Evaluating QoO in Challenging Network Conditions

A simulation study

This document provides supporting analysis for the QoO metric specified in [QoO-ID]. It contains results from simulation experiments designed to test QoO's sensitivity to sampling frequency, measurement accuracy, and application-specific requirements.

Quality of Outcome (QoO) is a network quality score that measures how well specific applications will work on a given internet connection. The full specification for QoO has been published as an internet draft. QoO is computed based on latency and packet loss measurements. In version 00 of the internet draft, we specify that the metric is independent of the method by which the latency and packet loss samples are collected. This document tests this assertion by exploring QoO's sensitivity to sampling frequency and accuracy through simulation.

We investigate the performance of QoO in challenging scenarios that stress its design assumptions. The scenarios are purposefully set up to be challenging and involve cases where it is well known that other methodologies such as speedtests nearly always fail to discover network quality degradation.

This allows researchers to use the results and methodology to investigate the relationship between how often network quality is measured (sampling frequency) and the accuracy of the QoO score.

Code for reproducing the results can be found here: https://github.com/domoslabs/qoosim

Method

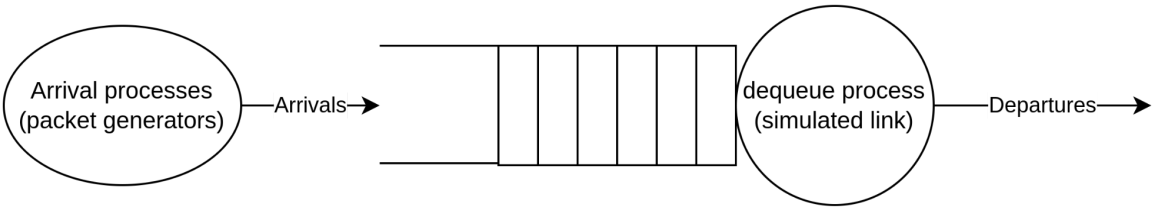
It is necessary to use simulation or real-world tests for this study, and not just mathematical analysis of long-tailed statistical distributions, because time-series of network latency are self-correlated. Latency spikes are typically caused by queuing, and these queues do not fill or drain instantaneously. An effect of this is that latency samples taken back-to-back at a high enough frequency tend to measure almost the same delays. This effect is not captured by analysis of stationary distributions.

This section describes the simulation method and the simulated scenarios.

Simulation

We implement a discrete-event simulator to generate end-to-end latency traces. The simulator employs a simple FIFO queue model, with packet arrivals driven by configurable random processes (Poisson, fixed interval, or bursty) to emulate diverse application traffic patterns. Packet departure, representing network service, is modeled by removing the head-of-line packet and

introducing a waiting time before the next packet departs. This waiting time can be fixed or drawn from a distribution of inter-departure times, allowing us to simulate specific network behaviors like WiFi access delays.



Scenarios

To evaluate QoO under different network conditions, we define a set of scenarios:

- **WiFi Access Delay:** Simulates delays due to contention on a WiFi channel.
- **Bufferbloat (Short and Long-Lived):** Emulates latency spikes caused by bufferbloat, with variations in spike duration.
- **Temporary Service Outage:** Models brief periods of service unavailability.

Each scenario is parameterized by the random processes governing traffic arrival and network delays. We run multiple simulations per scenario to capture a range of potential outcomes due to the inherent randomness.

Ground Truth and Sampling

For each simulation run, we generate a high-resolution "ground truth" latency trace by interpolating the simulated queueing delays. This trace represents the ideal, continuous measurement of network latency. To examine the impact of sampling frequency, we sub-sample this ground truth trace at various rates and patterns, and to evaluate the impact of measurement inaccuracy, we introduce Gaussian noise to the sampled values.

QoO Computation and Analysis

QoO scores are computed for each sub-sampled and noise-injected trace. We then analyze the spread and accuracy of these QoO scores compared to the ground truth QoO, to assess the sensitivity of QoO to sampling frequency, measurement accuracy, and variations in application requirements. Specifically, we investigate:

1. **Measurement Frequency:** We evaluate the impact of sub-sampling the ground truth trace at different rates on the accuracy and variability of QoO scores.
2. **Measurement Accuracy:** We examine how adding Gaussian noise with varying standard deviations to the sampled latency values affects QoO scores.
3. **Requirement Specification:** We investigate QoO's sensitivity to changes in application requirements by varying the network requirement parameters (NRP, NRPOU, and percentiles) and observing the resulting scores.
4. **Measurement duration (Number of samples):** We investigate how the measurement duration, which, combined with the sampling rate, defines the number of samples collected before the QoO score is calculated, affect the accuracy and variability of QoO scores.

Detailed scenario descriptions

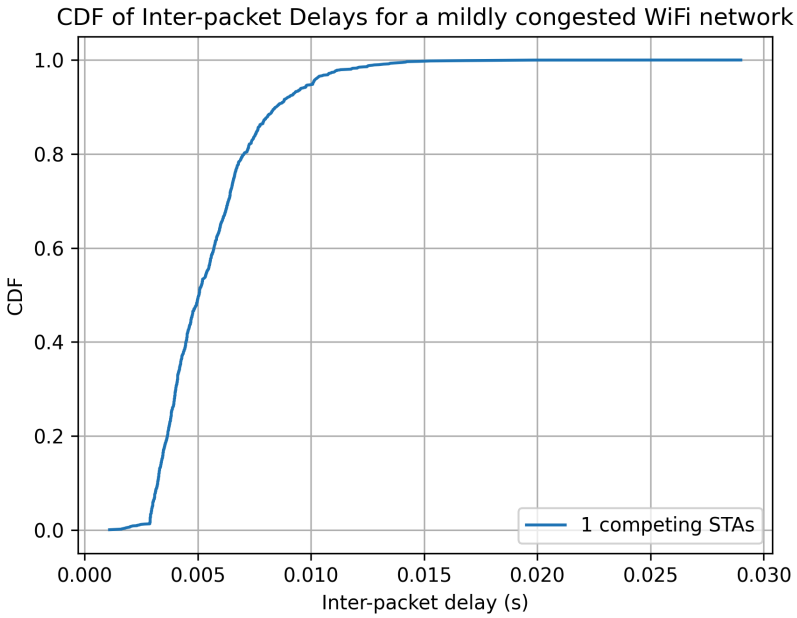
The simulator must be set up with specific scenarios. We attempt to simulate a few common failure modes in networks. We aim to make these scenarios realistic while keeping the simulation simple. The scenarios are defined in terms of the parameters of the random processes that generate traffic and delays in the simulated system. We run each scenario several times to sample a reasonable set of possible realizations.

Each of the simulated traces are stored and reused for analysis of how sensitive QoO scores are concerning measurement frequency, measurement accuracy, and application requirement specifications respectively.

WiFi access delay

This scenario simulates delays experienced by a device contending for access to a WiFi channel with other active stations. The simulation incorporates a probability distribution of inter-packet delays based on empirical data [WiFi QA], reflecting the variability in channel access times. The inter-packet delay distribution used is shown in the figure below.

- **Departure process:** Random samples from the distribution shown in the plot below, which is an empirical sample of WiFi access delays with a single always-on competing station.
- **Base load:** 170 packets per second of Poisson traffic.



Bufferbloat

We implement two bufferbloat scenarios, one with short-lived (1 second) bufferbloat, and one with longer-lived (5 second) bufferbloat. For a detailed description of bufferbloat, please refer to [Bufferbloat].

- **Departure process:** Constant 2.5 millisecond packet processing time, supporting a maximum load of 400 packets per second.
- **Base load (applies to both scenarios):** 200 packets per second of Poisson background traffic.
- **Short-Lived Bufferbloat:** Simulates brief (<1 second) latency spikes due to buffer congestion. This is simulated by adding a bursty traffic flow that temporarily overloads the queue by adding 350 packets per second of Poisson load for 1 second every 30 seconds.
- **Long-Lived Bufferbloat:** Emulates extended (~5 second) latency spikes. This is simulated by adding a bursty traffic flow that temporarily overloads the queue by adding 220 packets per second of Poisson load for 5 seconds every 30 seconds.

Both scenarios introduce a bursty traffic flow that periodically overloads the queue, causing temporary latency spikes. Both scenarios also include a background flow generating traffic well below the sustainable rate of the link.

Temporary service outage

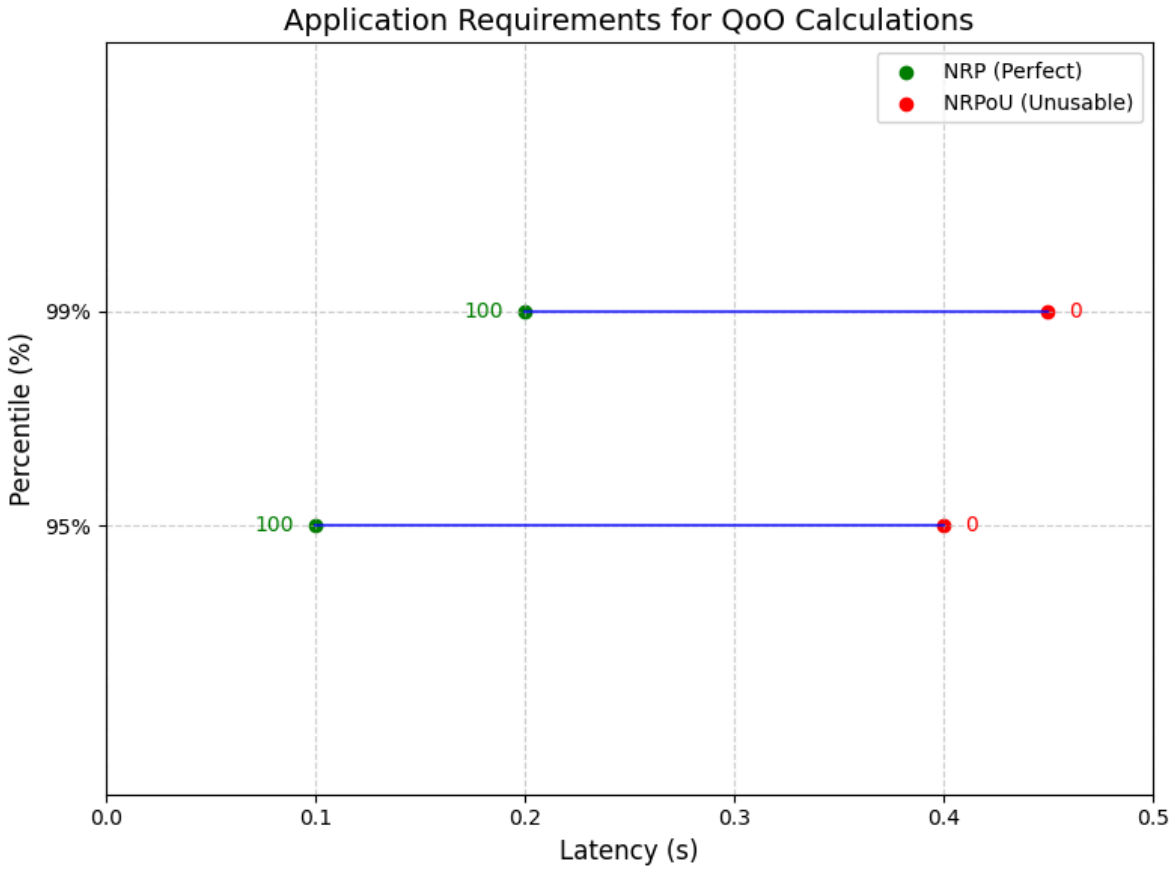
This scenario simulates a situation where the end-to-end service is temporarily unavailable for short periods at random times. We simulate this by adding a small probability of spending a full second to process a single packet. This simulates a 1 second service outage happening a random intervals.

