IP Performance Measurement
Internet-Draft
Intended status: Informational
Expires: 5 January 2026

B. I. T. Monclair
M. Olden
CUJO AI

4 July 2025

Quality of Outcome
draft-ietf-ippm-qoo-04

Abstract

This document introduces the Quality of Outcome (QoO) framework, a novel approach to network quality assessment designed to align with the needs of application developers, users, and operators.

By leveraging the Quality Attenuation metric, QoO provides a unified method for defining and evaluating application-specific network requirements while ensuring actionable insights for network optimization and simple quality scores for end-users.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://github.com/getCUJO/QoOID. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-ippm-qoo/.

Discussion of this document takes place on the IP Performance Measurement Working Group mailing list (mailto:ippm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ippm/. Subscribe at https://www.ietf.org/mailman/listinfo/ippm/.

Source for this draft and an issue tracker can be found at https://github.com/getCUJO/QoOID.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

This document introduces the Quality of Outcome network score. Quality of Outcome is a network quality score designed to be easy to understand, while at the same time being objective, adaptable to different network quality needs, and allowing advanced analysis to identify the root cause of network problems. This document defines a network requirement framework that allows application developers to specify their network requirements, along with a way to create a simple user-facing metric based on comparing application requirements to measurements of network performance. The framework builds on Quality Attenuation [TR-452.1], enabling network operators to achieve fault isolation and effective network planning through composability [RFC6049].

Quality Attenuation is a network quality metric that meets most of the design goals set out in the requirements section of this document; it can capture the probability of a network satisfying application requirements, it is composable, and it can be compared to a variety of application requirements. However, interpreting raw Quality Attenuation values is difficult for end-users and application developers, raising the challenge of how to simplify the entailed information without losing too much in terms of precision and accuracy. The part that is yet missing is how to present Quality

Attenuation results to end-users and application developers in an understandable way. A per-application, per application-type, or per-SLA approach is most appropriate here.

Taking a probabilistic approach is key because the network stack and application's network quality adaptation can be highly complex. Applications and the underlying networking protocols make separate optimizations based on their perceived network quality over time and saying something about an outcome with absolute certainty is practically impossible. It is possible, however, to make educated guesses on the probability of outcomes.

This document proposes representing network quality as a minimum required throughput and set of latency and loss percentiles. Application developers, regulatory bodies and other interested parties can describe network requirements in the same manner. This document defines a distance measure between perfect and unusable. With some assumptions, we can use this distance measure to calculate something that can be simplified into statements such as "A Video Conference has a 93% chance of being lag free on this network" all while making it possible to use the framework both for end-to-end test and analysis from within the network.

The work proposes a minimum viable framework, and often trades precision for simplicity. The justification for this is to ensure adoption and usability in many different contexts such as active testing from applications and monitoring from network equipment. To counter the loss of precision, is it necessary to combine measurement results with a description of the measurement approach that allow for analysis of the precision.

2. Overview

Quality of Outcome (QoO) produces simple percentage scores that represent the probability an application will work well on a network. For example: "Video conferencing has a 94% chance of being lag-free on this network."

QoO works by comparing measured network performance against application-specific requirements. Applications define two thresholds:

* Perfect performance (NRP): Network conditions where the app works optimally

* Unusable performance (NRPoU): Network conditions where the app becomes unusable

QoO calculates where measured network performance falls between these thresholds, expressing this as a probability percentage.

The key innovation is using dual thresholds rather than binary pass/fail criteria, providing meaningful scores even when networks aren't perfect while accounting for different application sensitivities.

QoO measurements are mathematically composable, enabling network operators to isolate performance bottlenecks across different network segments for precise fault diagnosis and network planning.

The remainder of this document explains the detailed requirements, mathematical foundations, and implementation considerations for this framework.

3. Terminology

This document introduces several new terms and concepts that are central to the Quality of Outcome framework:

Network: In the context of this document, a network refers to the communication infrastructure that connects endpoints, including all intermediate devices, links, and protocols that affect the transmission of data between a source and destination. This encompasses both the physical infrastructure and the logical protocols that govern data transmission.

Quality Attenuation: A network quality metric that represents packet loss as infinite (or excessively delayed) latency, providing a unified approach to measuring both latency and loss characteristics of network performance [TR-452.1].

Quality of Outcome (QoO): A network quality framework and metric designed to assess network performance based on the probability that applications will achieve successful outcomes. QoO provides a unified approach that serves the needs of end-users, application developers, and network operators.

QoO Score: A numerical value derived from the QoO framework that represents the likelihood of application success on a given network, typically expressed as a percentage.

Network Requirements for Perfection (NRP): The network performance characteristics at which an application achieves optimal performance and user experience.

Network Requirements Point of Unusableness (NRPoU): The network performance threshold below which an application becomes unusable or fails to provide acceptable user experience.

Composability: The mathematical property that allows network quality measurements to be combined across different network segments or decomposed to isolate specific network components for analysis and troubleshooting.

Accuracy and Precision: In this document, "accuracy" refers to how close measurements are to the true value, while "precision" refers to the consistency and repeatability of measurements. These terms are used with their standard statistical meanings and are not interchangeable.

4. Motivation

This section describes the features and attributes a network quality framework must have to be useful for different stakeholders: application developers, end-users, and network operators/vendors. At a high level, end-users need an understandable network metric. Application developers require a network metric that allows them to evaluate how well their application is likely to perform given the measured network performance. Network operators and vendors need a metric that facilitates troubleshooting and optimization of their networks. Existing network quality metrics and frameworks typically address the needs of one or two of these stakeholders, but the authors have yet to find one that bridges the needs of all three. Examples include throughput metrics that operators use to prove regulatory compliance but with little relevance to application performance, or subjective Quality of Experience (QoE) metrics that are understandable to users but difficult for operators to collect at meaningful scale.

