testRTC

Deliver Better WebRTC Services

Performance Whitepaper

A comparative analysis of Vonage Video API

SPONSORED BY: VONAGE

Table of Contents

testRTC

Overview

WebRTC is the enabler of many of our real time communication interactions. While the technology itself is open, the quality derived from a solution is greatly dependent on the infrastructure and the specific implementation of each vendor.

Vonage, a global leader in cloud communications, commissioned testRTC, the world's most powerful WebRTC testing and monitoring platform, to conduct an analysis of its Vonage Video API quality and performance, in order to figure out its comparative quality versus other Video API vendors.

testRTC develops and licenses a testing and monitoring service designed for WebRTC-based communications. As part of the service, the company offers a robust WebRTC testing and analysis platform, and devises the test scripts and methodologies to conduct objective, repeatable test scenarios.

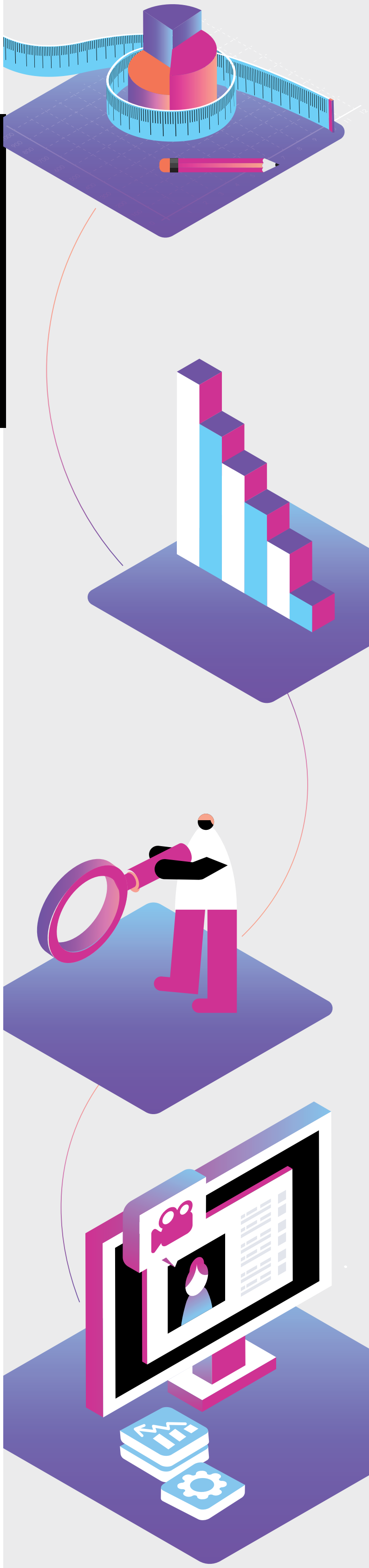
As part of this project, testRTC has written the applications used in front of the Video API vendors, defined and implemented test scenarios using the testRTC platform, conducted the testing and analyzed the results.

Throughout the process, the goal and intent was to understand the behavior of each platform and the resulting media quality associated with it. testRTC tested 3 different vendors, over 4 use cases, with 3 different group sizes. This resulted in a total of 36 scenarios, each of which was executed multiple times to validate repeatability as well as stability of the scenario.

Our testing shows that Vonage has put a lot of energy and attention to the call quality on its platform, especially in large group calls scenarios. In many cases, the Vonage implementation matched the behavior expected to optimize the experience for multiple users. This is done by taking into account total bitrate and CPU consumption associated with more media streams that need to be processed.

We are certain that the out of the box, naive implementation that we've created for the purpose of this analysis can be optimized and further improved across all vendors. Our intent wasn't to provide these optimizations ourselves, but rather to see what optimizations are provided by the vendors directly.

In this whitepaper, we will first review the scenarios and testing environment. From there, we will go through a thorough analysis of the results. In the analysis, we look at CPU consumption, bitrate, jitter and packet loss. Special focus is also given to screen sharing and limited bandwidth consumption, where we will try to look deeper at how the various vendors overcome challenges of these use cases.



Assessing Media Quality in WebRTC

When assessing the media quality of a WebRTC application, there are many different aspects that need to be considered.

