TESSABYTE™

THROUGHPUT TEST

Help Documentation

Contents

Introduction

Tessabyte Throughput Test is an application for testing the performance of a wireless or wired network. This utility continuously sends TCP and UDP data streams across your network and computes important metrics, such as upstream and downstream throughput values, packet loss, jitter, and round-trip time, and displays the results in both numeric and chart formats. Tessabyte Throughput Test supports both IPv4 and IPv6 connections and allows the user to evaluate network performance depending on the Quality of Service (QoS) settings.

In addition to this core functionality, the application can generate reports in HTML and text formats, conduct pre-scheduled tests, and allows customization of TCP and UDP payloads.

Tessabyte Throughput Test can be used for:

- **Network Capacity Testing:** Assess the maximum data transfer rate that a network link can support.
- **Link Quality Analysis:** Determine the quality of individual network links.
- **Testing Network Congestion:** Simulate high traffic loads and observe how the network behaves under congestion.
- **WLAN Performance Evaluation:** Test wireless networks for speed and reliability.
- **Quality of Service (QoS) Assessment:** Evaluate how different traffic types are prioritized in the network.
- **Firmware and Hardware Updates Validation:** Verify the impact of network device firmware or hardware upgrades.
- **Network Topology Planning:** Understand how network design choices impact performance.
- **Troubleshooting Network Issues:** Identify the root causes of performance degradation.
- **Comparative Benchmarking:** Compare the performance of different network components.
- **Security Assessment:** Evaluate the impact of security measures on network performance.
- **Load Balancing Verification:** Test the effectiveness of load balancing in a network.
- **SLA Compliance Verification:** Ensure that the network meets its Service Level Agreements.
- **Hardware Compatibility Testing:** Verify that all network devices work seamlessly together.

System Requirements

Tessabyte Throughput Test can run on computers with the following minimal system requirements:

- Operating systems:
  Windows 10 and 11. **Windows Server** 2019, 2022, and 2025.
  macOS Monterey, Ventura, Sonoma, and Sequoia.
  Linux Ubuntu, Debian, Fedora, and Red Hat running on x86-64. Other Linux distributions may be supported, but have not been tested. On **Linux, only the server** component is available.

  The **Windows, macOS,** and **Linux** versions are **interoperable**; for example, you can run the server on macOS and the client on a Windows machine or vice versa.

- 8 GB of RAM.
- 100 MB of free disk space.

Installation and Configuration

To perform a throughput test, the application uses two components: a server and a client. The server component of the application listens for connections from clients, while the client connects to the server. Once a connection is established, the client and server exchange data in both directions, and the client component computes and displays the network metrics. The server component can accept connections from multiple clients.

Installation

When you install the application, both the server and client components are installed. You can then run either one, depending on how you plan to perform the tests. In a WLAN setup, the server component should run on the wired side of the network, while the client component should run on a WLAN client. In this type of setup, "downstream" refers to the data flow from the wired side of the network, through the access point, to the client, while "upstream" refers to the data flow from the client, through the access point, to the wired side. For wired LANs, it doesn't matter which of the two computers acts as the server and which as the client.

Configuring the Server on Windows and macOS

The server component of the application (illustrated below) has two key configurable options: the **port** on which it listens for incoming connections and the **maximum number of concurrent client connections**. By default, the server listens on port 32500.

If you want to change the default parameters, make sure to click the **Apply** button after modifying them.

You can also check the **Prevent system sleep** box to prevent the computer from going into sleep or hibernate mode while the application is running. Use the **Dark theme** box to switch between the dark and light user interface themes (Windows only; on macOS, the application uses the currently active visual style).

The middle panel lists the IP addresses and connection times of the currently connected clients. The lower panel displays the IP addresses that the application is listening on for inbound connections (it listens on both IPv4 and IPv6 addresses). The same panel displays a log of client connections and disconnections as they occur.
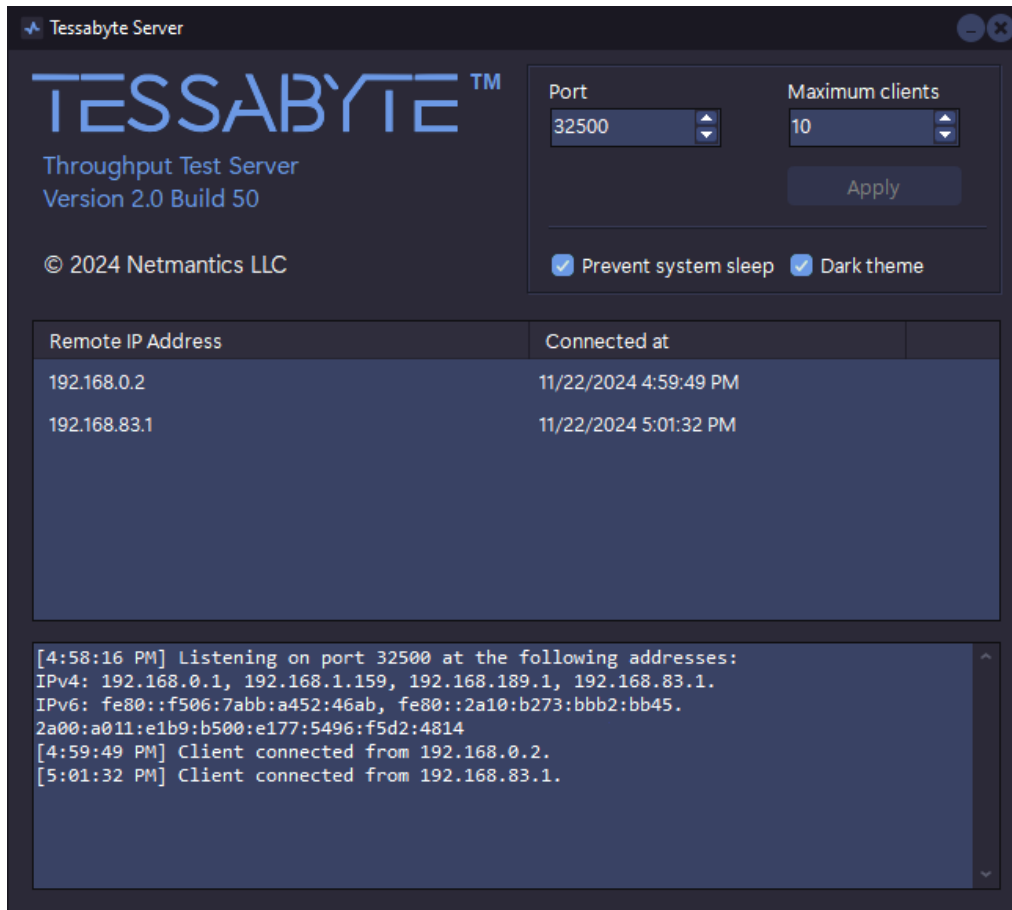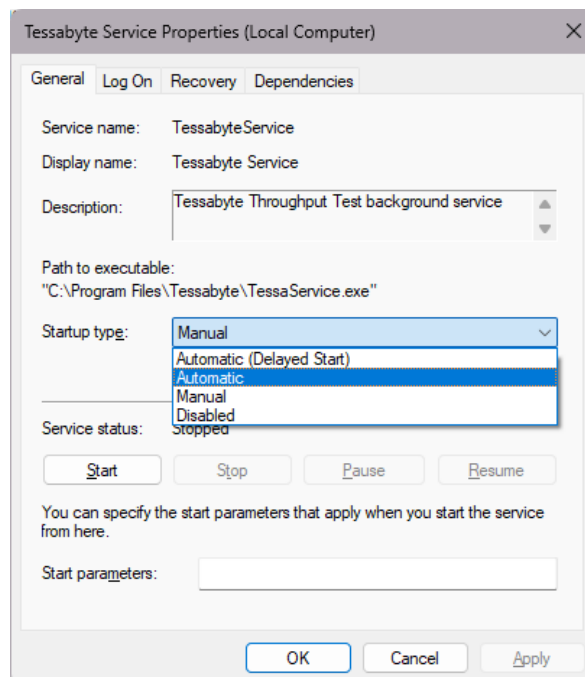
Running the Server as a Windows System Service

The server component is available both as a standard application with a user interface, as shown above, and as a non-interactive Windows system service. This chapter is intended **for advanced users** who are familiar with the concept of system services. If you are not, you can simply run the standard Tessabyte server application whenever you need to conduct a test.

