Requirements for a Network Quality Framework Useful for Applications, Users, and Operators

Abstract

This document describes the features and attributes a network quality framework must have to be useful for different stakeholders. The stakeholders included are developers of Applications, 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.

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/AppQualityMetricID/draft-teigen-ippm-app-quality-metric-reqs.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-teigen-ippm-app-quality-metric-reqs/.

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/AppQualityMetricID.

Status of This Memo

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

1. Introduction
2. Design Goal
3. Requirements,
   3.1. Requirements for end-users
   3.2. Requirements from Application and Platform Developers
   3.3. Requirements for Network Operators and Network Solution Vendors
4. Discussion of other performance metrics
   4.1. Average Peak Throughput
   4.2. Average Latency
   4.3. 99th Percentile of Latency
   4.4. Variance of latency
   4.5. Inter-Packet Delay Variation (IPDV)
   4.6. Packet Delay Variation (PDV)
   4.7. Trimmed Mean of Latency
   4.8. Round-trips Per Minute
   4.9. Quality Attenuation
   4.10. Summary of performance metrics
5. Conclusion
6. Conventions and Definitions
1. Introduction

This document 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. 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.

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)

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.

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.

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.

4. 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.

4.1. 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.

4.2. 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.

4.3. 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.

4.4. 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.

4.5. 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.
4.6. 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.

4.7. 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.

4.8. 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.

4.9. 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].

4.10. 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.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Capture probability of general applications working well</th>
<th>Easy to articulate Application requirements</th>
<th>Composable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average latency</td>
<td>Yes for some applications</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Variance of latency</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IPDV</td>
<td>Yes for some applications</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PDV</td>
<td>Yes for some applications</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Average Peak Throughput</td>
<td>Yes for some applications</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>99th Percentile of Latency</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Trimmed mean of latency</td>
<td>Yes for some applications</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Round Trips Per Minute</td>
<td>Yes for some applications</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Quality Attenuation</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1

5. Conclusion

We describe requirements for a framework which is useful for end-users, network operators, vendors, and applications. Our brief survey of existing performance metrics concludes that none of the metrics we looked at meet all of the requirements at once. This clearly presents an opportunity. For instance, RPM does a great job of improving the visibility of network quality issues beyond
throughput but is inherently about end-to-end tests and is not designed to help network operators monitor, test, and understand their networks from within. Quality Attenuation [TR-452.1], on the other hand, is a great tool for understanding the performance of a network from within but is challenging to use and understand for end-users or application developers.

The requirements described here may be impossible to meet entirely in practice. Still, aiming for a framework and metrics that meet the requirements is a worthwhile goal. A solution that meets all of these requirements may help improve the Internet by strengthening incentives to deploy solutions that materially affect the quality of user experiences.

6. 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] when, and only when, they appear in all capitals, as shown here.

7. Security Considerations

TODO Security

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References


9.2. Informative References


Acknowledgments

The authors would like to acknowledge Gavin Young, Kevin Smith, Peter Thompson, Brendan Black, Gino Dion, Mayur Sarode, Greg Mirsky, Olav Nedreid, Karl Magnus Kalvik, Knut Joar Strommen, Hans Petter Dalsklev, Jakub Kozlowski, Wim De Ketelaere, William Hawkins, and Ian Wheelock for their comments, reviews, and contributions.

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