These aspects include:

User devices

Different users use different devices to connect to services. To be able to conduct proper performance analysis it was important to standardize the devices and mediums used for the test. testRTC selected browsers since these are the most common environments today for WebRTC and because they can be automated and profiled without direct access to the source code of the tested application.

Networks

Networks are dynamic in nature. They fluctuate based on the local use of devices using them as well as traffic in the WAN (Wide Area Network). To conduct comparative analysis, the network conditions have to be controllable.

Predictability and repeatability

Repeating certain scenarios should have similar results, providing predictability to the system. This enables testing across vendors and use cases with high level of confidence in the results collected.

Analysis

The analysis itself needs to be objective, relying on KPIs (Key Performance Indicators that are highly measurable). The KPIs used directly correlate to the media quality perceived by the users.

When Vonage set out to conduct this performance analysis, we strived to create a stable and repeatable environment that enables an objective analysis. By using testRTC, planning the test environment and the test processes, we reached our goals.

Test Environment

Our plan was to test Vonage Video APIs and other well known video API platforms in a set of predefined scenarios. We focused on how these platforms behave on the network in the various scenarios.

For the tests themselves we used the reference application provided by Vonage and developed simple reference applications for the other API platforms. We aimed to make the UI as close as possible and as simple as possible to reduce its effect on performance.

Vonage

Vendor B

Vendor C

For the performance testing itself, we've used the testRTC platform, configuring it in the following manner:

- A testing probe was assigned for each user/browser/participant
- Each probe was configured with 4 vCPUs and ~2 Gb of memory, giving it ample processing power and resources for the tasks involved
- Chrome 88 was used
- Camera sources were identical with VGA resolution. Since we aimed for larger calls, we wanted to optimize for bandwidth and screen layouts. Using HD would be counterproductive in this case
- All participants were allocated and launched from Google Cloud's us-west1 data center (located in The Dalles, Oregon, North America)
- For each test run conducted, we've allocated fresh machines, starting from scratch
- Each test scenario was executed for a period of five minutes. This enabled us to analyze bitrate ramp up, resiliency and network variations. Using shorter periods of time wouldn't yield accurate results in our scenarios, and longer periods of time wouldn't produce any additional data
- Each test scenario was tested with 2, 4 and 8 participants

Test Environment (cont'd)

The following test scenarios were selected and conducted for the performance analysis:

1

Normal conditions - all participants join with audio and video. The network is configured to run without any throttling.

2

Screen sharing dynamic content - first participant to join shares a YouTube video with dynamic content.

3

Screen sharing static content - first participant to join shares a YouTube video with static content (a slide deck).

4

Limited bandwidth conditions - first participant to join has its network configured to dynamically limit available bandwidth to 500kbps on both incoming and outgoing traffic for a period of 100 seconds in the middle of the scenario.

Each scenario was tested multiple times with different numbers of participants. testRTC wanted to make sure results aren't random in nature and found the results to be reproducible in nature.

The main metrics testRTC focused on were bitrate, packet loss, jitter and CPU consumption.

Performance Results and Analysis

testRTC conducted the tests for each scenario multiple times, taking into account potential variability in network conditions and infrastructure. In all cases and for all vendors, we've seen stable results across the same scenario when executed multiple times.

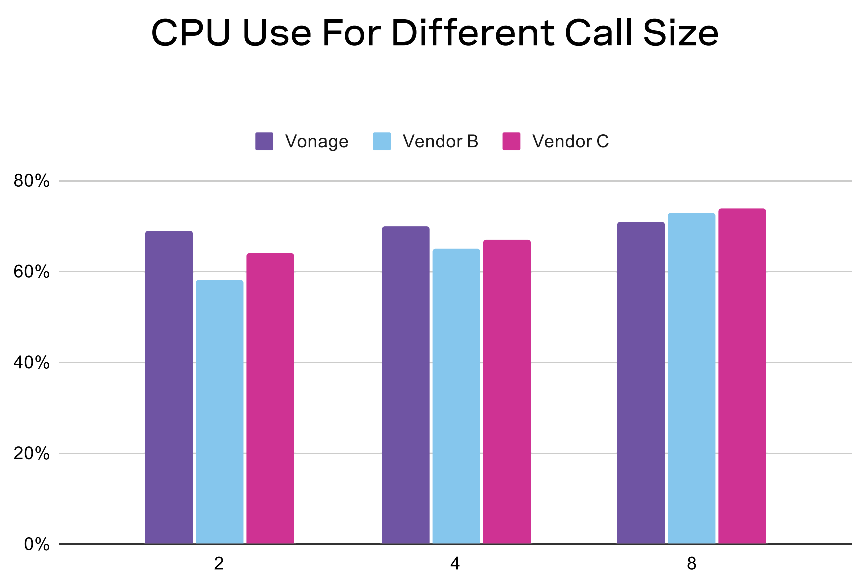
We then looked to analyze the results collected, looking at performance indicators selected. What we were looking for in our results were:

