TestDevLab

apiMind

Video App Benchmarking 2024

apiMind | Google Meet | Jitsi

Google Meet

jitsi

Introduction to TestDevLab

- 10 years in business
- 500 employees, 8 offices across 4 countries (Latvia, Estonia, North Macedonia, Spain)
- Clients include both startups and Fortune 500 companies
- Products that we test are being used by 4.5 billion people every day
- We offer QA services, testing labs (such as Audio/ Video quality testing) and products
- ISO 27001 certified
- >2500 actual devices to test against

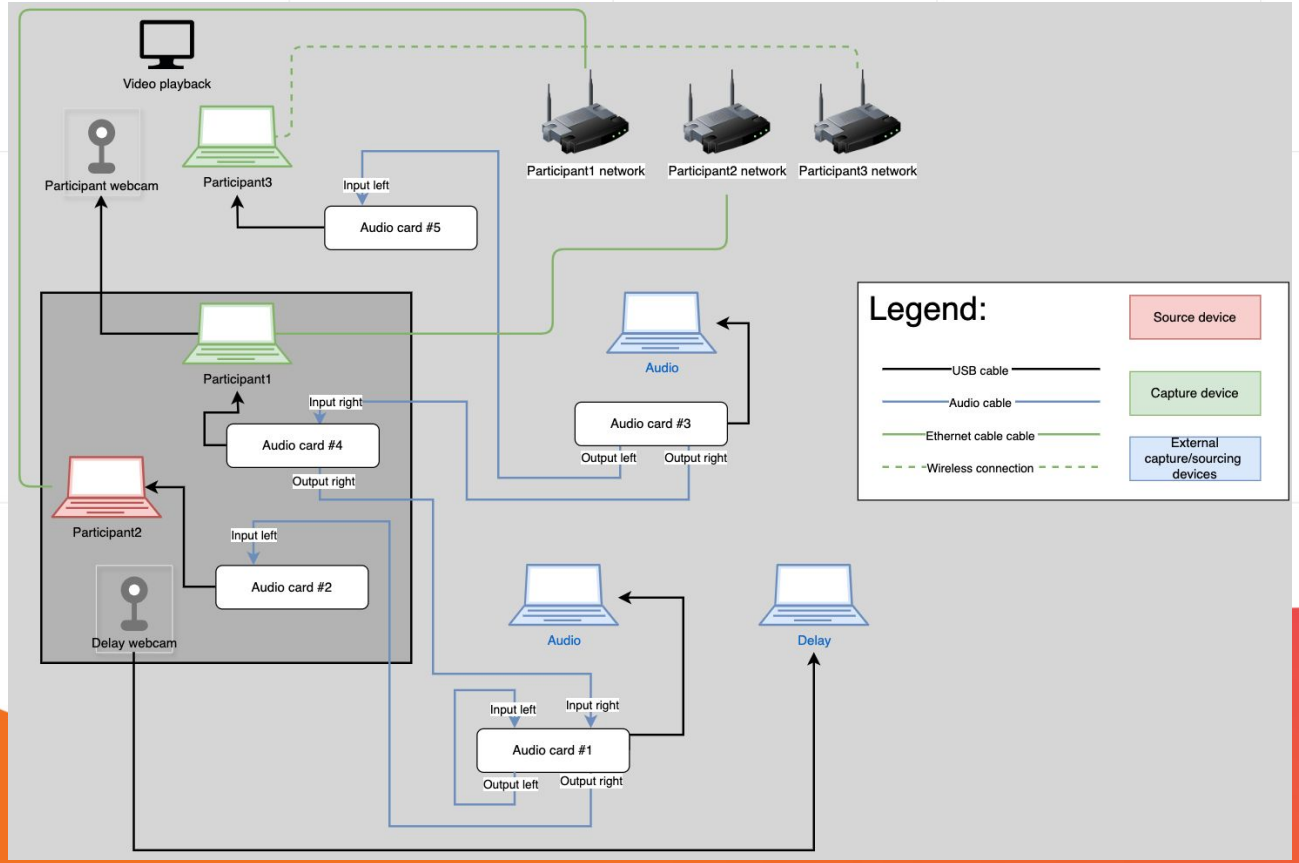
Trusted by: Microsoft WorldRemit skype Telefónica twilio orange and others

What we could offer

- Functional/Regression Testing
- Accessibility tests
- Performance Benchmarking (Battery/CPU/GPU/RAM/)
- Load Testing
- VOIP communications
- Video Conferencing/Streaming/VOD (video on demand) Testing
- iOT (internet of things)
- Automation / manual testing

Benchmark Program Goals

- Benchmark api Mind quality vs Google Meet & Jitsi
- Review behavior in different network conditions (Changing BW, Changing PL, Changing Latency & Jitter)



Three participants connect to he call.

Playback of reference files(audio and video) start on the sender side.

Playback of files (audio and video) start on both receiver sides.

At the same time screen and audio recording starts on the constrained participant.

Testing process & Schema

Benchmark Test Scope

Applications

api Mind

Google Meet

Jitsi

Platforms

Sender:

WinChrome

Receiver:

WinChrome

Network Constraints

Sender:

None

Receiver:

Changing Bandwidth tests

Unlimited->2M->500K->200K ->500K->2M->Unlimited

Changing Packet loss tests

Unlimited->10%->20%->20%->20%->10%->Unlimited

Changing Latency & Jitter tests

0/0-100/30-500/90-1500/270-500/90-100/30-0/0

Each limitation lasts 60 seconds which sums up to 7 min long tests

Test device/app versions

Katana GF66 11UD i7-11800H, 8GB, 512GB SSD, GeForce RTX 3050 Ti 4GB

Google Chrome 126.0.6478.127

Metrics explanation

Audio metrics

- **POLQA** - (Perceptual Objective Listening Quality Analysis) Full reference audio quality measurement standard in MOS scale. Documentation link
- **Audio Delay** - End to end latency between the audio signal being sent and getting received
- **VISQOL** - (Virtual Speech Quality Objective Listener) is an objective, full-reference metric for perceived audio quality. It uses a spectro-temporal measure of similarity between a reference and a test speech signal to produce a MOS-LQO (Mean Opinion Score - Listening Quality Objective) score. Documentation link
- **Audio and Video Synchronization** - The difference in milliseconds between audio and video signals being received that were sent at the same time.

Metrics explanation

Video metrics

- **VQTDL** - NO-REFERENCE ALGORITHM FOR VIDEO QUALITY ASSESSMENT DEVELOPED BY TESTDEVLAB. Video Quality Testing with Deep Learning—or VQTDL—is a no-reference algorithm for video quality assessment. This solution produces image quality predictions that correlate well with human perception and offers good performance under diverse circumstances, such as various network conditions, platforms and applications.

- **Full reference metrics:**
  - **VMAF** - full reference video quality metric developed by Netflix
  - **PSNR** - Peak signal to noise ratio Documentation link
  - **SSIM** - Structural similarity index measure Documentation link
- **FPS** - Frames per second, shows how fluid the video is
- **Video Delay** - End to end latency between the video frames being sent to them getting received.
- **Freezes count** - The count of each individual freeze that appears.
- **Freezes between** - The average time between two freezes.
- **Freezes total time** - The sum of values from all freeze's length.
- **Freezes average time** - The time calculated by (Freezes total time/Freezes count)

VQTDL - our own machine learning algorithm

VQTDL: is based on a convolutional neural network with Resnet50 as a backbone. Which is a 50 layer neural network with very rich feature representation. Moreover it uses a transformer encoder to handle different resolutions which translates into a much more robust algorithm for IQA. Prediction values are more stable and closer to the subjective than BRISQUE. Scores from 1 to 5

VQTDL	
>4	Video is very clear.
3.6 - 4	Video looks fairly good, although it's not great in most cases.
3 - 3.6	Video will have many artefacts and low resolution.
2.3 - 3	Poor video quality
<2.3	Very bad, not acceptable in most cases.

Documentation link

FPS

- **FPS:** calculated using QR codes - the combination of qr codes is changing 30 times per second

QR codes - used to calculate FPS

VMAF Image Evaluation

- Full Reference
- Represents the quality difference between two videos
- Developed and maintained by Netflix

