This is the calculated QoS metric derived from the collected sample of size n. It is the sample average, given by,

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{3}$$

C. Deriving the sample size for Provable Guarantee

For Bernoulli random variables where $a_i = 0$ and $b_i = 1$, the term $\sum_{i=1}^{n}(b_i - a_i)^2$ simplifies to $\sum_{i=1}^{n}(1-0)^2 = n$. The inequality becomes:

$$P(|\hat{\mu} - \mu| > \epsilon) \leq 2 \cdot \exp(-2n\epsilon^2) \tag{4}$$

The inequality is inverted to determine the minimum required sample size (n_{req}) needed to guarantee a desired confidence level ($1 - \delta$) with a target error margin (ϵ).

1) Set the upper bound equal to the desired failure probability (δ):

$$2 \cdot \exp(-2n_{req}\epsilon^2) = \delta \tag{5}$$

2) Solve for the minimum required sample size n_{req}:

$$n_{req} = \frac{1}{2\epsilon^2}\ln\left(\frac{2}{\delta}\right) \tag{6}$$

By collecting at least n_{req} samples, the system achieves the provable statistical guarantee that the true QoS metric (μ) is within $\pm\epsilon$ of the measured metric ($\hat{\mu}$), with a probability of at least $1 - \delta$.

IV. System Design

The VeriStat framework illustrated in Fig. 1, enables mathematically verifiable (QoS) compliance monitoring by combining programmable telemetry with statistical guarantees derived from Hoeffding's inequality. The design consists of three cooperative components: the (CPE), the Provider Network Telemetry Plane, and the Central Collector. Each component fulfills a distinct function: source-controlled sampling, conditional in-network telemetry, and statistical verification grounded in the theoretical model developed in Section III.

In conventional telemetry systems, performance metrics such as delay or packet loss are inferred solely from packets successfully received by the collector. Packet drops or reordering introduce ambiguity regarding the actual number of packets sampled at the source, creating uncertainty in the effective sample size. To eliminate this ambiguity, VeriStat introduces a source-maintained counter that certifies each initiated sampling trial, enabling the collector to accurately determine when the analytically derived requirement n_{req} has been satisfied and preserving the validity of the statistical verification process.

A. Source-Controlled Sampling and Tagging

The Customer Premise Equipment (CPE), implemented on a P4-programmable device, initiates the measurement process by performing source-controlled sampling and tagging. This design confines telemetry initiation to the network edge, ensuring scalability and minimizing per-flow state in the provider network.

1) Traffic Classification and Pseudo-Random Sampling: Each incoming packet is classified into a traffic class (e.g., voice, video, best-effort), which determines its associated sampling probability and verification profile. Sampling is implemented using a pseudo-random number generator (PRNG). For every packet, the P4 pipeline computes a hash value; if it falls below the current sampling threshold T_p, the packet is selected for telemetry participation. The Sampling Threshold T_p is dynamically set based on the current experimental or operational requirements. This sampling logic can be achieved using standard P4 primitives which enable pseudo-random packet selection and dynamic threshold adjustment for probabilistic telemetry [13]. The continuous sampling process runs independently of the collector's state, feeding the data stream until the required quantity (n_{req}) is accumulated.

2) VeriStat Sample Counter and Tag Generation: To maintain verifiable sampling progress under packet loss or reordering, the CPE maintains a class-specific VeriStat Sample Counter (VSC). Each time a packet is selected, the counter increments, and its value is encoded as the VeriStat Sample ID within the packet's metadata. This identifier certifies the total number of independent Bernoulli trials initiated at the source and decouples the statistical stopping condition from the actual delivery outcome. The selected packets are marked with a lightweight VeriStat Tag and forwarded into the provider domain for telemetry processing.

B. Provider Network: Conditional Telemetry Insertion

Packets carrying the VeriStat Tag traverse the provider's programmable switches configured for conditional telemetry insertion. Each switch detects the tag and selectively performs (INT) operations, appending key QoS metadata such as switch identifier, ingress and egress timestamps, and queue occupancy. Non-tagged packets bypass the telemetry pipeline and follow standard forwarding paths. This conditional approach ensures that telemetry processing and memory overhead are proportional only to the sampled subset of packets, maintaining scalability even under high traffic volumes. The telemetry-enriched packets are exported to the Central Collector for aggregation and analysis.