That said, if you prefer to run the Tessabyte server in a non-interactive, unattended mode under the system account and start it immediately after the system boots up, you can easily do so. During installation, Tessabyte creates a new system service named **Tessabyte Service**. The service is installed with the **Manual** startup type, meaning it won't start automatically unless you configure it to do so. Like all other Windows services, **Tessabyte Service** is accessible in the **Services** section of the **Computer Management** panel in Windows, as illustrated below.
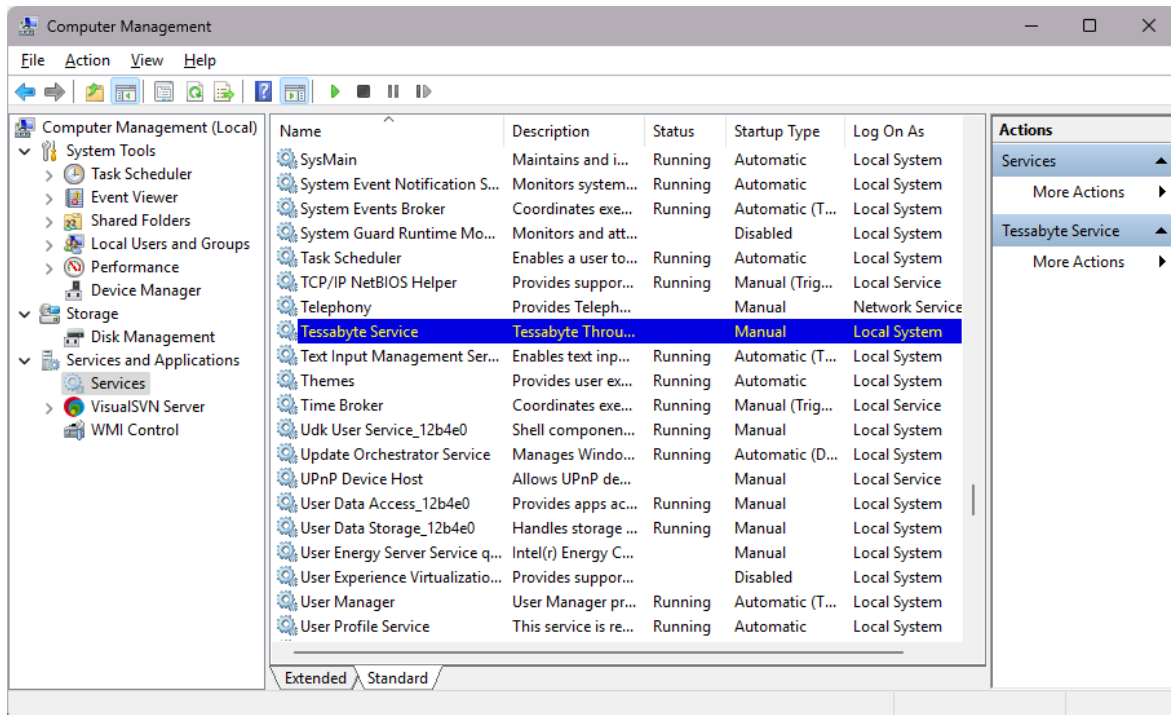
You can double-click the service to change the startup type to **Automatic** if you want it to launch automatically after the system boots. Then, click **Start** to start the service immediately:

A few additional tips for using this system service:

- You obviously should not run the standard Tessabyte Server application while the service is running, as the listening ports will already be in use by the service process.
- You can change the default port number (32500) and maximum number of clients (5) by modifying the following registry key: HKEY_LOCAL_MACHINE\SOFTWARE\TessabyteService. Stop the service, edit the registry values, and then start it again.
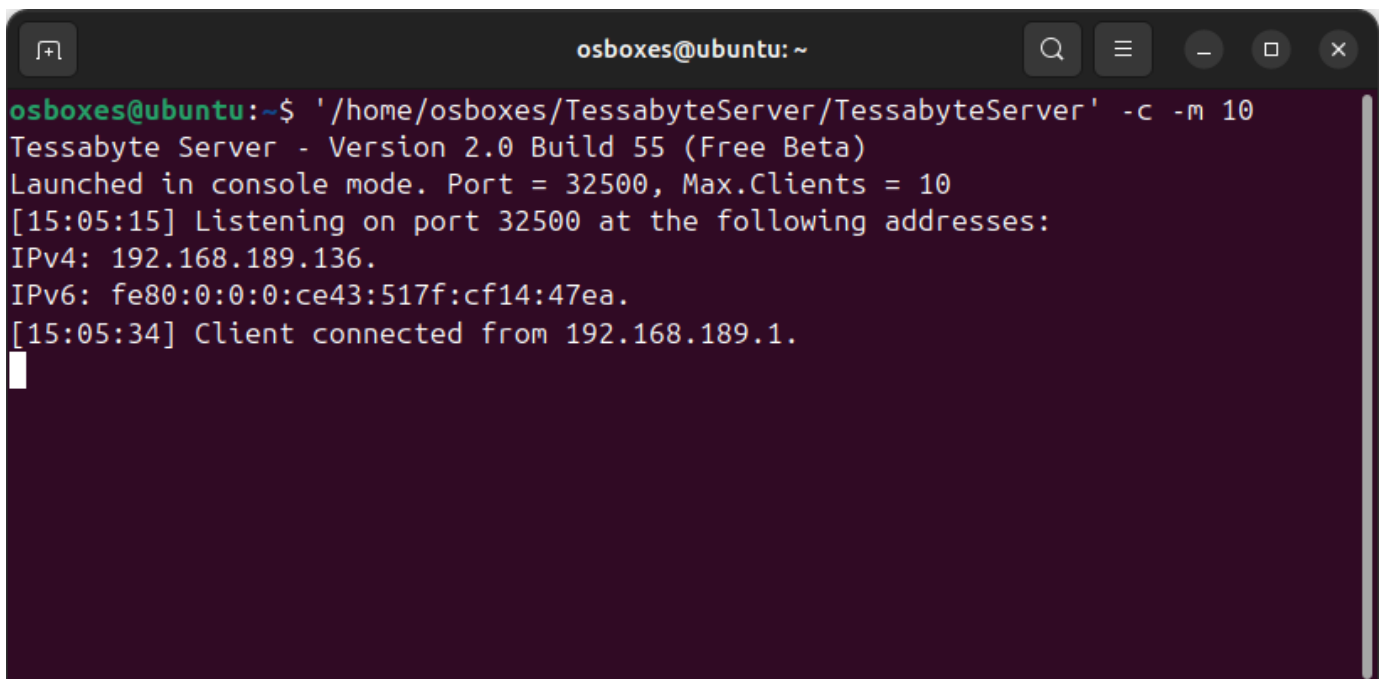
Configuring the Server on Linux

The Linux version includes only the server component. The server can run as a standard system daemon or as a console utility. The following command-line arguments are supported:

- **-c** – Forces the application to run in console mode rather than as a daemon.

- **-p** – Specifies the listening port (between 1024 and 65535). If not specified, the default value is 32500.

- **-m** – Sets the maximum number of concurrent client connections (between 1 and 100). If not specified, the default value is 5.

Example:

```
./TessabyteServer -c -p 41200 -m 20
```

When running in console mode, Tessabyte prints logs to the terminal window:

Without the **"-c"** argument, Tessabyte runs silently as a daemon. If you want it to continue running after the user logs out, consider the following two options.