- **CPU consumption.** The lower CPU consumption is the better the experience is for the user and the more types of devices the service will be able to support
- **Bitrate.** The lower the bitrate on the incoming and outgoing streams, the bigger the calls that can be supported and the less network resources are consumed. Here it is important to make sure bitrates aren't too low, as that would negatively affect media quality
- **Jitter.** The lower the jitter values observed the more stable the media stream is
- **Packet loss.** We were aiming for no packet loss on all scenarios. The only scenario where packet loss was observed as part of the test itself was when we limited the available network bandwidth
- **Resolution and frame rate.** We've focused on these parameters when looking at the generated screen sharing video streams in the relevant scenarios. Our goal here is to maintain the original screen resolution at a high frame rate
- **Ramp up time.** When we limited the bandwidth available, we wanted to see how much time it takes for a service to ramp up its bitrate once the bandwidth limit was removed. The faster the ramp up the better

CPU Consumption

CPU use on WebRTC clients is important for media quality. The higher the CPU consumption, the more energy is consumed. On mobile devices this translates to shorter battery life and devices heating up. In both desktop and mobile it means less CPU available for other tasks that need to be performed by the application and in edge cases of high CPU consumption can lead to lip sync issues, packet losses and loss of connectivity.

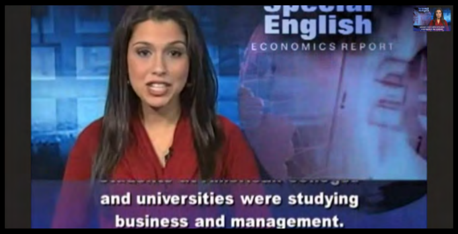
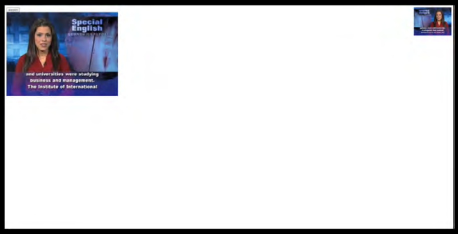
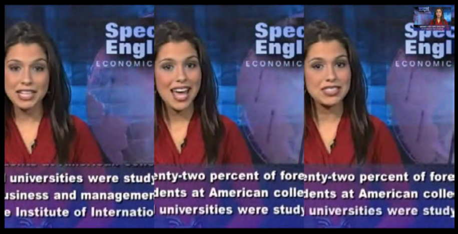
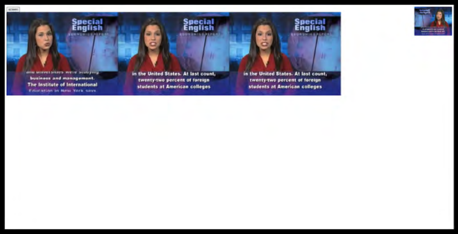
We started off by reviewing the CPU consumption of the participants during the test scenarios.



The graph above shows the average CPU consumption for each participant in our normal conditions test scenario with different numbers of participants - 2, 4 and 8.

