Workgroup: IP Performance Measurement
Internet-Draft: draft-ietf-ippm-qoo-00
Published: 21 March 2024
Intended Status: Informational
Expires: 22 September 2024
Authors: B. I. Teigen
Domos
M. Olden
Domos

Quality of Outcome

Abstract

This document describes a new network quality framework named Quality of Outcome (QoO). The QoO framework is unique among network quality frameworks because it is designed to meet the needs of application developers, users and operators. This is achieved by basing the framework on Quality Attenuation, defining a simple format for specifying application requirements, and defining a way to compute a simple and intuitive user-facing metric.

The framework proposes a way of sampling network quality, setting network quality requirements and a formula for calculating the probability for the sampled network to satisfy network requirements.

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://domoslabs.github.io/QoOID/draft-olden-ippm-qoo.html. 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/domoslabs/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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 September 2024.

Copyright Notice

Table of Contents

1. Introduction

This document explores how the quality attenuation metric and framework [TR-452.1] can be extended to meet the full set of requirements described in the Motivation section. The goal is to define 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. Basing this framework on quality attenuation [TR-452.1] ensures that network operators get the fault-isolation and network planning benefits of composability.

Quality attenuation is a network quality metric that meets most of the criteria set out in the requirements; 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. The part that is yet missing is how to present quality attenuation results to end-users and application developers in an understandable way. We believe a per-application, per application-type, or per-SLA approach is appropriate here. The challenge lies in specifying how to simplify enough without losing too much in terms of precision and accuracy.

We believe the probabilistic approach is key as 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 will be practically impossible. We can however make educated guesses on the probability of outcomes.

We propose 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. We propose a formula for a distance measure between perfect and unusable. This distance measure can, with some assumptions, 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, we require some parameters that allow for analysis of the precision.

2. Motivation

This section describes the features and attributes a network quality framework must have to be useful for different stakeholders. The stakeholders included are Application developers, End-Users, and Network Operators and Vendors. At a high level, End-Users need an understandable network metric. Application developers need 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 we have yet to find one that bridges the needs of all three.

2.1. Introduction

This section aims to describe the features a network performance framework must have to be understandable to end-users, useful for application developers, and actionable for network operators. One of the key motivations behind this initiative 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, the incentives to deploy them have remained relatively

weak. By creating a unifying framework for assessing network quality, we aim to strengthen these incentives significantly.

Bandwidth 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]. Unfortunately, it is complicated to benchmark the quality of network transport. 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 huge strides in making a better network quality metric that is far closer to application outcomes than bandwidth is, and the latter is successful at being relatively relatable/understandable to end-users.

As pointed out in [RPM], “Our networks remain unresponsive, not from a lack of technical solutions, but rather a lack of awareness of the problem.” The lack of awareness means a lack of incentives for operators to invest in improving network quality (beyond increasing the throughput). While Open Source solutions exist, vendors rarely implement them. And it all boils down to the lack of a universally accepted network quality framework that captures how well applications are likely to work.

A recent IAB workshop on measuring internet quality for end users identified this important point: Users mostly care about application performance (as opposed to network performance). Among the conclusions is 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]. One of the requirements we set out here is, therefore, to be able to answer this question: "Will an application work properly?". An answer to this question requires a few things; First, we must acknowledge that the internet is stochastic (from the point-of-view of any given client), and we can never answer this question with certainty. Second, different applications have different needs and adapt differently to varying network conditions. Any framework aiming to answer this question must be able to cater to the needs of different applications. Thirdly, end users are individuals with different perception of, and levels of tolerance for, degradation of network conditions and the resulting effect on application experience.

2.2. Design Goal

The overall goal 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.

2.3. Requirements

This section describes the three main requirements and the motivation for each.

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 the network links and computational steps involved in making the application function.

These delays in turn 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 put different patterns of load on the network. To provide an answer to whether or not applications will work well or fail, a network quality framework must therefore be able to compare measurements of network performance to many different application requirements.

Flexibility in describing application requirements and the ability to capture the delay characteristics of the network in enough detail to compute how likely application success is with satisfactory accuracy and precision are necessary conditions.