- **Departure process:** A 6000/6001 chance of spending 2.5 milliseconds per packet, and a 1/6001 chance of spending a full second processing a packet.
- **Base load:** 100 packets per second of Poisson load

The network requirement

We use the same network requirement specification for all of the experiments, except for those testing sensitivity to requirement specification. The chosen requirement is what we would consider a reasonable guess at a network requirement for a video conferencing application.

As defined in [QoO-ID], a network requirement for QoO consists of two thresholds defined in terms of percentiles and latency values: The Network Requirements for Perfection (NRP) and the Network Requirements Point of Unusability (NRPOU).



Results

Impact of Measurement Frequency

How does measurement frequency affect the spread of QoO scores?

To answer this question, we must understand how sampling the same signal at different rates affects QoO scores.

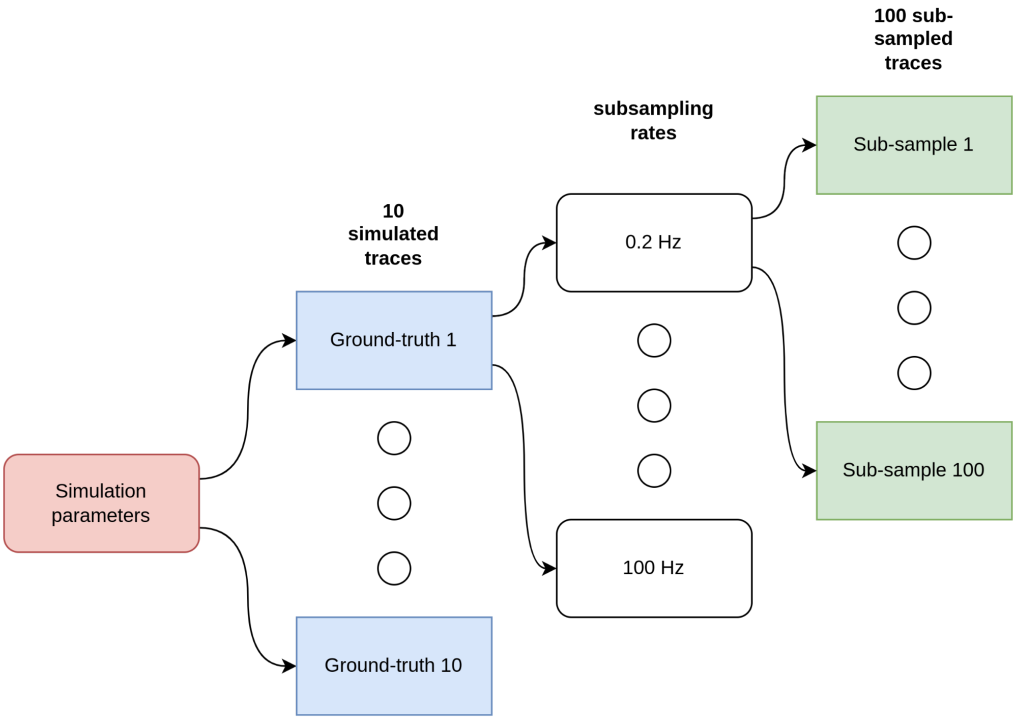
To estimate lower sampling frequencies, we sub-sample and interpolate the ground truth traces using different frequencies and sampling patterns. Then we compute the QoO score of each sub-sampling, and finally, we analyze the spread of these QoO scores.

We run a set of 10 simulations for each of the scenarios described above. We then generate a ground truth latency trace by interpolating the simulated queueing delays. This results in a ground truth trace with an effective sampling rate of 1000 Hz. This signal is then sub-sampled at several different rates, and the sub-sampled traces are used to compute QoO. We also compute a ground-truth QoO score using the 1000 Hz ground truth. We note that our 1000 Hz 'ground truth' is an approximation of continuous latency, which is then sub-sampled at lower rates to simulate different real-world measurement strategies.

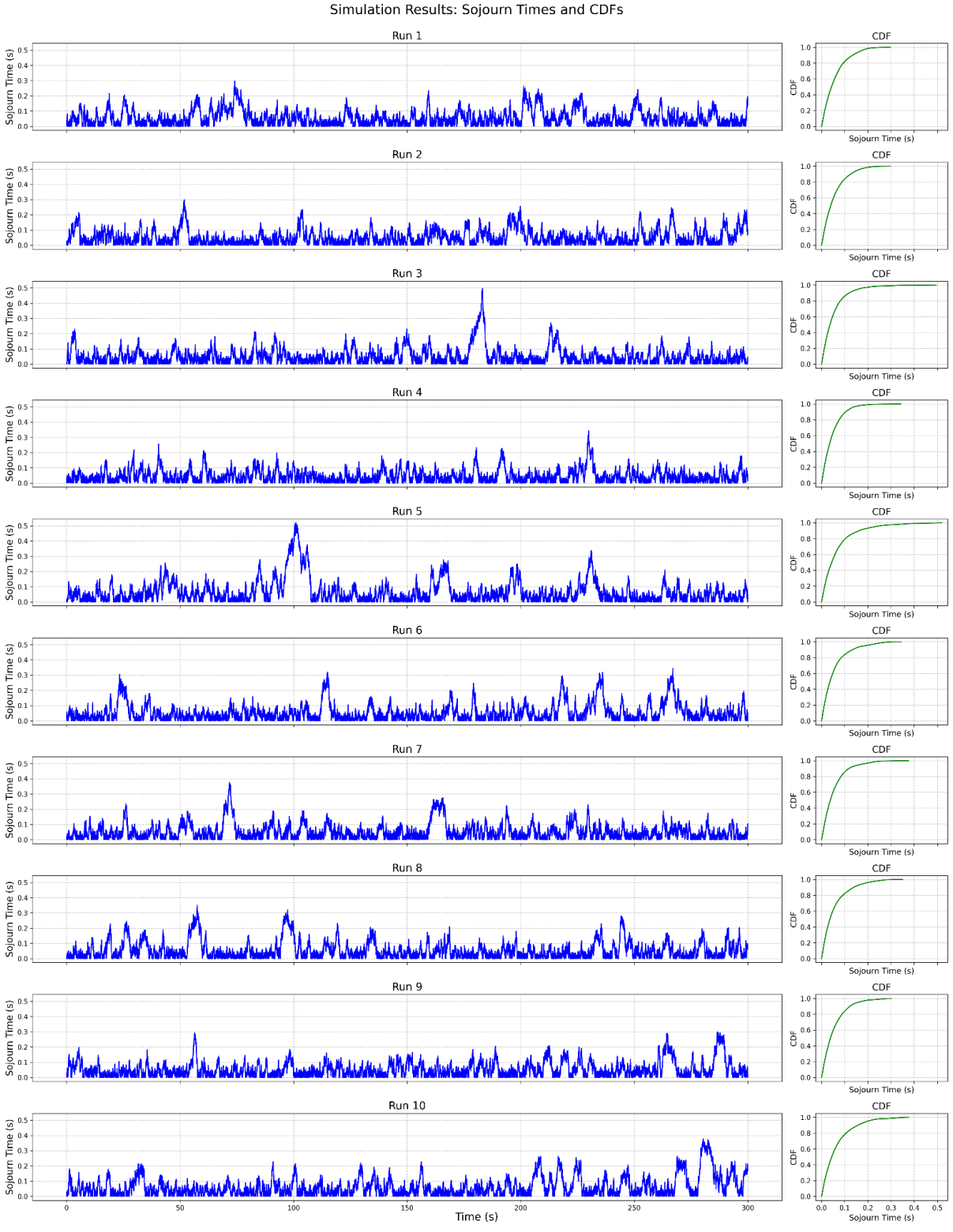
How to read the results

There are two levels of random sampling shown in the plots here:

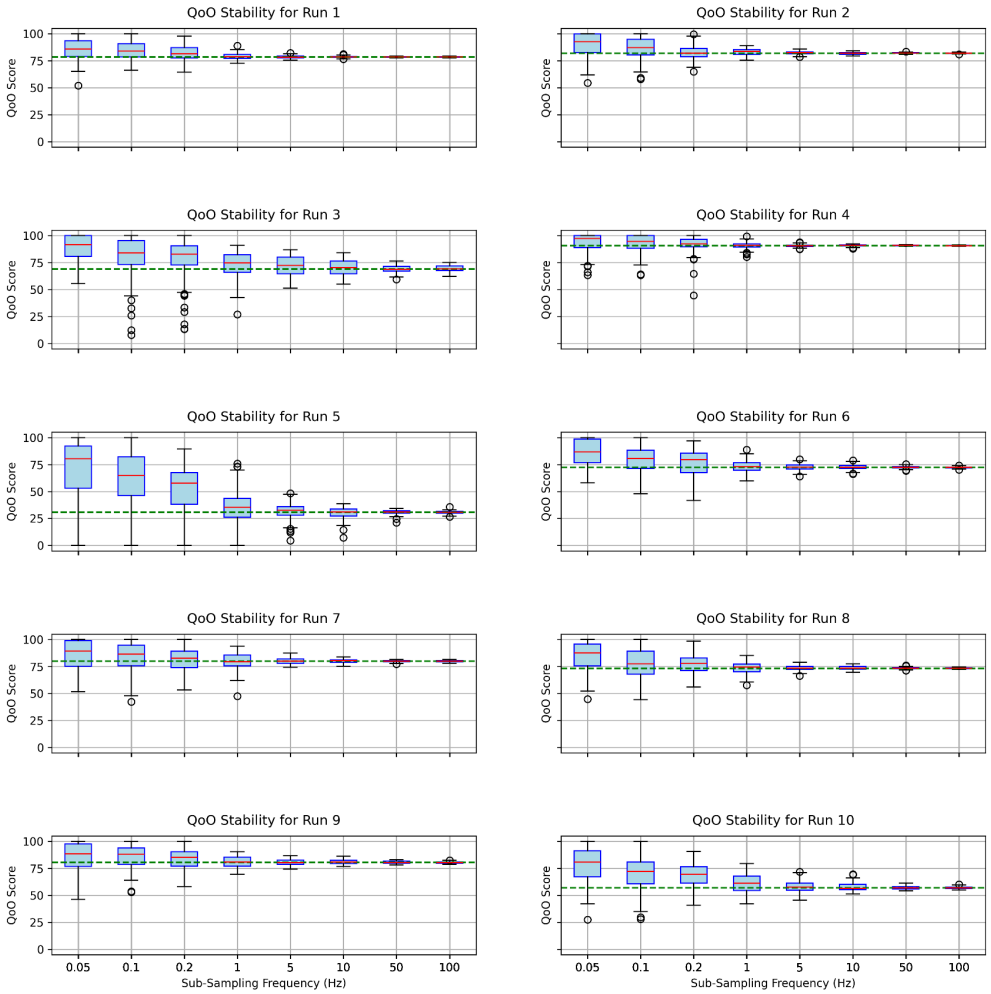
1. First, we run each simulated scenario 10 times, generating 10 different ground-truth latency traces. Each of the 10 runs use the same simulation parameters except for the random seed.
2. Second, for each of the rates we sub-sample at, we generate 100 different sub-sampled traces. This is done by sub-sampling using a poisson process (where inter-sample delays are determined by randomly sampling a negative exponential function) to determine the sub-sampling points. This generates 100 different versions approximating the ground truth, sub-sampled at the desired rate.



