BlogGeek.Me

Top 7 WebRTC

Video Quality Metrics and KPIs

Table of Contents

Top 7 WebRTC Video Quality Metrics and KPIs

WebRTC is a great technology that is used by interactive video applications - be it live broadcasting, group meetings or one-to-one interactions and consultations.

Figuring out video quality in such applications is paramount to maintain user experience. It is also the first step towards observability and optimization of the application.

This eBook details and explains the top 6 video quality metrics and KPIs (key performance indicators) that you must understand and track in your WebRTC application.

WebRTC Video Quality Metrics and KPIs

Overview

Managing and optimizing media quality in WebRTC is no easy feat. The first step towards that goal is to better understand what are the metrics that need to be monitored and what their values mean.

In WebRTC there are hundreds of different metrics you have access to via the getStats() interface. Many of them are important in specific troubleshooting scenarios. What I want to do here is to cover the basics. For that purpose, I've selected the top 7 most important metrics that you must understand, monitor and strive to optimize.

I consider these as the KPIs (Key Performance Indicators) that will guide you towards a better WebRTC application. By measuring these metrics, monitoring their behavior over time and then working towards improving their values, you will be able to offer higher media quality for your users.

The first 4 KPIs relate to both voice and video - these are **bitrate, jitter, packet loss** and **round trip time.** The final 3 KPIs are video specific - these are **resolution, frame rate** and **freeze count.**

1. Bitrate

The number of bits per second that are being actively sent or received over the network.

Higher bitrate = Higher quality

Bitrate is the number of bits per second we invest in the given media we are sending, be it voice and video. The higher the bitrate, the higher the quality you will be able to achieve.

Modern codecs are used to encode (compress) and decode (decompress) media that is sent over the network. Without this compression, we wouldn't be able to use video communication. Different codecs provide different compression rates.

Video requires a lot higher bitrates than voice. That's because video carries a lot more information in it.

While higher bit rates means higher quality, this comes with a price:

- Using more network resources that might not be always available or need to be shared with others.
- Higher bit rates also suggest more CPU and memory invested in the encoding and decoding processes. Resources which might be of better use elsewhere or not available

My suggestion?

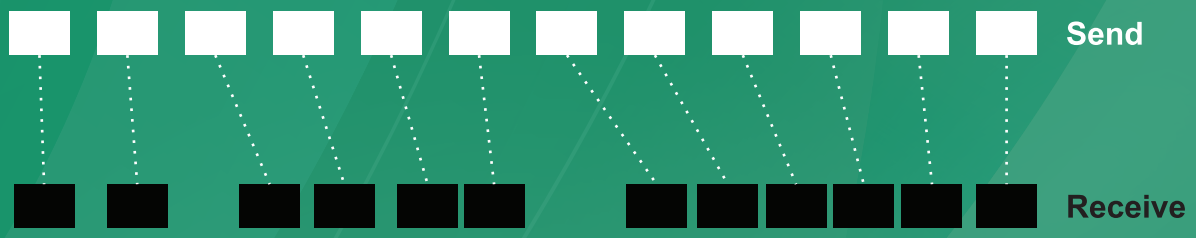
- Figure out what bitrates your application requires to operate well for your user base.
- Measure the average used in your application and monitor it over time.
- Search and focus on outliers - both on the lower end of bitrates but also at the higher end of bitrates.

2. Jitter

The variation in time delay between when media packets are sent and when they are received.

Lower is better; 30ms or less is acceptable

You need to play back media at the same rate it was originally sampled. Failing to do so will speed up or slow down the original content, which ruins the quality. When sending media as packets over a network, there is no guarantee that the same sampling rate will be maintained by the network. That difference between the rate and which the packets are received versus sent is measured as jitter.



Higher jitter values cause headaches to real time media applications. They require the application to either delay playout of the media or to drop packets that arrive too late altogether. Both of these lead to reduction in quality.

