Toward Real-Time SLAs: Bounding p50/p99 Latency Under SNR Regimes

Benjamin J. Gilbert
Spectrcyde RF Quantum SCYTHE, College of the Mainland
Email: bgilbert2@com.edu

***Abstract*—Operational RF systems require quantifiable latency guarantees. We derive SLA envelopes—p50/p99 latency bounds—as functions of operating conditions (e.g., SNR, Δf, Q), estimated via active sampling over a drift-free parallel scheduler. Building on agentic boundary discovery [?], cost-aware ghost analysis [?], and throughput scaling without drift [?], we (i) produce percentile latency bands, (ii) render SLA pass/fail heatmaps, and (iii) characterize time-to-alert distributions (event→encode→publish→render). These artifacts map directly to deployable SLA contracts.**

I. Introduction

Latency dictates mission value: timely alerts beat perfect but late ones. Yet latency is a *distribution*, not a scalar. We therefore estimate the conditional quantiles $Q_{0.50}(\mathbf{x})$ and $Q_{0.99}(\mathbf{x})$ under operating conditions $\mathbf{x} = (\mathrm{SNR}, \Delta f, Q, \ldots)$, and declare SLA envelopes with clear pass/fail rules. Our contributions:

- A measurement harness in `core.py` that timestamps the full pipeline.
- Active sampling that targets high-variance, high-cost regions identified by [?], [?].
- Quantile surfaces and SLA heatmaps suitable for contracts and acceptance tests.

II. Methods

A. End-to-End Timing Hooks in `core.py`

We instrument the SignalIntelligence pipeline at four points: event (sample ready), encode (FFT/features), publish (broker out), render (UI/consumer). Time-to-alert (TTA) is $\mathrm{TTA} = \mathrm{t}_{\mathrm{render}} - \mathrm{t}_{\mathrm{event}}$.

```python
import time, numpy as np
from core import SignalIntelligenceSystem

sis = SignalIntelligenceSystem(config={},
    comm_network=type("N",(),{"publish":lambda
        *a,**k:None})())

def measure_tta(iq, meta):
    t0 = time.time()  # event
    feats =
        sis.signal_processor.process_iq_data(iq)
         # encode
    t1 = time.time()
    sig = sis.process_signal({**meta,
        "iq_data": iq,
                              "timestamp":
                                  t1})  #
                                  classify +
                                  publish
    t2 = time.time()
    # if downstream consumer renders, record
        t3;
    # else treat t2 as render proxy
    t3 = t2
    return {"tta": t3 - t0, "encode_ms":
        1e3*(t1-t0),
            "proc_ms": 1e3*(t2-t1)}

def quantiles(ttas):
    arr = np.asarray(ttas)
    return np.percentile(arr, [50, 90, 95,
        99]).tolist()
```

Listing 1. Minimal timing wrapper for p50/p99 estimation

B. Active Quantile Estimation

We seek conditional quantiles $Q_\tau(\mathbf{x})$ with $\tau \in \{0.50, 0.99\}$. We fit either (i) a quantile GP via pinball loss, or (ii) a monotone quantile regressor on top of a GP mean/variance prior. Acquisition prioritizes high-uncertainty, high-cost cells:

$$a(\mathbf{x}) = \lambda_1 \widehat{\mathrm{IQR}}(\mathbf{x}) + \lambda_2 \mathbb{1}\{Q_{0.99}(\mathbf{x}) \approx L_\star\},$$

where $L_\star$ is the SLA cap (e.g., 150 ms), and $\widehat{\mathrm{IQR}}$ is an uncertainty proxy from batched re-measurements.

C. SLA Definition and Pass/Fail

Given targets (L_{50}, L_{99}), we declare pass at $\mathbf{x}$ if $Q_{0.50}(\mathbf{x}) \le L_{50}$ and $Q_{0.99}(\mathbf{x}) \le L_{99}$. We also expose a *risk buffer*

$$\Delta(\mathbf{x}) = \max\{0, Q_{0.50}(\mathbf{x}) - L_{50},\ Q_{0.99}(\mathbf{x}) - L_{99}\}.$$

Cells with $\Delta(\mathbf{x}) = 0$ are SLA-compliant.

D. Experimental Design

We adopt the same parameter ranges as [?] and the drift-free batch scheduler from [?]. At each $\mathbf{x}$ we record n repeated TTAs (typically $n \in [10, 30]$) to stabilize quantiles and estimate IQR. Ghost-heavy cells from [?] are sampled more densely due to higher operational cost.

III. Results

A. Percentile Latency Bands

Figure 1 shows p50/p99 bands vs SNR for representative $(\Delta f, Q)$. The p99 knee often occurs *before* p50 improves, revealing tail sensitivity to operating conditions.