A key motivation for the Quality of Outcome (QoO) framework is to bridge the gap between the technical aspects of network performance and the practical needs of those who depend on it. While solutions exist for many of the problems causing high and unstable latency in the Internet, such as bufferbloat mitigation techniques [RFC8290] and improved congestion control algorithms [RFC8033], the incentives to deploy them have remained relatively weak. A unifying framework for assessing network quality, can serve to strengthen these incentives significantly.

Bandwidth alone is necessary but not sufficient for high-quality modern network experiences. Idle latency, working latency, jitter, and unmitigated packet loss are major causes of poor application outcomes. The impact of latency is widely recognized in network

engineering circles [BITAG], but benchmarking the quality of network transport remains complex. Most end-users are unable to relate to metrics other than Mbps, which they have long been conditioned to think of as the only dimension of network quality.

Real Time Response under load tests [RRUL] and Responsiveness [RPM] make significant strides toward creating a network quality metric that is far closer to application outcomes than bandwidth alone. The latter, in particular, is successful at being relatively relatable and understandable to end-users. However, as noted in [RPM], "Our networks remain unresponsive, not from a lack of technical solutions, but rather a lack of awareness of the problem." This lack of awareness means operators have little incentive to improve network quality beyond increasing bandwidth. Despite the availability of open-source solutions, vendors rarely implement them. A universally accepted network quality framework that successfully captures how well applications are likely to perform may help to increase the willingness of vendors to implement such solutions.

A recent IAB workshop on measuring internet quality for end-users identified a key insight: users care primarily about application performance rather than network performance. Among the conclusions was the statement, "A really meaningful metric for users is whether their application will work properly or fail because of a lack of a network with sufficient characteristics" [RFC9318]. Therefore, one critical requirement of a meaningful framework is its ability to answer the question, "Will networking conditions stop an application from working properly?"

Answering this question requires several considerations. First, the Internet is inherently stochastic from the perspective of any given client, so absolute certainty is unattainable. Second, different applications have varying needs and adapt differently to network conditions. A framework aiming to answer this question must accommodate such diverse application requirements. Third, end-users have individual tolerances for degradation in network conditions and the resulting effects on application experience. These variations must be factored into the framework’s design.

4.1. Design Goal

The overall goal of this document is to describe the requirements for an objective network quality framework and metric that is useful for end-users, application developers, and network operators/vendors alike.

4.2. Requirements

This section outlines the three main requirements and their motivation.

In general, all stakeholders ultimately care about the success of applications running over the network. Application success depends not just on bandwidth but also on the delay of network links and computational steps involved in making the application function. These delays depend on how the application places load on the network, how the network is affected by environmental conditions, and the behavior of other users sharing the network resources.

Different applications have different needs from the network, and they place different patterns of load on it. To determine whether applications will work well or fail, a network quality framework must compare measurements of network performance to a wide variety of application requirements. Flexibility in describing application requirements and the ability to capture the delay characteristics of the network in sufficient detail are necessary to compute application success with satisfactory accuracy and precision.

How can operators take action when measurements show that applications fail too often? The framework must support spatial composition [RFC6049], [RFC6390] to answer this question. Spatial composition allows results to be divided into sub-results, each measuring the performance of a required sub-milestone that must be reached in time for the application to succeed.

To summarize, the QoO framework and "meaningful QoO metric" should have the following properties:

1. *Capture a set of network performance metrics which provably correlate to the application quality of a set of different applications as perceived by users.* (Useful for end-users and application developers.)

2. *Compare meaningfully to different application requirements.*

3. *Compose.* Allow operators to isolate and quantify the contributions of different sub-outcomes and sub-paths of the network. (Useful for operators and vendors.)

4.2.1. Requirements for end-users

The quality framework should facilitate a metric that is based on objective QoS measurements, correlated to application quality and relatively understandable for an end-user. A middle ground between objective QoS metrics (Throughput, packet loss, jitter, mean latency) and subjective but understandable QoE metrics (MOS, 5-star ratings). The ideal framework should be objective, like QoS metrics, and understandable, like QoE metrics.

If these requirements are met, the end-user can understand if a network is a likely source of impairment for what they care about: the outcomes of applications. Examples are how quickly a web page loads, the smoothness of a video conference, or whether or not a video game has any lag.

Each end-user will have an individual tolerance of session quality, below which their quality of experience becomes personally unacceptable. However it may not be feasible to capture and represent these tolerances _per user_ as the user group scales. A compromise is for the quality of experience framework to place the responsibility for sourcing and representing end-user requirements onto the application developer. Application developers should perform user-acceptance testing (UAT) of their application across a range of users, terminals and network conditions to determine the terminal and network requirements that will meet the end-user quality threshold for an acceptable subset of their end-users. Some real world examples where ’acceptable levels’ have been derived by application developers include (note: developers of similar applications may have arrived at different figures):

* Remote music collaboration: <20ms one-way latency [JamKazam]

* Online gaming: 6Mb/s downlink throughput and 30ms RTT to join a multiplayer game (based on requirements for popular online games)

* Virtual reality: <20ms RTT from head motion to rendered update in VR (based on human perception studies for VR applications)

Performing this UAT helps the developer understand what likelihood a new end-user has of an acceptable Quality of Experience based on the application’s existing requirements towards the network. These requirements can evolve and improve based on feedback from end-users, and in turn better inform the application’s requirements towards the network.

4.2.2. Requirements from Application and Platform Developers

The framework needs to give developers the ability to describe the network requirements of their applications. The format for specifying network requirements must include all relevant dimensions of network quality so that different applications which are sensitive to different network quality dimensions can all evaluate the network accurately. Not all developers have network expertise, so to make it easy for developers to use the framework, developers must be able to specify network requirements approximately. Therefore, it must be possible to describe both simple and complex network requirements. The framework also needs to be flexible so that it can be used with different kinds of traffic and that extreme network requirements which far exceed the needs of today's applications can also be articulated.

