API CUSTOM DAILY

How to build a network statistics panel in a live video call

Help your call participants resolve network issues with Daily's "getNetworkStats()" method



Jess Mitchell
May 4, 2023 / 7 min read



When building software, we as developers often focus on the "happy path" through an app's UI. The happy path refers to the intended path a user takes through an app with the predicted outcome. In other words, the user does what we think they will, and the app responds how we want it to. If only that were the only path!

The other side of building software is when things don't go as planned. We know that even the most reliable software needs to handle less-than-ideal situations. Poor network conditions, for example, can negatively impact a user's experience of an app or even prevent them from being able to use it. In a video call app, it can cause an increase in latency, which makes the video lag or freeze. And even for the calmest person, a laggy video call can be a horribly frustrating experience.

In this post, we'll cover how to help video call participants understand when their network conditions are the culprit for video call issues. Primarily, we'll look at:

- How you can update your video call UI to help users know what their network's status is.
- How Daily handles poor internet connections in our Daily Prebuilt video component's design.
- What the `daily-js` and `react-native-daily-js` `getNetworkStats()` method is and what data it returns.
- How to build features similar to Daily Prebuilt's using the `getNetworkStats()` method.

This post will be an overview of how to approach these topics, rather than a coding tutorial.

Video call features to improve user experience during a poor internet connection

An important part of app UI (user interface) design is clearly communicating information that the user might find helpful. In a video call, if there's a delay in the audio or video being sent, the call participant might wonder, "Is this app broken? Is there a setting I can change? Is it my internet? Do I have too many apps open?" Ultimately, they'll want to know, "Is there something I can do to fix this?"

When poor network conditions is the problem, there are two main ways to help your users:

1. Let them know the issue is caused by their network connection (issues with either packet loss or low available bandwidth).
2. Provide some options for what they can do to help improve the video call experience given the circumstances.

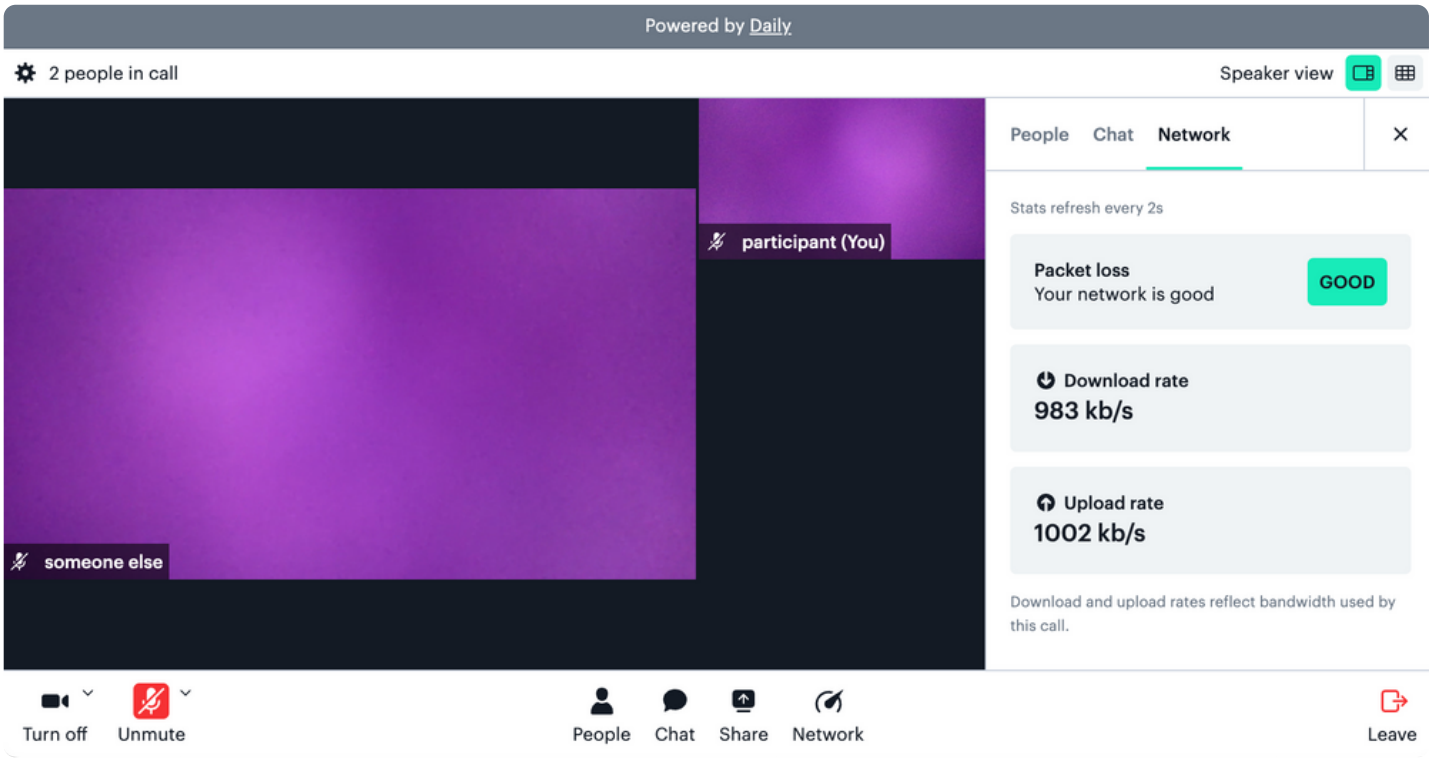
Let's look at the first point: communicating the problem.