VMAF	
80-100	Excellent
60-80	Good
40-60	Fair
20-40	Poor
0-20	Bad

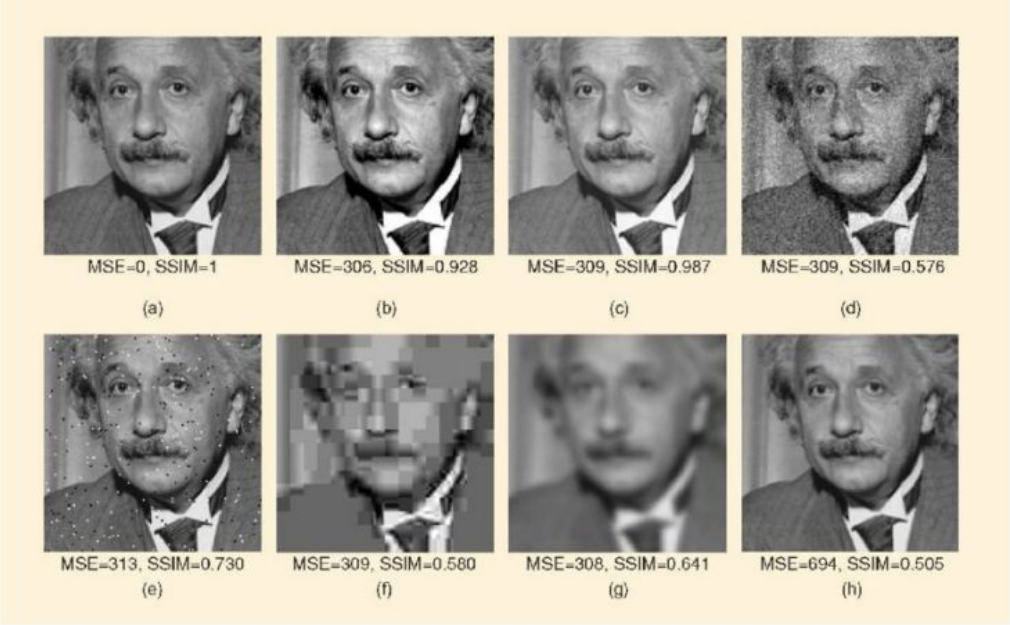
Documentation link

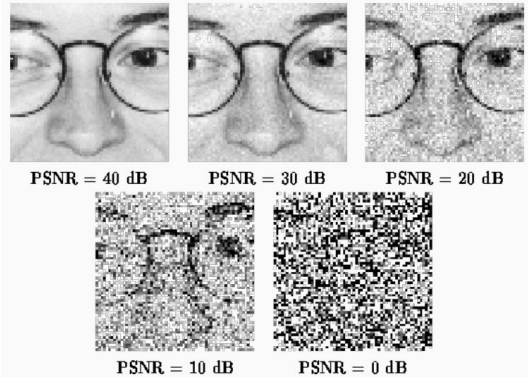
Original

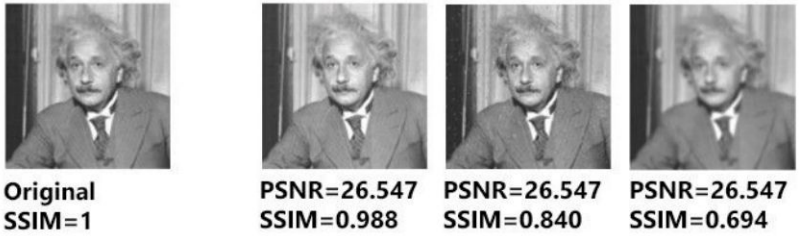
Degraded

Full reference metrics explanation

- Full reference Video Analysis compares the original reference video with a degraded one to get different video quality metrics







Metrics explanation

Network metrics

- Sender trace
- Receiver trace

Performance metrics

- **CPU Utilization** - Percentage of total CPU used by the specified process.
- **GPU Utilization** - Percentage of total GPU used by the specified process.
- **RAM Utilization** - Total Memory used by the specified process.

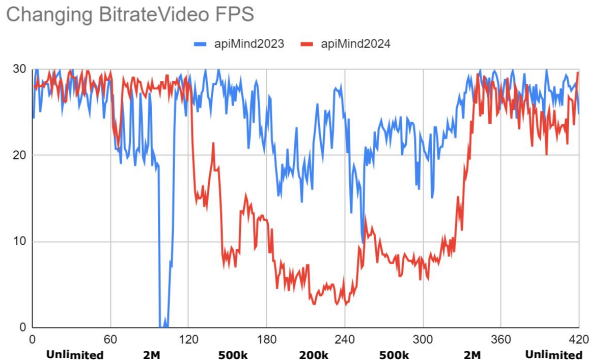
Changes vs 2023

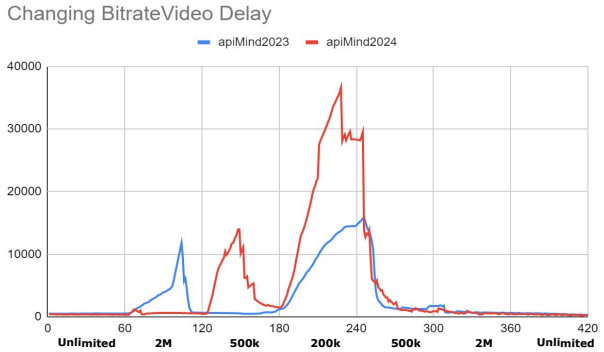
Scenario changes

- Packet loss scenario
  - Previous packet loss scenario (each condition 60 seconds)
    - 0->10->20->45->20->10->0
  - New packet loss scenario (each condition 60 seconds)
    - 0->10->20->20->20->10->0
- Marker change (to be able to analyze apiMind - sides cut off)
- 3 Users in call (Instead of 2)
- Extra users send audio during call (lower amount of audio quality data points)

Changing bitrate changes (1)

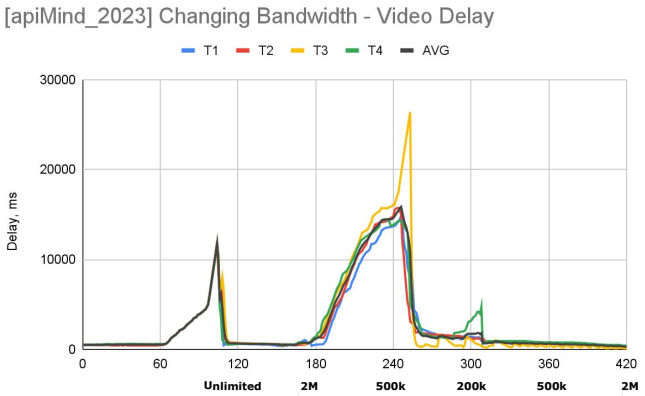
- Better bitrate handling when going to 2M limitation (minor impact)
- Worse handling when going to 500k and 200k
- Slower FPS recovery when bitrate is enabled again

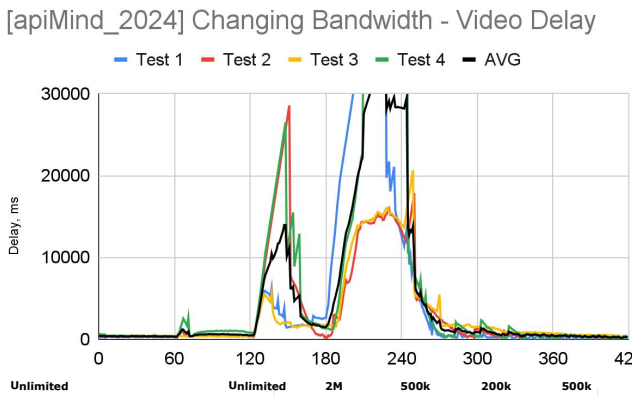


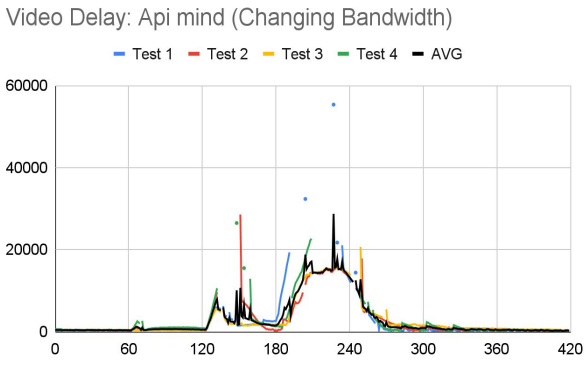


Changing bitrate changes (2)

- Individual results show that on 200k the bitrate handling 2/4 tests was similar as 2023
- However other 2/4 tests had much worse handling with delay going over 30 seconds
- To note - 200k delay going up to 15 seconds is also not good

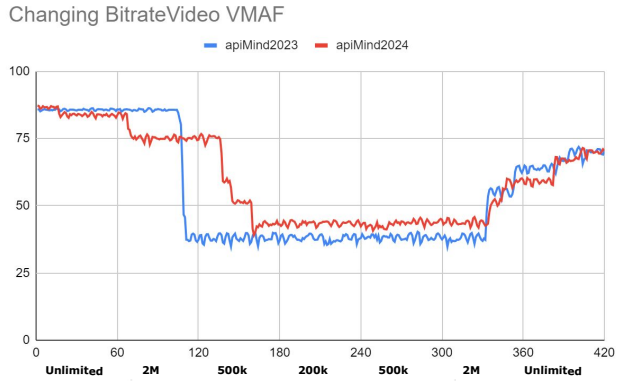






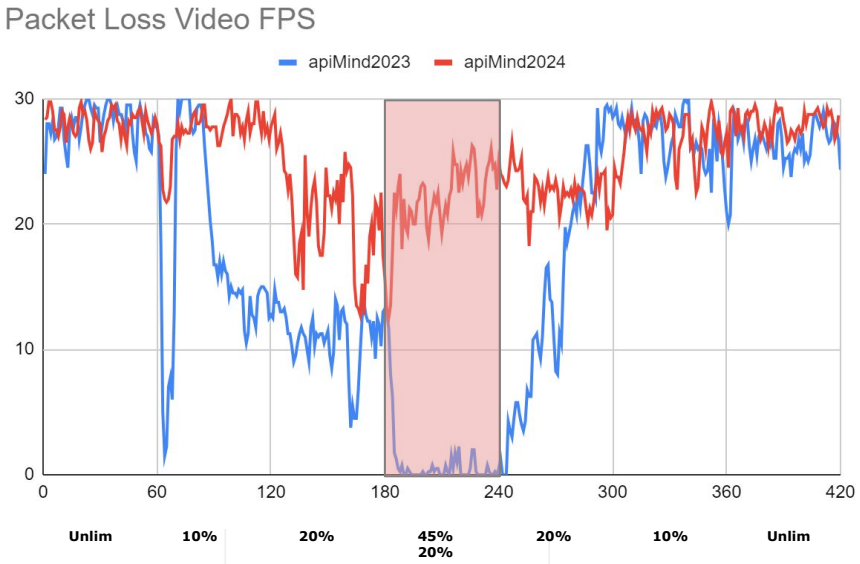
Changing bitrate changes (3)