Vonage CPU consumption in the 1:1 scenario (2 participants) is higher than the other vendors but then gets lower than the other vendors as the number of participants increases to 8.

testRTC traced back the reason to the difference of rendering: Vonage reference UI renders bigger video tiles in 2-way and 4-way calls than the other vendors:

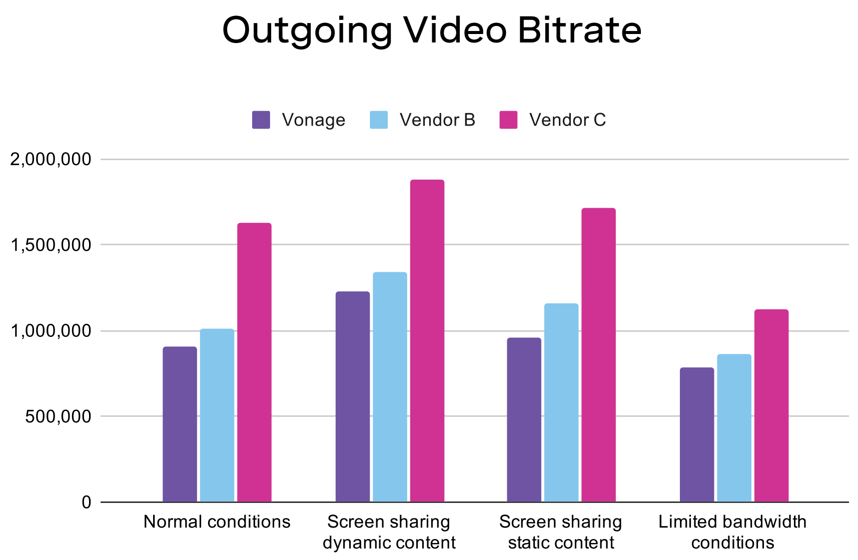
As the call size increases, the difference in CPU consumption between the vendors reduces. This relates solely to the application processing and not the API vendor itself.

We have found all vendors analyzed to offer similar CPU consumption values across all test scenarios: all test results showed CPU consumption of 68-76%, with screen sharing scenarios taking roughly 4% higher CPU consumption than the normal conditions and limited bandwidth scenarios across all vendors.

Bitrate

Video calls consume bitrate. A good service would try to consume as little bitrate as possible while maintaining media quality, especially as the meeting size grows.

In our analysis, we focused on video bitrates, looking at outgoing and incoming video bitrates separately.



Outgoing Video Bitrate

The outgoing video bitrate of the various vendors was quite different from one another, each taking a different approach.

From the graph above, we can deduce that the content affects the actual bitrate. The dynamic screen sharing content required from all services to increase their bitrate, more than the rest of the test scenarios used.

Vonage and Vendor B tried using lower bitrate than Vendor C on the uplink. This makes perfect sense, especially considering the webcam source configured for normal and limited bandwidth conditions has VGA resolution, where 800kbps is enough for good video quality.

All 3 vendors use simulcast with 2 layers in their solution. Each had a different bitrate configuration:

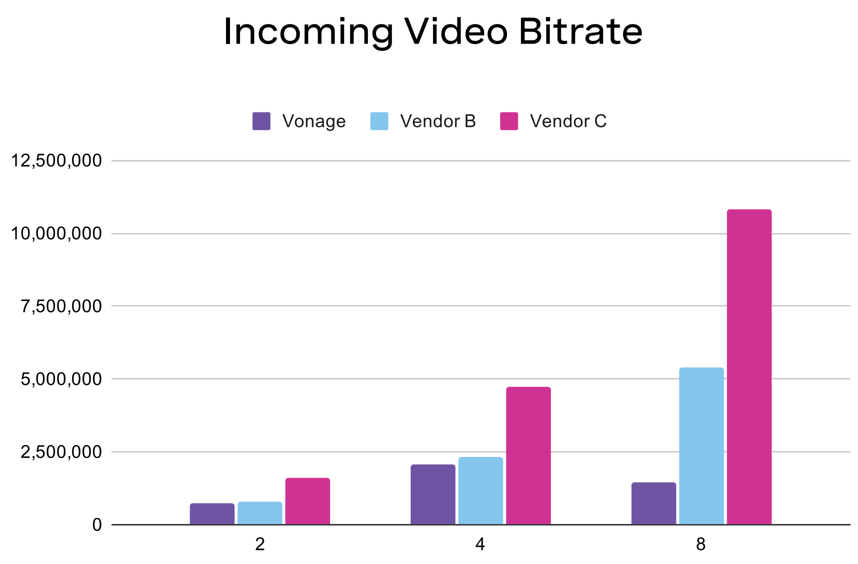
	Vonage	Vendor B	Vendor C
Layer 1	200 kbps	200 kbps	50 kbps
Layer 2	750 kbps	800 kbps	1,750 kbps