How Daily communicates network issues in video calls

Daily offers several SDKs and tools for developers building their own video call software, but we're also our own customer. The Daily Prebuilt video call component can be embedded into any web page in just a few lines of code. It is also a useful example of how we recommend handling common video call features.

One tool Daily Prebuilt's UI offers is an optional Network panel. This can be turned on for any Daily room in the Daily dashboard or with the `enable_network_ui` option when creating or editing a Daily room with the REST API.

When turned on (it's off by default!), the Network panel is made available in Daily Prebuilt calls for that room.



This allows us to communicate the information we have in our API to the call participant who might be trying to understand what's going on if a video call starts lagging. They can check for packet loss, as well as the current download and upload rate. If the packet loss is bad (i.e., if it's high), the call quality will worsen. In extreme cases, the call will look like it's frozen.

> 💡 Read Daily CEO's blog post on our Global Mesh Network to learn more about packet loss.

Communicating this information helps the call participant know if their network is the issue or at least rule it out as the culprit.

Daily provides all of the information seen in the Network panel with our own public APIs. This specific information is made available by the `getNetworkStats()` method available in Daily's JavaScript SDKs. (More on this method below.)

Now that we've seen an example of communicating network issues with users, let's look at how we can help them resolve the issues so they don't just give up on the video call.