If these requirements are met, developers of applications or platforms can state or test their network requirements and evaluate if the network is sufficient for a great application outcome. Both the application developers with networking expertise and those without can use the framework.

4.2.3. Requirements for Network Operators and Network Solution Vendors

From an operator perspective, the key is to have a framework that lets operators find the network quality bottlenecks and objectively compare different networks and technologies. The framework must support mathematically sound compositionality ('addition' and 'subtraction') to achieve this. Why? Network operators rarely manage network traffic end-to-end. If a test is purely end-to-end, the ability to find bottlenecks may be gone. If, however, measurements can be taken both end-to-end (e.g., a-b-c-d-e) and not-end-to-end (e.g., a-b-c), the results can be subtracted to isolate the areas outside the influence of the network operator. In other words, the network quality of a-b-c and d-e can be separated. Compositionality is essential for fault detection and accountability.

By having mathematically correct composition, a network operator can measure two segments separately, perhaps even with different approaches, and add them together to understand the end-to-end network quality.

For another example where composition is useful, look at a typical web page load sequence. If web page load times are too slow, DNS resolution time, TCP round-trip time, and the time it takes to establish TLS connections can be measured separately to get a better idea of where the problem is. A network quality framework should support this kind of analysis to be maximally useful for operators.

The quality framework must be applicable in both lab testing and monitoring of production networks. It must be useful on different time scales, and it can't have a dependency on network technology or OSI layer.

If these requirements are met, a network operator can monitor and test their network and understand where the true bottlenecks are, regardless of network technology.

5. Background

The foundation of the framework is Quality Attenuation [TR-452.1]. This work will not go into detail about how to measure Quality Attenuation, but some relevant techniques are:

* Active probing with TWAMP Light [RFC5357] / STAMP [RFC8762] / IRTT [IRTT]
* Latency Under Load Tests
* Speed Tests with latency measures
* Simulating real traffic
* End-to-end measurements of real traffic
* TCP SYN ACK / DNS Lookup RTT Capture
* On-Path Telemetry methods (IOAM, AltMark)
* Estimation

Quality Attenuation represents quality measurements as distributions. Using Latency distributions to measure network quality is nothing new and has been proposed by various researchers/practitioners [Kelly][RFC8239][RFC6049]. Similar to the One-Way Loss Metric for IP Performance Metrics (IPPM) [RFC7680], which defines packet loss in terms of packets that fail to arrive within a specified time threshold, the Quality Attenuation approach extends this concept by viewing packet loss as infinite (or too late to be of use e.g. > 5 seconds) latency [TR-452.1]. The novelty of TR-452.1 lies in its unified treatment of latency and loss within a single distributional framework, enabling mathematical composition of network segments.

Latency Distributions can be gathered via both passive monitoring and active testing. The active testing can use any type of traffic, such as TCP, UDP, or QUIC. It can be applied across different layers of the protocol stack and is network technology independent, meaning it

can be gathered in an end-user application, within some network equipment, or anywhere in between. Passive methods rely on observing and time-stamping packets traversing the network. Examples of this include TCP SYN and SYN/ACK packets and the QUIC spin bit.

A key assumption behind the choice of latency distribution is that different applications and application categories fail at different points of the latency distribution. Some applications, such as downloads, have lenient latency requirements when compared to real-time application. Video Conferences are typically sensitive to high 90th percentile latency and to the difference between the 90th and the 99th percentile. Online gaming typically has a low tolerance for high 99th percentile latency. All applications require a minimum level of throughput and a maximum packet loss rate. A network quality metric that aims to generalize network quality must take the latency distribution, throughput, and packet loss into consideration.

Two distributions can be composed using convolution [TR-452.1].

5.1. Discussion of other performance metrics

Numerous network performance metrics and associated frameworks have been proposed, adopted, and, at times, misapplied over the years. The following is a brief overview of several key network quality metrics.

Each metric is evaluated against the three criteria established in the requirements section. Throughput is related to user-observable application outcomes because there must be _enough_ bandwidth available. Adding extra bandwidth above a certain threshold will, at best, receive diminishing returns (and any returns are often due to reduced latency). It is not possible to compute the probability of application success or failure based on throughput alone for most applications. Throughput can be compared to a variety of application requirements, but since there is no direct correlation between throughput and application performance, it is not possible to conclude that an application will work well even if it is known that enough throughput is available.

Throughput cannot be composed.

5.1.1. Mean Latency

Mean latency relates to user-observable application outcomes in the sense that the mean latency must be low enough to support a good experience. However, it is not possible to conclude that a general application will work well based on the fact that the mean latency is good enough [BITAG].

Mean latency can be composed. If the mean latency of links a-b and b-c is known, then the mean latency of the composition a-b-c is the sum of a-b and b-c.

5.1.2. 99th Percentile of Latency

The 99th percentile of latency relates to user-observable application outcomes because it captures some information about how bad the tail latency is. If an application can handle 1% of packets being too late, for instance by maintaining a playback buffer, then the 99th percentile can be a good metric for measuring application performance. It does not work as well for applications that are very sensitive to overly delayed packets because the 99th percentile disregards all information about the delays of the worst 1% of packets.

It is not possible to compose 99th-percentile values.

5.1.3. Variance of latency

The variance of latency can be calculated from any collection of samples, but network latency is not necessarily normally distributed, and so it can be difficult to extrapolate from a measure of the variance of latency to how well specific applications will work.

The variance of latency can be composed. If the variance of links a-b and b-c is known, then the variance of the composition a-b-c is the sum of the variances a-b and b-c.

5.1.4. Inter-Packet Delay Variation (IPDV)

The most common definition of IPDV [RFC5481] measures the difference in one-way delay between subsequent packets. Some applications are very sensitive to this because of time-outs that cause later-than-usual packets to be discarded. For some applications, IPDV can be useful in assessing application performance, especially when it is combined with other latency metrics. IPDV does not contain enough information to compute the probability that a wide range of applications will work well.