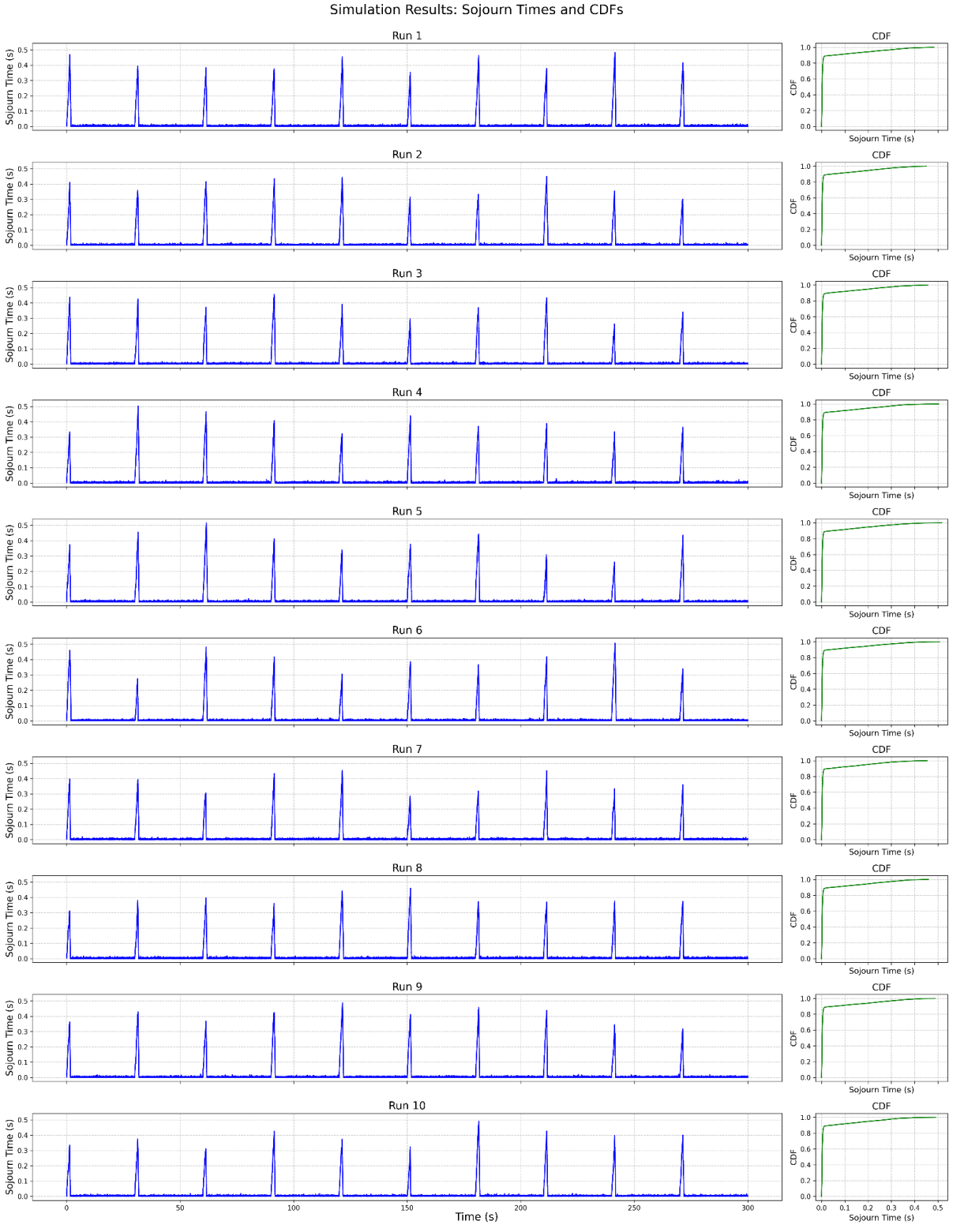
WiFi results



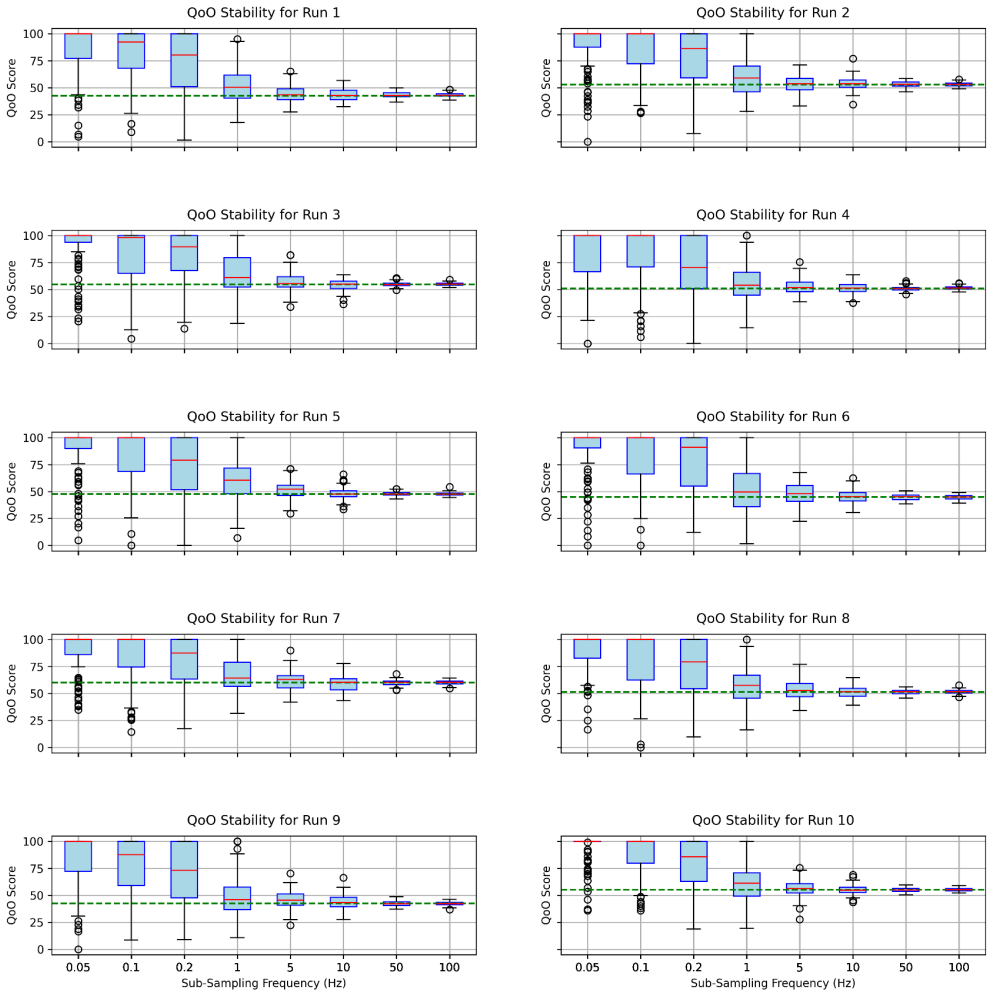
QoO Stability Analysis (Boxplots)



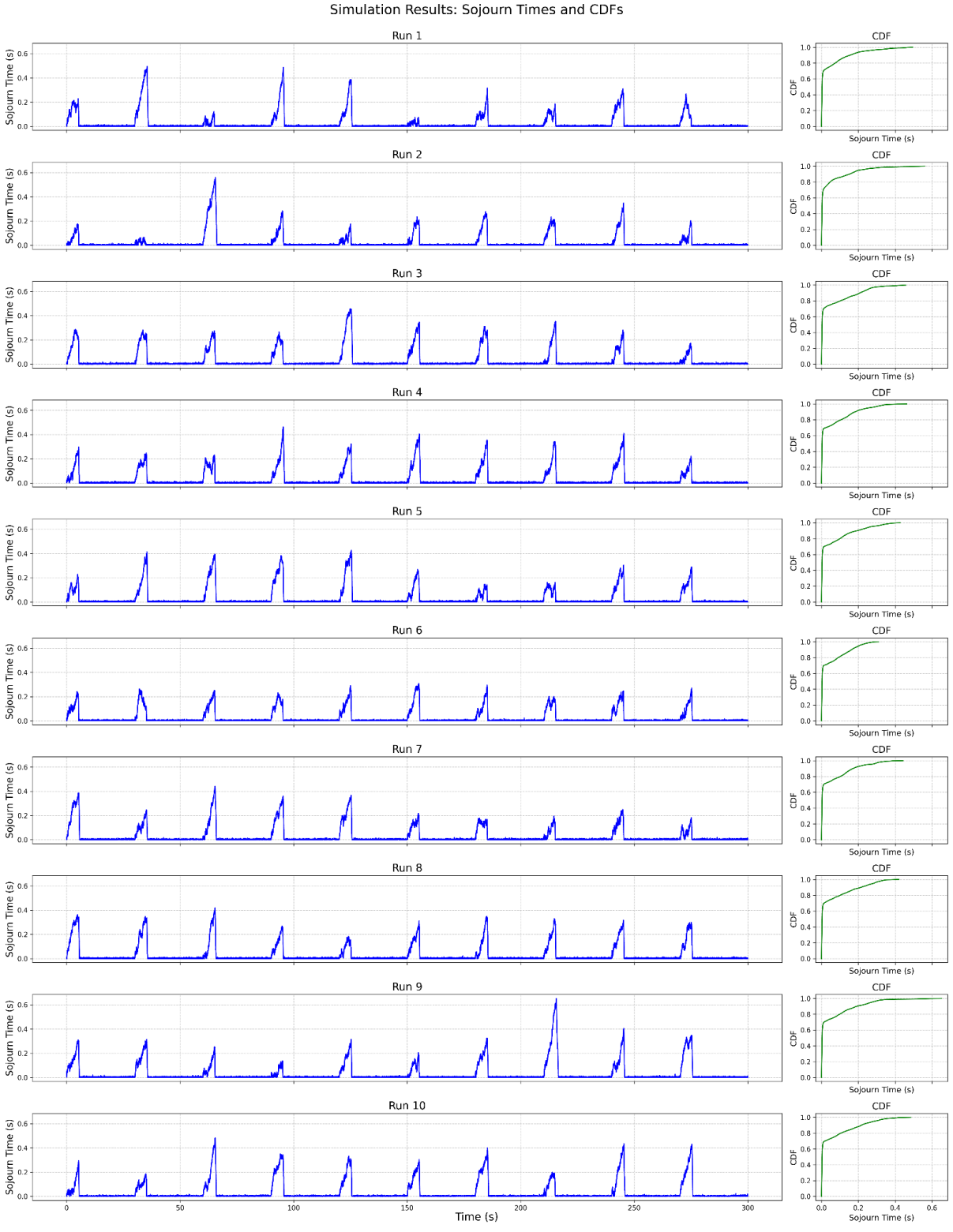
Short-lived (<1 second) bufferbloat spike results