While Vonage and Vendor B had a lower layer of 200 kbps and limited the higher layer, Vendor C opted for having a very low bitrate of 50 kbps on its lower layer and a considerably higher bitrate on the higher layer. The approach Vendor C took creates a wide gap between the lower and the higher layer which doesn't leave much flexibility for optimizations or variability in the video subscriber capabilities. In the case of VGA resolutions, this is also wasteful in resources.

A developer using Vendor C would need to carefully optimize the application to configure the layers better.

Incoming Video Bitrate

The graph to the right shows the variance across incoming video bitrate observed between the vendors when using 2, 4 and 8 participants. The scenario selected is the normal conditions, but similar results were visible across the scenarios.



Vonage tended to use lower incoming bitrates than the rest of the vendors, no matter the size of the meeting. Vendor B raises the bitrate, reaching above 5mbps for an 8-way video call while vendor C raises the bitrate to above 10mbps for the 8-way video call.

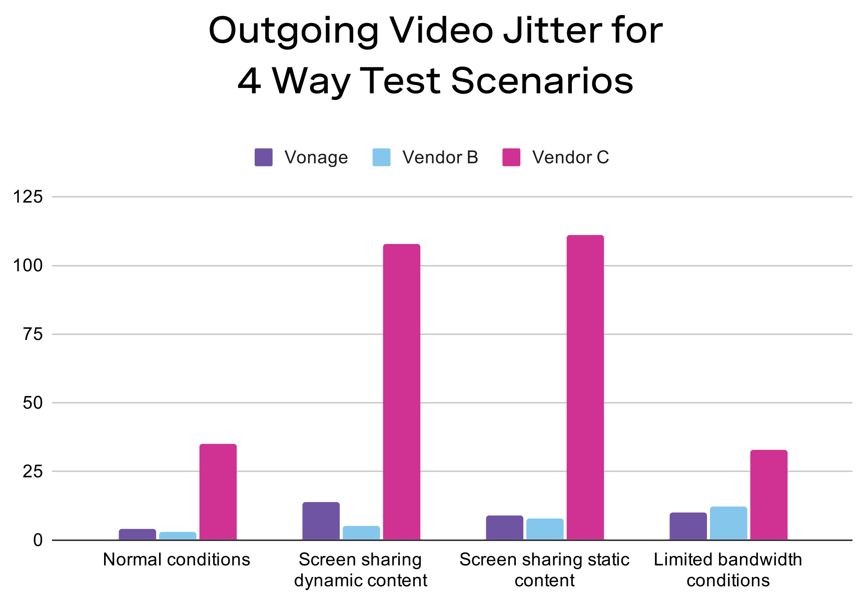
Aiming for 4mbps or more in incoming video bitrate isn't the best decision for most users today. It eats up on the bandwidth as well as CPU resources of the receiving machine, and in many cases can cause the session to fail altogether.

Interestingly, Vonage uses more incoming video bitrate for a 4-way call than they do for an 8-way call. This is probably due to a threshold being passed at some point between 4 and 8 participants which affects how they spread the bitrate budget between the available streams.

Jitter and Packet Loss

Jitter and packet loss values observed across the vendors were similar and reasonable.

For all our non-traffic-shaping scenarios we've seen no packet loss reported, which was aligned with our expectations. Jitter was low and within reasonable bounds.



One outlier we did observe was how Vendor C reports back the outgoing jitter values for video.

We decided to plot the graph arbitrarily for the outgoing video jitter on the 4-way video call in the various test scenarios. 2-way and 8-way scenarios showed the same trends. It can be observed that the jitter reported by Vendor C is considerably higher than the rest, especially for the screen sharing test scenarios.

Screen Sharing

With screen sharing, we've opted for running the exact same scenario but in two variations: dynamic content and static content.

The dynamic content was based on the popular Big Buck Bunny video:

Copyright 2008, Blender Foundation / www.bigbuckbunny.org

For the static content, testRTC chose a slide deck explaining webrtc-internals.