Option A: Disable systemd's killing behavior

By default, **systemd** terminates all processes launched as part of a user session when the user logs out—even if they were properly detached from the terminal. To change this behavior, execute the following command:

```
loginctl enable-linger $USER
```

If you want to undo this setting later:

```
loginctl disable-linger $USER
```

Option B: System-wide systemd service running as root

Step 1: Create and edit the service file:

```
sudo nano /etc/systemd/system/tessadaemon.service
```

Step 2: Add the following content to the service file:

```
[Unit]
Description=TessaDaemon system-wide service
After=network.target

[Service]
ExecStart=/path/to/TessabyteServer
Restart=always
RestartSec=5
Type=forking
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

Step 3: Reload **systemd**, enable, and start the daemon:

```
sudo systemctl daemon-reload
sudo systemctl enable tessadaemon
sudo systemctl start tessadaemon
```

Firewalls and NAT

On Windows, the application installer automatically creates a Windows Firewall rule that allows it to accept connections. If you use a third-party firewall or a firewall on macOS or Linux, ensure it is configured to allow incoming connections for this application. Both TCP and UDP connections must be allowed. Since the application uses random ports for streaming data back to the client, the best strategy is generally to configure the firewall to allow all connections to and from the server, regardless of the port and protocol.

While Tessabyte Throughput Test is primarily designed for testing LAN and WLAN throughput between local nodes, it can be used for testing WAN throughput. However, you should bear in mind that you may encounter issues with NAT. Generally, a computer located behind a NAT cannot receive incoming TCP connections and/or UDP streams on an arbitrary, unsolicited port without some additional configuration or protocol assistance. This means that the server must be installed on a computer with a public IP address and that if the clients are behind a NAT, they may be limited to TCP tests only. A preferable solution is using IPv6, as it generally doesn't require NAT.

macOS System Permissions

When you install the application and use it for the first time on macOS Sequoia, the operating system will ask you for a confirmation that looks like this for the client and the server, respectively:
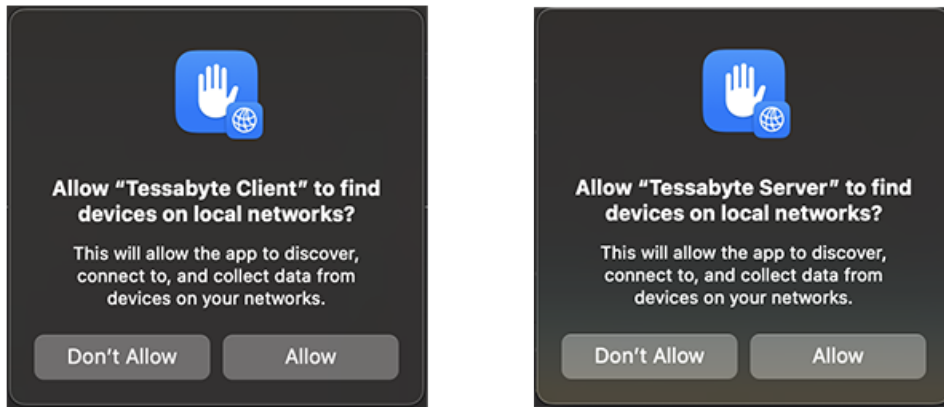
These dialogs typically appear after you click "Connect" in the client application for the first time and/or when the server application accepts a client connection for the first time. You **must grant** this permission to both the client and server applications; otherwise, they will not function properly. You may need to re-launch the applications after granting permission. To verify that the permissions were granted, open macOS **Settings ➔ Privacy & Security ➔ Local Network**, as shown below.
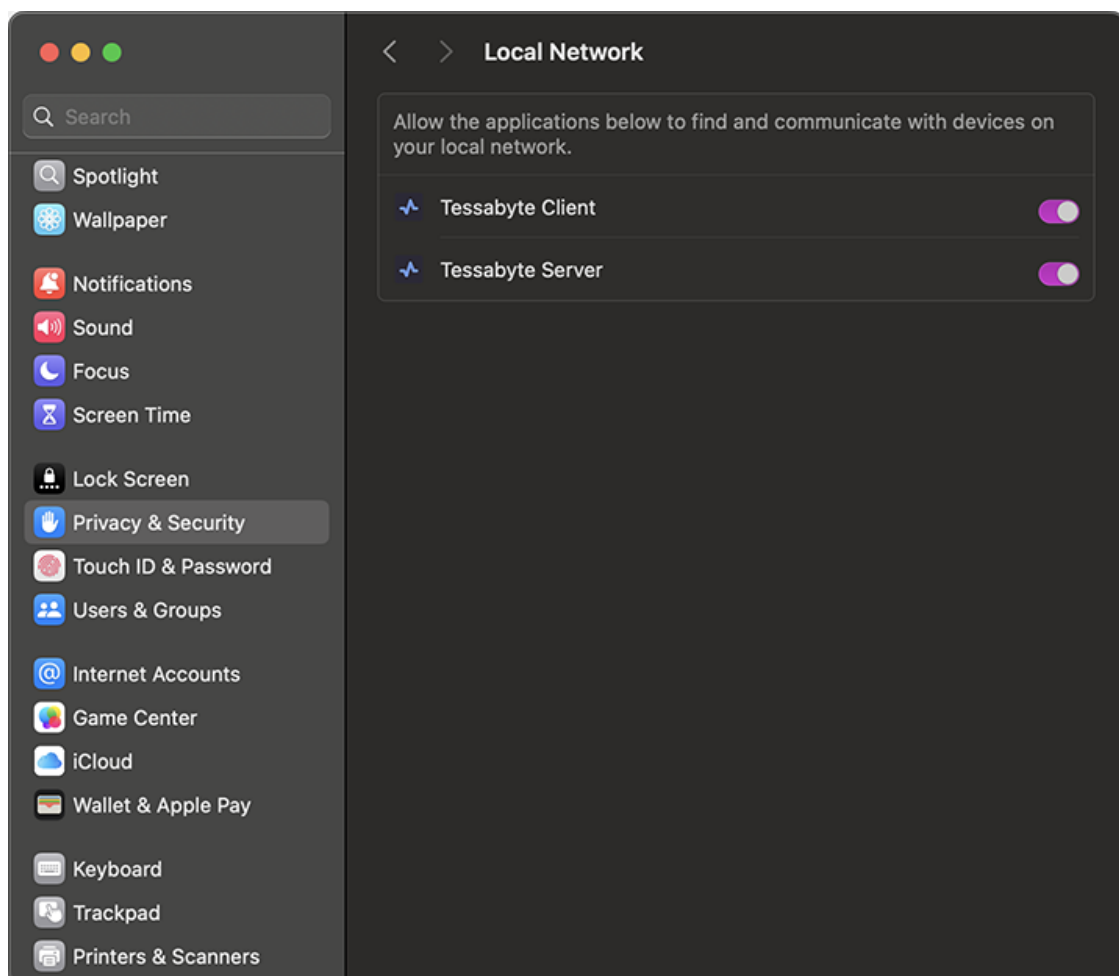
Dashboard – Performing Tests

To begin throughput testing, you need to launch both the client and server on different computers, as described in the previous chapter. In the client window, enter the **hostname, IPv4 address, or IPv6 address** of the server.

> Note: On macOS, link-local IPv6 addresses may require a zone index. If you encounter the "No route to host" error, you may need to include a zone index in the address. For example, instead of fe80::6a5b:35ff:fed1:4633 (which does not contain a zone index), use fe80::6a5b:35ff:fed1:4633%en0.

Additionally, if you changed the default **port number** on the server side, make sure to adjust the port number accordingly.

Click **Connect**, and the client will attempt to connect to the server. If the connection is successful, continuous throughput testing will commence and will continue until you click **Disconnect**.
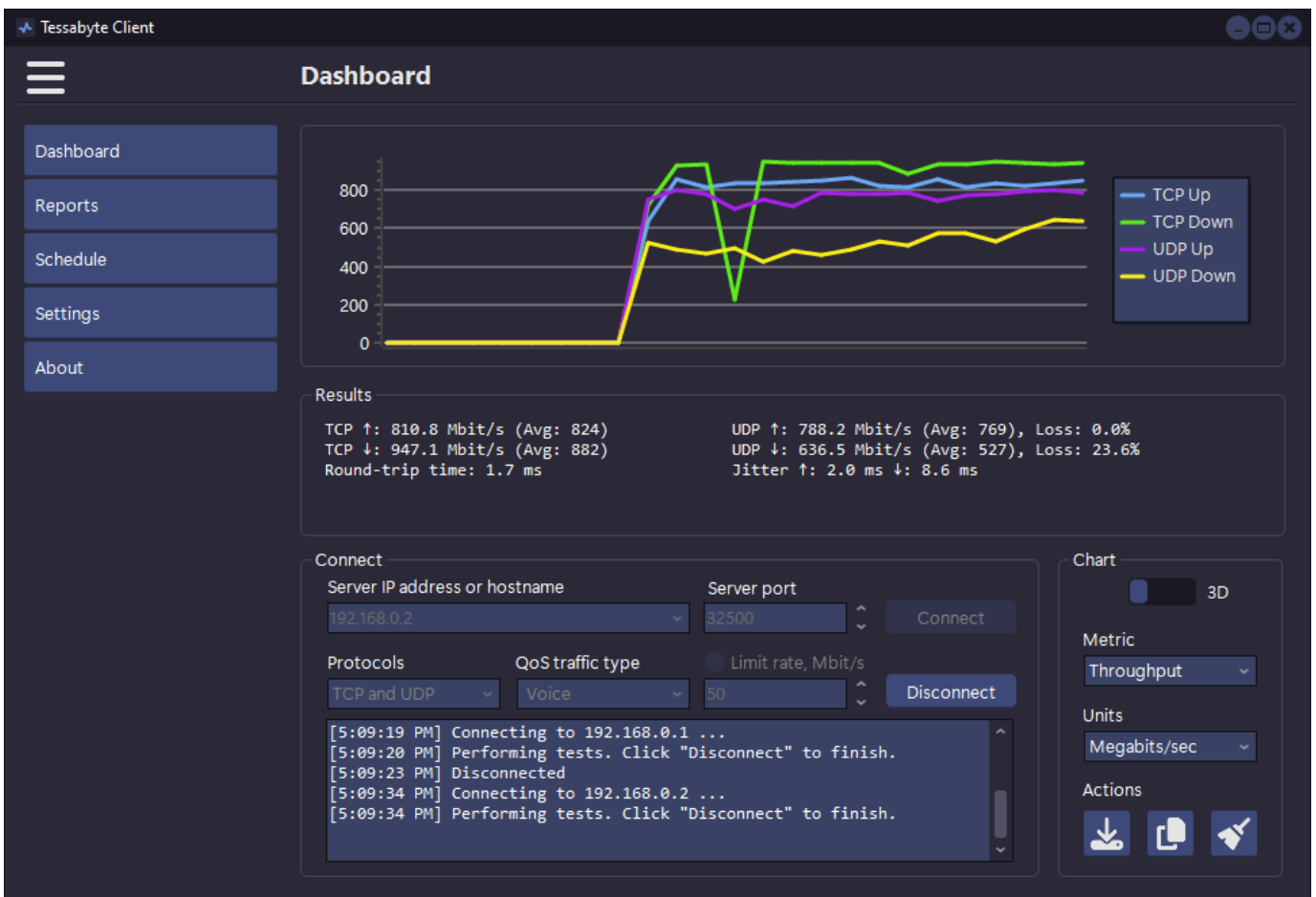
The client window displays TCP and UDP upstream and downstream throughput values (both current and average), loss percentage for UDP streams, round-trip time (RTT), and jitter. The same data is illustrated by a dynamically updated chart.

The **Protocols** drop-down list offers three options for testing your network link: **TCP and UDP**, **TCP Only**, and **UDP Only**, which you can select depending on your testing requirements. You can also configure the **QoS traffic type** and **Limit bitrate**, which will be discussed in detail below.

The **Chart** panel contains several interface elements that control how the data is presented. The **3D** switch toggles the **3D** view on and off, the **Metric** drop-down list allows you to select which data subset to display: **Throughput**, **UDP Packet Loss**, **RTT,** or **Jitter** (alternatively, you can switch between the respective charts using the **Ctrl + 1..4** keyboard shortcut). Finally, the **Units** drop-down list can be used to change the measurement units: **gigabits per second**, **gigabytes per second, megabits per second**, **megabytes per second**, **kilobits per second**, or **kilobytes per second**.

Please note a few important points:
- In the context of data rates, decimal prefixes are typically used with their standard SI interpretation. In other words, in this application, one megabit equals 1000^2 bits rather than 1024^2 bits.
- **Jitter** is measured according to the widely accepted method outlined in RFC 3550.