IPDV cannot be composed.

5.1.5. Packet Delay Variation (PDV)

The most common definition of PDV [RFC5481] measures the difference in one-way delay between the smallest recorded latency and each value in a sample.

PDV cannot be composed.

5.1.6. Trimmed Mean of Latency

The trimmed mean of latency is the mean computed after the worst x percent of samples have been removed. Trimmed means are typically used in cases where there is a known rate of measurement errors that should be filtered out before computing results.

In the case where the trimmed mean simply removes measurement errors, the result can be composed in the same way as the mean latency. In cases where the trimmed mean removes real measurements, the trimming operation introduces errors that may compound when composed.

5.1.7. Round-trips Per Minute

Round-trips per minute [RPM] is a metric and test procedure specifically designed to measure delays as experienced by application-layer protocol procedures such as HTTP GET, establishing a TLS connection, and DNS lookups. It, therefore, measures something very close to the user-perceived application performance of HTTP-based applications. RPM loads the network before conducting latency measurements and is, therefore, a measure of loaded latency (also known as working latency) well-suited to detecting bufferbloat [Bufferbloat].

RPM is not composable.

5.1.8. Quality Attenuation

Quality Attenuation is a network performance metric that combines latency and packet loss into a single variable [TR-452.1].

Quality Attenuation relates to user-observable outcomes in the sense that user-observable outcomes can be measured using the Quality Attenuation metric directly, or the Quality Attenuation value describing the time-to-completion of a user-observable outcome can be computed if the Quality Attenuation of each sub-goal required to reach the desired outcome are known [Haeri22].

Quality Attenuation is composable because the convolution of Quality Attenuation values allows us to compute the time it takes to reach specific outcomes given the Quality Attenuation of each sub-goal and the causal dependency conditions between them [Haeri22].

5.1.9. Summary of performance metrics

This table summarizes the properties of each of the metrics surveyed.

The column "Capture probability of general applications working well" records whether each metric can, in principle, capture the information necessary to compute the probability that a general application will work well, assuming measurements capture the properties of the end-to-end network path that the application is using.

Metric	Capture probability of general applications working well	Easy to articulate Application requirements	Composable
Mean latency	Yes for some applications	Yes	Yes
Variance of latency	No	No	Yes
IPDV	Yes for some applications	No	No
PDV	Yes for some applications	No	No
Mean Peak Throughput	Yes for some applications	Yes	No
99th Percentile of Latency	No	No	No
Trimmed mean of latency	Yes for some applications	Yes	No
Round Trips Per Minute	Yes for some applications	Yes	No
Quality Attenuation	Yes	No	Yes

Table 1

Explanations:

* "Captures probability" refers to whether the metric captures enough information to compute the likelihood of an application succeeding.

* "Articulate requirements" refers to the ease with which application-specific requirements can be expressed using the metric.

* "Composable" means whether the metric supports mathematical composition to allow for detailed network analysis.

6. Sampling and Network Requirements

6.1. Sampling requirements

To ensure broad applicability across diverse use cases, this framework deliberately avoids prescribing specific conditions for sampling such as fixed time intervals or defined network load levels. This flexibility enables deployment in both controlled and real-world environments.

At its core, the framework requires only a latency distribution. When measurements are taken during periods of network load, the result naturally includes latency under load. In scenarios such as passive monitoring of production traffic, capturing artificially loaded conditions may not always be feasible, whereas passively observing the actual network load may be possible.

Modeling the full latency distribution may be too complex to allow for easy adoption of the framework, and reporting latency at selected percentiles offers a practical compromise between accuracy and deployment considerations. A commonly accepted set of percentiles spanning from the 0th to the 100th in a logarithmic-like progression has been suggested by others [BITAG] and is recommended here: [0th, 10th, 25th, 50th, 75th, 90th, 95th, 99th, 99.9th, 100th].

The framework is agnostic to traffic direction but mandates that measurements specify whether latency is one-way or round-trip.

Importantly, the framework does not enforce a minimum sample count. This means that even a small number of samples (e.g., 10) could technically constitute a distribution— though such cases are clearly insufficient for statistical confidence. The intent is to balance rigor with practicality, recognizing that constraints vary across devices, applications, and deployment environments.

To support reproducibility and enable confidence analysis, each measurement must be accompanied by the following metadata:

* Description of the measurement path
* Timestamp of first sample
* Total duration of the sampling period
* Number of samples collected
* Sampling method, including:
  - Cyclic: One sample every N milliseconds (specify N)
  - Burst: X samples every N milliseconds (specify X and N)
  - Passive: Opportunistic sampling of live traffic (non-uniform intervals)

These metadata elements are essential for interpreting the precision and reliability of the measurements. As demonstrated in [QoOSimStudy], low sampling frequencies and short measurement durations can lead to misleadingly optimistic or imprecise Quality of Outcome (QoO) scores.

6.2. Describing Network Requirements

This work builds upon the work already proposed in the Broadband Forum standard called Quality of Experience Delivered (QED/TR-452) [TR-452.1], which defines the Quality Attenuation metric. In essence, QoO describes network requirements as a list of percentile and latency requirement tuples. In other words, a network requirement may be expressed as: The network requirement for this app quality level/app/app category/SLA is "at 4 Mbps, 90% of packets needs to arrive within 100 ms, 100% of packets needs to arrive within 200ms". This list can be as simple as "100% of packets need to arrive within 200ms" or as long as you would like. For the sake of simplicity, the requirements percentiles must match one or more of the percentiles defined in the measurements, i.e., one can set requirements at the [0th, 10th, 25th, 50th, 75th, 90th, 95th, 99th, 99.9th, 100th] percentiles. Packet loss must be reported as a separate value.