Our goal was to see how the various vendors cope with different content types for screen sharing, focusing on resolution and frame rate. We configured our screen to 1080p resolution: 1920x1080 pixels.

All vendors sent screen sharing alongside the camera feed in the session. We then checked the receiver metrics on the screen sharing channel and got these values:

Content type	Participants	Vonage	Vendor B	Vendor C
Dynamic	2	1080p@11fps	1080p@12fps	VGA@3fps
	4	1080p@10fps	1080p@8fps	VGA@3fps
	8	1080p@6fps	1080p@3fps	VGA@3fps
Static	2	1080p@24fps	1080p@11fps	1080p@5fps
	4	1080p@16fps	1080p@9fps	1080p@3fps
	8	1080p@11fps	1080p@5fps	1080p@3fps

The measurements are taken from the second participant in each test scenario, looking at the average frames per second and bitrate. The resolution was stable throughout the test scenarios for all vendors and use cases.

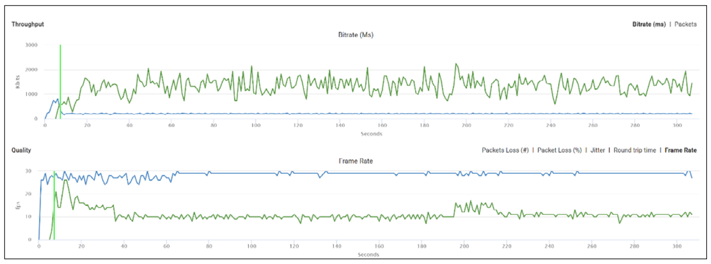
Vonage seemed to fare better than the other vendors the more participants we had in a session. Vendor C didn't handle the dynamic content well, with only 3 frames per second for the 2, 4 and 8 participants test scenarios.

Screen Sharing (cont'd)

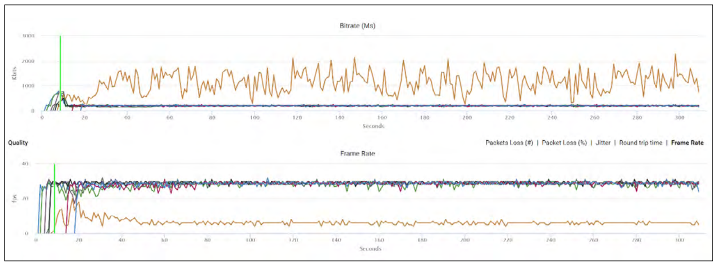
The reason behind the behavior of each vendor can be seen in the media diagrams below.

In these diagrams, we capture the incoming video bitrates and frame rates in the dynamic content scenarios with 2 and 8 participants for one of the viewer participants in each test.

Vonage



Vonage, 2 participants

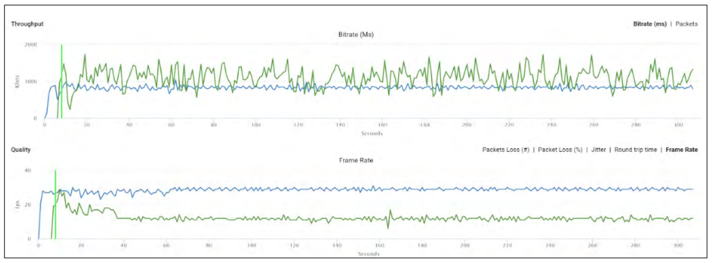


Vonage, 8 participants

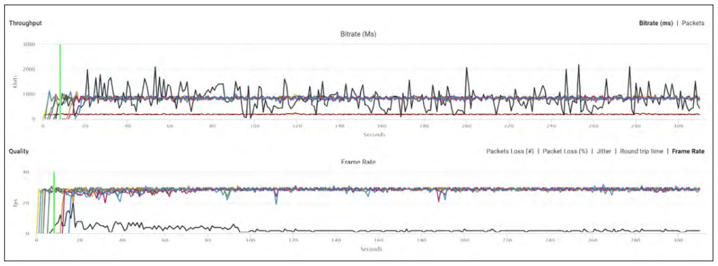
The green line on the left graphs and the orange line on the right graphs indicate the screen sharing video stream.