- If you select **TCP Only** as the testing protocol, the UDP-based metrics—**UDP Packet Loss** and **Jitter**—will not be available.

The **Actions** group of buttons offers quick chart actions: you can **save** the chart image, **copy** it to the clipboard, or **clear** the chart.

The status log window at the bottom displays messages about the current application's status.

QoS Testing

Advanced users might want to use the **QoS traffic type** control to specify the **Quality of Service traffic type** associated with the TCP and UDP data streams sent and received by the application. A description of QoS and related standards and technologies, such as WMM, 802.11e, DSCP, and 802.11p, is beyond the scope of this manual. However, in brief, there are two reasons why you may want to use this functionality:

- To check how different QoS traffic types affect throughput. In a properly designed WLAN using enterprise-class APs, throughput values for high-priority traffic should exceed those for normal-priority traffic.
- To verify end-to-end QoS network design. In a properly designed LAN or WLAN, QoS-tagged traffic must traverse the entire network from the source to the destination through both wireless and wired segments, which may use different technologies and protocol implementations. When testing this scenario, you should use Tessabyte Throughput Test for generating QoS-tagged traffic, and then use packet capture and analysis tools to inspect the packets and verify the QoS or DSCP values in the captured packets.

The table below summarizes different QoS traffic types that you can use. Please note that not all the QoS types available in the application and described below have corresponding WMM access categories. In practice, this means that when you run Tessabyte Throughput Test on a WLAN client and select a QoS type with no WMM mapping, your Wi-Fi adapter driver might fail to QoS-tag packets altogether.

QoS Type	Description
Best Effort	Flow traffic has the same network priority as regular traffic not associated with QoS. This traffic type is the same as not specifying priority, and as a result, the DSCP mark and 802.1p tag are not added to the sent traffic. Corresponds to the WMM AC-BE access category. On macOS, packets are tagged with Class Selector CS0.
Background	Flow traffic has a network priority lower than that of **Best Effort**. This traffic type could be used for traffic of an application doing data backup. Sent traffic will contain a DSCP mark with a value of 0x08 and an 802.1p tag with a value of 2. Corresponds to the WMM AC-BK access category. On macOS, packets are tagged with Class Selector CS1.
Excellent Effort	Flow traffic has a network priority higher than **Best Effort**, yet lower than **AudioVideo**. This traffic type should be used for data traffic that is more important than normal end-user scenarios, such as e-mail. Sent traffic will contain a DSCP mark with value of 0x28 and 802.1p tag with a value of 5. This doesn't correspond to any WMM access category. On macOS, packets are tagged with Class Selector CS2.
AudioVideo	Flow traffic has a network priority higher than **Excellent Effort**, yet lower than **Voice**. This traffic type should be used for A/V streaming scenarios such as MPEG2 streaming. Sent traffic will contain a DSCP mark with a value of 0x28 and an 802.1p tag with a value of 5. Corresponds to the WMM AC-VI access category. On macOS, packets are tagged with Class Selector CS4.

| Voice	Flow traffic has a network priority higher than **AudioVideo**, yet lower than **Control**. This traffic type should be used for real time voice streams such as VOIP. Sent traffic will contain a DSCP mark with a value of 0x38 and an 802.1p tag with a value of 7. Corresponds to the WMM AC-VO access category. On macOS, packets are tagged with Class Selector CS5.
Control	Flow traffic has the highest network priority. This traffic type should only be used for the most critical of data. For example, it may be used for data carrying user inputs. Sent traffic will contain a DSCP mark with a value of 0x38 and an 802.1p tag with a value of 7. This does not correspond to any WMM access category. On macOS, packets are tagged with Class Selector CS7.

Limiting Bitrate

Advanced users might want to check the **Limit rate** box and then specify the maximum bitrate to be used for testing (also referred to as the "target bitrate"). By default, Tessabyte measures the maximum achievable network throughput, but there are certain testing scenarios in which you might not want to fully saturate the bandwidth. For example, your network provider might be using a Committed Information Rate (CIR). Under a CIR, a user is guaranteed a specific amount of bandwidth based on a Service-Level Agreement (SLA), so the user might need to test only, say, 100 Mbps on a 1000 Mbps network. Another scenario is when you want to avoid slowing down network connectivity for other users sharing the same network link.

To limit bitrate, check the **Limit rate** box and specify the bitrate you want Tessabyte to achieve, in megabits per second. For example, if you specify "5000", the target bitrate would be 5 Gbps. It's important to remember that while Tessabyte attempts to keep the throughput rate as close to the specified limit as possible, it cannot match the exact rate with 100% accuracy—especially for UDP. Therefore, you should expect some fluctuations, typically within 5–7% of the target rate.