How can operators take action when measurements show that applications fail too often? We can answer this question if the measured metric(s) support spatial composition [RFC6049], [RFC6390]. Spatial composition gives us the ability to divide results into sub-results, each measuring the performance of a required sub-milestone that must be reached in time for the application to succeed.

To summarise, the framework and "meaningful metric" we're looking for should have the following properties:

1. Capture the information necessary to compute the probability that applications will work well. (Useful for End-users and Application developers)

2. Compare meaningfully to different application requirements.

3. Compose. So that operators can isolate and quantify the contributions of different sub-outcomes and sub-paths of the network. (Useful for Operators and Vendors)

2.3.1. Requirements for end-users

The quality framework should facilitate a metric that is objective, relatable, and relatively understandable for an end-user. We are looking for a middle ground between objective QoS metrics (Throughput, packet loss, jitter, average 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 can reliably deliver 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: 28ms latency note-to-ear for direct monitoring, <2ms jitter

*Online gaming: 6Mb/s downlink throughput and 30ms RTT to join a multiplayer game

*Virtual reality: <20ms RTT from head motion to rendered update in VR

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.

2.3.2. 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. We can only expect some developers to 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.

2.3.3. 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, we could measure end-to-end (e.g., a-b-c-d-e) and not-end-to-end (e.g., b-c-d-e) and subtract, we can isolate the areas outside the influence of the network operator. In other words, we could get the network quality of a-b and b-c-d-e separately. 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 spatial composition is useful, we can look at a typical web page load sequence. If we measure web page load times and find they are too often too slow, we may then separately measure DNS resolution time, TCP round-trip time, and the time it takes to establish TLS connections 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.

3. 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]
- Varying Latency Under Load Tests
- Varying Speed Tests with latency measures
- Simulating real traffic
- End-to-end measurements of real traffic
- TCP SYN ACK / DNS Lookup RTT Capture
- 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]. The novelty of the Quality Attenuation metric is to view packet loss as infinite (or too late to be of use e.g. > 3 seconds) latency [TR-452.1].

Latency Distributions can be gathered via both passive monitoring and active testing. The active testing can use any type of IP traffic, such as TCP, UDP, or QUIC. It is OSI Layer and network technology independent, meaning it can be gathered in an end-user application, within some network equipment, or anywhere in between.

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, typically downloads, have lenient latency requirements. Video Conferences typically are 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 minumum 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].

3.1. Discussion of other performance metrics

Many network performance metrics and frameworks for reasoning about them have been proposed, used, and abused throughout the years. We present a brief description of some of the most relevant metrics.

For each of the metrics below, we discuss whether or not they meet each of the three criteria set out in the requirements.

3.1.1. Average Peak Throughput

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 we know that enough throughput is available.

Throughput cannot be composed.

3.1.2. Average Latency

Average latency relates to user-observable application outcomes in the sense that the average 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 average latency is good enough [BITAG].

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

3.1.3. 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.

3.1.4. 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.

3.1.5. Inter-Packet Delay Variation (IPDV)

The most common definition of IPDV [RFC5481] measures the difference in one-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.

3.1.6. Packet Delay Variation (PDV)

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

PDV cannot be composed.

3.1.7. Trimmed Mean of Latency

The trimmed mean of latency is the average 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 average latency. In cases where the trimmed mean removes real measurements,

the trimming operation introduces errors that may compound when composed.

3.1.8. 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.

3.1.9. 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 we know the quality attenuation of each sub-goal required to reach the desired outcome [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 [Haeri22].

3.1.10. Summary of performance metrics

This table summarizes the properties of each of the metrics we have 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. We assume 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
Average 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
Average 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

4. Sampling requirements

To reach the design goal of being useful in the contexts laid out in the Motivation section, this work imposes no requirement on the time period or the network loading situation. This choice has pros and cons. Latency under load is extremely important, but average or median latency has a role too. However, a network quality metric that does not take latency under load into account is bound to fail at predicting application outcome.

This framework only requires a latency distribution. If the sampling is done while the network is loaded, latency under load will be part of the distribution, which is encouraged, but is not always possible, for example when passively monitoring the latency of real traffic.

It takes quite a few samples to have a statistically significant distribution. Modeling a distribution may be a challenging software engineering task, hence we need to sample the latency distribution at certain percentiles. A list of 10 percentiles in a logarithmic-esque fashion has already been suggested in industry [0th, 10th, 25th, 50th, 75th, 90th, 95th, 99th, 99.9th, 100th] and seems adequate. We propose to define a shared set of percentile values to report.

