NETMETRİK Quality Framework (NQF) Methodology

Version 2.0 — As-Built (Implemented System) Publication date: 20 June 2026
Organization: NETMETRİK — independent internet measurement platform based in Türkiye

> **About this version.** This document describes the *actually running* (as-built) version of the NETMETRİK measurement engine. Every method described here is implemented in the production measurement client (`SpeedEngine`) and is verifiable against the source code. Planned but not-yet-implemented techniques are listed separately and explicitly labeled in §9 **Roadmap**; they are never presented as current methodology. This distinction is a direct requirement of our "transparent and verifiable" principle.

Abstract

The NETMETRİK Quality Framework (NQF) is a consumer measurement methodology that treats internet connection quality as a time-dependent distribution of measurements rather than a single static number. It runs entirely as a browser-based client, using HTTP/2 streaming for download and WebSocket-first transport for upload. Bandwidth is reported as the **P30–P90 trimmed mean** of samples collected over a fixed 15-second test. Responsiveness is computed via an RFC 9097-inspired **RPM (Responsiveness Per Minute)** metric based on real latency under load; bufferbloat is measured as the difference between loaded and idle latency. Results are additionally mapped to concrete usage scenarios (4K video, cloud gaming, VoIP). The methodology is calibrated to share the same fundamental statistical reporting approach as the industry reference Speedtest.net (v2.11); differences are stated explicitly in §7.

---

1. Motivation

Expressing internet connection quality with a single static number ("100 Mbps") fails to capture the variable behavior of modern networks. Real user experience depends on the stability of throughput over time, on how latency behaves under load, and on where bottlenecks occur.

NQF aims to answer not only "How fast is my internet?" but also "How stable is my connection, how responsive is it under load, and what digital experiences can I run smoothly on this line?" As an independent measurement authority based in Türkiye, NETMETRİK aims to provide a transparent and reproducible methodology.

2. Assessment of Existing Approaches

Mainstream measurement tools each make distinct design choices with corresponding limitations:

- **Ookla (Speedtest.net):** The global industry reference. Performs a multi-connection, short-duration, saturation-oriented measurement. Its reporting methodology (trimmed mean) is the approach we also build upon. Its commonly cited drawback is a tendency to pull the reported value toward the peak.
- **Cloudflare Speed Test:** Offers a realistic 90th-percentile (p90) reporting and integrates RPM. Still reduces performance to a single summary value.
- **M-Lab (NDT):** Uses single-stream TCP analysis. Scientifically conservative, but does not fully reflect the real performance of modern multi-stream applications.
- **Fast.com:** Netflix-CDN oriented; ISP CDN peering agreements can influence results (CDN bias).

NQF adopts Ookla's proven statistical reporting foundation and, on top of it, adds responsiveness under load (RPM, bufferbloat) and usage-scenario mapping as first-class outputs.

3. System Architecture

3.1. Test Servers (NetmetrikServer)

Test load is provided by stateless servers written in Go. The servers only produce/consume measurement traffic and hold no user state:

- `HEAD /ping` — for RTT (round-trip time) measurements.
- `GET /download` — streaming test payload sent to the client.
- `POST /upload` and a WebSocket upload channel — reception and acknowledgment of binary data from the client.

3.2. Transport Layer (As-Built)

The implemented system uses two transport protocols:

- **Download:** Streaming `GET` over HTTP/2 (H2). H2's multiplexing allows multiple concurrent streams over a single connection.
- **Upload:** Primarily **WebSocket** binary frames; if WebSocket cannot be established, it falls back to **HTTP/2 streaming** `POST`. The selected transport is reported in the result record via the `transport_ul` field (`ws` or `h2`).

> Idle and loaded latency measurements are taken over a separate TCP connection (WebSocket probe, or HTTP `HEAD` fallback) to avoid being multiplexed onto the same H2 connection as download/upload traffic. This prevents loaded latency from being distorted by H2 head-of-line effects.