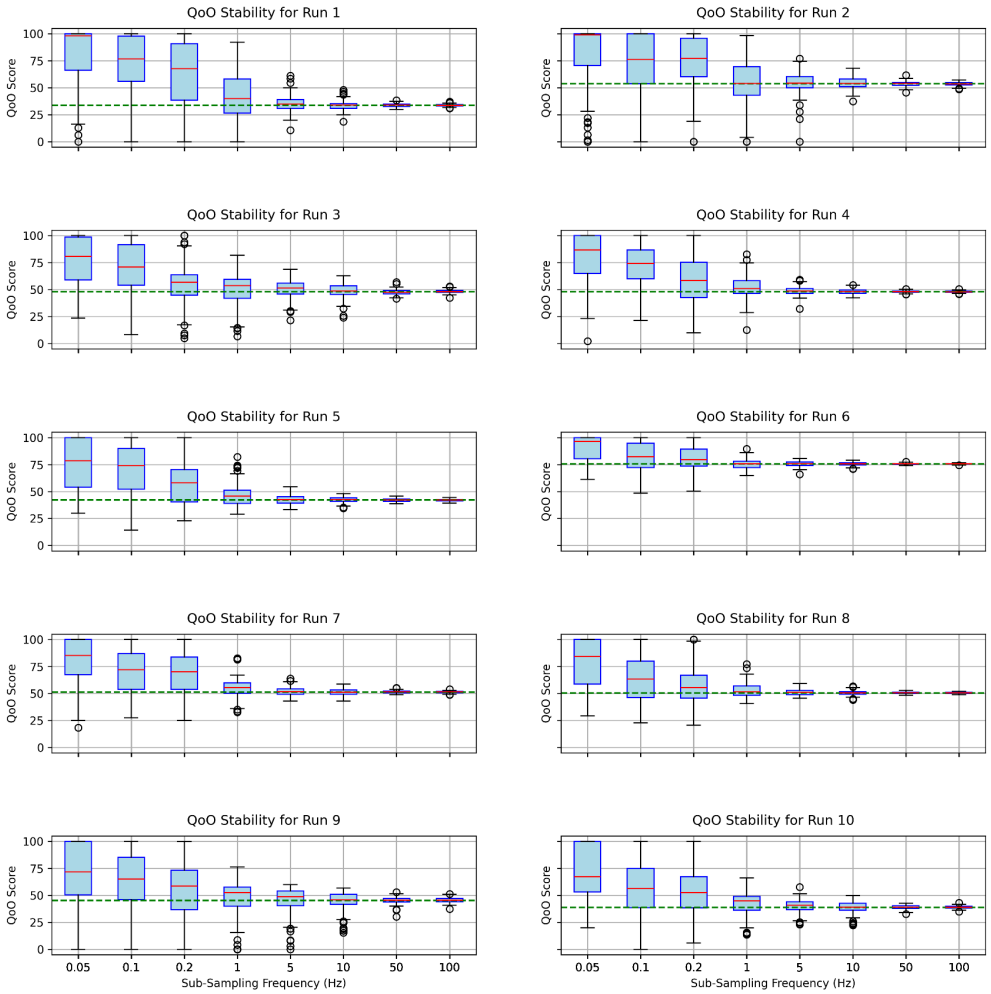
QoO Stability Analysis (Boxplots)



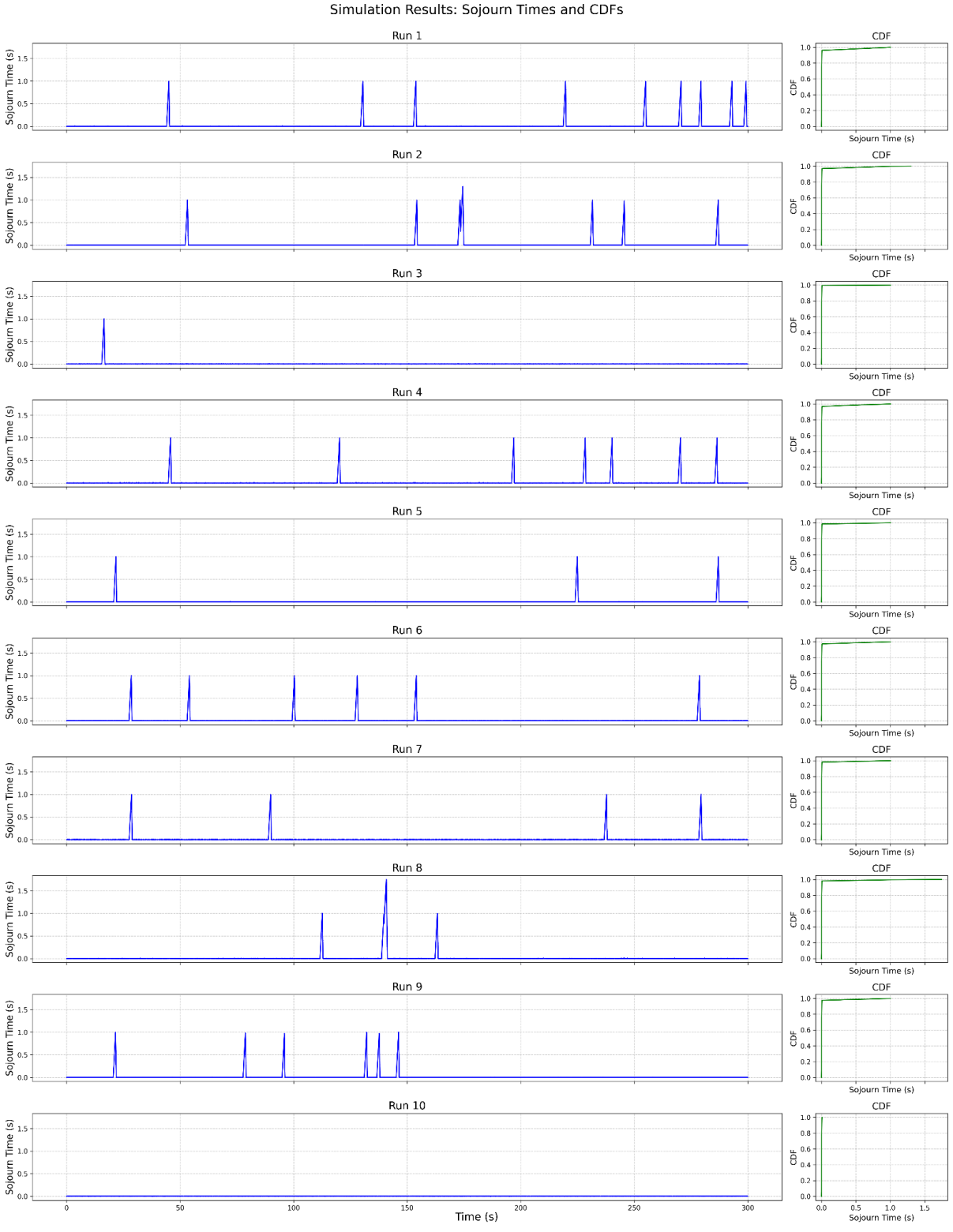
Long-lived (~5 second) bufferbloat spike results

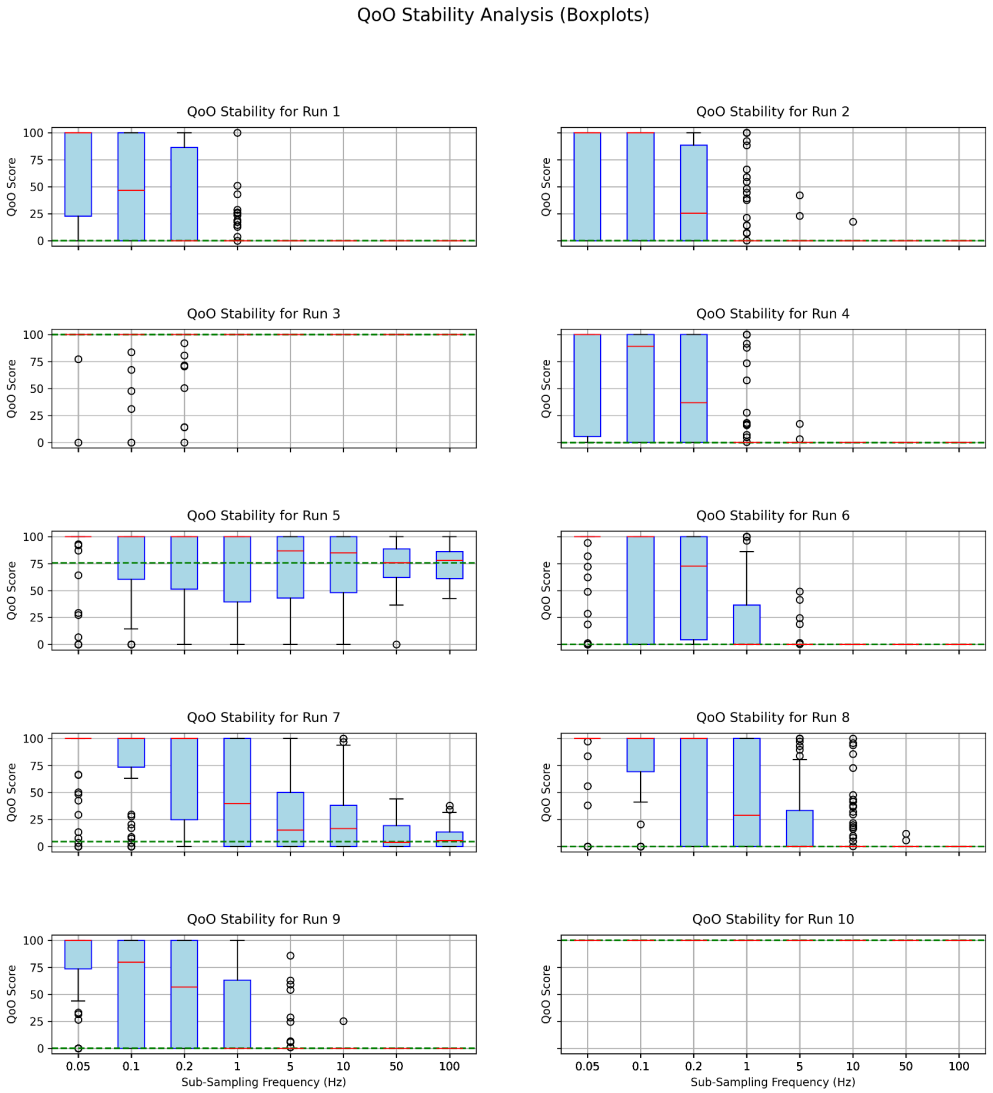


QoO Stability Analysis (Boxplots)



Service outage results





Summary of measurement frequency findings

The main pattern we see is that sampling at slow rates compared to the duration of the latency spikes we're trying to detect can result in poor accuracy of QoO scores. This effect is stronger when the spikes are rare, short-lived events, with the service outage scenario being the most extreme case. In our simulations we have set the requirements so that QoO starts to deteriorate when latency spikes above 100 milliseconds are frequent enough. QoO is captured very well with a sampling rate of 5-10 Hz (one sample every 100-200 milliseconds on average), and for slower (less frequent) sampling rates the QoO scores are more variable. A notable exception is the simulated WiFi link, where slower rates perform much better compared to the buffferbloat and service outage scenarios. This is likely due to the more consistent variability of the WiFi latency, making it more likely that less-frequent samples will observe latency spikes.