Note that if you set a very low rate limit—e.g., 1 Mbps—you should expect long TCP testing cycles, unless you specify a custom data chunk size. For example, it takes over a minute to transfer 20 megabytes of data over a link limited to 1 Mbps. Also note that if your limit is above the maximum network link speed (e.g., a limit of 20,000 Mbps (20 Gbps) for a 1 Gbps connection), setting a limit will have no effect.

Generating Reports

Tessabyte Throughput Test records the results of all tests conducted to a compact database. Using the **Reports** page, you can generate a report containing all the measurements for a specific IP address or host, or for all the hosts that have been tested during a given time frame.
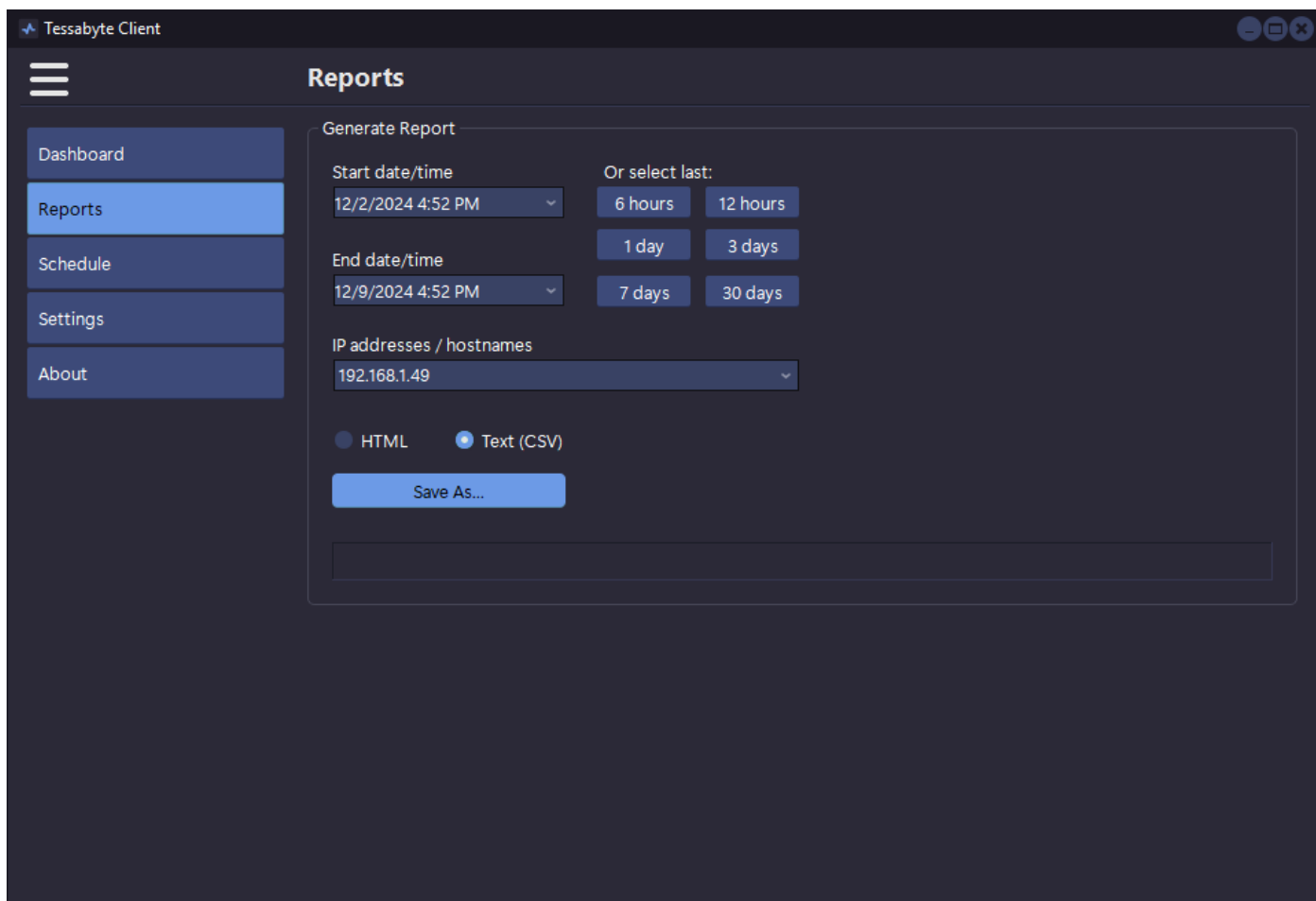
To generate a report, first select the **Start date/time** and **End date/time**, or use the quick selection buttons on the right to set the time range to a specific number of days or hours. Then, select the **IP address or hostname** for which you'd like to generate a report. Finally, choose the report format (**HTML** or **CSV**) and click **Save As** to save it.

Some report settings are customizable; refer to the **Customizing Settings** chapter for more information.

Scheduling Tasks

Throughput tests can be run as scheduled jobs, without initiating them manually. The **Schedule** page provides an interface for creating such automatic tasks.
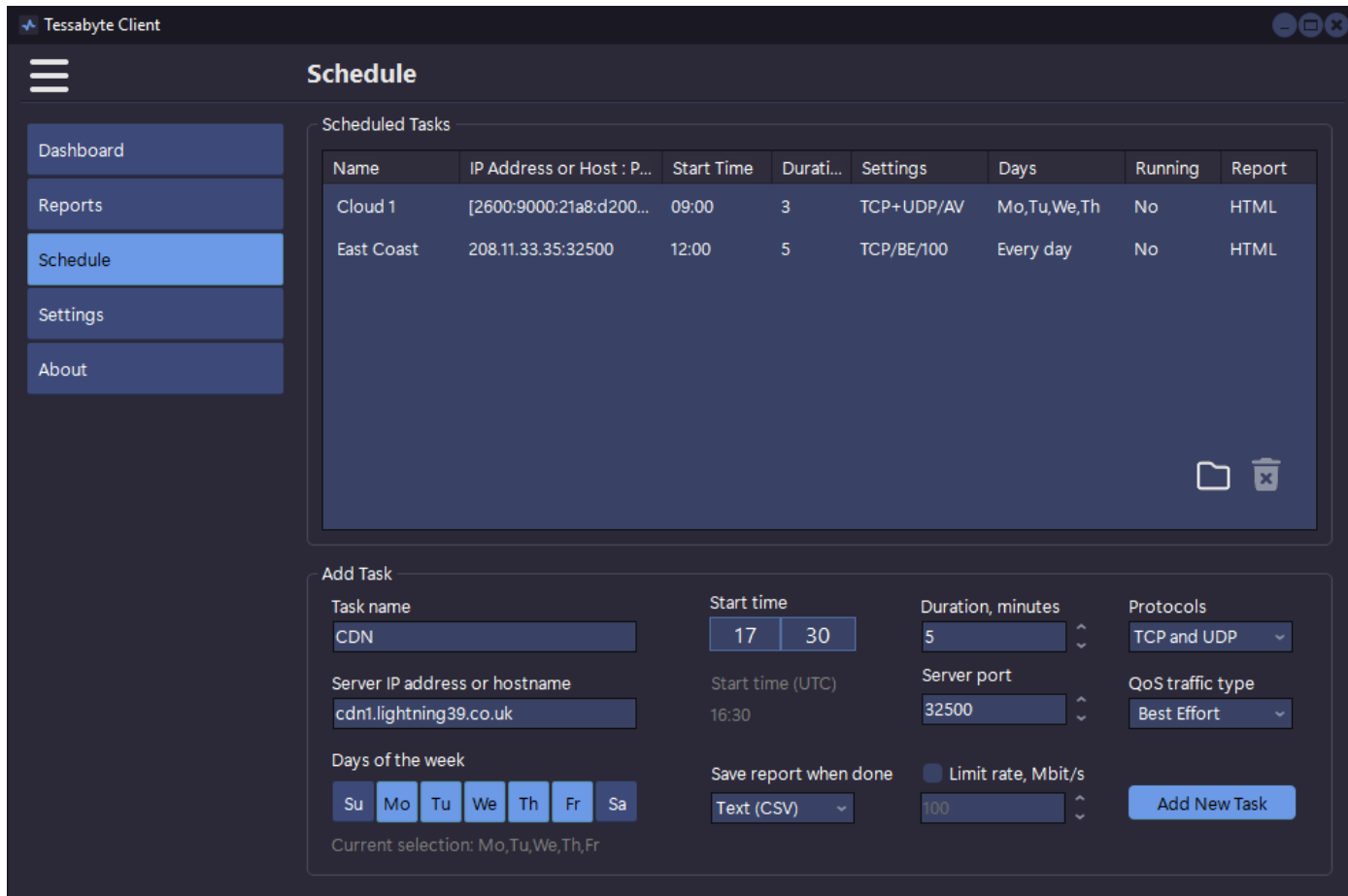
To create a new task, enter a unique task name, the IP address or hostname of the server, select the days of the week by toggling the corresponding buttons, choose the start time and duration (to help you select the correct time, a hint with the corresponding UTC time is displayed below the start time field), and select protocols and QoS to be used. If required, change the default port number and/or limit the bitrate. Upon test completion, the application can generate and save an HTML or CSV report, so choose the preferred format and click **Add New Task**. The new task will be added to the task list in the upper part of the page. The **Running** column indicates the current status and changes from **No** to **Yes** when a task is running.

If a task is scheduled to run while the application is busy conducting a manually initiated test, the manual test will be interrupted in favor of the scheduled task. To delete a task, select it from the list and click the **trashcan** button. To open the folder where reports are saved, click the **folder** button. Reports are named using the task name and end time, for example, **"CDN 26-Dec-2024 15-59-58.htm"**. Additionally, if you hover the mouse over a recently completed task, a tooltip window will prompt you to double-click the task to open the latest report.

A few other important things to remember when working with scheduled tasks:

- Note that the application must be running when the scheduled task's time and day arrive. There is no mechanism to automatically launch the application to conduct the tests.
- The application uses the port configured on the **Dashboard** page to connect to the server.
- Prior to version 2.0 Build 58, Tessabyte used the protocols (TCP and UDP, TCP Only, UDP Only) and QoS settings configured on the **Dashboard** page to connect to the server. In version 2.0 Build 58, we added the option to use specific settings for each test, unrelated to the current dashboard settings. For the tasks created earlier, the **Settings** column will show "Default" and the old behavior (using the **Dashboard** settings) will be applied.

Some report settings are customizable; refer to the **Customizing Settings** chapter for more information.

Customizing Settings

The Settings page lists a few configurable options that allow you to customize the application behavior.
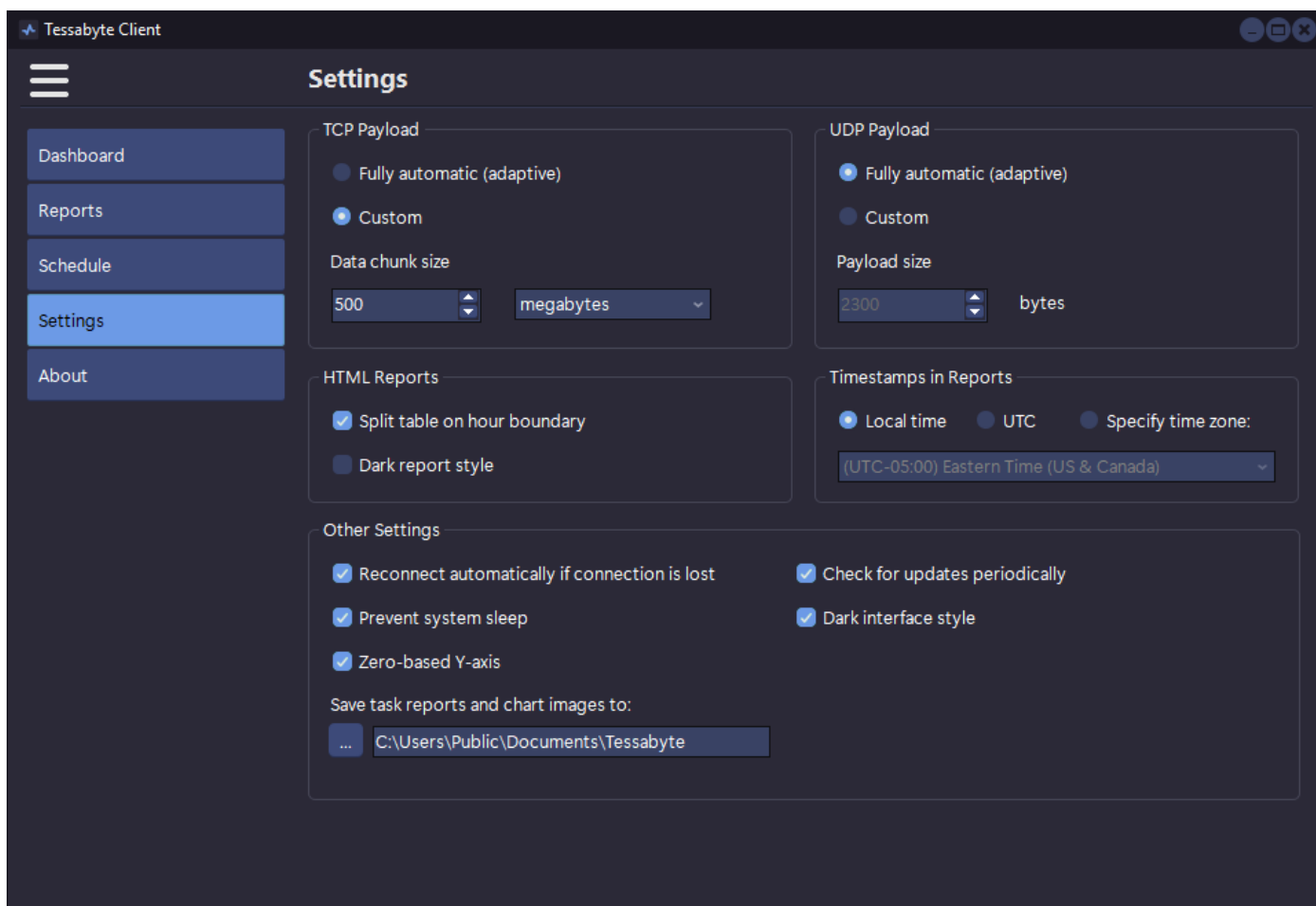
TCP Payload

The **TCP Payload** panel determines how the application selects the data size to be sent and received during each TCP testing cycle (chunk size). By default, Tessabyte employs an **automatic, adaptive** mechanism to determine the chunk size based on multiple factors: connection type (wired or wireless), connection bandwidth, and the results of the previous testing cycle. Generally, the goal is to select a chunk size that can be sent and received within a reasonable amount of time (typically between 0.5 and 1.0 seconds). However, you can select a **custom** data chunk size that best meets your specific testing requirements. Note that selecting a chunk size that is too small will typically produce imprecise results with a volatile chart line, while selecting a chunk size that is too large will result in a slow testing cycle, causing long delays before the next chart update.

For example, consider a 1 Gbps link between a client and a server. Such a link provides about 110 Megabytes per second of application-level throughput under ideal conditions. If you set the chunk size to 1 Megabyte, the chunk would be transferred in just 9 milliseconds, which is too fast to precisely measure actual throughput due to several factors, such as network stack buffering and OS task

scheduling. On the other hand, if you set the chunk size to 2,000 Megabytes, you will get high-quality measurements, but each TCP testing cycle would take 18 seconds × 2 = 36 seconds, as the application needs to measure both uplink and downlink speeds.

UDP Payload