4. Server Selection and Latency Measurement

4.1. Server Selection

Configured test nodes are measured in parallel (up to 10 nodes — consistent with the Speedtest v2.11 default). Multiple samples are taken per node, nodes are sorted by **median RTT**, and the node with

the lowest median RTT is selected as the winner. All test traffic is then pinned to this single winning node (single-host pinning).

4.2. Idle (Unloaded) Latency

- **5 samples** per node, taken at **50 ms** intervals.
- The reported **ping** is the **minimum** of the samples — this strips operating-system scheduling jitter and reflects the line's true base latency.
- **Jitter** is computed as the mean of absolute differences between consecutive samples.
- An **IQM** (interquartile mean, p25–p75) is additionally reported as a stability measure.

4.3. Loaded Latency

During download and upload tests, after the ramp-up phase, latency samples are continuously taken at **50 ms** intervals. The median and IQM of loaded latency are computed separately for download and upload.

5. Bandwidth Measurement

Both download and upload tests run for a **fixed 15 seconds**.

5.1. Download (HTTP/2)

- **Initial concurrency:** 4 parallel streams.
- **Concurrency adaptation:** The stream count is set by the formula `clamp(round(Mbps / 6), 4, 6)` based on measured throughput (~6 Mbps per stream capacity assumption; consistent with Speedtest v2.11 `calculateDesiredConnections`). The upper bound is 6 streams in a single-host topology.
- **Probe phase:** A pre-measurement is taken in a 1-second window between 1.0–2.0 s.
- **Ramp-up and discarded samples:** Samples taken during the probe + settle period (~2.5–3.0 s) are **excluded** from the final computation to eliminate TCP slow-start effects.
- **Sampling interval:** 500 ms.

5.2. Upload (WebSocket-first, HTTP/2 fallback)

Primary — WebSocket multi-socket:

- Binary frames are sent over **4 parallel sockets** (consistent with Speedtest's 4 parallel POST behavior).
- **Chunk size:** 64 KB; backpressure via `bufferedAmount`, target buffer 4 MB.
- **Aggregation:** Deltas derived from server-side cumulative byte reports are written into **250 ms** wall-clock buckets. Wall-clock bucketing prevents server reporting delays from inflating the measurement.

Fallback — HTTP/2 streaming POST:

- 32 KB chunk per stream (stays below the H2 initial stream window of ~64 KB, preventing WINDOW_UPDATE stalls).
- Scales to a higher stream count in a high bandwidth-delay-product (high-BDP) regime.

5.3. Statistical Aggregation and Reporting

For both download and upload, the reported value is computed in two steps:

1. **Bucketize → 20 buckets:** High-frequency samples (~50–500 ms) are downsampled into 20 buckets (~750 ms effective resolution). This prevents TCP-cwnd micro-fluctuations from over-influencing the trimming step and approximates Speedtest's own reporting resolution.
2. **P30–P90 trimmed mean:** Bucketed samples are sorted; the bottom **30%** (slow-start and cwnd dips) and the top **10%** (burst-credit spikes) are discarded; the arithmetic mean of the remaining middle **60%** is taken as the reported throughput.

If the sample count is below 5 (degraded measurement), the system falls back to the median — this is a fault-tolerance behavior, not a calibration. The raw **median** and **peak** are also computed as contextual values.

> **Design rationale.** Reporting near the upper band (e.g., a top-30% mean) captures burst credit and can produce values above the line's physical limit. P30–P90 trimming cuts the spikes and yields the *sustainable* middle band, which converges with the cumulative byte counter.

6. Responsiveness Under Load

6.1. Bufferbloat

Bufferbloat is the queuing delay added when the line is saturated, computed as:

```
worstLoaded = max(loadedDownload, loadedUpload)
bufferbloat = max(0, worstLoaded − unloadedPing)
```

6.2. RPM (Responsiveness Per Minute) — RFC 9097-inspired

RPM is NQF's primary consumer-side responsiveness score. It represents the number of full round trips achievable in one minute while the line is fully loaded, and is computed from the worst-case loaded latency:

```
RPM = min( round( 60000 / max(loadedLatencyMs, 10) ), 2000 )
```

The floor (10 ms) prevents division by near-zero; the ceiling (2000 RPM) represents loaded latency of ~30 ms and below. A high RPM indicates a fluid, instantly responsive connection; a low RPM (e.g., <200) indicates serious bufferbloat.