Applications do of course have throughput requirements, and thus a complete framework for application-level network quality must also take capacity into account. Insufficient bandwidth may give poor application outcomes without necessarily inducing a lot of latency.

Therefore, the network requirements should include a minimum throughput requirement. A fully specified requirement can be thought of as specifying the latency and loss requirements to be met while the end-to-end network path is loaded in a way that is at least as demanding of the network as the application itself. This may be achieved by running the actual application and measuring delay and loss alongside it, or by generating artificial traffic to a level at least equivalent to the application traffic load.

Whether the requirements are one-way or two-way must be specified. Where the requirement is one-way, the direction (uplink or downlink) must be specified. If two-way, a decomposition into uplink and downlink measurements may be specified.

Until now, network requirements and measurements are what is already standardized in the BBF TR-452 (aka QED) framework [TR-452.1]. The novel part of this work is what comes next. A method for going from Network Requirements and Network Measurements to probabilities of good application outcomes.

To do that it is necessary to make articulating the network requirements a little bit more complicated. A key design goal was to have a distance measure between perfect and unusable, and have a way of quantifying what is ’better’.

The requirements specification is extended to include the quality required for perfection and a quality threshold beyond which the application is considered unusable.

This is named Network Requirements for Perfection (NRP). As an example: At 4 Mbps, 99% of packets need to arrive within 100ms, 99.9% within 200ms (implying that 0.1% packet loss is acceptable) for the outcome to be perfect. Network Requirements Point of Unusability (NRPoU): If 99% of the packets have not arrived after 200ms, or 99.9% within 300ms, the outcome will be unusable.

Where the NRPoU percentiles and NRP are a required pair then neither should define a percentile not included in the other - i.e., if the 99.9th percentile is part of the NRPoU then the NRP must also include the 99.9th percentile.

The derivation of NRP and NRPoU values requires standardized testing conditions to ensure consistency and accuracy. Application developers should publish their testing methodologies, including the network conditions, hardware configurations, and measurement procedures used to establish these thresholds. Without such standardization, the overall accuracy and precision of QoO metrics may be reduced due to variations in testing approaches across different applications and developers.

6.3. Creating network requirement specifications

A detailed description of how to create a network requirement specification is out of scope for this document, but this section will provide a rough outline for how it can be achieved. Additional information about this topic can be found in [QoOAppQualityReqs].

When searching for an appropriate network requirement description for an application, the goal is to identify the points of perfection and uselessness for the application. This can be thought of as a search process. Run the application across a network connection with adjustable quality. Gradually adjust the network performance while observing the application-level performance. The application performance can be observed manually by the person performing the testing, or using automated methods such as recording video stall duration from within a video player.

Establish a baseline under excellent network conditions. Then gradually add delay, packet loss or decrease network capacity until the application no longer performs perfectly. Continue adding network Quality Attenuation until the application fails completely. The corresponding network quality levels are the points of perfection and unusability.

7. Calculating Quality of Outcome (QoO)

The QoO metric calculates the likelihood of application success based on network performance, incorporating both latency and packet loss. There are three key scenarios:

* The network meets all the requirements for perfection. Probability of success: 100%.

* The network fails one or more criteria at the Point of Unusableness (NRPoU). Probability of success: 0%.

* The network performance falls between perfection and unusable. In this case, a QoO score is computed. The QoO score is calculated by taking the worst score derived from latency and packet loss.

Latency Component The latency-based QoO score is computed as follows:

QoO_latency = min_{i}(min(max((1 - ((ML_i - NRP_i) / (NRPoU_i - NRP_i))) * 100, 0), 100))

Where:

* ML_i is the Measured Latency at percentile i.

* NRP_i is the Network Requirement for Perfection at percentile i.

* NRPoU_i is the Network Requirement Point of Unusableness at percentile i.

Packet Loss Component Packet loss is considered as a separate, single measurement that applies across the entire traffic sample, not at each percentile. The packet loss score is calculated using a similar interpolation formula, but based on the total measured packet loss (MLoss) and the packet loss thresholds defined in the NRP and NRPoU:

QoO_loss = min(max((1 - ((MLoss - NRP_Loss) / (NRPoU_Loss - NRP_Loss))) * 100, 0), 100)

Where:

* MLoss is the Measured Packet Loss.

* NRP_Loss is the acceptable packet loss for perfection.

* NRPoU_Loss is the packet loss threshold beyond which the application becomes unusable.

Final QoO Calculation The overall QoO score is the minimum of the latency and packet loss scores:

QoO = min(QoO_latency, QoO_loss)

Example Requirements and Measured Data:

* NRP: 4 Mbps {99%, 250 ms, 0.1% loss}, {99.9%, 350 ms, 0.1% loss}

* NRPoU: {99%, 400 ms, 1% loss}, {99.9%, 401 ms, 1% loss}

* Measured Latency: 99% = 350 ms, 99.9% = 352 ms

* Measured Packet Loss: 0.5%

* Measured Minimum Bandwidth: 32 Mbps / 28 Mbps

The QoO calculation proceeds as follows:

Latency Component: QoO_latency = min((min(max((1 - (350 ms - 250 ms) / (400 ms - 250 ms)) * 100, 0), 100), (min(max((1 - (352 ms - 350 ms) / (401 ms - 350 ms)) * 100, 0), 100))) = min(33.33, 96.08) = 33.33

Packet Loss Component: QoO_loss = min(max((1 - (0.5% - 0.1%) / (1% - 0.1%)) * 100, 0), 100) = 55.56

Final QoO Calculation: QoO = min(33.33, 55.56) = 33.33

In this example, the application has a 33% chance of meeting the quality expectations on this network, considering both latency and packet loss.