The **UDP Payload** panel determines how the application selects the UDP datagram size to be sent and received during each UDP testing cycle. By default, Tessabyte employs an **automatic, adaptive** mechanism to determine the datagram size based on multiple factors: connection type (wired or wireless), connection bandwidth, and the results of the previous testing cycle. The goal is to select a datagram size that ensures the maximum possible UDP data transfer rate while minimizing packet loss and avoiding fragmentation. Normally, you should not need to select a **custom** UDP payload size, but you may use this option if required for your specific testing needs.

Note that while the maximum theoretical UDP datagram size is 65,535 bytes (minus the size of the IP and UDP headers), in practice, applications rarely use datagrams that exceed the MTU size (about 1,500 bytes). Exceeding the MTU size can lead to UDP packet fragmentation and packet loss. In other words, while you can use a custom size of, say, 4,000 or even 64,000 bytes, you should not expect reliable delivery of such UDP data streams. Additionally, note that on macOS, the maximum supported UDP datagram size by default is 9,216 bytes.

HTML Reports

The options on the **HTML Reports** panel determine two parameters of HTML reports generated by the application. The **Split table on hour boundary** option paginates tables in HTML reports, splitting them by hour. The **Dark report style** option toggles the dark visual theme of the reports on and off.

Timestamps in Reports

The options on this panel determine what kind of time stamps are included in CSV and HTML reports generated by the user. You can configure Tessabyte to use **Local Time** (your computer's clock), **UTC** (Coordinated Universal Time), or a **specific time zone** by selecting one from the drop-down list. For example, if your computer is in the Central European Time zone, your reports can be timestamped using the US Mountain Time zone, should you wish to do so. Please note that the time zones in the drop-down list indicate their standard UTC offset (winter time) for consistency. Tessabyte will still apply the correct offset if daylight saving time is currently active in the corresponding time zone.

Other Settings

Reconnect automatically if connection is lost – when this option is on, the application attempts to

reconnect to the server whenever a connection is lost.

Prevent system sleep – prevents the computer from going into sleep or hibernate mode while the application is running.

Check for updates periodically – when this option is enabled, the application will connect to our website every few days to check if a new version is available.

Dark interface style – switches between the dark and light user interface themes (Windows only; on macOS, the application uses the currently active visual style).

Zero-based Y-axis – toggles between two modes: one where the chart's Y-axis minimum is always set to zero, and another where the Y-axis minimum is dynamic and adjusts based on the actual data.

Save task reports and chart images to – allows you to select a folder where reports generated after running scheduled tasks are saved automatically. The same folder is used to save chart images when the "quick save" button is clicked on the dashboard page.

Understanding Test Results

During each testing cycle, the application performs several tests: sending and receiving TCP data, sending and receiving UDP data, and sending and receiving a time probe packet. Based on these tests, it computes TCP and UDP upstream and downstream throughput values (current, for the latest test, and averaged, for all tests), as well as the round-trip time.

When all tasks in a cycle are completed, a new cycle automatically begins. If the **Protocols** selection on the **Dashboard** is set to **TCP Only** or **UDP Only**, then the UDP part or TCP part, respectively, is skipped. Naturally, if the UDP part is skipped, all the UDP-based metrics—UDP throughput, UDP packet loss, and jitter—will not be available.

Throughput

Throughput (also often referred to as "goodput") is the amount of application-layer data delivered from the client to the server (upstream) or from the server to the client (downstream) per second. Protocol overhead is not included. For example, when we refer to a TCP throughput rate of 1 Mbps, it means that 125 Kbytes of actual data payload were sent between two network nodes during one second, excluding TCP, IP, and Ethernet or 802.11 headers.

Packet Loss

Packet loss is applicable to UDP tests only, because in TCP, all packets must be acknowledged, and no data loss can occur. UDP loss is calculated as the percentage of data lost during transmission. For example, let's interpret the following result:

`UDP Down: 60.00 Mbps (Avg: 55), Loss: 40.0%`

This means that during the latest test cycle, the server sent 10 megabits of data in 100 milliseconds (which translates to 100 megabits per second; actual data amounts and durations may vary—this is just an example), and the client received 6 megabits in 100 milliseconds, while 4 megabits were lost en route.

Some UDP loss is quite common and doesn't necessarily indicate a problem with your network. Refer to the **Dealing with Common Problems** chapter for a detailed discussion.

Round-Trip Time