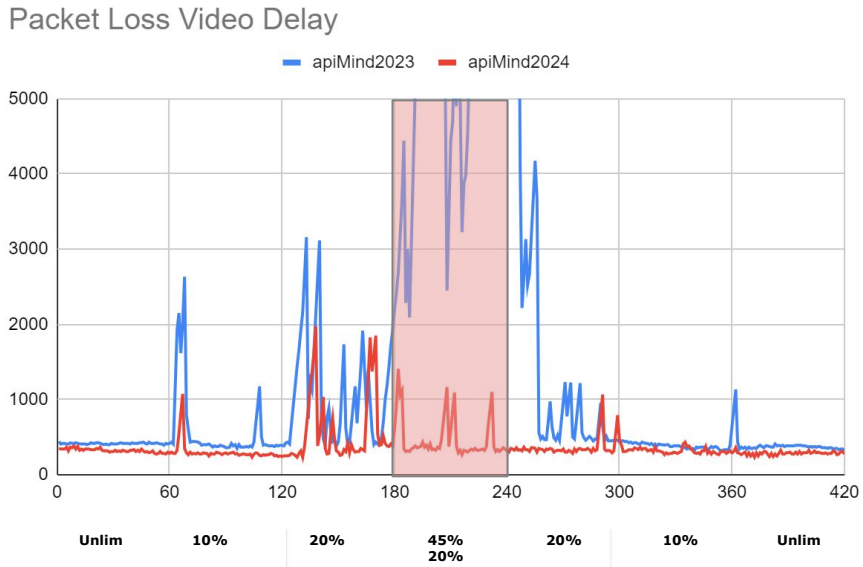
- For VMAF it looks like there are more quality levels that were used at start
- On VMAF side recovery seems similar to 2023 (reminder that FPS recovery was slower)



Changing packet loss changes (1)

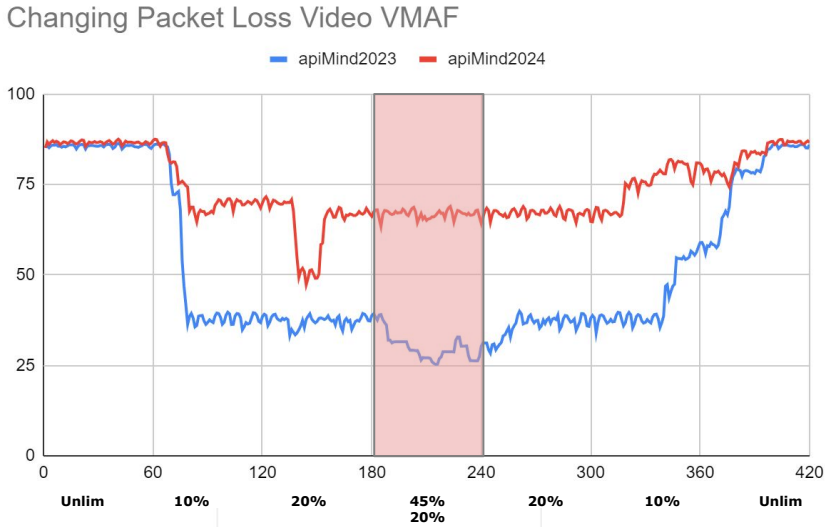
- **NOTE: In 2024 tests we don't have 45% packet loss limitation - instead 20% packet loss is used at that point.**
- Even taking in account changes in condition - frame rate and delay stability is better in 2024





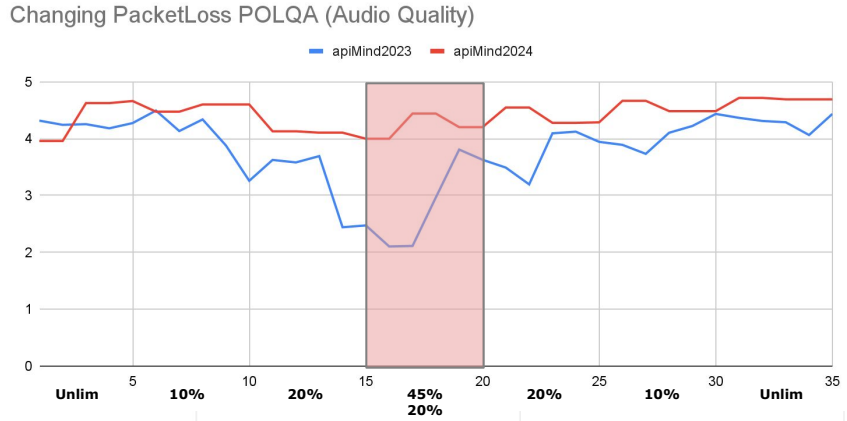
Changing packet loss changes (2)

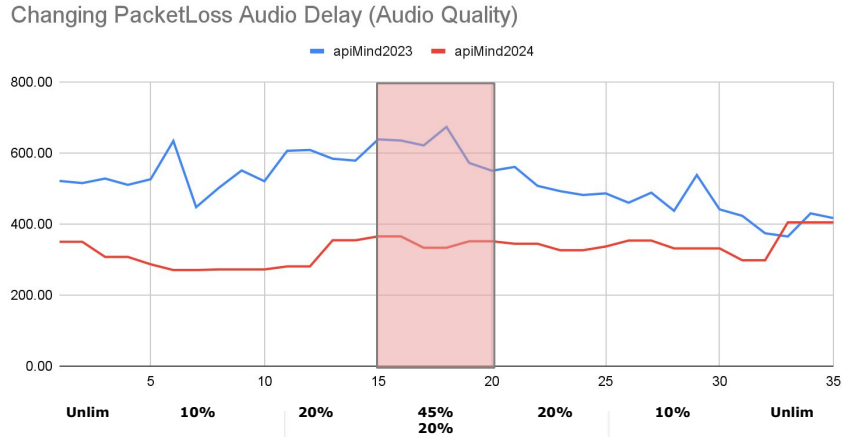
- **NOTE: In 2024 tests we don't have 45% packet loss limitation - instead 20% packet loss is used at that point.**
- Video quality during packet loss has increased



Changing packet loss changes (3)

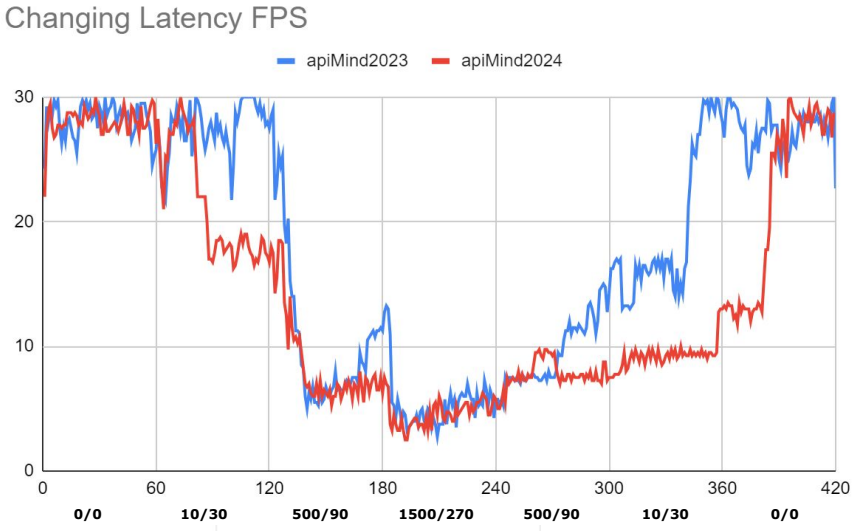
- **NOTE: In 2024 tests we don't have 45% packet loss limitation - instead 20% packet loss is used at that point.**
- Audio quality has increased at 20% packet loss (POLQA changes become noticeable <~3.7 POLQA score)
- Audio delay baseline also is lower and stays lower during the call

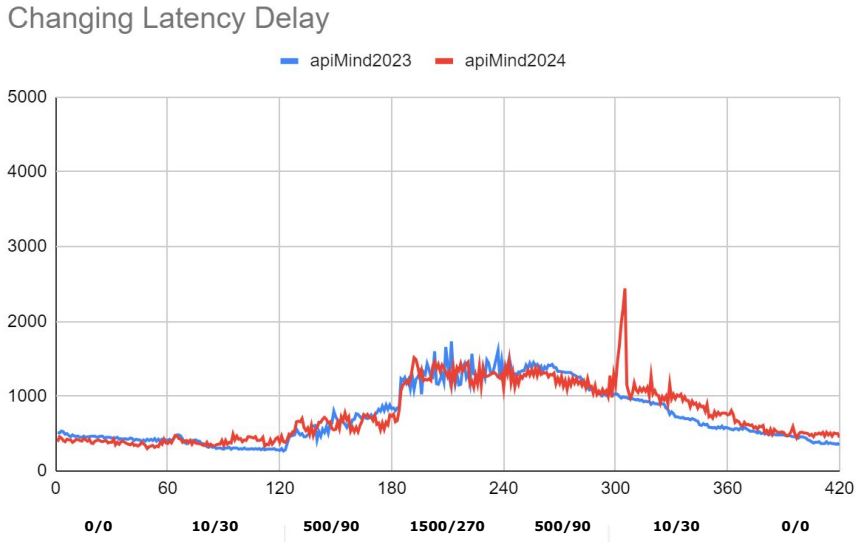




Changing latency/jitter (1)