C. QoS Verification and Compliance Reporting

The Central Collector acts as VeriStat's statistical verification engine. It continuously aggregates telemetry data, evaluates compliance against QoS guarantees, and produces verifiable compliance reports with defined confidence and accuracy bounds. Algorithm 1 describes the verifiable QoS process. In lossy network environments, direct reliance on

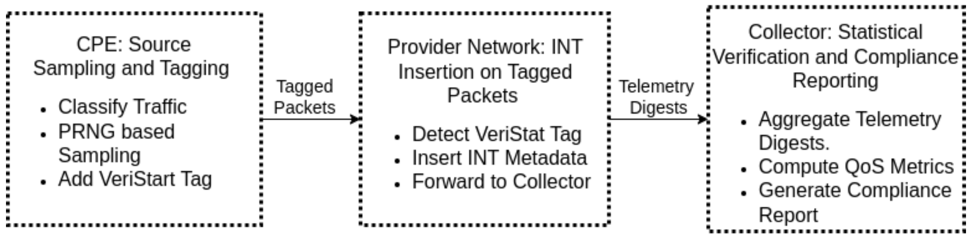
CPE: Source Sampling and Tagging
• Classify Traffic
• PRNG based Sampling
• Add VeriStart Tag
Tagged Packets
Provider Network: INT Insertion on Tagged Packets
• Detect VeriStat Tag
• Insert INT Metadata
• Forward to Collector
Telemetry Digests
Collector: Statistical Verification and Compliance Reporting
• Aggregate Telemetry Digests.
• Compute QoS Metrics
• Generate Compliance Report

Fig. 1. Overview of the VeriStat framework and its main components.

received packets can distort the true sample size, undermining the guarantees established by Hoeffding's inequality, which assumes that the number of independent trials equals the analytically required sample size (n_{req}).

VeriStat solves this issue through a cooperative counting mechanism between the CPE and the Collector. The Collector interprets the VeriStat Sample ID, embedded by the CPE in each sampled packet, as the authoritative indicator of sampling progress. This decouples the verification process from the number of packets actually received, ensuring that packet loss or reordering does not affect the correctness of the statistical inference. The Collector continuously aggregates the QoS metrics (e.g., packet loss, delay) of the incoming VeriStat-tagged packets and monitors the highest Sample ID observed. When this value meets or exceeds the analytically derived requirement n_{req}, the verification trigger is activated. At that point, the Collector proceeds to compute the empirical estimate of the QoS parameter, constructs its confidence interval, and evaluates compliance against the predefined SLA threshold. This cooperative mechanism guarantees that verification decisions are based on the true number of initiated trials, preserving the rigor of the statistical confidence bound even in the presence of network imperfections.

1) Formal Verification and Compliance Reporting: Once the verification trigger is activated, the Collector computes the empirical estimate $\hat{\mu}$ of the QoS metric, such as loss rate or delay compliance. Using Hoeffding's bound, the Collector constructs a confidence interval around $\hat{\mu}$ and compares it against the SLA requirements. If the interval lies within the predefined accuracy ε at the confidence level $(1-\delta)$, the network is deemed compliant; otherwise, a QoS violation is reported. Each compliance report is timestamped, stored, and optionally signed for auditability. After report generation, the monitoring cycle resets automatically, enabling continuous, low-overhead QoS verification with provable statistical guarantees.

Algorithm 1 Verifiable QoS with Statistical Guarantees

```
Require: QoS target $\mu$, error margin $\varepsilon$, confidence $1-\delta$
Ensure: Statistical verification of compliance within $(\varepsilon, \delta)$ bounds
 1: Compute required sample size: $n \leftarrow \frac{1}{2\varepsilon^2} \ln\left(\frac{2}{\delta}\right)$
 2: while system active do
 3:     Receive incoming packets at ingress
 4:     for each packet $p$ do
 5:         if $H(p.\text{5tuple} \,\|\, p.\text{seq}) < T_p$ then
 6:             Tag $p$ with $(sample\_id)$
 7:         end if
 8:     end for
 9:     Forward packets through network
10:     P4 switches extract telemetry for tagged packets
11:     Collector aggregates digests $\{X_i\}_{i=1}^{n}$
12:     if $n \geq n_{\text{req}}$ then
13:         Compute empirical mean $|\hat{\mu} = \frac{1}{n}\sum X_i$
14:         if $|\hat{\mu} - \mu| \leq \varepsilon$ then
15:             QoS compliance verified with confidence $1-\delta$
16:         else
17:             Generate Non Compliance report.
18:         end if
19:     end if
20: end while
```