The framework is flexible when it comes to the direction of traffic that is being sampled, but does require that it is noted whether the latency distribution is measured one-way or two-way. The framework does not require an explicit throughput measurement, but does require a note on the maximal observed throughput in the time period.

By not requiring a specific number of samples, this framework allows taking 10 samples and calling it a distribution, which of course is not ideal. On the other hand, making the framework overly complex and difficult to adhere to using real-world equipment and applications is likely to reduce willingness to adopt the framework. Constraints will vary for different network equipment and applications.

To make sure we can trust measurements from others and analyze their precision, we require:

*Timestamp of first sample

*Duration of the sampling period

*Number of samples

*Type of measurement:

-Cyclic (a sample every Nth ms) - Specify N

-Bursts (X samples every Nth ms) - Specify X and N

-Passive (observing traffic and therefore unevenly sampled)

By requiring the report of these variables, we ensure that the network measurements can be analyzed for precision and confidence.

5. 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]. In essence, it 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. The last specified percentile

marks the acceptable packet loss. I.e. if the 99th percentile is the highest percentile defined, 1% packet loss (100-99) is inferred.

Applications do of course have throughput requirements. With classical TCP and typical UDP flows, latency and packet loss would be enough, as they are bound to create some latency or packet loss when ramping up throughput if subsequently they become hindered by insufficient bandwidth. However, we cannot always rely on monitoring latency exclusively, as low bandwidth may give poor application outcomes without necessarily inducing a lot of latency. Therefore, the network requirements should include a minimum throughput requirement.

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 we need 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’.

We extend the requirements 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 Requirement Point of Unusableness (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.

6. Calculating Quality of Outcome (QoO)

At this point we have everything we need to calculate the quality of the application outcome (QoO). There are 3 scenarios:

1. The network meets all the requirements for perfection. There is a 100% chance that the application is not lagging because of the network

2. The network does meet one of the criteria of the Point of Unusableness. There is a 0% chance that the application will work well, and it's because of the network

3. The network does not meet NRP but is not beyond NRPoU.

1 and 2 require nothing more from the framework. For 3, we will now specify the calculation to translate these distances to a 0 to 100 measure. We use the percentile pair where the measured latency is the closest to the NRPoU as the application is only as good as its weakest link.

Mathematically:

QoO = min_{i}(min(max((1-((ML-NRP)/(NRPoU-NRP))) * 100, 0), 100))

Where

ML = Measured Latency in percentiles and milliseconds
NRP = Network Requirement for Perfection, defined as minimum throughput and percentiles and milliseconds
NRPoU = Network Requirement Point of Unusableness in percentiles and milliseconds
and i iterates over the list of percentiles and milliseconds

Essentially, where on the relative distance between Network Requirement for Perfection (NRP) and Network Requirement Point of Unusableness (NRPoU) the Measured Latency (ML) lands, normalized to a percentage.

6.1. Example requirements and measured latency:

NRP: 4 Mbps {99%, 250 ms},{99.9%, 350 ms} NRPoU: {99%, 400 ms}, {99.9%, 401 ms} Measured Latency: .... 99% = 350ms, 99.9% = 352 ms Measured Minumum bandwidth: 32 Mbps / 28 Mbps

Then the QoO is defined:

QoO

```
= 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
```

In this example, we would say: This application/SLA/application category has a 33% chance of being lag-free on this network. Note that packet loss is included as infinite latency, so if there is enough packet loss to breach the highest percentile requirement then the QoO is 0.

7. How to find network requirements

A key advantage of having a measurement that stretches between perfect and unusable, as opposed to having a binary (Good/Bad) or other low resolution (Superbad/Bad/OK/Great/Supergreat) metrics, is that we have some leeway. The leeway is useful, for instance: a lower than 20% chance of lag free experience is intuitively not good and a greater than 90% chance of lag free experience is intuitively good --- meaning we don’t have to find perfection for making the QoO metric useful.