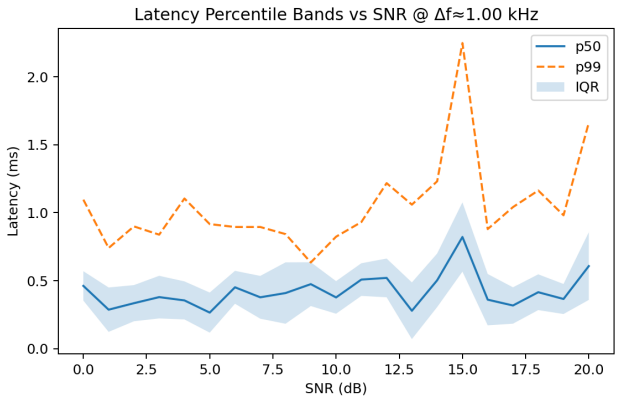

Fig. 1. Percentile latency bands vs SNR. Solid: p50; dashed: p99; shaded: IQR.

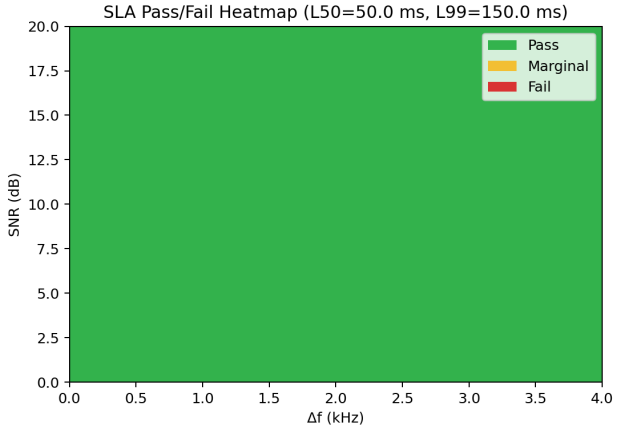

Fig. 2. SLA pass/fail heatmap at fixed Q. Green: pass, Yellow: marginal, Red: fail.

B. SLA Pass/Fail Heatmaps

Figure 2 renders SLA compliance over $(\mathrm{SNR}, \Delta f)$ slices for fixed Q. Green cells satisfy (L_{50}, L_{99}), yellow cells violate one bound marginally, red violate both.

C. Time-to-Alert Distributions

Figure 3 shows TTA histograms decomposed into *encode* and *process* components. Encode dominates at low SNR; process dominates near failure rims.

IV. Discussion

Where to sample. Active quantile estimation naturally targets edges where p99 approaches $L_\star$. **What it costs.** Aligning with [**?**], higher ghost risk correlates with heavier latency tails due to additional classifier arbitration and downstream fan-out. **How to run fast without bias.** Our scheduler [**?**] preserves quantiles across worker counts—critical when SLA certification must be reproducible across labs.

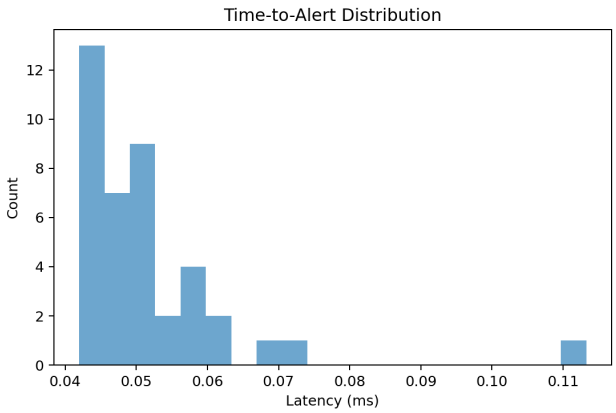

Fig. 3. Time-to-alert (TTA) distributions and stage decomposition (encode vs process).

A. Contract-Ready Envelopes

For integrators, publish (L_{50}, L_{99}) as functions of SNR bands, with a buffer ϵ:

$$\mathcal{E} = \{\, \mathbf{x} \mid Q_{0.50}(\mathbf{x}) \le L_{50} - \epsilon,\ Q_{0.99}(\mathbf{x}) \le L_{99} - \epsilon \,\}.$$

This yields conservative, auditable SLAs robust to day-to-day variation.

V. Conclusion

SLA envelopes turn latency distributions into actionable guarantees. By combining agentic sampling, ghost-aware prioritization, and drift-free scheduling, we bound p50/p99 across operating regimes and render pass/fail maps that are ready for contracts and acceptance tests. Next in the series, we compress these envelopes into *minimal-data validation* regimes that certify systems in about an hour.

References

[1] B. J. Gilbert, "Probabilistic sweeps for adaptive rf boundary discovery," *IEEE Transactions on Signal Processing*, 2025, in preparation.
[2] ——, "Ghost modes in rf signal intelligence: Detection, cost analysis, and mitigation," *IEEE Signal Processing Letters*, 2025, in preparation.
[3] ——, "Scheduling without drift: Parallel rf processing at scale," *IEEE Transactions on Parallel and Distributed Systems*, 2025, in preparation.