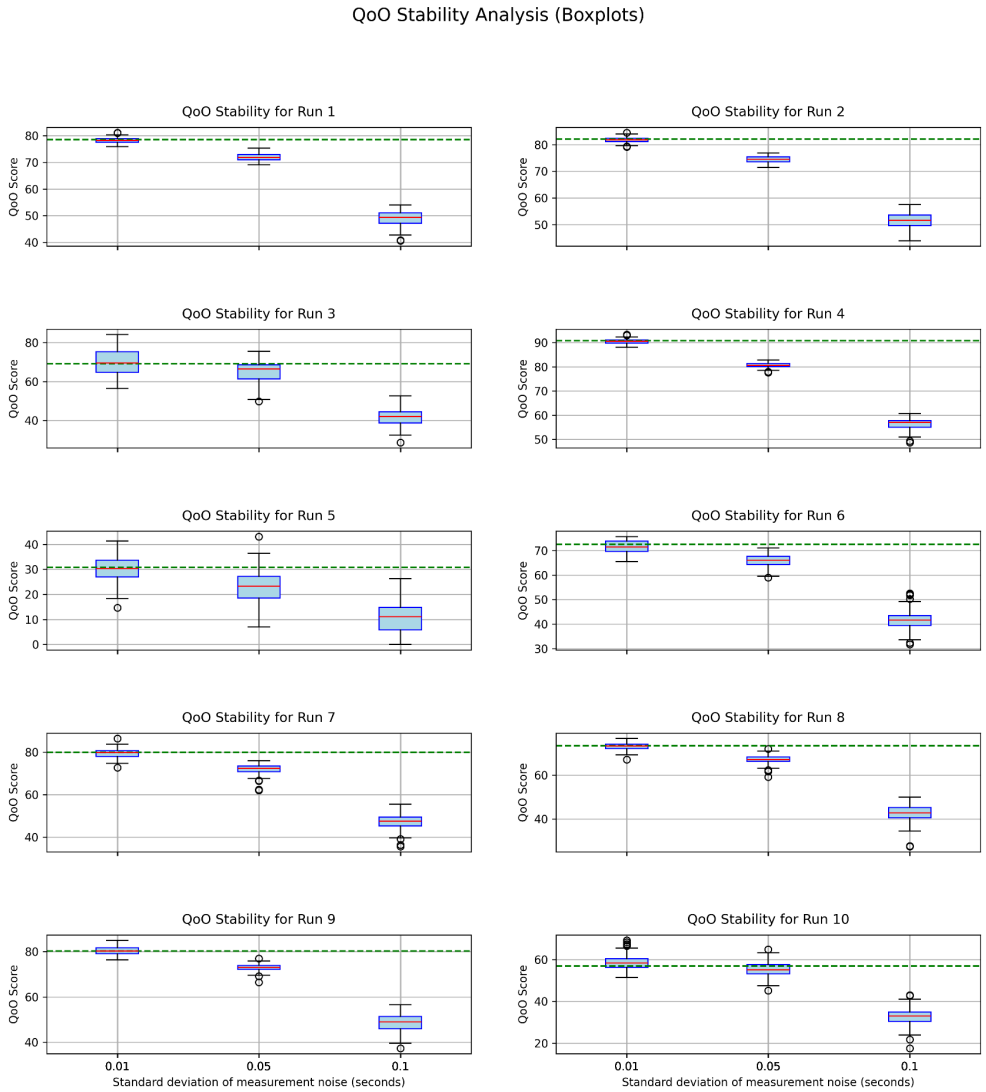
Another noticeable trend is that too-low sampling rates tend to result in overly optimistic QoO scores. This is likely an effect of, by chance, not sampling the latency spikes.

Impact of Measurement Accuracy

How does the accuracy of each latency sample affect the spread of QoO scores?

To understand this, we use a sub-sampling rate of 10Hz and add zero-mean gaussian noise to each of the sampled points. We vary the standard deviation of the noise to emulate different degrees of measurement inaccuracies.

WiFi results



Summary of measurement accuracy findings

The main effect we see is that noise on a scale approaching the scale of the application requirements (here, 100-400 milliseconds), has a large effect on the QoO scores. Even 50 millisecond standard deviation noise has a significant effect in our simulations.

Another pattern is that white noise only makes QoO worse. This makes sense, because our application requirement only cares about the worst 95th and 99th percentiles of latency. For the QoO score, it does not matter that the best-case latencies improve (when noise is negative), only that the worst-case worsens.

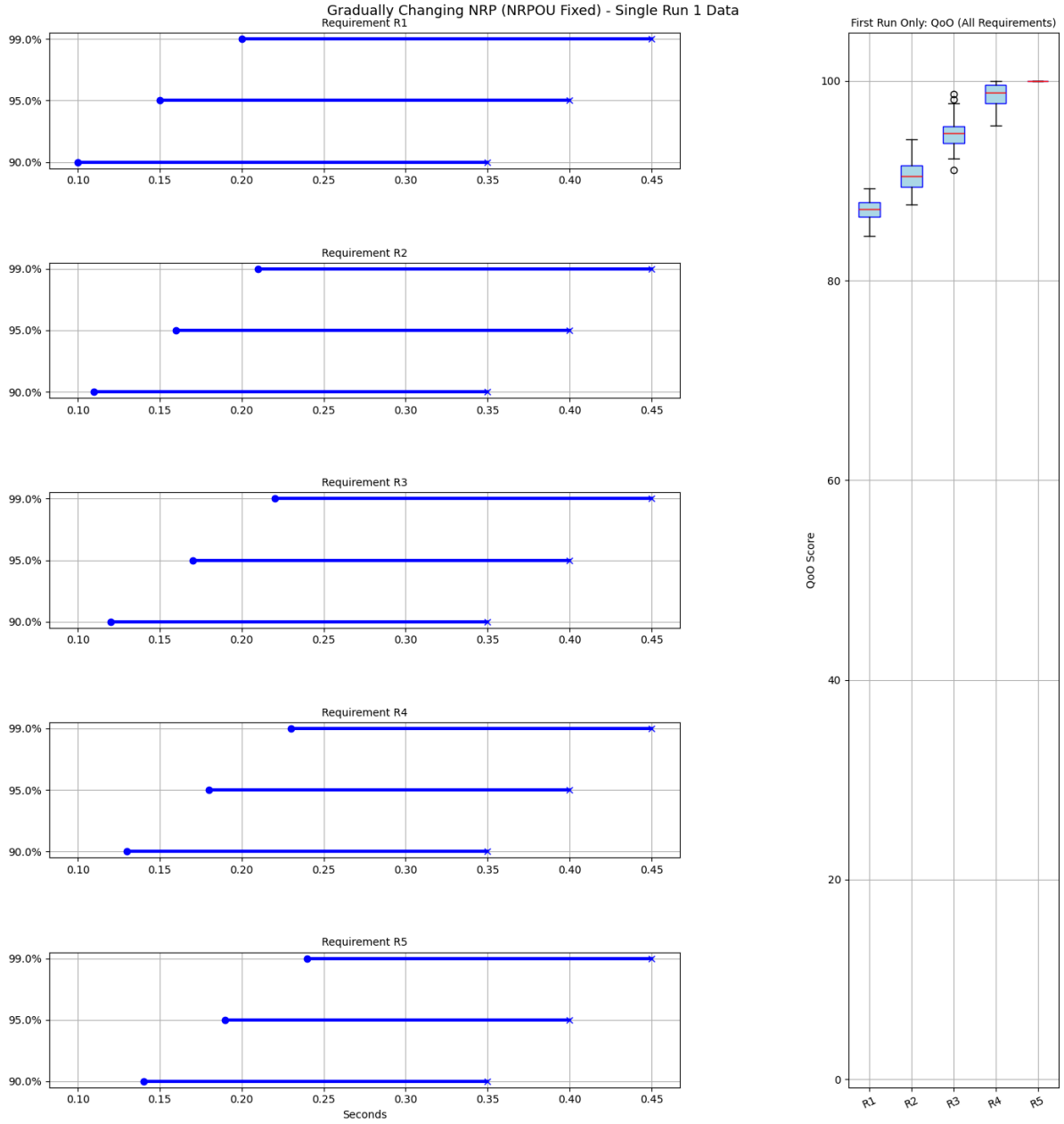
Impact of Requirement Specification

Do the details of the application requirement specification affect the QoO score? How sensitive is the score to small changes in the requirement spec?