V. Experimental Setup and Results

Experiments were conducted using the *Mininet* network emulator with *BMv2 P4 software switches*, emulating a single-operator domain. The topology (shown in Fig. 2) consists of two P4 switches (S_1, S_2) and two hosts (h_1, h_2), where h_1 acts as the ingress sampler and traffic source (CPE), while h_2 serves as the collector and QoS verifier. The python script written in the host (h_1) generates continuous stream of packets and randomly tag the packets with varying sampling rates. Sampled packets are tagged at the ingress using the IPv4 DSCP field (DSCP = 46, Expedited Forwarding), encoded in the IPv4 TOS/DSCP field. Each switch is programmed to identify tagged packets, extract telemetry fields (timestamps, port identifiers, and sequence numbers), and export digests to the collector. Programmable switches match this DSCP value and insert telemetry via a custom header. The collector module on h_2 aggregates digests, computes packet loss, and applies the Hoeffding-based statistical verification model in real time.

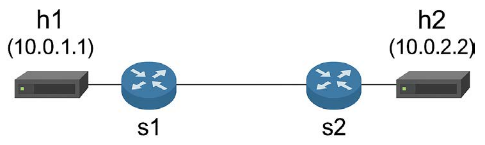

Fig. 2. PoC Implementation: Mininet topology for QoS verification

The objective of the experiments is to evaluate VeriStat's

TABLE I
EXPERIMENTAL RESULTS — COMPLIANCE SCENARIO (HOEFFDING SAMPLING: $n = 18{,}445$, SAMPLE RATE = 10%).

Round	Sampled Received	Sampled Lost	Sampled Loss (%)	95% Confidence Interval	Compliance Status
1	18,269	176	0.954%	[−0.046%, 1.954%]	Compliant
2	18,284	161	0.873%	[−0.127%, 1.873%]	Compliant
3	18,262	183	0.992%	[−0.008%, 1.992%]	Compliant
4	18,257	188	1.019%	[0.019%, 2.019%]	Compliant
5	18,256	189	1.025%	[0.025%, 2.025%]	Compliant
6	18,254	191	1.035%	[0.035%, 2.035%]	Compliant
7	18,262	183	0.992%	[−0.008%, 1.992%]	Compliant
8	18,247	198	1.073%	[0.073%, 2.073%]	Compliant
9	18,267	178	0.965%	[−0.035%, 1.965%]	Compliant
10	18,267	178	0.965%	[−0.035%, 1.965%]	Compliant

TABLE II
EXPERIMENTAL RESULTS — NON-COMPLIANCE SCENARIO (UNDER-SAMPLED: $n = 10{,}000$, SAMPLE RATE = 10%).

Round	Sampled Received	Sampled Lost	Sampled Loss (%)	95% Confidence Interval	Compliance Status
1	9,895	105	1.050%	[0.050%, 2.050%]	Compliant
2	9,891	109	1.090%	[0.090%, 2.090%]	Compliant
3	9,920	80	0.800%	[−0.200%, 1.800%]	Compliant
4	9,759	241	2.410%	[1.410%, 3.410%]	**Non-compliant**
5	9,907	93	0.930%	[−0.070%, 1.930%]	Compliant
6	9,733	267	2.670%	[1.670%, 3.670%]	**Non-compliant**
7	9,901	99	0.990%	[−0.010%, 1.990%]	Compliant
8	9,912	88	0.880%	[−0.120%, 1.880%]	Compliant
9	9,896	104	1.040%	[0.040%, 2.040%]	Compliant
10	9,758	242	2.420%	[1.420%, 3.420%]	**Non-compliant**

capability to verify packet loss with provable confidence and minimal overhead. For all experiments, the link between S_1 and S_2 was configured with a 1.00% drop rate. Mininet leverages the Linux kernel's NetEm module, which supports precise, probabilistic packet loss; therefore, introducing a 1.00% drop rate in Mininet mimics the real network behavior With target accuracy $\varepsilon = 0.01$ and confidence $1 - \delta = 0.95$, the required sample size is derived via Hoeffding's inequality:

A. Calculating the Required Sample Size

To illustrate the efficiency of the VeriStat framework, consider a representative scenario where the objective is to verify packet loss with an accuracy of $\pm 1\%$ at a 95% confidence level. In this case, the permissible error margin is set to $\varepsilon = 0.01$, and the corresponding failure probability is $\delta = 0.05$. Substituting these parameters into the Hoeffding-based sample size formula yields:

Substituting these values into the n_{req} formula:

$$n_{req} = \frac{1}{2(0.01)^2} \ln\left(\frac{2}{0.05}\right)$$

$$n_{req} = \frac{1}{0.0002} \ln(40)$$

$$n_{req} \approx 5000 \times 3.6888 \approx 18,445 \text{ samples}$$

Thus, a total of approximately $18{,}445$ samples are sufficient to ensure, with 95% confidence, that the empirically measured loss rate $\hat{\mu}$ deviates from the true loss rate μ by no more than $\pm 1\%$. Importantly, this required number of samples is independent of the total traffic volume or flow duration, depending solely on the desired accuracy and confidence parameters.

B. QoS Compliance Verification

Table 1 presents the results of ten independent trials where the collector was programmed to aggregate samples until the required statistical minimum of $n_{req} = 18,445$ was met. The objective was to validate the sufficiency of the calculated sample size. Across all ten runs, the measured sample loss rate ($\hat{\mu}$) consistently clustered tightly around the ground truth of 1.00%. In every trial, the observed loss fell within the expected range of $[0.00\%, 2.00\%]$. Crucially, the resulting 95% confidence interval, defined as $[\hat{\mu} - 1.0\%, \hat{\mu} + 1.0\%]$, successfully included the true loss rate of 1.00% This validates that the combination of P4-based PRNG sampling and the Hoeffding calculation effectively provides a reliable, auditable compliance report with the guaranteed 95% confidence. Note that since packet loss is a non-negative metric, any negative lower bounds in the computed confidence intervals can be treated as zero, since the loss values cannot be less than 0%.

C. Non Compliance QoS Verification

Table 2 presents 10 trials conducted under identical network conditions, but utilizing an insufficient sample size of $n = 10,000$ (approximately 54% of n_{req}). The samples exhibited high statistical volatility, leading to a breakdown of the confidence guarantee. The runs that resulted in non-compliance are Rounds 4, 6, and 10. These results perfectly demonstrate the necessity of n_{req}. The three out of the ten runs (30%) resulted in measured sampled loss rates that were outside the expected range of $[0.0\%, 2.0\%]$ (e.g., $2.410\%, 2.670\%, 2.420\%$). For these non-compliant runs, the calculated 95% confidence

TABLE III
ANALYSIS OF MEASURED LOSS AT DIFFERENT SAMPLING RATES

Sample Rate(%)	Total Packets Sent	Avg. Packets Dropped	Avg. Loss (%)
5%	368,900	182.33	0.989%
10%	184,450	186.00	1.009%
15%	122,966	176.33	0.956%
20%	92,225	193.00	1.046%

interval completely excluded the true loss rate of 1.00%. The observed failure rate of 30% significantly exceeds the acceptable failure probability of $\delta = 5\%$. This confirms that $n = 10,000$ provides unreliable data and proves that the explicit determination of the sample size via Hoeffding's inequality is a crucial feature that distinguishes this verifiable method from conventional probabilistic telemetry.

D. Effect of Varying Sampling Rates

The experimental analysis of varying sampling rates validates the scalability of the VeriStat framework. For a fixed statistical requirement ($n_{\text{req}} = 18{,}445$, $\varepsilon = 1\%$, $\delta = 0.05$), the sampling rate (p) influences only the telemetry overhead and time-to-report, not the accuracy of compliance verification.

Experiments conducted with $p \in \{5\%, 10\%, 15\%, 20\%\}$ as shown in Table III, confirmed that all measured average loss values ($\hat{\mu}$) remained statistically consistent with the true loss rate of 1.0%, demonstrating that accuracy depends solely on the achieved sample size. Higher sampling rates reduced the reporting delay but increased resource consumption, while lower rates offered minimal overhead at the cost of slower verification. This highlights VeriStat's ability to balance operational efficiency and verification latency without compromising its formal statistical guarantees. This flexibility allows network operators to choose the lowest feasible sampling rate to maximize infrastructure scalability without compromising the ϵ and δ statistical guarantee.