Round-trip time (RTT) is the length of time it takes for a data packet to be sent from the client to the server and back. The application uses TCP packets for RTT measurements, which is normally a bit slower than the standard ICMP ping.

Jitter

Jitter refers to the variation in packet arrival times, measured as the difference between successive packets' delay. High jitter can cause issues in real-time applications like VoIP and video streaming, where consistent timing is crucial. Tessabyte calculates jitter using the widely accepted method outlined in RFC 3550, which defines it as the mean deviation of packet delay variation. To illustrate the concept, consider the following chart:
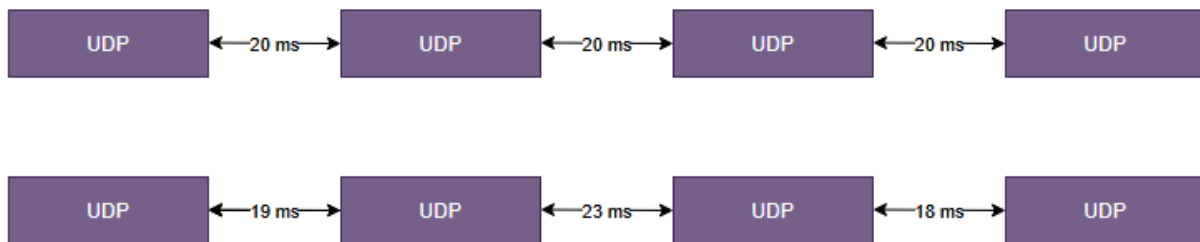
On the sender's side, an application, such as a VoIP phone, transmits UDP packets every 20 milliseconds. On the receiver's side, an application receives these UDP packets. In an ideal world, the inter-packet intervals on the receiver's side remain equal at 20 milliseconds. However, in reality, these intervals deviate from the expected 20 milliseconds —some packets arrive faster, while others may be delayed. This variation is known as jitter. Lower jitter values indicate a more stable network connection, while higher values may suggest congestion or network instability.

Recommended Jitter Limits for Different Applications

Application	Acceptable Jitter (ms)
VoIP	<30 ms (Ideal: <10 ms)
Video Conferencing (Zoom, Teams, etc.)	<40 ms
Online Gaming	<30 ms (Ideal: <10 ms)
Live Streaming (Twitch, YouTube Live, etc.)	<50 ms
Standard Video Streaming (Netflix, YouTube, etc.)	<100 ms

Dealing with Common Problems

Sharing Bandwidth with Other Clients

As the Tessabyte Throughput Test server can handle multiple clients concurrently, when more than one client is connected to the server, the bandwidth is inevitably shared among them. Unless the server has a very wide bandwidth connection that far exceeds the clients' bandwidth, you should expect that multiple concurrent client connections will affect the per-client throughput.

For example, consider a local LAN segment with a 1 Gbps link between a server and a few client computers. If you run a Tessabyte client on just one of these client computers, you can expect a maximum throughput of about 0.9 Gbps. If three clients are connected, there will inevitably be time periods when all three are sending a TCP stream to the server, as the testing cycles (TCP Up ➜ TCP Down ➜ UDP Up ➜ UDP Down) are not synchronized. During such times, throughput might drop to approximately 0.3 Gbps per client. However, when only one client is sending data at the given moment, throughput may reach 0.9 Gbps. To summarize, the chart may appear volatile when multiple client connections are active.

Note that the client's log window on the Dashboard page keeps you informed about the number of currently connected clients whenever this number changes. This way, you can always determine whether the connection is exclusively yours or if other clients are sharing the bandwidth with you.

Low TCP Throughput

If you observe consistently low throughput compared to what you expect from the network link between the server and the client, then first of all—congratulations! You've used the right tool to discover the problem. However, after a brief celebration, let's dig into the issue. The most common reasons for a throughput below the expected level are:

- **Wi-Fi networks:** If you are measuring throughput over a Wi-Fi network, it's important to remember that the maximum PHY rate advertised by the wireless AP or router is only a theoretical maximum and is never reached in practice. In real-life scenarios, many constraints limit Wi-Fi throughput, such as:
  - Client capabilities: For example, only 2 MIMO streams versus 4 MIMO streams supported by the AP, or a 160 MHz channel width versus 320 MHz supported by the AP.
  - Protocol overhead: The overhead caused by protocol headers.
  - Signal strength: Weak signals reduce throughput.
  - RF noise: Radio frequency interference can degrade performance.
  - Medium contention: When multiple clients are connected, they compete for the shared wireless medium, further reducing throughput.

  This leads us to the next point.

- **Competing traffic (congestion):** Network traffic competes for available bandwidth, which can lead to congestion. This issue is particularly significant in Wi-Fi networks, where multiple clients share a small number of radio channels and employ collision avoidance mechanisms to manage access. By contrast, modern wired networks—such as those using gigabit switches—are designed with sufficient internal capacity to handle the combined bandwidth of all ports simultaneously. For example, a 4-port gigabit switch can theoretically support up to 4 Gbps of aggregate traffic (1 Gbps per port, in both directions). That said, while traffic congestion is less common on wired local segments, it can still occur in certain scenarios.
- **CPU or system bottlenecks:** High CPU usage or poor NIC offloading performance limits throughput. Modern NICs support offloading tasks (e.g., checksum, segmentation), and missing these optimizations leads to CPU overload.
- **Intermediate network devices:** Routers, firewalls, or proxy servers in the path can throttle or delay traffic. Misconfigurations, traffic shaping, or policy-based throttling can restrict throughput.