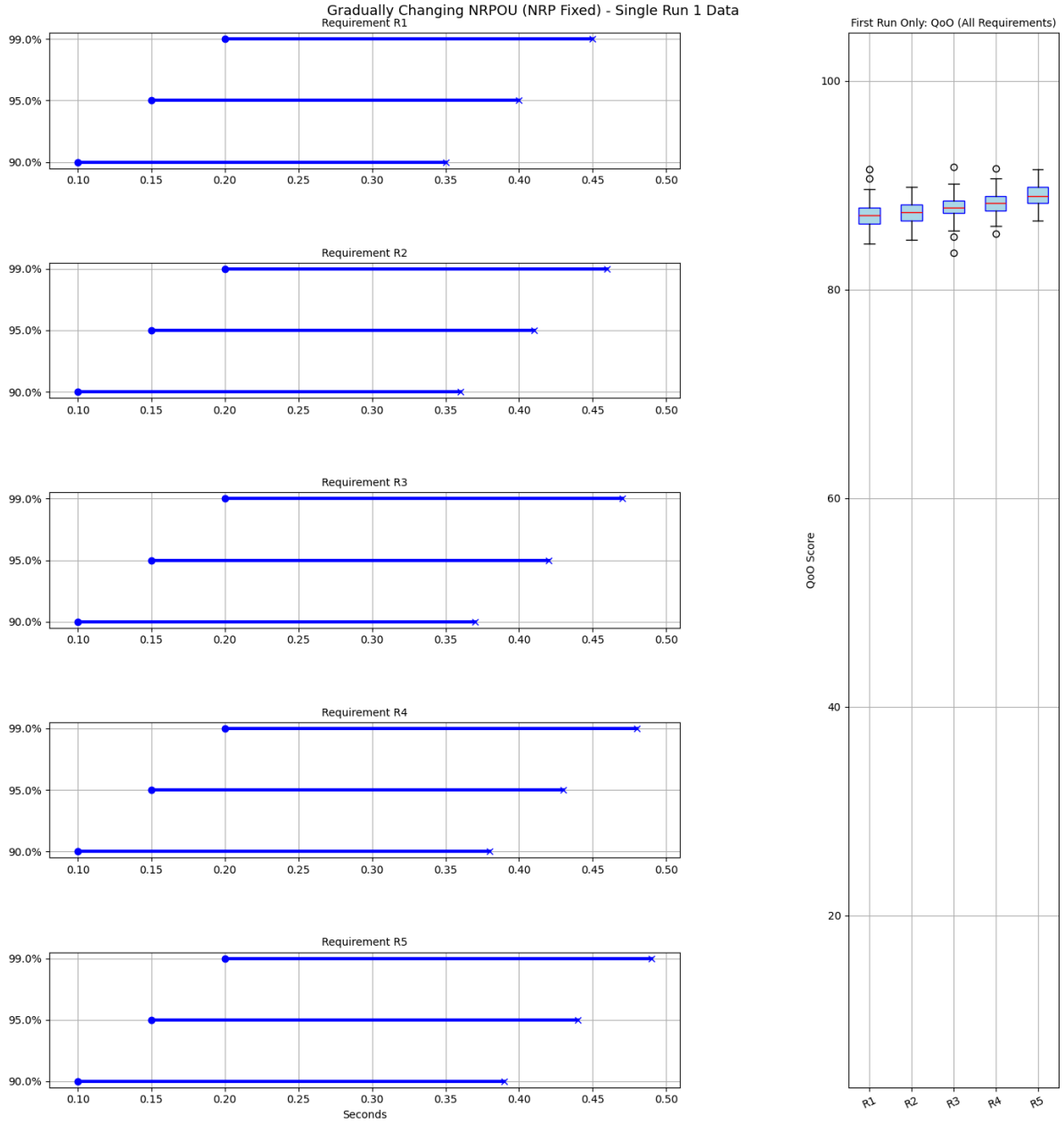
To answer these questions, we run a series of tests with similar but slightly different QoO requirements, and plot their scores for each of our 4 scenarios (WiFi, Short- and long-lived bufferboat and service outage).

Results

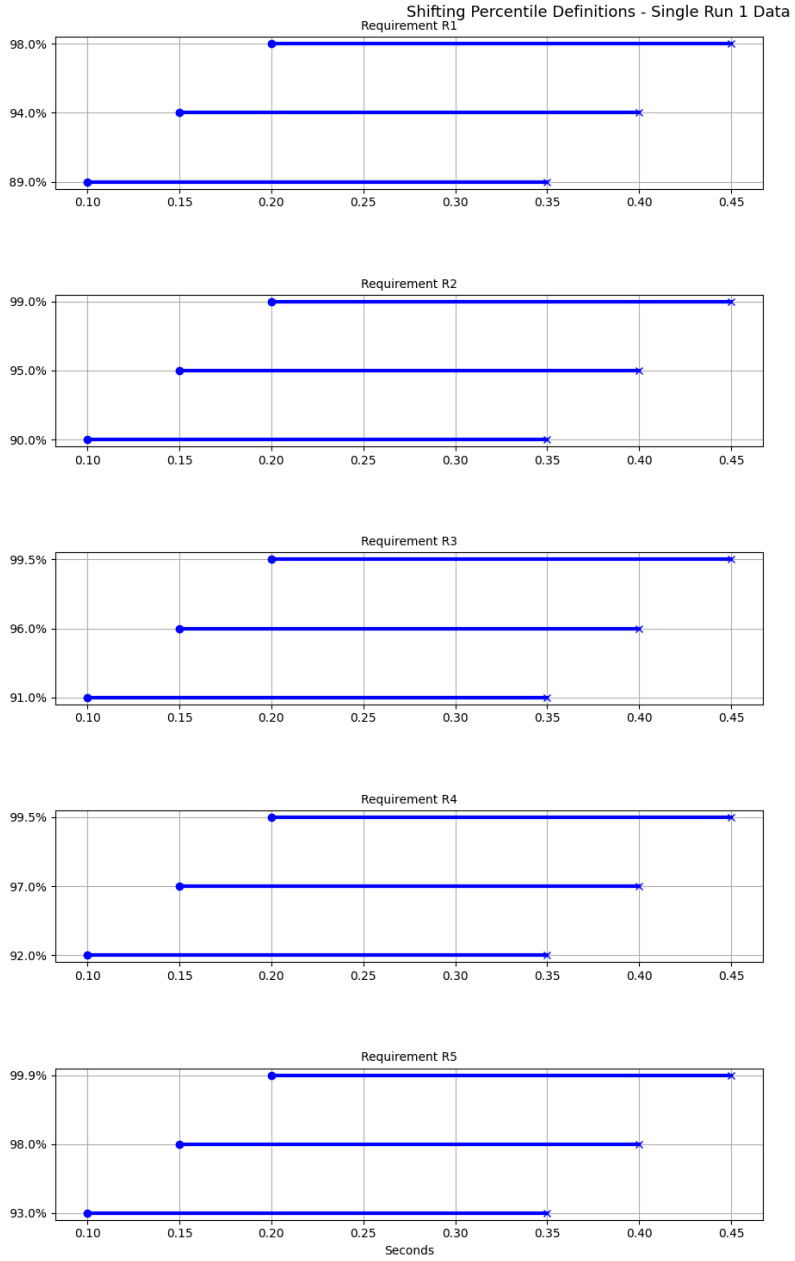
The following plot shows the impact of changing the NRP while keeping the other parts of the network requirements fixed. The plots show a sequence of five different network requirements. We gradually shift the NRP by adding 10 millisecond increments to each of the NRP percentiles, making the requirement less demanding with each step.

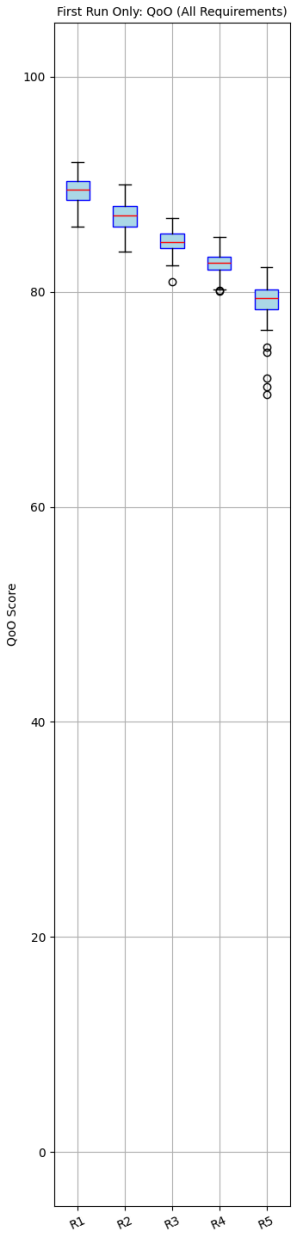


The following plot shows the impact of changing the NRPOU while keeping the other parts of the network requirements fixed. The plots show a sequence of five different network requirements. We gradually shift the NRPOU by adding 10 millisecond increments to each of the NRPOU percentiles, making the requirement less demanding with each step.



The following plot shows the impact of changing the percentiles while keeping the other parts of the network requirements fixed. The plots show a sequence of five different network requirements, where we gradually shift the percentiles upwards, making the requirement increasingly demanding with each step.





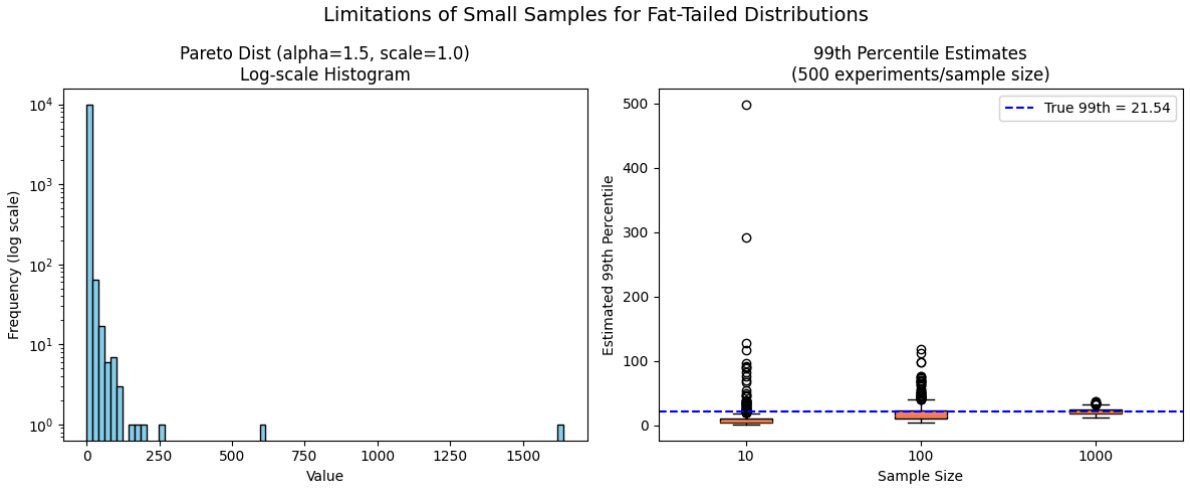
Summary of requirement specification findings

QoO scores can be sensitive to the application requirement specification. Especially the NRP and the choice of percentiles can have a large impact on the resulting QoO scores.

One way to mitigate this is to use a large difference between the NRP and NRPOU values for a given percentile. That way, small changes in the NRP, NRPOU, or offsets to the latency samples, will not have as large an impact on the QoO scores. Nevertheless, these results indicate that care must be taken when application requirements are defined.