VI. DISCUSSION

The experimental results definitively validated the necessity and sufficiency of the sample size (n_{req}) derived from Hoeffding's inequality. The $n < n_{req}$ trials confirmed the statistical risk of under-sampling, observing a 30% failure rate in compliance compared to the guaranteed $\delta \leq 5\%$. Conversely, the $n = n_{req}$ trials demonstrated that the 95% confidence bound reliably contains the true loss rate (μ_{actual}), thereby validating the method's rigor. However, Hoeffding's inequality provides a universally applicable, but often conservative, bound because it ignores the variance (σ^2) of the random variables. For high-quality network services, such as enterprise VoIP true packet loss (μ) is very small ($\mu \ll 1\%$), Hoeffding's conservative nature leads to a higher n_{req} than is strictly necessary.

To achieve operational efficiency in near-zero-loss networks, we plan to apply Bernstein's inequality to compute n_{req}, which incorporates variance and yields a tighter bound. Bernstein's inequality refines the bound by incorporating the low variance ($\sigma^2 = \mu(1-\mu)$) characteristic of small loss probabilities. This results in a tighter, less conservative sample size. By introducing a reliable a priori estimate of μ (e.g., historical average or SLA target) into the n_{req} calculation, Bernstein's inequality enables the system to maintain the same ϵ and δ guarantee with significantly reduced telemetry overhead, transforming the solution into an operationally optimized tool for highly scalable P4 environments.

VII. CONCLUSION

This paper presented a novel framework for Verifiable QoS with Provable Statistical Guarantees implemented within a P4-programmable data plane. By integrating Pseudorandom Number Generation sampling with the non-asymptotic concentration bounds of Hoeffding's inequality, we move network monitoring toward auditable, mathematically provable SLA compliance verification. The experimental results confirmed the method's robustness by validating the sufficiency and necessity of the sample size boundary, and demonstrated the scalability gains achieved by decoupling accuracy from sampling rate. Future work will focus on implementing Bernstein's inequality to optimize the number of required samples based on low-variance traffic characteristics, further minimizing telemetry costs for critical QoS applications. We also plan to validate the framework on larger, multi-hop topologies and extend verification to continuous metrics, including latency and jitter, to ensure comprehensive SLA compliance.

REFERENCES

[1] E. Boyd et al., "The perfSONAR Network Measurement Toolkit," IEEE Commun. Mag., vol. 54, no. 2, pp. 34–40, 2016.
[2] B. H. Cantor et al., "Two-Way Active Measurement Protocol (TWAMP)," IETF RFC 5357, 2008.
[3] N. Duffield et al., "A Framework for Passive Packet Measurement," IEEE Trans. Netw. Serv. Manag., vol. 7, no. 4, pp. 365–379, 2009.
[4] R. Enns et al., "Streaming Network Telemetry: Towards Real-Time Visibility," IEEE Commun. Standards Mag., vol. 3, no. 4, pp. 18–25, 2019.
[5] P. Bosshart et al., "In-band Network Telemetry (INT)," Intel White Paper, 2015.
[6] B. Gafni et al., "In-situ OAM (IOAM): Operational Considerations," IETF RFC 9197, 2022.
[7] Papadopoulos, Konstantinos, Panagiotis Papadimitriou, and Chrysa Papagianni. "Deterministic and probabilistic p4-enabled lightweight in-band network telemetry." IEEE Transactions on Network and Service Management 20.4 (2023): 4909-4922.
[8] Ben Basat, Ran, et al. "PINT: Probabilistic in-band network telemetry." 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.
[9] D. Sanvito et al., "P4DM: Data Plane Delay Monitoring Using P4," IEEE NOMS, 2020.
[10] M. Barnes et al., "P4-perfSONAR: Extending Network Telemetry with Programmable Data Planes," IEEE Trans. Netw. Serv. Manag., 2022.
[11] G. Antichi et al., "Lean: Efficient Loss Detection in the Data Plane," ACM SOSR, 2021.
[12] J. Suh et al., "NetVision: Efficient Network Telemetry via Sketch Compression," IEEE INFOCOM, 2022.
[13] Y. Liu et al., "INT-Source: Topology-Adaptive In-band Network-wide Telemetry," IEEE/IFIP NOMS, 2023.
[14] T. Song et al., "PathLens: Scalable Path-Level Telemetry in Programmable Networks," IEEE TNSM, vol. 20, no. 1, pp. 455–469, 2023.
[15] K. Papadopoulos, P. Papadimitriou and C. Papagianni, "Deterministic and Probabilistic P4-Enabled Lightweight In-Band Network Telemetry," in IEEE Transactions on Network and Service Management, vol. 20, no. 4, pp. 4909-4922, Dec. 2023