In a screen sharing scenario, Vonage immediately lowers the bitrate of all incoming video streams to the lower available layer (200kbps), giving priority to the screen sharing video stream. This enables them to invest higher bitrate and CPU resources for screen sharing once the size of the meeting grows to 8 participants.

Vendor B



Vendor B, 2 participants



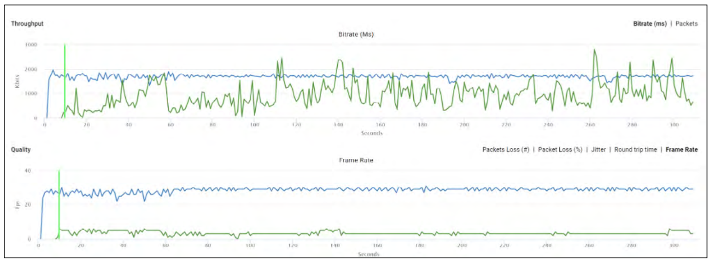
Vendor B, 8 participants

The green line on the left graphs and the black line on the right graphs indicate the screen sharing video stream.

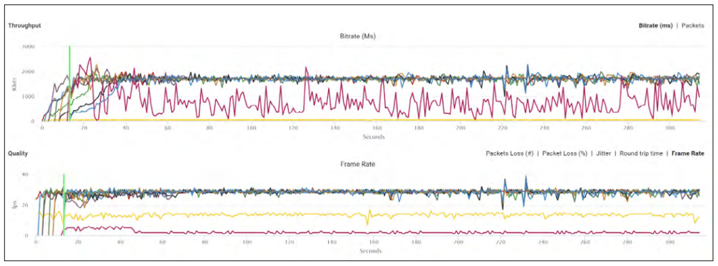
Vendor B starts nicely in the 2 participants graph, but doesn't prioritize screen sharing over other video inputs automatically. On the 8 participants scenario that is felt by a lower frames per second.

Screen Sharing (cont'd)

Vendor C



Vendor C, 2 participants



Vendor C, 8 participants

The green line on the left graphs and the magenta line on the right graphs indicate the screen sharing video stream.

Vendor C also doesn't prioritize screen sharing over other video inputs automatically. Along with its generally higher bitrate per incoming video stream, this means it starts at a low 3-5 frames per seconds where it stays for all test scenarios.

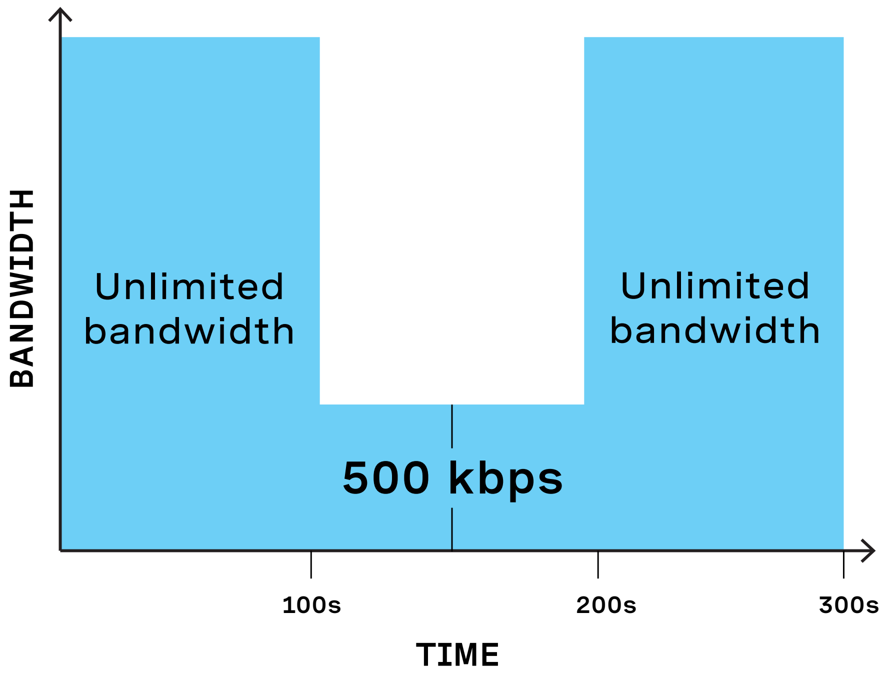
For the dynamic content scenarios, Vendor C also needs to reduce the resolution further from 1080p to VGA to be able to maintain screen sharing functionality.