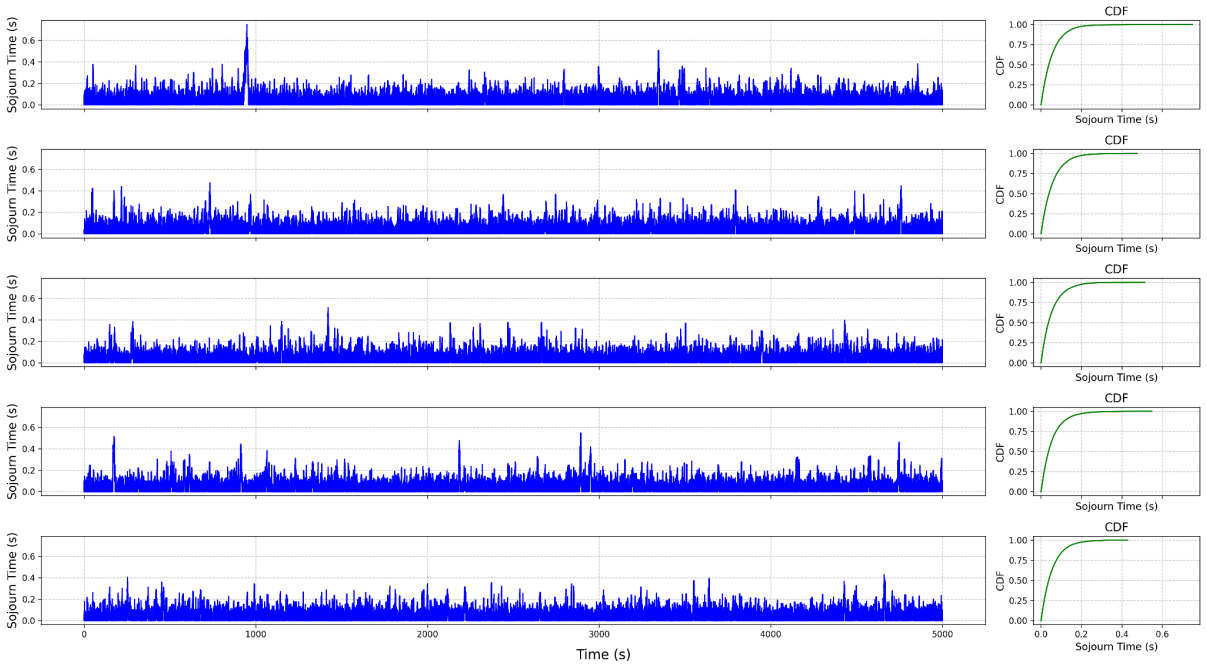
Impact of Measurement Duration (Number of Samples)

Accurately estimating high-percentile values for fat-tailed distributions requires enough samples. The following plot shows an example for the Pareto distribution. The trend is that low-sample-number estimates of the 99th percentile tends to underestimate the real value. This effect might explain why low sampling rates tend to over-estimate the QoO score.

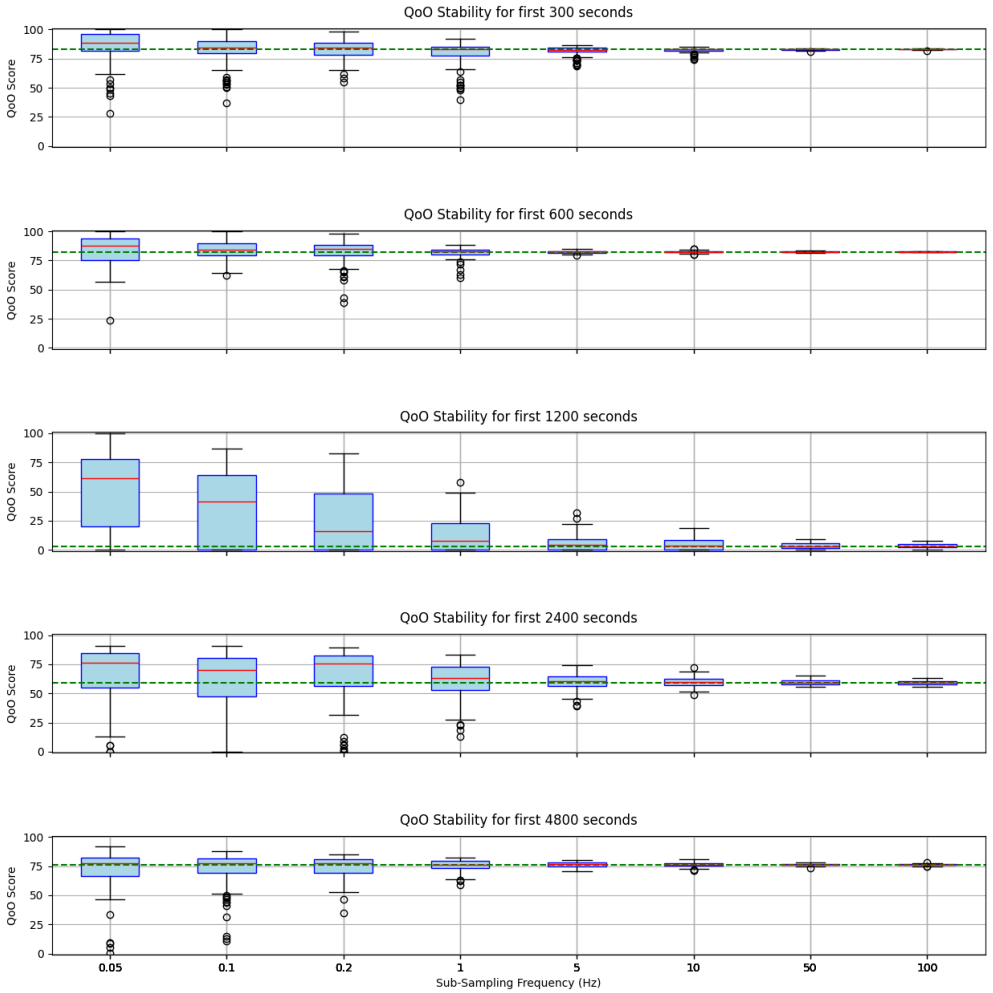


The results in the measurement frequency section assume a fixed measurement interval of 300 seconds. This has the effect that low sampling rates result in few samples, which could contribute to less accurate QoO results. To test this hypothesis, we run a set of longer simulations and evaluate the different sampling rates on these as well. We use the WiFi scenario for this. The only change from the WiFi scenario described above is to increase the duration of the simulation.

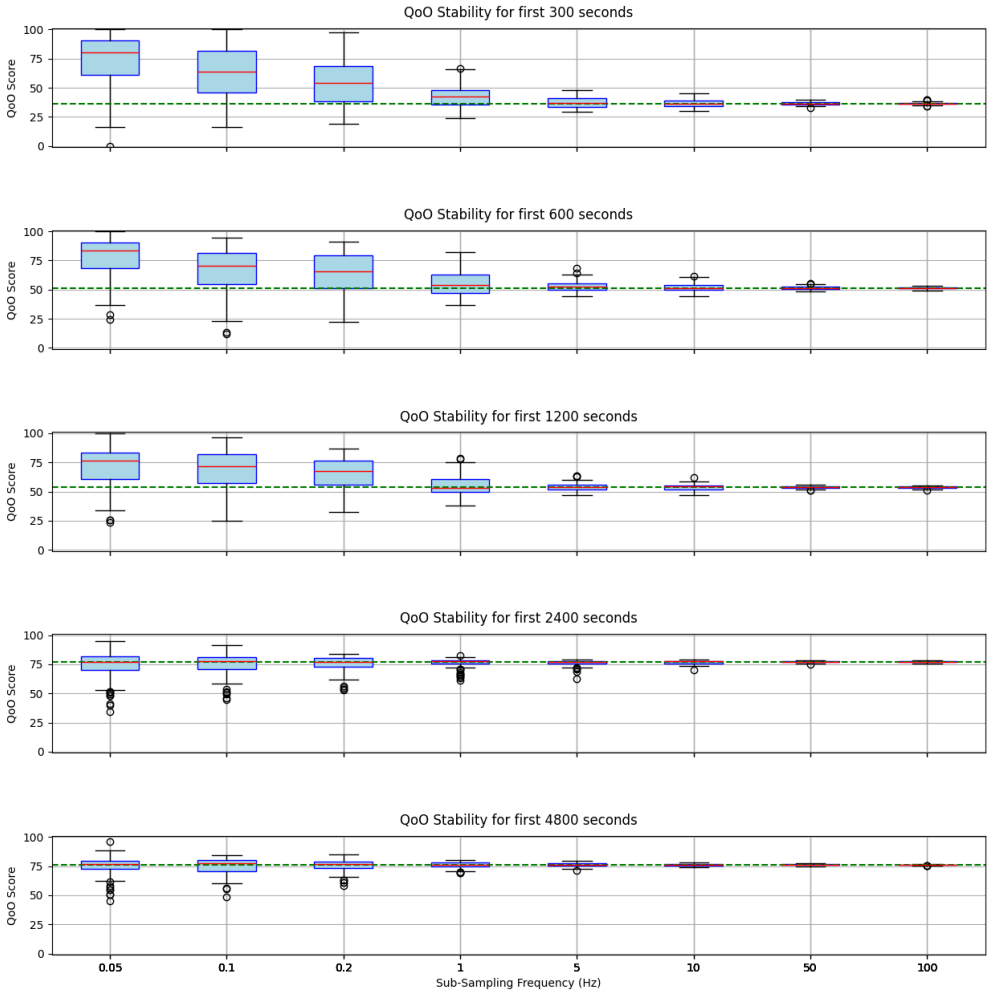
These plots show some examples where we try different sampling *duration*s in addition to different sampling rates. If increasing the sampling duration is a viable way of improving QoO scores, then we should expect the longer-duration measurements to have less variability and higher accuracy. This seems to be the case for runs number 2, 3, 4 and 5, but the result is not entirely consistent. Run number 1 does not show the same effect as clearly. This is likely due to the one short-lived latency spike in the period from 600-1200 seconds causing a sudden change in the QoO. This illustrates that longer-duration sampling does improve the stability of QoO results, but this method is not robust to occasional short-lived latency spikes.


WiFi Simulation Results: Sojourn Times and CDFs
Sojourn Time (s)
CDF
Time (s)
Sojourn Time (s)

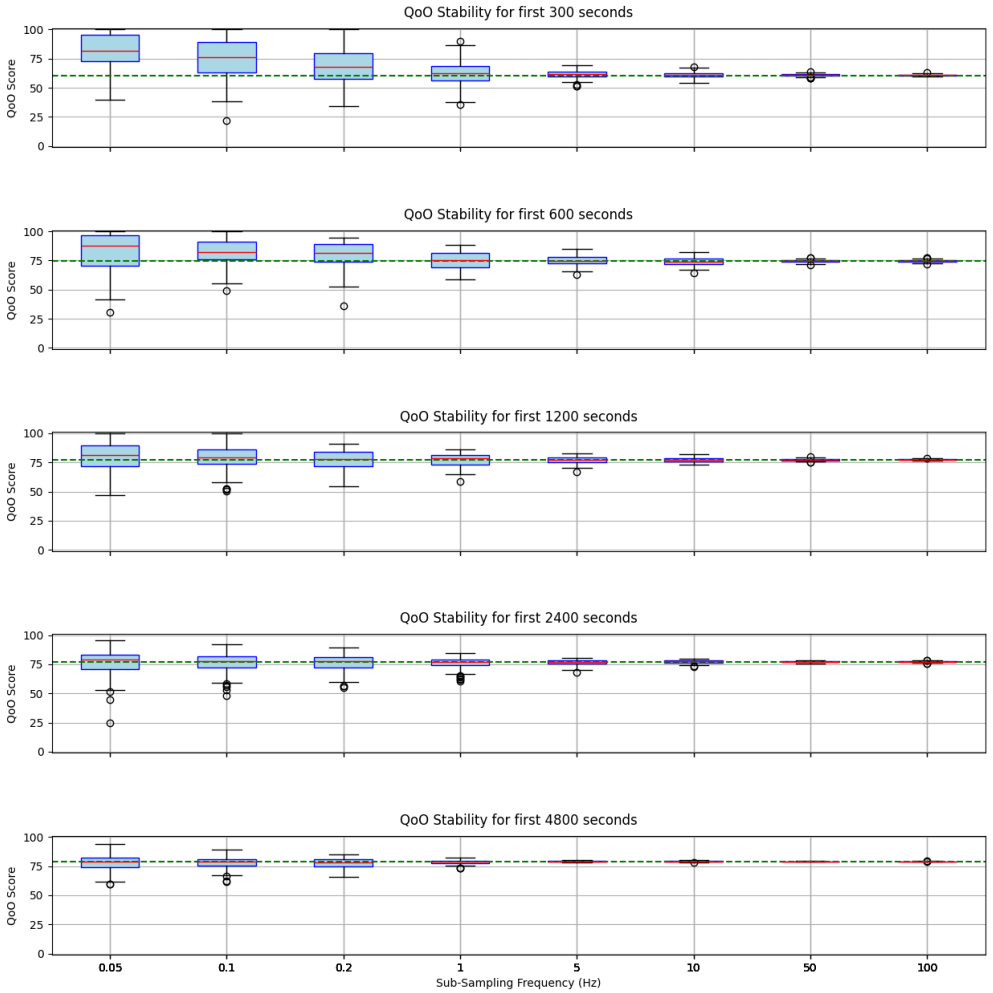
QoO Stability Analysis for wifi_simulation_run_5000_seconds_1.npy