- **Faulty cabling:** Damaged cables can cause signal degradation, resulting in packet loss and frequent retransmissions. Poorly shielded cables or cables not meeting category specifications (e.g., Cat5e, Cat6) may suffer from electromagnetic interference or crosstalk. Finally, modern Ethernet devices use auto-negotiation to determine the best possible link speed (e.g., 1 Gbps, 100 Mbps), so faulty cabling or poor-quality cables may cause the link to fall back to a lower speed, such as 100 Mbps instead of 1 Gbps.
- **Thermal throttling**: If a network card overheats, it may throttle and reduce its processing speed. This can cause lower throughput and packet loss, especially at high data rates. Overheating in switches or routers can also cause them to throttle their packet-processing speed, introduce latency, or temporarily drop packets to maintain thermal stability.

Note that this is by no means a comprehensive list; there are many other potential causes of low throughput, such as high network latency and small TCP window size, packet fragmentation, etc.

Abnormally High TCP Throughput

Sometimes, the reported throughput rate might exceed the theoretical maximum. For example, while testing a 5 Gbps link between two computers, you might observe a spike in TCP downlink or uplink throughput that exceeds 5 Gbps. This is typically caused by buffering at the operating system level, where a relatively large chunk of data is accumulated before being "pushed" to the receiving application.

If you observe such spikes, the best solution is to use a custom TCP payload size. The payload size should be large enough to take at least half a second to transfer over your network link. To learn how to use a custom TCP payload size, refer to the **Customizing Settings** chapter.

Network Connection Down After a Few Testing Cycles

This might not be a good sign for your network infrastructure. Any throughput testing tool generates a considerable load on the network hardware, sometimes to the extent that the hardware fails to reliably handle the high load. While the specific reason may vary (for example, the adapter chips might overheat, triggering a self-protection mechanism such as thermal shutdown), this is often an indication that the infrastructure being tested is unable to pass the stress test.

High Uplink and/or Downlink UDP Loss

Generally, non-zero loss of UDP packets is totally normal. Unlike TCP, the UDP protocol is connectionless and does not guarantee delivery. Several factors may contribute to such loss:

- **Network congestion**: Network devices (routers, switches) have limited buffer space. When the network is overloaded, packets are dropped.
- **Packet rate exceeds hardware capacity**: If the sending node transmits packets faster than the receiving node or intermediary hardware (e.g., routers) can process, packets will be dropped.
- **Buffer overflows**: Both sending and receiving nodes have network buffers. If these buffers overflow due to a high packet arrival rate, packets are dropped.
- **Misconfigured network parameters**: A Maximum Transmission Unit (MTU) mismatch can cause packets larger than the MTU to be fragmented or dropped. Ensure you are not using a custom UDP packet size that exceeds the MTU (configured on the **Settings** page). Some hardware may drop fragmented UDP packets. For example, if your custom packet size is 2,000 bytes, your UDP packets will likely be fragmented.
- **QoS settings**: UDP traffic may be deprioritized in favor of other protocols, such as TCP or UDP with a higher QoS tag. Use the QoS control to check how a high-priority QoS tag affects performance.
- **Slow/old computer**: Try running Tessabyte on a different computer. Your current computer may be old, slow, or its CPU might be loaded by other processes.

High Downlink UDP Loss in WLANs

High downstream UDP loss is quite common when running the client on computer with a Wi-Fi link. UDP traffic is not acknowledged, meaning the sender can transmit as much traffic as the network can handle without "caring" about how much of it is lost. If you run the server on the wired side of the network, a typical computer equipped with a gigabit adapter can send hundreds of megabits per second. This data will first reach a switch, which might be the first bottleneck, and then the access point, which is also often a bottleneck. In a multi-client environment, even modern 802.11be APs cannot provide a gigabit downlink to all clients simultaneously. As a result, many UDP packets might be lost en route. However, this is the only way to determine the maximum downstream UDP

throughput.

100% Downlink UDP Loss

A 100% downlink UDP loss is typically caused by a firewall or NAT issue. This means that the UDP data sent from the server cannot reach the client. During UDP testing, the client sends upstream UDP traffic to the server from an arbitrary UDP port to the server port (32500 by default). The return downstream traffic originates from an arbitrary server port and is sent to a port on the client side, determined by the following rule: (server port, 32500 by default) + 1. If this port is unavailable, the next available port is selected. While this information can help you configure a firewall or port forwarding rule, UDP traffic cannot easily traverse NATs. If you want to test UDP, you should avoid using NATs or consider using IPv6, which generally eliminates the need for NAT.

About

Tessabyte Throughput Test is brought to you by Netmantics LLC.

Comments? Questions? Feature requests? Visit us at **www.netmantics.com**.