Implementation Note: Sensitivity to Sampling Accuracy

Based on the simulation results in [QoOSimStudy], overly noisy or inaccurate latency samples can artificially inflate worst-case percentiles, thereby driving QoO scores lower than actual network conditions would warrant. Conversely, coarse measurement intervals can miss short-lived spikes entirely, resulting in an inflated QoO. Users of this framework should consider hardware/software measurement jitter, clock offset, or other system-level noise sources when collecting data, and configure sampling frequency and duration to suit their specific needs.

8. How to find network requirements

A key advantage of a measurement that spans the range from perfect to unusable, rather than relying on binary (Good/Bad) or other low-resolution (Terrible/Bad/OK/Great/Excellent) metrics, is the flexibility it provides. For example, a chance of lag-free experience below 20% is intuitively undesirable, while a chance above 90% is intuitively favorable—demonstrating that absolute perfection is not required for the QoO metric to be meaningful.

However, it remains necessary to define points representing unusableness and perfection. There is no universally strict threshold at which network conditions render an application unusable. For perfection, some applications may have clear definitions, but for others, such as web browsing and gaming, lower latency is always preferable. To assist in establishing requirements, it is recommended that the Network Requirements for Perfection (NRP) be set at the point where further reductions in latency do not result in a perceivable improvement in end-user experience.

Someone who wishes to make a network requirement for an application in the simplest possible way, should do something along these lines:

* Simulate increasing levels of latency
* Observe the application and note the threshold where the application stops working perfectly
* Observe the application and note the threshold where the application stops being useful at all

Someone who wishes to find sophisticated network requirements might proceed in this way:

* Set thresholds for acceptable fps, animation fluidity, i/o latency (voice, video, actions), or other metrics capturing outcomes that directly affects the user experience
* Create a tool for measuring these user-facing metrics
* Simulate varying latency distribution with increasing levels of latency while measuring the user facing metrics

Note that when observing application performance, it’s important to recognize that different users may have different tolerance levels for application degradation. The thresholds established should represent acceptable performance for the target user base, which may require user studies or market research to determine appropriate values.

A QoO score at 94 can be communicated as "John’s smartphone has a 94% chance of lag-free Video Conferencing", however, this does not mean that at any point of time there is a 6% chance of lag. It means there is a 6% chance of experiencing lag during the entire session/ time-period, and the network requirements should be adjusted accordingly.

The reason for making the QoO metric for a session is to make it understandable for an end-user, an end-user should not have to relate to the time period the metric is for.

8.1. An example

Example.com’s video-conferencing service requirements can be translated into the QoO Framework. For best performance for video meetings, they specify 4/4 Mbps, 100 ms latency, <1% packet loss, and <30 ms jitter. This can be translated to an NRP:

NRP example.com video conferencing service: At minimum 4/4 Mbps. {0p=70ms,99p=100ms}

For minimum requirements example.com does not specify anything, but at 500ms latency or 1000ms 99p latency, a video conference is very unlikely to work in a remotely satisfactory way.

NRPoU {0p=500,99p=1000ms}

Of course, it is possible to specify network requirements for Example.com with multiple NRP/NRPoU, for different quality levels, one/two way video, and so on. Then one can calculate the QoO at each level.

8.2. Adaptive Applications

Many modern applications are adaptive, meaning they can adjust their behavior based on network conditions. For example, video streaming applications may reduce bit rate when bandwidth is limited, or increase buffer size when latency is high.

For adaptive applications, there are typically different levels of "perfection" rather than a single absolute threshold. A video streaming application might provide:

* Excellent quality: 4K resolution, low latency

* Good quality: 1080p resolution, moderate latency

* Acceptable quality: 720p resolution, higher latency

* Minimum quality: 480p resolution, high latency

The QoO framework can accommodate this by defining multiple NRP (Network Requirements for Perfection) thresholds corresponding to different quality levels. The framework can then compute the probability that the application will achieve each quality level, providing a more nuanced view of application performance than a simple binary success/failure metric. Another, less complex approach at the cost of reduced fidelity in the QoO score, is to set the threshold for perfection at where Excellent quality can be delivered, and the threshold for unusability where not even minimum quality can be delivered. This approach assumes that the app can perform adaptations without frustrating the users, for instance that quality level can be lowered without significant video or audio interruptions.

Application developers implementing adaptive applications should consider publishing quality profiles that define network requirements for different adaptation levels, enabling more accurate QoO assessment.

9. Insights

9.1. Insights from Simulation Results

While the QoO framework itself places no strict requirement on sampling patterns or measurement technology, a recent simulation study [QoOSimStudy] examined the metric's real-world applicability under varying conditions of:

1. *Sampling Frequency*: Slow sampling rates (e.g., <1 Hz) risk missing rare, short-lived latency spikes, resulting in overly optimistic QoO scores.

2. *Measurement Noise*: Measurement errors on the same scale as the thresholds (NRP, NRPoU) can distort high-percentile latencies and cause artificially lower QoO.

3. *Requirement Specification*: Slightly adjusting the latency thresholds or target percentiles can cause significant changes in QoO, especially when the measurement distribution is near a threshold.

4. *Measurement Duration*: Shorter tests with sparse sampling tend to underestimate worst-case behavior for heavy-tailed latency distributions, biasing QoO in a positive direction.

From these findings, we deduce the following guidelines for practical application:

* Calibrate the combination of sampling rate and total measurement period to capture fat-tailed distributions of latency with sufficient accuracy.

* Avoid significant measurement noise where possible (e.g., by calibrating time sources, accounting for clock drift).

* Thoroughly test application requirement thresholds so that the resulting QoO scores accurately reflect application performance.

These guidelines are _non-normative_ but reflect empirical evidence on how QoO performs.

9.2. Insights from user testing

A study involving 25 participants tested the Quality of Outcome (QoO) framework in real-world settings [QoOUserStudy]. Participants used specially equipped routers in their homes for 10 days, providing both network performance data and feedback through pre- and post-trial surveys.