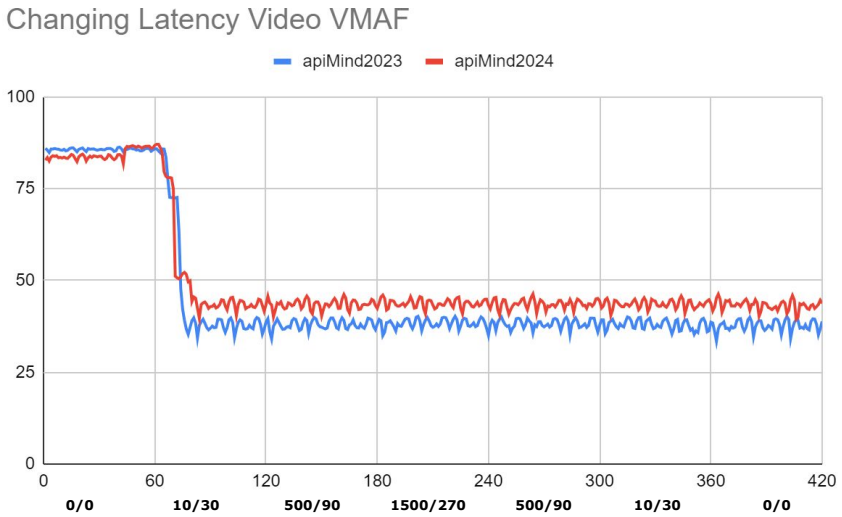
- Slightly earlier impact on FPS, however Latency behavior very similar (it takes time for latency to decrease even after limitation is removed)





Changing latency/jitter (2)

- Video quality impact very similar and starts already at 10ms jitter and 30ms delay.
- Worth noting that delay and jitter should not directly affect video performance, however it is heavy indication on overall network performance



Competitive analysis - Key findings

Summary of findings

Video

#1: Video quality under unconstrained network conditions
#2 apiMind Low FPS with 200k network
#3 Jitsi turns off video in bad network conditions
#4 Meet low FPS in Jitter/Latency condition
#5 apiMind does not recover to original quality when user has Unlimited network

Performance

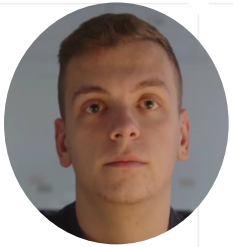
#8: High device performance usage on Jitsi

Audio & Network

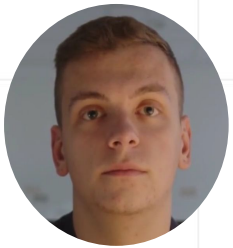
#6 apiMind drops audio in 200k condition
#7 apiMind has highest receiver bitrate on unlimited network

Video

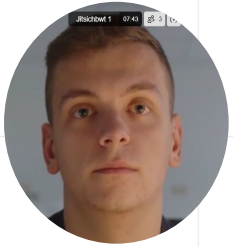
#1: Video quality under unconstrained network conditions



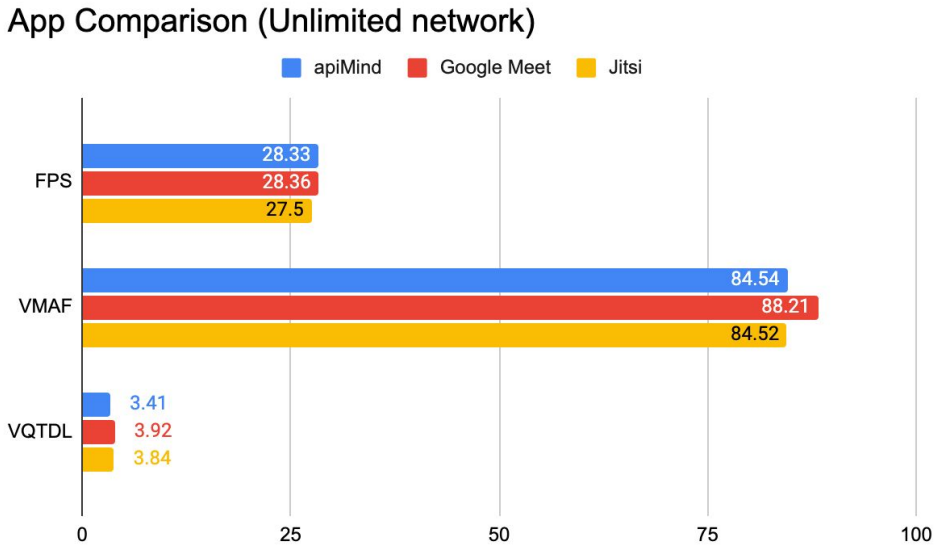
GoogleMeet
Test1 36th sec
VMAF - 88
VQTDL - 3.95



apiMind
Test1 36th sec
VMAF - 86
VQTDL - 3.43



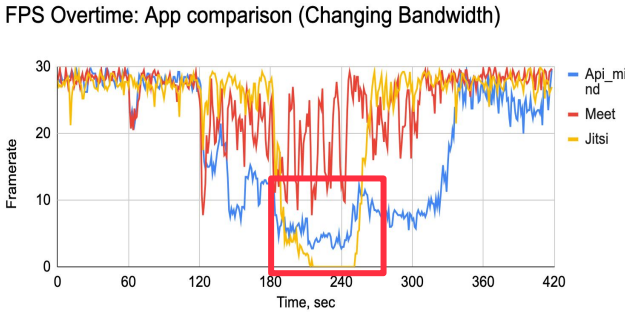
Jitsi
Test1 36th sec
VMAF - 84
VQTDL - 3.88

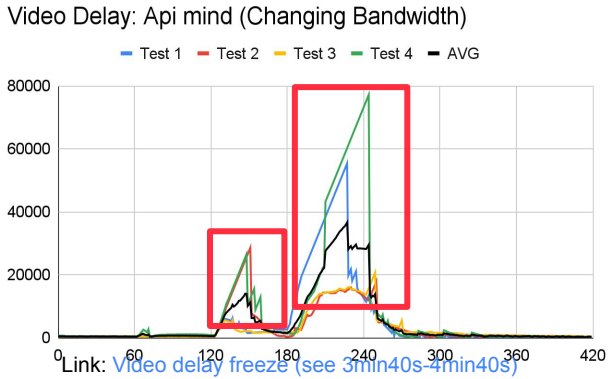


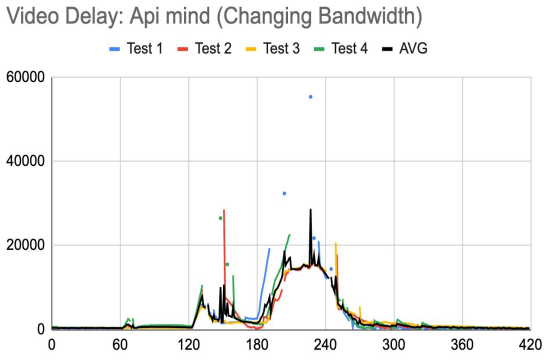
Under unconstrained network conditions,
the video quality in **apiMind**, **Google Meet** & **Jitsi** is similar.

Video

#2 apiMind Low FPS with 200k network







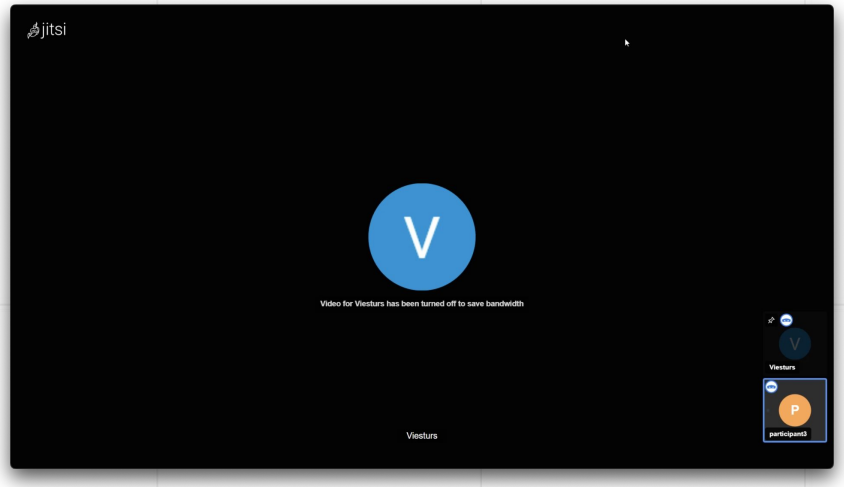
FPS	Api_mind	Meet	Jitsi
Unlimited	28.33	28.36	27.50
2mbps	27.35	27.18	27.37
500kbps	13.85	20.86	26.52
200kbps	**5.38**	**17.83**	4.66
500kbps	7.73	22.57	18.34
2mbps	17.05	27.28	27.63
Unlimited	24.87	28.62	27.62