My suggestion?

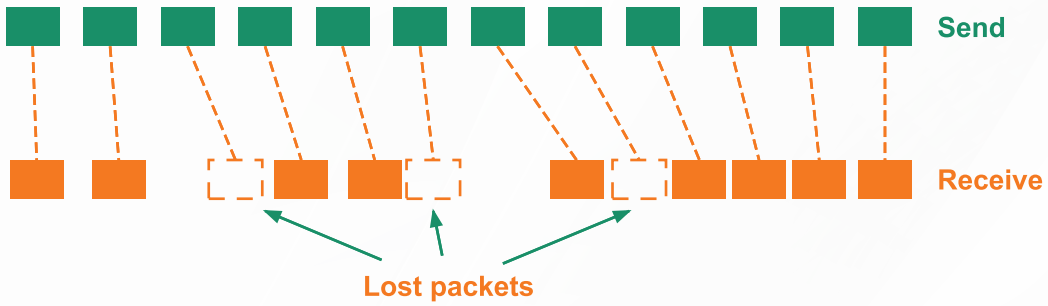
- Monitor jitter behavior of the users in your application
- Try to see if jitter goes higher based on specific geographies, service providers, network types or devices
- Focus on the 99 percentile of jitter values when optimizing performance

3. Packet Loss

Packet loss occurs when one or more transmitted data packets fail to arrive at their destination.

Lower is better

Lost packets mean the media decoder needs to fill in the gaps. With video, this almost always means losing a full frame that spans multiple packets. It can also lead to multiple frame losses and the need to reset the encoder on the sender side. There are many reasons for packets to be "lost in transmission". WebRTC uses UDP, which is unreliable. It does so to maintain low latency. The downside is that packets might be lost and that needs to be dealt with.



The higher the packet loss, the lower the expected media quality is going to be. Your goal is to reduce packet loss to the lowest possible value (zero, though don't expect to reach there on all of your calls).

In general, there are two types of packet losses that may occur, and each has its own type of handling:

- **Congestion:**
  Trying to send more than the network can handle. The solution here is better bandwidth estimation and a more timely reduction in bitrate when the need arise.

- **Corruption:**
  Random packet losses that occur from time to time that are harder to predict. Here, better network resiliency is the way to go.

Figuring out the type of packet loss can help us understand both the nature of the network and the quality that we can derive from it with our application.

My suggestion?

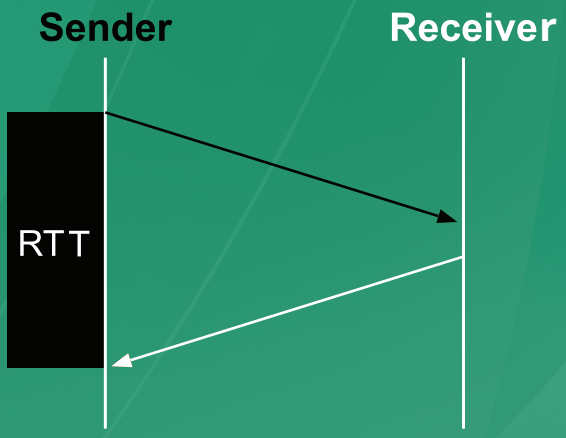
- Track packet loss values in your service.
- Figure out if there are endemic issues or sporadic ones. These will guide you to the solutions needed.
- For the outliers, search for network issues related to the user's environment.

4. Round Trip Time (RTT)

Round Trip time is the time it takes from the moment a request is sent over the network until a response is received.

Lower is better; 300ms or less is acceptable

WebRTC is used for live real time communication. If the round trip time is high, it means the time it takes a packet to go in one direction and a response to arrive is long. Which leads to a poor experience. Think of it as a long distance call 30 years ago, when you had to wait to really be certain the person on the other side finished talking before answering or risk “double talk” - one person speaking on top of the other.