Participants found QoO metrics more intuitive and actionable than traditional metrics (e.g., speed tests). QoO directly aligned with their self-reported experiences, increasing trust and engagement.

These results provide supporting evidence for QoO’s value as a user-focused tool, bridging technical metrics with real-world application performance to enhance end-user satisfaction.

10. Known Weaknesses and open questions

A method has been described for simplifying the comparison between application network requirements and Quality Attenuation measurements. This simplification introduces several artifacts, the significance of which may vary depending on context. The following section discusses some known limitations.

Volatile networks - in particular, mobile cellular networks - pose a challenge for network quality prediction, with the level of assurance of the prediction likely to decrease as session duration increases. Historic network conditions for a given cell may help indicate times of network load or reduced transmission power, and their effect on throughput/latency/loss. However: as terminals are mobile, the signal bandwidth available to a given terminal can change by an order of magnitude within seconds due to physical radio factors. These include whether the terminal is at the edge of cell, or undergoing cell handover, the interference and fading from the local environment, and any switch between radio bearers with differing signal bandwidth and transmission-time intervals (e.g. 4G and 5G). This suggests a requirement for measuring Quality Attenuation to and from an individual terminal, as that can account for the factors described above. How that facility is provisioned onto individual terminals, and how terminal-hosted applications can trigger a Quality Attenuation query, is an open question.

10.1. Missing Temporal Information in Distributions.

The two latency series (1,200,1,200,1,200,1,200,1,200) and (1,1,1,1,1,200,200,200,200,200) have identical distributions, but may have different application performance. Ignoring this information is a tradeoff between simplicity and precision. To capture all information necessary to perfectly capture outcomes quickly gets into extreme levels of overhead and high computational complexity. An application's performance depends on reactions to varying network performance, meaning nearly all different series of latencies may have different application outcomes.

10.2. Subsampling the real distribution

Additionally, it is not feasible to capture latency for every packet transmitted. Probing and sampling can be performed, but some aspects will always remain unknown. This introduces an element of probability. Absolute perfection cannot be achieved; rather than disregarding this reality, it is more practical to acknowledge it. Therefore, discussing the probability of outcomes provides a more accurate and meaningful approach.

10.3. Assuming Linear Relationship between Perfect and Unusable (and that it is not really a probability)

One can conjure up scenarios where 50ms latency is actually worse than 51ms latency as developers may have chosen 50ms as the threshold for changing quality, and the threshold may be imperfect. Taking these scenarios into account would add another magnitude of complexity to determining network requirements and finding a distance measure (between requirement and actual measured capability).

10.4. Binary Bandwidth threshold

Choosing this is to reduce complexity, but it must be acknowledged that the applications are not that simple. Network requirements can be set up per quality level (resolution, fps etc.) for the application if necessary.

10.5. Arbitrary selection of percentiles

A selection of percentiles is necessary for simplicity, because more complex methods may slow adoption of the framework. The 0th (minimum) and 50th (median) percentiles are commonly used for their inherent significance. According to [BITAG], the 90th, 98th, and 99th percentiles are particularly important for certain applications. Generally, higher percentiles provide more insight for interactive applications, but only up to a certain threshold— beyond which

applications may treat excessive delays as packet loss and adapt accordingly. The choice between percentiles such as the 95th, 96th, 96.5th, or 97th is not universally prescribed and may vary between application types. Therefore, percentiles must be selected arbitrarily, based on the best available knowledge and the intended use case.

11. Implementation status

Note to RFC Editor: This section must be removed before publication of the document.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

11.1. qoo-c

* Link to the open-source repository:

  https://github.com/getCUJO/qoo-c

* The organization responsible for the implementation:

  CUJO AI

* A brief general description:

  A C library for calculating Quality of Outcome

* The implementation’s level of maturity:

A complete implementation of the specification described in this document

* Coverage:

  The library is tested with unit tests

* Licensing:

  MIT

* Implementation experience:

  Tested by the author. Needs additional testing by third parties.

* Contact information:

  Bjrn Ivar Teigen Monclair: bjorn.monclair@cujo.com

* The date when information about this particular implementation was last updated:

  27th of May 2025

11.2. goresponsiveness

* Link to the open-source repository:

  https://github.com/network-quality/goresponsiveness

  The specific pull-request: https://github.com/network-quality/goresponsiveness/pull/56

* The organization responsible for the implementation:

  University of Cincinatti for goresponsiveness as a whole, Domos for the QoO part.

* A brief general description:

  A network quality test written in Go. Capable of measuring RPM and QoO.

* The implementation’s level of maturity:

  Under active development; partial QoO support integrated.

* Coverage:

The QoO part is tested with unit tests

* Licensing:

  GPL 2.0

* Implementation experience:

  Needs testing by third parties

* Contact information:

  Bjrn Ivar Teigen Monclair: bjorn.monclair@cujo.com

  William Hawkins III: hawkinwh@ucmail.uc.edu

* The date when information about this particular implementation was last updated:

  10th of January 2024

12. Security Considerations

The Quality of Outcome (QoO) framework introduces a method for assessing network quality based on probabilistic outcomes derived from latency, packet loss, and throughput measurements. While the framework itself is primarily analytical and does not define a new protocol, some security considerations arise from its deployment and use.

Measurement Integrity and Authenticity

QoO relies on accurate and trustworthy measurements of network performance. If an attacker can manipulate these measurements—either by injecting falsified data or tampering with the measurement process—they could distort the resulting QoO scores. This could mislead users, operators, or regulators into making incorrect assessments of network quality.

To mitigate this risk:

* Measurement agents can authenticate with the systems collecting or analyzing QoO data.

* Measurement data can be transmitted over secure channels (e.g., TLS) to ensure confidentiality and integrity.

* Digital signatures may be used to verify the authenticity of measurement reports.