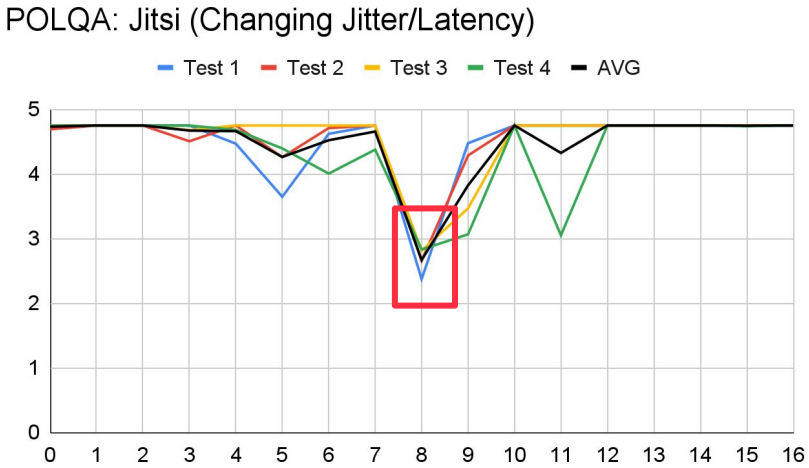
Link: Video delay freeze (see 3min40s-4min40s)

In changing Bandwidth when limitation switches to 200k, **apiMind** FPS drops to ~5 and viewer suffers constant freezes. It is the same for **Jitsi**.
Google Meet handles the limitation the best with ~17 FPS.

Video

#3 Jitsi turns off video in bad network conditions

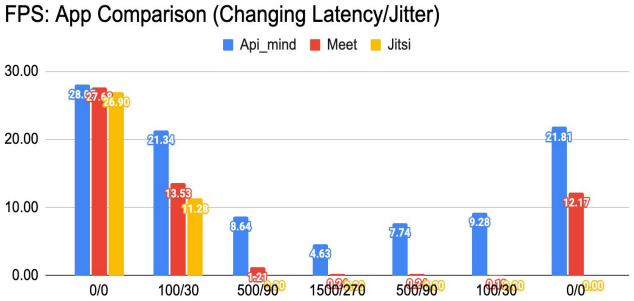




Jitsi turns off video when it detects low network. This is the screen the viewer sees under:
1) Changing Bandwidth - 200k
2) Changing Jitter/Latency - (500/90, 1500/270)

Audio passes through, but ~3s of 12s sample is dropped.

#4 Meet low FPS in Jitter/Latency condition

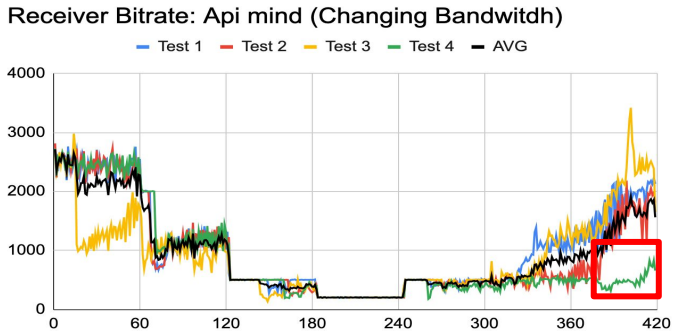


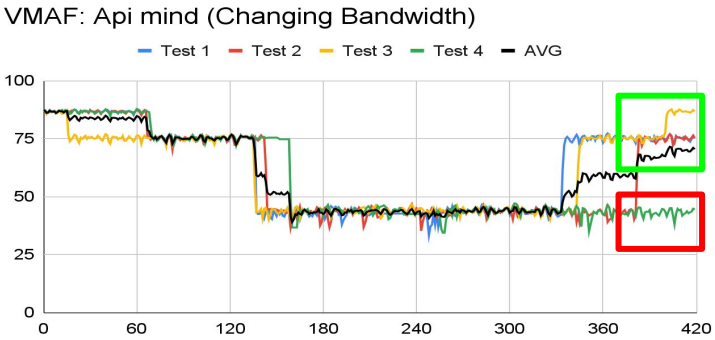
FPS	Api_mind	Meet	Jitsi
0/0	28.05	27.68	26.90
100/30	21.34	13.53	11.28
500/90	8.64	1.21	0.00
1500/270	4.63	0.24	0.00
500/90	7.74	0.24	0.00
100/30	9.28	0.19	0.00
0/0	21.81	12.17	0.00

When Jitter/Latency is applied, **Google Meet** FPS significantly drops.
It is similar for **Jitsi**, where video is dropped.
apiMind handles the limitation the best.

Video

#5 apiMind does not recover to original quality when user has Unlimited network





VMAF	Api_mind	Meet	Jitsi
Unlimited	84.54	88.21	84.52
2mbps	76.24	88.15	84.04
500kbps	55.88	72.90	69.62
200kbps	43.46	46.39	20.11
500kbps	43.33	67.40	42.09
2mbps	48.88	77.98	79.31
Unlimited	64.98	81.57	82.68

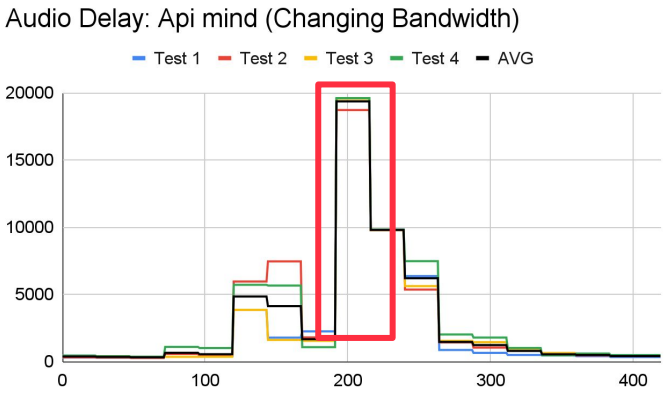
In changing Bandwidth when limitation switches back to Unlimited network, only 2/4 tests recover to original quality.

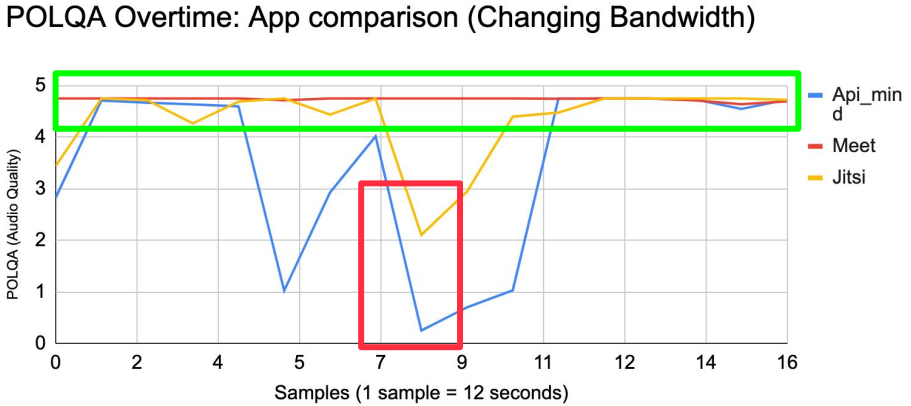
For tests that did not recover - Receiver Bitrate also did not go back to original values.

Video

Audio & Network

#6 apiMind drops audio in 200k condition

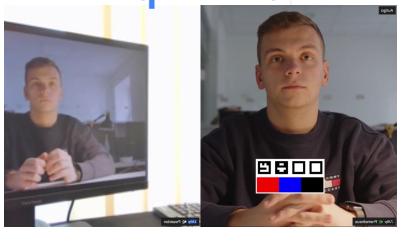




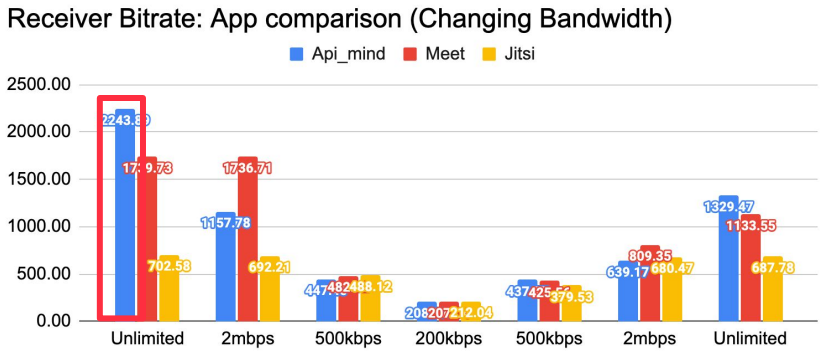
In Changing Bandwidth condition when limitation switches to 200k, **apiMind** drops entire 12s audio sample, and viewer does not hear what other user is speaking.
Google Meet & **Jitsi** does not have this issue.