High roundtrip usually means a routing issue in the network. There are other reasons as well, but this should be top of mind for you. Where you place your media servers and TURN servers can have a huge impact on the RTT values you will experience.

It is important to remember that there is no one-size-fits-all here - the location of your servers need to be aligned with where your users are, and that may be different than other applications.

My suggestion?

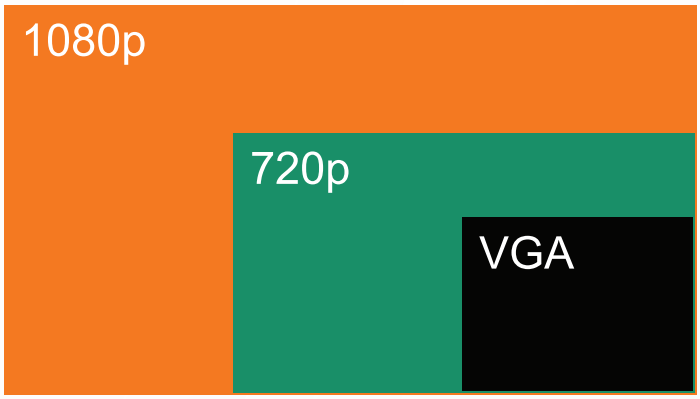
- Collect and monitor RTT values and work towards lowering RTT averages across your deployment
- Focus on each geography separately, deciding where to add more servers
- Consider using a third party TURN service such as Twilio Network Traversal Service or Cloudflare Calls TURN Service

5. Resolution

The number of pixels in a video frame, measured in width and height.

Higher is (mostly) better

The higher the resolution of the video received the higher the quality is going to be. Generally speaking, you should strive to reach higher video resolutions, which usually means increasing video bitrates.



Here's the thing:

- Higher resolution necessitates higher bitrate, which means more network resources used for the video stream.
- But if the display region on the receiver is of low resolution or physically small, then higher resolutions isn't going to help any or just won't be worth it.
- Higher resolution means higher CPU and memory requirements for both sender and receiver.
- Frequent fluctuations in resolution in a stream often leads to lower perceived video quality.

This all boils down to the need to better understand your application, its requirements and the desired behavior derived from it.

My suggestion?

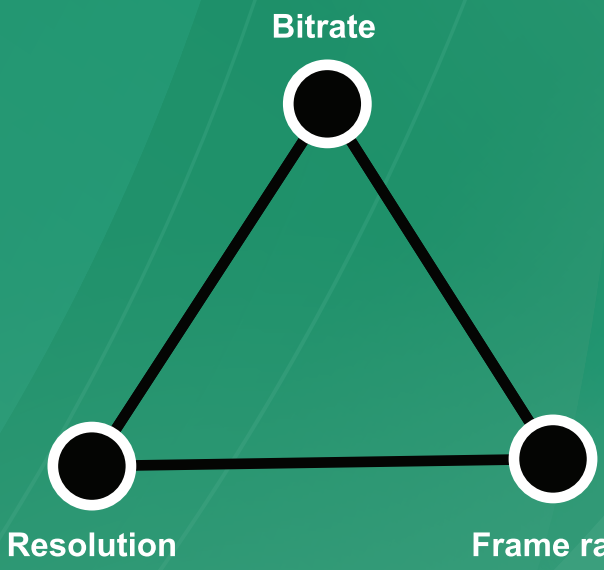
- Check the resolution used in your service
- Make sure resolutions fit well in what the receiver can handle and really needs
- Measure and monitor how often resolution changes not as a result of a layout change. Switching back and forth between resolutions every few seconds translates to poor user experience
- See if calls have too frequent fluctuations in resolutions in their session that cannot be explained by a change in screen layout

6. Frame Rate

The number of frames per second that are sent and received in a video stream.

Higher is better; 15-30 is acceptable

The higher the frame rate the more fluent the video will seem. That said, we need to understand the correlation between bitrate, resolution and frame rate.