Limited Bandwidth

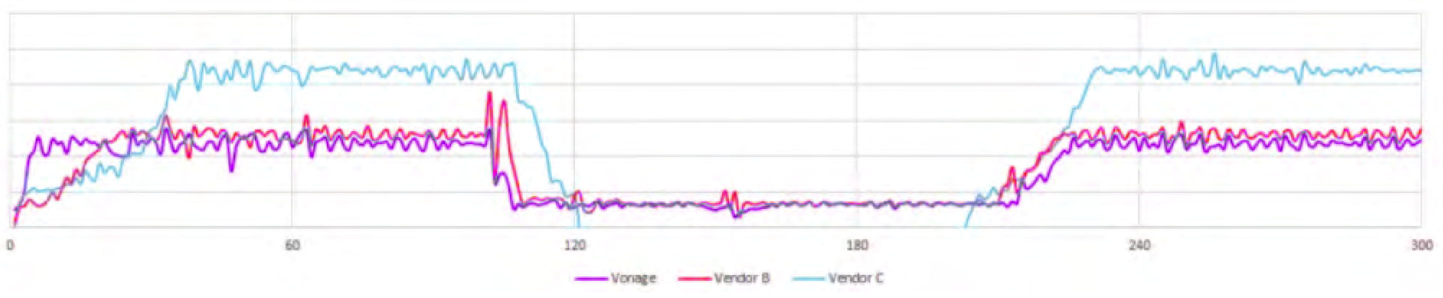
For this test scenario, we configured the first participant in each test run to dynamically change the network configuration it was using in the following manner:

Participant #1 starts off without any network limits, just like the rest of the participants in all test scenarios. After 100 seconds, participant #1 limits its bandwidth to 500 kbps both incoming and outgoing. After an additional 100 seconds, it removes that limit and proceeds as usual.

Our purpose in this test scenario is to check how each Video API vendor handles such network conditions - what happens during the span of the limited network conditions - and how much time it takes the service to recuperate and get back to previous conditions.



Limited Bandwidth (cont'd)



The graph above was taken from the testRTC console. It depicts the video bitrate of participant #1 in 4-way calls of the limited bandwidth scenario across the vendors.

Vonage and Vendor B were able to nicely cope with a reduction of bitrate to 500kbps, reaching back to 750kbps outgoing video bitrate within 15-25 seconds across the test scenarios of 2-way and 4-way calls. Both didn't behave well at 8-way calls, failing to reach back to the levels before the bandwidth limitation within a span of 100 seconds.

Vendor C didn't deal with the bandwidth limitation scenario well at all. Within 20 seconds of limiting the bandwidth, it lost incoming streams, both audio and video, opting to close their peer connections altogether, only to return and reconnect once bitrates came up again, after the 100 seconds period was over. During the bandwidth limitation, the participant didn't send out any media and received media only from a single other participant who monopolized its bitrate. Interestingly, a different strategy could have been employed here, since the lower simulcast layer used by Vendor C was only 50kbps.

Summary

This analysis shows how different vendors use WebRTC and the assumptions they take of network traffic and available compute resources.

testRTC picked four widely common use cases: normal video call, screen sharing of dynamic content, screen sharing of static content and video call under limited network conditions.

In all scenarios, the vendors took different approaches:

- Vendor C, for example, used simulcast with extremely different bitrates (50kbps and 1,750kbps). This left them exposed and inflexible to the changing conditions of larger meetings or limited network conditions.
- Vonage took the approach of reducing incoming bitrates aggressively by selecting the lower simulcast layer (configured to 200kbps in their case). This enables conducting larger meetings in front of a variety of devices at different network conditions;
- While Vendor B used a similar simulcast configuration, they didn't make use of it by reducing total incoming bitrates. This caused a reduction in performance in screen sharing scenarios.

When selecting a Video API vendor, it is important to validate its performance in the scenarios and use cases you expect in your service. Your mileage may vary from the tests testRTC had conducted and from the implementations used.