#7 apiMind has highest receiver bitrate on unlimited network

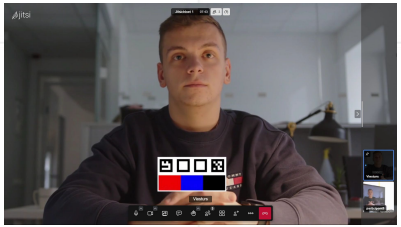
apiMind



Google Meet



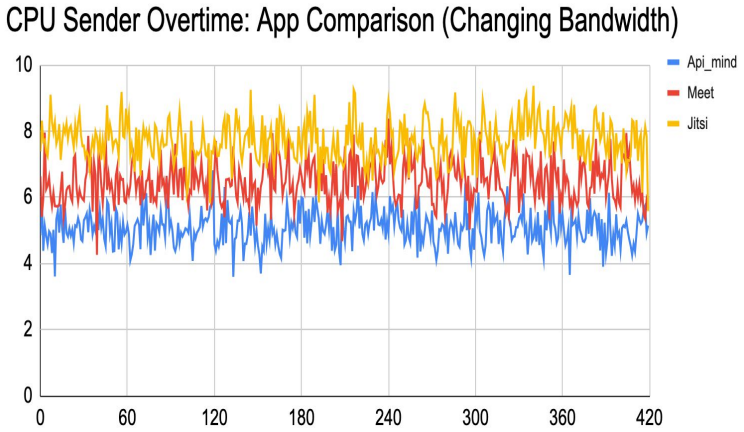
Jitsi

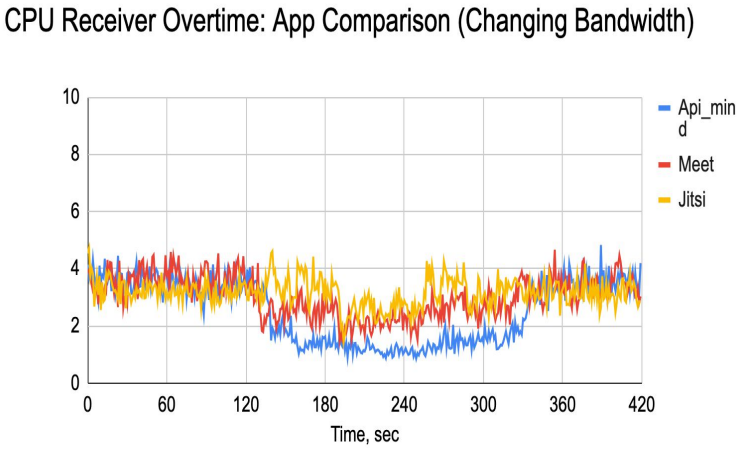


In all scenarios when limitation is “Unlimited”, **apiMind** has the highest Receiver Bitrate. Possible reason - receiver is watching two videos, which is not the case **Google Meet** & **Jitsi**.

Performance

#8: High device performance usage on Jitsi





Overall, in all scenarios **Jitsi** exhibits the hightest utilization of sender device performance.
apiMind - the lowest.
Google Meet - in the middle.
For receiver it is similar across all apps.

apiMind performance against Google Meet

Windows platform

	Changing Bitrate	Changing Packet Loss	Changing Jitter/Latency
Video quality	**Lower ⬇️** FPS - 28% VQTDL - 17% VMAF - 20% Video Delay - 1115%	**Lower ⬇️** FPS - 1% VQTDL - 8% VMAF - 9% Video Delay - 27%	**Higher ⬆️** FPS + 83% VQTDL - 1% VMAF + 3% Video Delay + 96%
Audio quality	**Lower ⬇️** POLQA - 25% Audio Delay - 1036%	**On par ↔️** POLQA - 1% Audio Delay - 90%	**Lower ⬇️** POLQA - 3% Audio Delay - 31%
Network	**On par ↔️** Receiver Bitrate - 1% Sender Bitrate - 1%	**Higher ⬇️** Receiver Bitrate + 61% Sender Bitrate - 2%	**Higher ⬇️** Receiver bitrate + 95% Sender Bitrate - 2%

apiMind performance against Jitsi

Windows platform

	Changing Bitrate	Changing Packet Loss	Changing Jitter/Latency
Video quality	**Lower⬇️** FPS - 22% VQTDL - 10% VMAF - 10% Video Delay - 98%	**Higher⬆️** FPS + 48% VQTDL + 12% VMAF + 33% Video Delay + 73%	**Higher⬆️** FPS + 165% VQTDL + 237% VMAF + 170% Video Delay +99%
Audio quality	**Lower⬇️** POLQA - 17% Audio Delay - 359%	**Higher⬆️** POLQA + 30% Audio Delay - 78%	**Lower⬇️** POLQA - 5% Audio Delay - 232%
Network	**Higher ⬇️** Receiver Bitrate + 68% Sender Bitrate + 142%	**Higher ⬇️** Receiver Bitrate + 467% Sender Bitrate + 134%	**Higher ⬇️** Receiver bitrate + 43% Sender Bitrate + 132%

TestDevLab

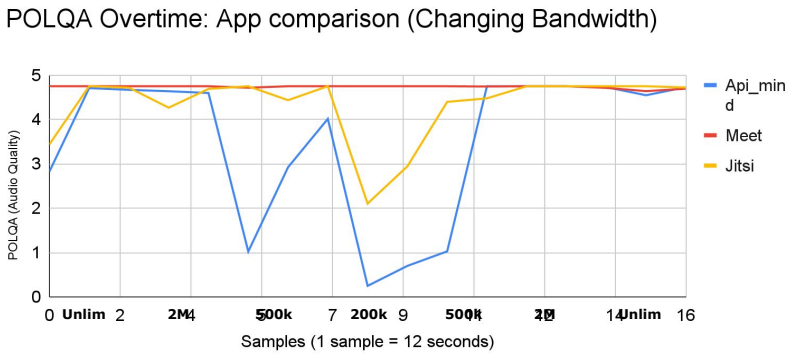
Full Result Summary

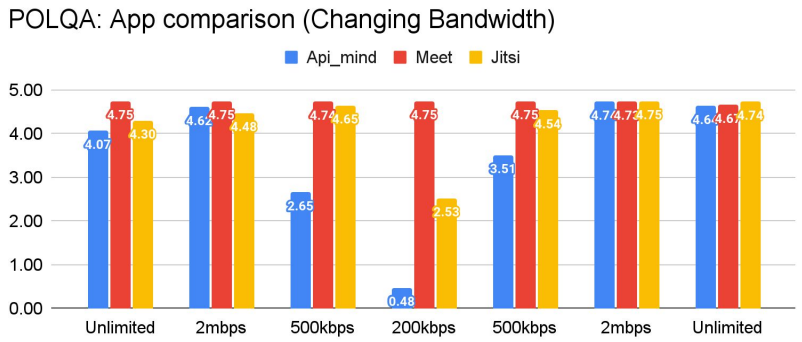
CHANGING BANDWIDTH

Changing Bandwidth Test Process

1. Sender creates a room
2. Receiver starts recording the screen and performance/delay data
3. Sender starts playing the video on OBS
4. Audio script along with network trace capture and “Changing Bandwidth” script are executed with conditions:
    1. Unlimited limitation enabled for 1 minute
    2. 2 Mbps limitation enabled for 1 minute
    3. 500Kbps limitation enabled for 1 minute
    4. 200Kbps limitation enabled for 1 minute
    5. 500Kbps limitation enabled for 1 minute
    6. 2 Mbps limitation enabled for 1 minute
    7. Unlimited limitation enabled for 1 minute
5. Test ends when the sender video reaches blue screen, delay video recording and network trace capturing is stopped
6. Receivers leave the room/call
7. Sender disconnects from the room/call and the chrome browser is restarted

POLQA comparison





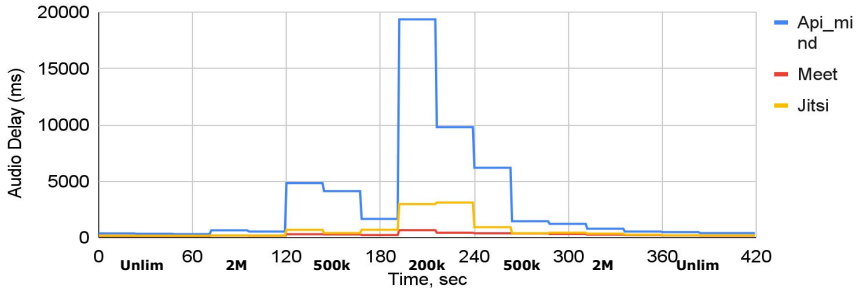
apiMind has poor POLQA when limitation is 500k and 200k, but recovers at 2mbps.

apiMind & **Jitsi** has muffled audio in the first audio sample.

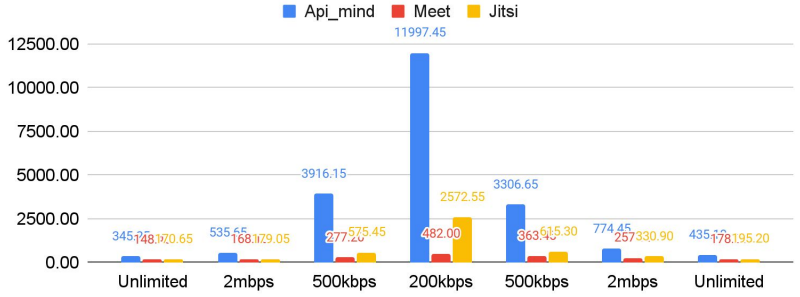
Google Meet has excellent POLQA on all limitations.