Bitrate is almost always decided for us based on the conditions of the network, capabilities of the devices and the layout we wish to display for users. Once the bitrate limit is defined, the video encoder needs to decide on the resolution and frame rate to use:

- The higher the resolution, the lower the frame rate can be.
- The higher the frame rate, the lower the resolution can be.

For an operations perspective, check the difference in frame rate between the sender and the receiver. Differences will usually be due to high packet loss or high RTT.

My suggestion?

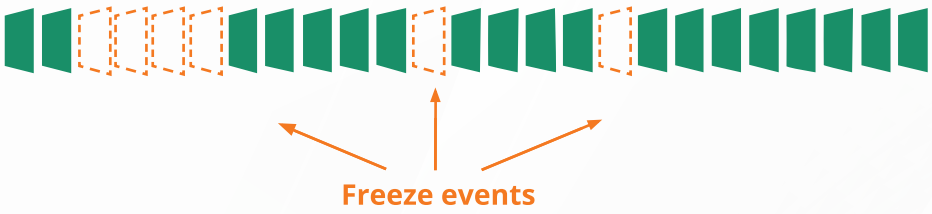
- Check the frame rate used in your service
- Understand fluctuations in sender and receiver values
- When using simulcast, see if two or three temporal layers yield better results for your use case

7. Freeze Count

The number of times the video froze for the viewer.

Lower is better

We've seen that video is measured by resolution and frame rate. The higher the frame rate, the more fluent the video will be. That said, due to packet loss, RTT and device resources, we might end up not displaying some of the frames originally encoded and sent. This leads to freezes in the viewed video - the opposite to having a fluent video experience.



Here are some of the reasons why your video may freeze in WebRTC:

- Packet losses cause frame drops which ends up as a video freeze.
- Packet losses cause a need to restart the video encoder with a keyframe request ending up with a longer video freeze.
- Frames arriving too late and being dropped ending up as a video freeze
- Not enough CPU or memory to decode frames in a timely fashion ending up as a video freeze.

My suggestion?

- Track and monitor freeze count values in WebRTC video calls
- Figure out the reasons behind them - there are cases where optimizing your code or infrastructure can reduce freezes from occurring

What's Next?

WebRTC offers a rich set of statistics and metrics that you can track. Understanding them requires a good grasp of WebRTC, and tracking them requires a robust monitoring solution.

Lucky for you, both are things I can help you

➤ Online WebRTC courses:

Check out my online WebRTC courses. The different courses can help you out skill up to the level you need.

To learn more, see https://webrtccourse.com

➤ WebRTC testing and monitoring:

TestRTC offers testing, monitoring and support services for those who develop and deploy WebRTC based services. I co-founded the company and after being acquired, I am now the Senior Director of Product Management for testRTC.

Sign up at https://testrtc.com

About the Author

My name is Tsahi Levent-Levi. I am a developer at heart. I have been working in the telecom and VoIP/UC industries for the past 25 years (and counting) in various roles: from a developer to project manager, product manager, CTO, CPO, co-founder and CEO. Most of that time, I was dealing with signaling and media products that were licensed to other developers who built their own products with them. This gives me a broad view of the market and an understanding of the challenges and opportunities that exist in the domain of VoIP and interactive video.

I came across WebRTC when it was first announced by Google and saw the potential in it. Since then, I have been watching the WebRTC space closely and writing about it on my blog: BlogGeek.me . From a hobby it became a "profession".

Today, I provide consulting services around WebRTC, CPaaS and ML/AI in communications, as well as offering an online course on WebRTC (webrtccourse.com)

I act as Senior Director of Product Management Cyara , after its successful acquisition of Spearline. Spearline acquired testRTC , a company developing a testing, monitoring and support platform for WebRTC applications. I was the Co-founder and CEO of testRTC priorto the acquisition.