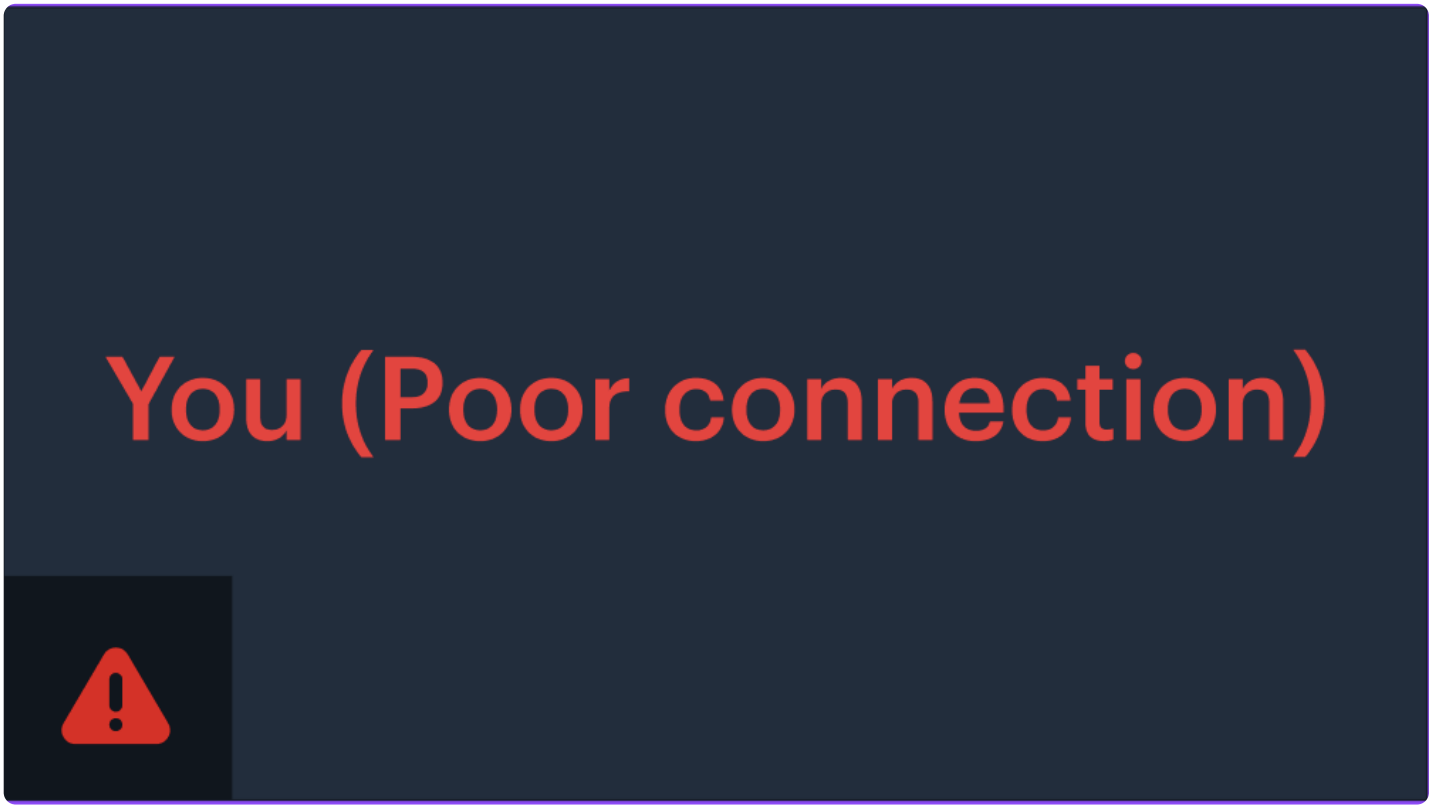
Providing users with actions to improve network issues

Video can be a pretty costly feature in terms of app performance. When a person is in a call and having network issues, we want to help them improve their situation without having to cancel their call. One main way we can do this is by encouraging users to temporarily turn off their video when network conditions are poor. This will allow the call to continue with audio until the connection improves.

It can be difficult at first to know what the issue is when your network conditions change. It could be a temporary lull that resolves itself, or it could be something where the call participant could take action to improve the issue. As an example of the latter, if you're on a sketchy WiFi connection and want to tether to your phone instead, you can turn your video off until that better network connection is available.

Knowing the root issue can vary, Daily Prebuilt handles these network issues gracefully by managing your video for you. When the connection is too poor to have an enjoyable video call experience, Daily Prebuilt will temporarily turn off your video and display a warning sign to let

you know about the issue. The call can continue with your audio and the video/audio of any other participants in the call.



This small change in the call allows call participants to continue speaking seamlessly, instead of the call going into a state where no one can hear or see each other.

Using Daily's `getNetworkStats()` method to handle variable network conditions

Now that we know why it can be good to communicate network condition data and how to help maintain a positive video call user experience in poor conditions, let's see how to actually accomplish these goals with Daily's client SDKs for JavaScript and React Native.

Daily provides an instance method in both `daily-js` and `react-native-daily-js` called `getNetworkStats()`.