Audio Delay comparison

Audio Delay Overtime: App comparison (Changing Bandwidth)

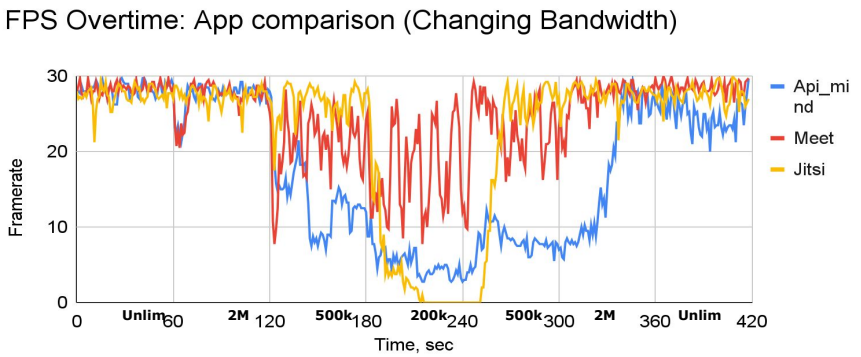


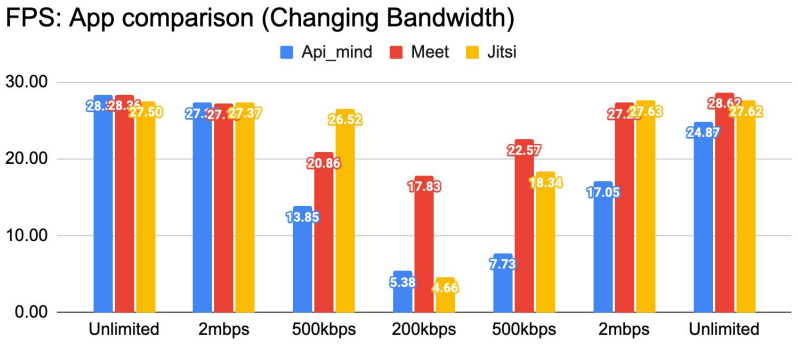
Audio Delay: App comparison (Changing Bandwidth)



apiMind has high audio delay in 500k and 200k limitation periods.
Jitsi audio delay increases in 200k limitation period.
Google Meet audio delay slightly increases in 200k limitations.

FPS comparison



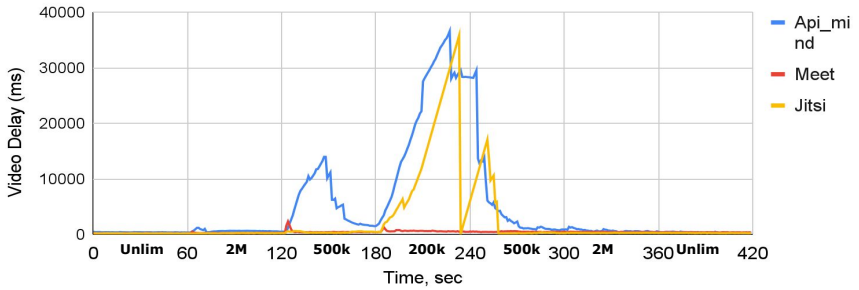


apiMind is on-par with **Google Meet** & **Jitsi** in Unlimited & 2mbps limitation periods.

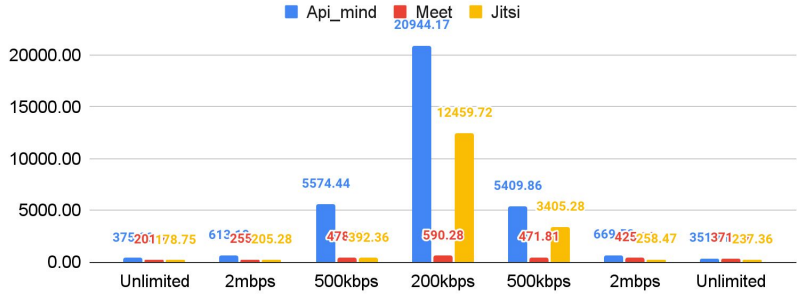
apiMind & **Jitsi** FPS drops when network is limited by 200k, but **apiMind** recovers slower.

Video Delay comparison

Video Delay Overtime: App comparison (Changing Bandwidth)

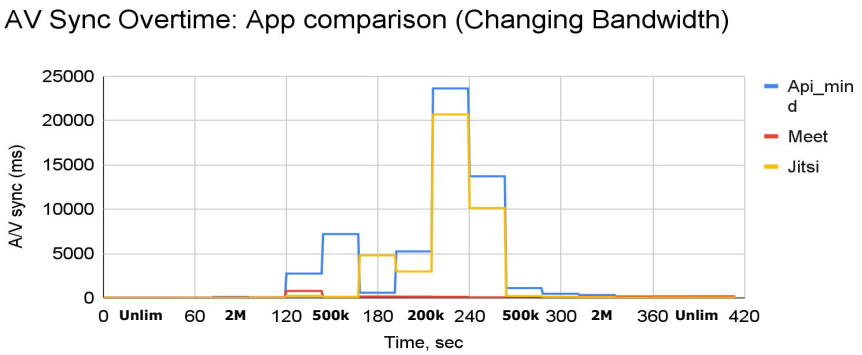


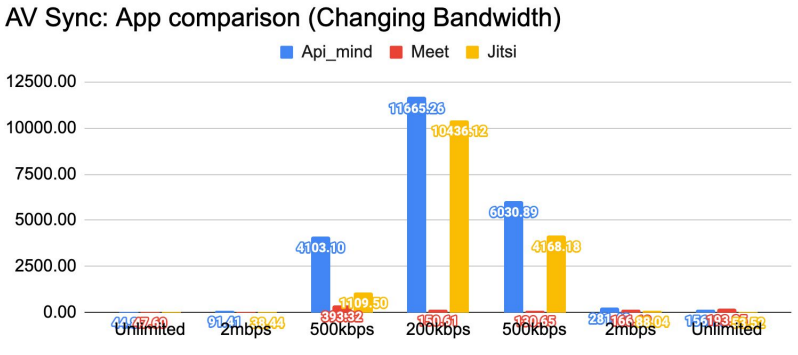
Video Delay: App comparison (Changing Bandwidth)



apiMind has the highest video delay.
apiMind & **Jitsi** start to have freezes in 500k & 200k limitations.
Google Meet has the lowest video delay across all limitations.
Jitsi in the middle.

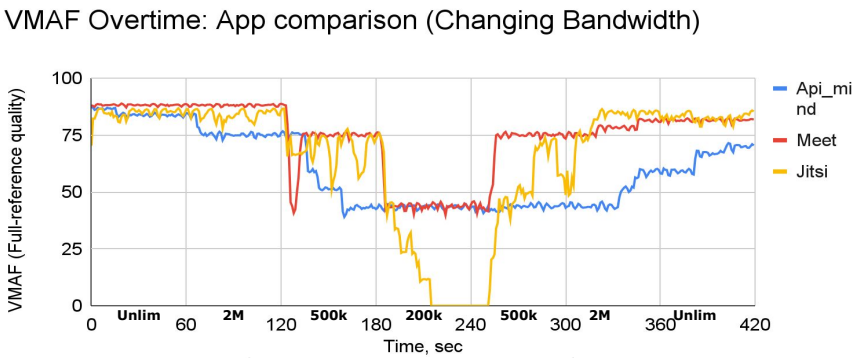
Audio and Video synchronization comparison

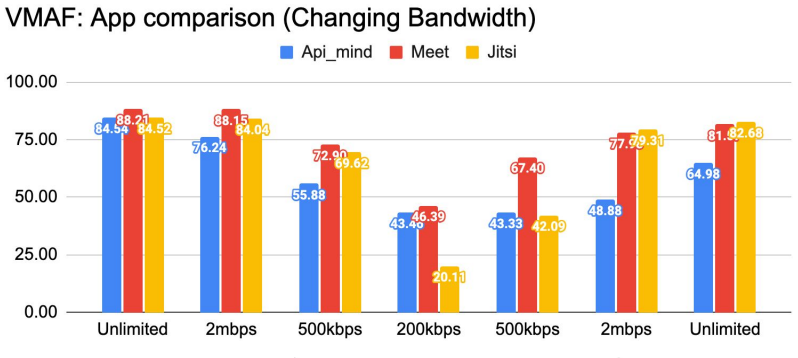




apiMind & **Jitsi** struggling in 500k & 200k limitations.
Google Meet has stable audio/video synchronization across all limitations.