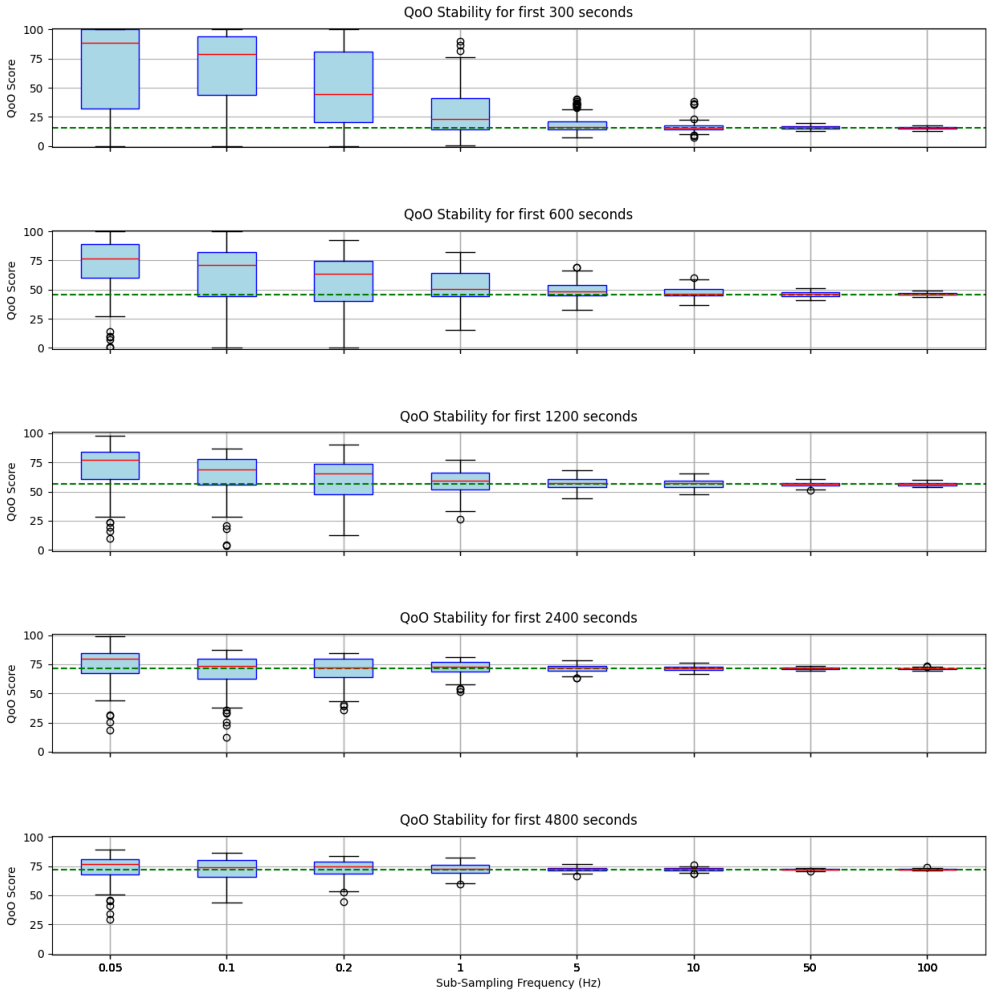
QoO Stability Analysis for wifi_simulation_run_5000_seconds_2.npy

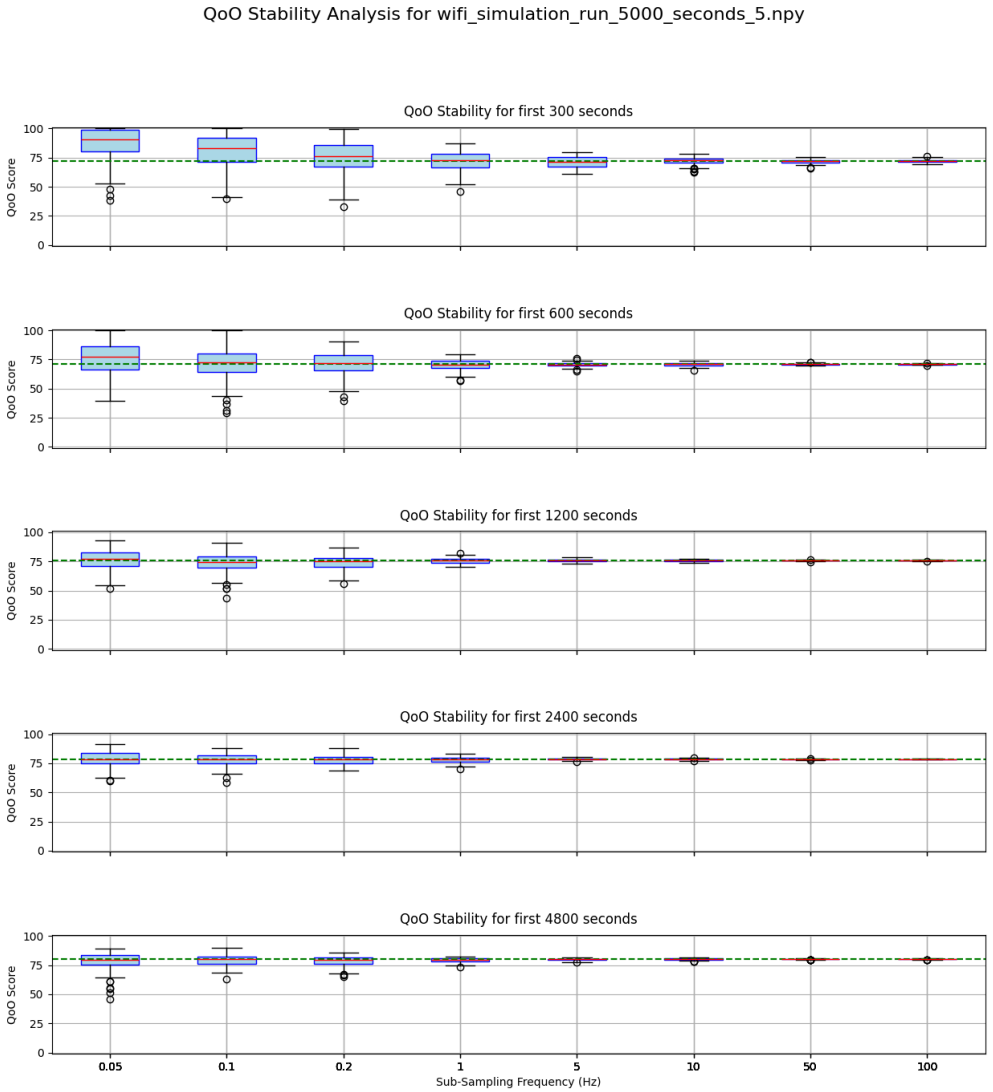


QoO Stability Analysis for wifi_simulation_run_5000_seconds_3.npy



QoO Stability Analysis for wifi_simulation_run_5000_seconds_4.npy





Summary of measurement duration findings

Increasing the measurement duration improves the QoO accuracy in some cases, but not as much as we might expect based on analysis of sampling effects from static fat-tailed distributions. The reason for this is very likely that network latency is “lumpy” in the sense that subsequent latency samples are correlated. This makes it more important to sample often because you never know when a group of outliers will appear. The outliers contain almost all of the relevant information for QoO, and if the sampling “skips over” a period of higher latency, then the QoO score will appear better than it should be.

Limitations

While our simulation-based study offers valuable insights into how sampling frequency, measurement accuracy, and application requirement specifications affect QoO scores, it inevitably abstracts away many real-world complexities. First, the simulator only models a single FIFO queue and does not account for more advanced queue management (e.g., Flow Queuing [FQ], CoDel [CoDel], L4S [L4S]) or multiple queues on different network segments.

Second, real networks often exhibit more complex cross-traffic patterns and additional delays or losses that our relatively simple models do not fully capture.

Third, our "ground truth" time series relies on artificially generated traffic processes (Poisson, fixed-interval, bursty) rather than traces collected from operational networks. This helps us isolate and analyze specific behavior, but it may not reflect the full variability or burstiness found in real deployments.

Fourth, while we incorporate noise in sub-sampling to reflect measurement inaccuracy, we have treated it primarily as uncorrelated, Gaussian noise. In real systems, measurement errors may exhibit temporal or spatial correlations (e.g., clock drift, system jitter), possibly affecting extreme-latency percentiles.

Finally, we have focused on latency as the main input to QoO. Thus our results do not include the impact of packet loss.

Conclusion

"Given the sampling pattern, and the latency distribution observed, what can we conclude about how the application layer will be affected?"

Looking at the results of our simulations, we see that at least three things must be in place before a QoO score can be trusted to represent the underlying reality of the network conditions and application performance.

First, the sampling rate vs. QoO accuracy trade-off must be managed according to the required level of precision in the QoO scores. The sampling rate must be high enough to catch enough of the cases where latency spikes to the levels that trigger a change in the QoO score. Very low sampling rates are likely to result in wrong QoO scores, and the error is biased toward producing overly optimistic results.

Second, the individual latency measurements must be accurate enough not to make problems seem worse than they are. If the measurement results contain large latency spikes caused by measurement error, the resulting QoO score will be lower than it should be.

Third, the application requirement must accurately reflect the actual requirement of the applications the score is supposed to represent. Inaccurate requirements will shift the QoO scores up or down on the scale, and if the stated requirements are far away from the real requirements, the score can become misleading.

References and Links

- **QoO Simulation Code:** https://github.com/domoslabs/qoosim
- **QoO Internet-Draft:** https://datatracker.ietf.org/doc/draft-ietf-ippm-qoo/
- Related Queue Management RFCs
  - [FQ] RFC8290 (Flow Queueing)
  - [CoDel] RFC8289 (CoDel)
  - [L4S] RFC9330 (L4S Architecture)
- [Bufferbloat] Bufferbloat and Beyond
- [WiFi QA] Opportunities and Limitations in Network Quality Optimization