This method is available on the main call object and usable after the participant joins a call. It will return the current network stats for the local participant and is invoked as follows:

```
callObject.getNetworkStats()
```

First, let's look at an example response object to see what information is returned to us:

```
{
  "stats": {
    "latest": {
      "recvBitsPerSecond": 1178543.4568373028,
      "sendBitsPerSecond": 1082780.0045457026,
      "timestamp": 1620311506953,
      "videoRecvBitsPerSecond": 1028897.7824626606,
      "videoRecvPacketLoss": 0,
      "videoSendBitsPerSecond": 987321.7967671315,
      "videoSendPacketLoss": 0,
“totalSendPacketLoss”: 0,“totalRecvPacketLoss”: 0,
    },
    "worstVideoRecvPacketLoss": 0.008333333333333333,
    "worstVideoSendPacketLoss": 0
  }
  "threshold": "good",
  "quality": 100
}
```


Here is that information again but with each key defined:

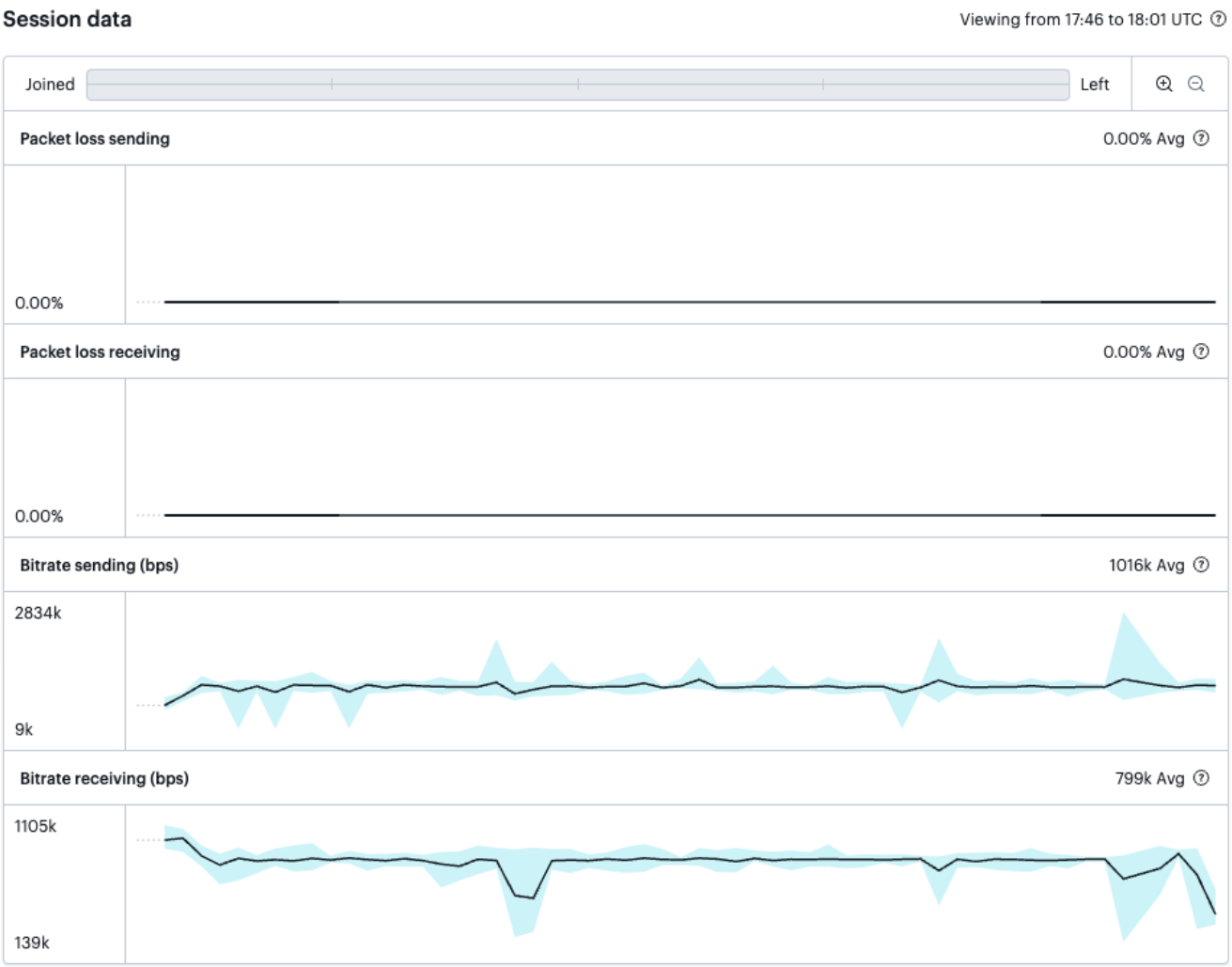
```
"stats": {
    "latest": {
      "recvBitsPerSecond": 0, // total data from video and/or audio being
received in bits per second
      "sendBitsPerSecond": 0, // total data from video and/or audio being sent
in bits per second
      "timestamp": 0, // a unix timestamp (seconds since the epoch)
      "videoRecvBitsPerSecond": 0, // total data from video being received in
bits per second
      "videoRecvPacketLoss": 0, // estimated data lost from the received video
over a two second period
      "videoSendBitsPerSecond": 0, // total data from video being sent in bits
per second
      "videoSendPacketLoss": 0 // estimated data lost from the video being sent
over a two second period
    “totalSendPacketLoss”: 0, // estimated data lost from the audio and video
being sent over a two second period
“totalRecvPacketLoss”: 0, // estimated data lost from the received audio and
```

```
video over a two second period

    },
    "worstVideoRecvPacketLoss": 0, // the highest videoRecvPacketLoss number
seen over a two second period during the call
    "worstVideoSendPacketLoss": 0 // highest videoSendPacketLoss number seen
over a two second period during the call
  }
  "threshold": "good", // an assessment of the current network quality
  "quality": 100 // a subjective calculation (by our expert engineers) of the
current network quality on a scale of 1–100
}
```