Nevertheless we have to find some points for unusableness and perfection. There is no strict definition of when the network is so bad that the application is unusable. For perfect, we may have a definition for some apps, but for apps like web browsing and gaming, lower latency is simply better. But to assist those who wish to make a requirement, we can say that if the end-user experience does not change when reducing the latency, the network quality is sufficient for the Network Requirements for Perfection (NRP) .

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.

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.

7.1. 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. Known Weaknesses and open questions

We have described a way of simplifying how the network requirements of applications can be compared to quality attenuation measurements. The simplification introduces several artifacts that may or may not be significant. If new information emerges that indicate other

tradeoffs are more fit for our purpose, we should switch before this Internet Draft moves further. In this section we discuss 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 indiviudal terminals, and how terminal-hosted applications can trigger a quality attenuation query, is an open question.

8.1. Missing Temporal Information in Distributions.

These two latency series: 1,200,1,200,1,200,1,200,1,200 and 1,1,1,1,1,200,200,200,200,200 will 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 we are getting into extreme computational complexity. As an application's performance is bound by how the developers react to varying network performance, meaning nearly all different series of latencies may have different application outcomes.

It will most likely be necessary to add a time-scale to the application requirement specifications.

8.2. Subsampling the real distribution

Additionally, we cannot capture latency on every packet that is sent. We can probe and sample, but there will always be unknowns. We are now in the realm of probability. Perfection is impossible, but instead of denying this, we should embrace it, which is why talking about the probability of outcomes is the way forward.

8.3. 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).

8.4. Binary Bandwidth threshold

Choosing this is to reduce complexity, but we do acknowledge that the applications are not that simple. The defence for this trade off is that insufficient bandwidth will cause queues and therefore latency, and it should be possible to see this. Additionally, network requirements can be set up per quality level (resolution, fps etc.) for the application. However, having too many network requirements also increases the complexity for users of the framework, and it is still unclear if this is the optimal tradeoff.

8.5. Low resolution on Packet Loss

To ensure simplicity, packet loss is described as infinite latency and the resolution will be bound to the percentiles we chose to sample. There is a good argument that some applications need higher resolution on packet loss for sufficiently describing application outcomes. If this good evidence is presented for this, packet loss should be measured separately and added to the QoO formula.

8.6. Arbitrary selection of percentiles

There is a need for a selection of percentiles, as we in the name of simplicity can’t use them all. But how should we select them? The 0th (minimal) and 50th (median) percentile have implicit usage by themselves. [BITAG] discusses that the 90th, 98th and 99th percentiles are key for some apps. In general the wisdom is that the higher percentiles are more useful for interactive applications, but only to a certain point. At this point an application sees it as packet loss and may adapt to it. Should we pick the 95th, 96th percentile, the 96.5th or the 97th? We don’t know, and as this is likely not universal across applications and applications classes, we simply have to choose arbitrarily, and to the best of our knowledge.

9. 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".

9.1. qoo-c

*Link to the open-source repository:

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

*The organization responsible for the implementation:

Domos

*A brief general description:

A C library for calculating Quality of Outcome

*The implementation's level of maturity:

A complete implentation of the specification described in this document

*Coverage:

The library is tested with unit tests

*Licensing:

GPL 2.0

*Implementation experience:

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

*Contact information:

Bjørn Ivar Teigen: bjorn@domos.no

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

10th of January 2024

9.2. 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:

In active development

*Coverage:

The QoO part is tested with unit tests

*Licensing:

GPL 2.0

*Implementation experience:

Needs testing by third parties

*Contact information:

Bjørn Ivar Teigen: bjorn@domos.no

William Hawkins III: hawkinwh@ucmail.uc.edu

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

10th of January 2024

10. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

11. Security Considerations

TODO Security

12. IANA Considerations

This document has no IANA actions.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

13.2. 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>.

[draft-teigen-ippm-app-quality-metric-reqs] "Requirements for a Network Quality Framework Useful for Applications, Users, and Operators", n.d..

[Haeri22] "Mind Your Outcomes: The ΔQSD 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>.

[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>.

[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>.

[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>.

[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>.

[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

TODO acknowledge.

Authors' Addresses

Bjørn Ivar Teigen
Domos
Gaustadalléen 21
0349
Norway

Email: bjorn@domos.no

Magnus Olden
Domos
Gaustadalléen 21
0349
Norway

Email: magnus@domos.no