7. Calibration Against Speedtest.net (v2.11) and Differences

The NQF engine is calibrated to share the same fundamental statistical reporting approach as the industry reference Speedtest.net v2.11. In the interest of honesty, the shared and divergent points are listed explicitly below.

Shared (calibrated):

- 15-second fixed test duration (download and upload).
- Reporting via P30–P90 trimmed mean.
- Download concurrency formula `clamp(round(Mbps/6), 4, 6)`.
- Maximum of 10 nodes measured in parallel; 50 ms ping sample interval.

Divergent (design choice):

- **Upload transport:** NQF is WebSocket-first (with H2 fallback); Speedtest uses HTTP POST (XHR) by default.
- **Server topology:** NQF uses single-host pinning (for reproducibility); Speedtest may use multiple servers.
- **Chunk sizes:** Transport-layer tuning (64 KB WS / 32 KB H2) differs from Speedtest's chunk sizes.

> In an independent three-run comparison, download throughput deviation was measured at the ±0.2% level. The full comparison data is published as an appendix to this document.

8. Application Capability Mapping

Raw metrics are mapped to concrete usage scenarios. Implemented thresholds:

Scenario	Condition
4K Video Streaming	Download > 25 Mbps
Cloud Gaming	RPM > 800 **and** Download > 30 Mbps **and** ping < 40 ms
Video Conferencing (VoIP)	RPM > 200 **and** ping < 150 ms

This profile directly answers "what can I run smoothly on this line?" instead of presenting an abstract number.

9. Limitations

- Measurement is performed at the browser layer; browser/OS scheduling and device CPU load can affect the upper bound. Minimum-RTT ping selection and sample trimming partially mitigate these effects.
- Single-host pinning relies on the assumption that the selected node can serve the full capacity.
- On gigabit-and-above lines, the single-host + 6-stream upper bound can impose a ceiling; this is addressed by increasing server capacity (not by round-robin).

- RPM and bufferbloat reflect network conditions at the moment of measurement and can vary across times of day.

10. Roadmap / Future Work (Not Yet Implemented)

> **Important.** The methods in this section are **not implemented** in the current production engine; they are design goals. They must not be confused with §3–§8 above.

- **Distribution modeling:** Kernel density estimation (KDE) of the throughput time series and bimodality detection (Hartigan dip test) to capture traffic-shaping/QoS signatures.
- **Spectral analysis:** Examination of structural fluctuations via the Welch periodogram.
- **Path decomposition:** Estimation of the bottleneck layer (Wi-Fi, modem, local exchange, peering) from DNS/TCP/TLS/TTFB timings.
- **HTTP/3 + QUIC transport:** A UDP-based, 0-RTT-capable primary transport layer.
- **Z-score-based outlier management:** Statistical flagging of tests corrupted by background activity.
- **Native clients (iOS/Android):** Higher accuracy via raw TCP/UDP sockets.
- **Open data and IETF draft:** Anonymized quarterly datasets and submission of the methodology as an Internet-Draft.

11. References

1. C. Paasch, R. Meyer, S. Cheshire, J. Iyengar, "Responsiveness under Working Conditions," RFC 9097, IETF, 2021.
2. Ookla, "Speedtest.net measurement methodology," (v2.11 web client, observed behavior).
3. Cloudflare, "Test your Internet speed — methodology."
4. M-Lab, "NDT (Network Diagnostic Tool) methodology."

This document is a living reference and is updated as the engine evolves. Every method described in the as-built sections (§3–§8) is verifiable against the production source code as of the publication date.