Risk of Misuse and Gaming

As QoO scores may influence regulatory decisions, service-level agreements (SLAs), or user trust, there is a risk that network operators or application developers might attempt to "game" the system. For example, they might optimize performance only for known test conditions or falsify requirement thresholds to inflate QoO scores.

Mitigations include:

* Independent verification of application requirements and measurement methodologies.

* Use of randomized or blind testing procedures.

* Transparency in how QoO scores are derived and what assumptions are made.

Privacy Considerations

QoO measurements may involve collecting detailed performance data from end-user devices or applications. Depending on the deployment model, this could include metadata such as IP addresses, timestamps, or application usage patterns.

To protect user privacy:

* Data collection should follow the principle of data minimization, only collecting what is strictly necessary.

* Personally identifiable information (PII) should be anonymized or pseudonymized where possible.

* Users should be informed about what data is collected and how it is used, in accordance with applicable privacy regulations (e.g., GDPR).

Denial of Service (DoS) Risks

Active measurement techniques used to gather QoO data (e.g., TWAMP, STAMP, synthetic traffic generation) can place additional load on the network. If not properly rate-limited, this could inadvertently degrade service or be exploited by malicious actors to launch DoS attacks.

Recommendations:

* Implement rate limiting and access control for active measurement tools.

* Ensure that measurement traffic does not interfere with critical services.

* Monitor for abnormal measurement patterns that may indicate abuse.

Trust in Application Requirements

QoO depends on application developers to define Network Requirements for Perfection (NRP) and Network Requirements Point of Unusableness (NRPoU). If these are defined inaccurately—either unintentionally or maliciously—the resulting QoO scores may be misleading.

To address this:

* Encourage peer review and publication of application requirement profiles.

* Where QoO is used for regulatory or SLA enforcement, require independent validation of requirement definitions.

13. IANA Considerations

This document has no IANA actions.

14. Informative References

[BITAG] BITAG, "Latency Explained", October 2022, <https://www.bitag.org/documents/BITAG_latency_explained.pdf>.

[Bufferbloat] "Bufferbloat: Dark buffers in the Internet", n.d., <https://queue.acm.org/detail.cfm?id=2071893>.

[Haeri22] "Mind Your Outcomes: The ∆QSD Paradigm for Quality-Centric Systems Development and Its Application to a Blockchain Case Study", n.d., <https://www.mdpi.com/2073-431X/11/3/45>.

[IRTT] "Isochronous Round-Trip Tester", n.d., <https://github.com/heistp/irtt>.

[JamKazam] Wilson, D., "What is Latency and Why does it matter?", n.d., <https://jamkazam.freshdesk.com/support/solutions/articles/66000122532-what-is-latency-why-does-it-matter-?>.

[Kelly] Kelly, F. P., "Networks of Queues", n.d., <https://www.cambridge.org/core/journals/advances-in-applied-probability/article/abs/networks-of-queues/38A1EA868A62B09C77A073BECA1A1B56>.

[QoOAppQualityReqs] stensen, T., "Performance Measurement of Web Applications", n.d., <https://assets.ycodeapp.com/assets/app24919/Documents/U6TlxIlbcl1dQfcNhnCleziJWF23P5w0xWzOARh8-published.pdf>.

[QoOSimStudy] Monclair, B. I. T., "Quality of Outcome Simulation Study", n.d., <https://github.com/getCUJO/qoosim>.

[QoOUserStudy] Monclair, B. I. T., "Application Outcome Aware Root Cause Analysis", n.d., <https://assets.ycodeapp.com/assets/app24919/Documents/LaiW4tJQ2kj4OOTiZbnf48MbS22rQHcZQmCriih9-published.pdf>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <https://www.rfc-editor.org/rfc/rfc5357>.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <https://www.rfc-editor.org/rfc/rfc5481>.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, DOI 10.17487/RFC6049, January 2011, <https://www.rfc-editor.org/rfc/rfc6049>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <https://www.rfc-editor.org/rfc/rfc6390>.

[RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <https://www.rfc-editor.org/rfc/rfc7680>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <https://www.rfc-editor.org/rfc/rfc8033>.

[RFC8239] Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <https://www.rfc-editor.org/rfc/rfc8239>.

[RFC8290] Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm", RFC 8290, DOI 10.17487/RFC8290, January 2018, <https://www.rfc-editor.org/rfc/rfc8290>.

[RFC8762] Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple Two-Way Active Measurement Protocol", RFC 8762, DOI 10.17487/RFC8762, March 2020, <https://www.rfc-editor.org/rfc/rfc8762>.

[RFC9318] Hardaker, W. and O. Shapira, "IAB Workshop Report: Measuring Network Quality for End-Users", RFC 9318, DOI 10.17487/RFC9318, October 2022, <https://www.rfc-editor.org/rfc/rfc9318>.

[RPM] "Responsiveness under Working Conditions", July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ippm-responsiveness>.

[RRUL] "Real-time response under load test specification", n.d., <https://www.bufferbloat.net/projects/bloat/wiki/RRUL_Spec/>.

[TR-452.1] Broadband Forum, "TR-452.1: Quality Attenuation Measurement Architecture and Requirements", September 2020, <https://www.broadband-forum.org/download/TR-452.1.pdf>.

Acknowledgments

The authors would like to thank Will Hawkins, Stuart Cheshire, Jason Livingood, Olav Nedrelid, Greg Mirsky, Tommy Pauly, Marcus Ihlar, Tal Mizrahi, Ruediger Geib, Mehmet kr Kuran, Michael Welzl, Kevin Smith, Luis Miguel Contreras Murillo, Ike Kunze, Guiseppe Fioccola and Neil Davies for their feedback and input to this document.

Authors' Addresses

Bjrn Ivar Teigen Monclair
CUJO AI
Gaustadallen 21
0349
Norway
Email: bjorn.monclair@cujo.com

Magnus Olden
CUJO AI
Gaustadallen 21
0349
Norway
Email: magnus.olden@cujo.com