The `threshold` value, for example, is one value we show in the Network panel.

With these values, you can build any interface you think would help your video call participants. It could be a panel like the one in Daily Prebuilt or even a live graph of network changes over time. The graphs in the Daily dashboard are an example of how this could look. Graphs similar to the dashboard's could be built using the `/logs` endpoint, rather than `getNetworkStats()`. (Note: You need to use the `/logs` endpoint over `getNetworkStats()` any time you're trying to retrieve session data after the call has ended.)



A basic code example polling `getNetworkStats()`

Since network conditions change over time, it is most common to poll the `getNetworkStats()` method every 15 seconds or so.

As a final step, let's look at a basic example of how you could poll `getNetworkStats()` while in a Daily video call.

```
// Create an HTML element that can be used as a container before running this
block of JS
const targetEl = document.getElementById('callframe');
const callframe = DailyIframe.createFrame(
  targetEl,
);
callframe.join({url: room_url});
```

```
// Start with an empty array to store network stats over time.
const statsArray = [];

const getStats = async () => {
    const currentStats = await callframe.getNetworkStats();

    // Add stats to stats array after the promise resolves
    statsArray.push(currentStats);
}
// Start interval for fetching stats that will run every 15 seconds until you
leave the call
const interval = setInterval(() => getStats(), 15000);
```

Once the interval is initialized, `getNetworkStats()` will be called every 15 seconds, and the results will be added to `statsArray`. As the array is added to, you can decide how to display that data to your customers, such as a graph or Network panel that regularly updates itself.

Conclusion

In this post, we went through how to use Daily's `getNetworkStats()` call object instance method to display network data to call participants. To learn more, read our call quality guide and infrastructure blog posts. To see a demo example of `getNetworkStats()`, try our Daily Prebuilt demo app.

Never miss a story

Get the latest direct to your inbox.

Email*

Country*

Join peerConnection - our video/WebRTC developer community.

By submitting, you consent to allow Daily.co to store and process the personal data from this form to fulfill your request.

Subscribe



daily

Interactive Live Streaming for Dummies: Daily.co Special Edition

Get your free copy

Platform

Video SDK

Interactive Live Streaming

Live Streaming

Recordings

Audio SDK

Daily Prebuilt

Security

Developers

Get Started

Client SDKs

Reference docs

Mobile docs

Prebuilt

Get API key

Company

Customers

Community

Announcements

Pricing
Security center
Partners
In the news
Startup program

Q&A
Workshops
Github
Join us

Use Cases

Telehealth
Live Commerce
EdTech
Event Tech
Creator Technology
Virtual Collaboration
Social and Gaming
Read case studies

Dev Resources

Blog
Guides
Demos
Support
API status
Changelog

Contact us

Contact Us
Live chat
Support
Japan

Resources

Blog
Resource center
Events
Twillio alternative
Vonage alternative
Agora alternative
Videosaurus
Site index

Security & Privacy

Security center
Privacy policy
Terms of service
Cookie policy