VMAF comparison



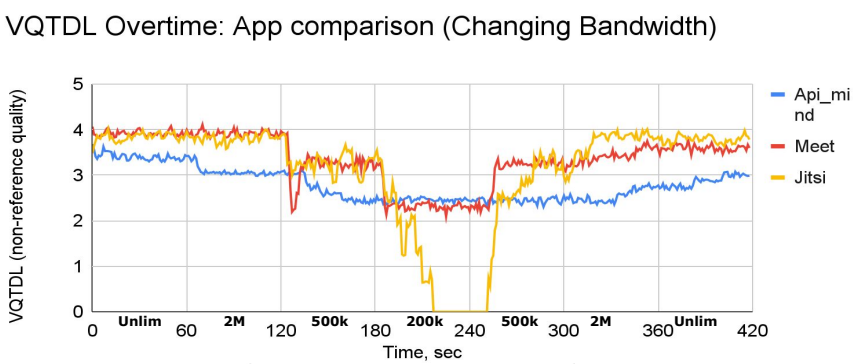


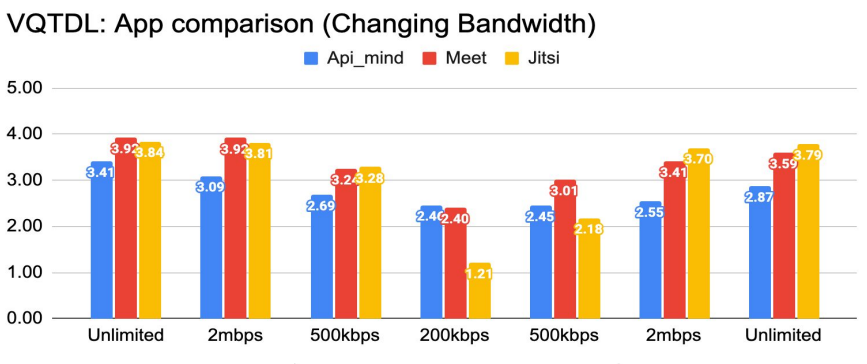
apiMind is on-par with **Google Meet** & **Jitsi** in unlimited network.

apiMind has lower, but stable video quality at low bandwidth periods & never recovers to original quality.

Jitsi turns off video in 200k limitation.

VQTDL comparison

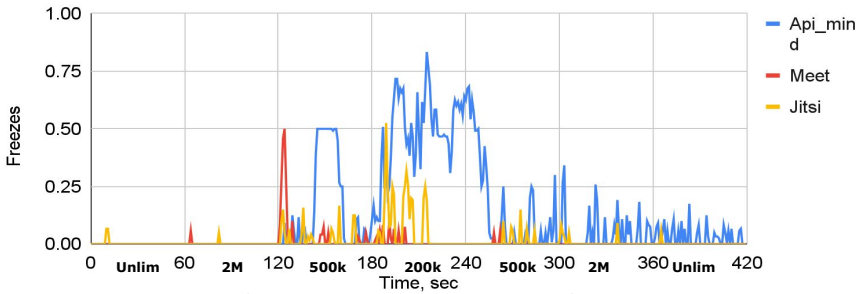




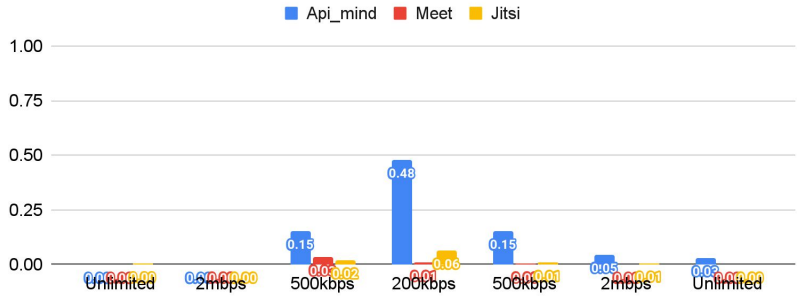
apiMind has lower average VQTDL value in all conditions compared to **Google Meet** & **Jitsi**.
Jitsi turns off video in 200k limitation.

Freeze count comparison

Freezes count Overtime: App comparison (Changing Bandwidth)



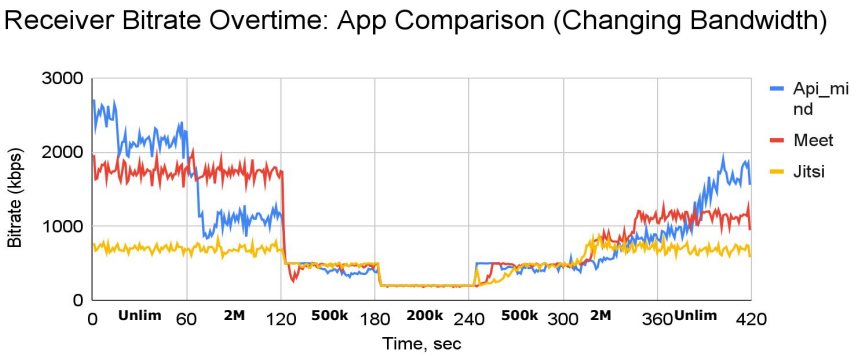
Freezes count: App comparison (Changing Bandwidth)

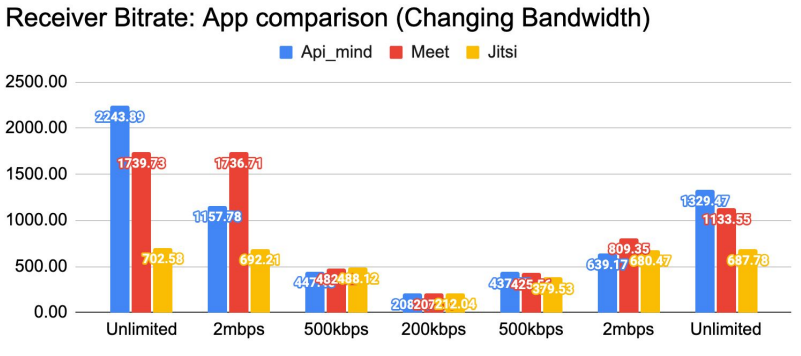


apiMind & **Jitsi** has some freezes in 500k limitation.

apiMind has frequent freezes in 200k limitation.

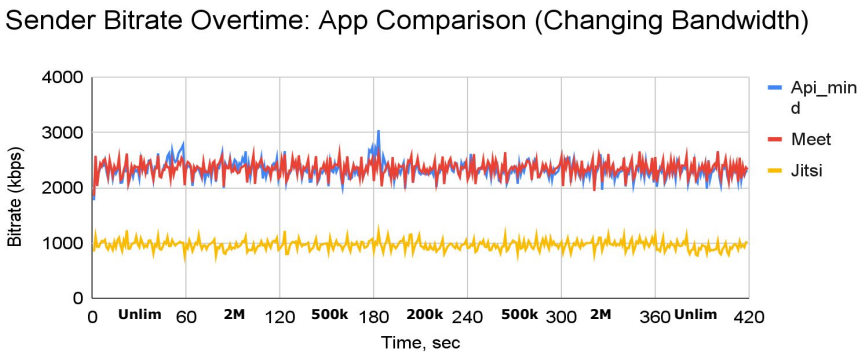
Receiver bitrate comparison

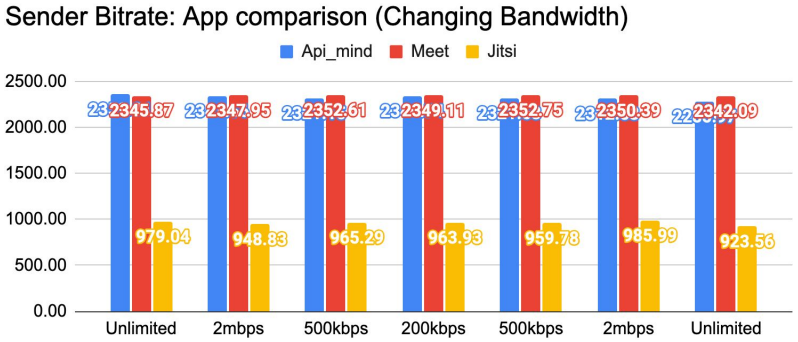




apiMind has higher Receiver bitrate consumption at baseline compared to **Google Meet**.
Jitsi has the lowest Receiver bitrate.

Sender bitrate comparison

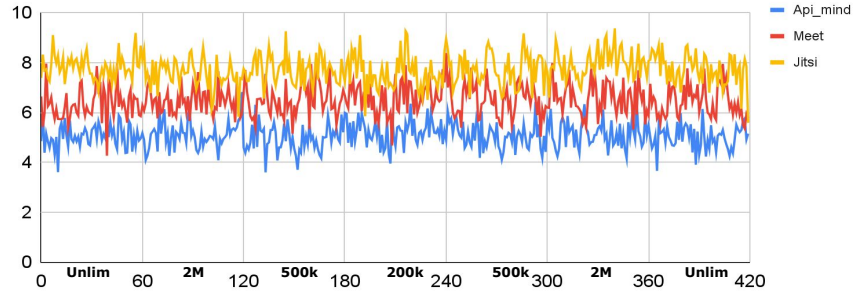




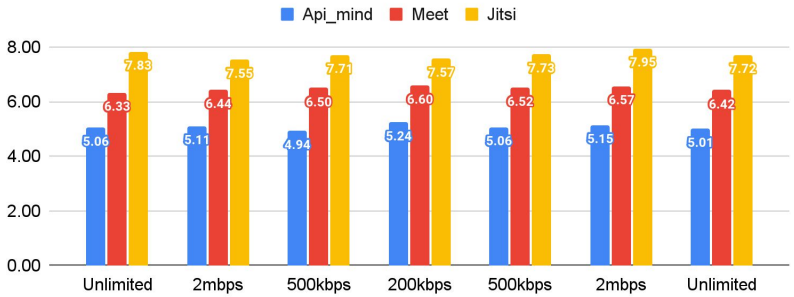
For all apps Sender bitrate does not adapt to Receiver bitrate and is constant throughout the tests.

CPU comparison

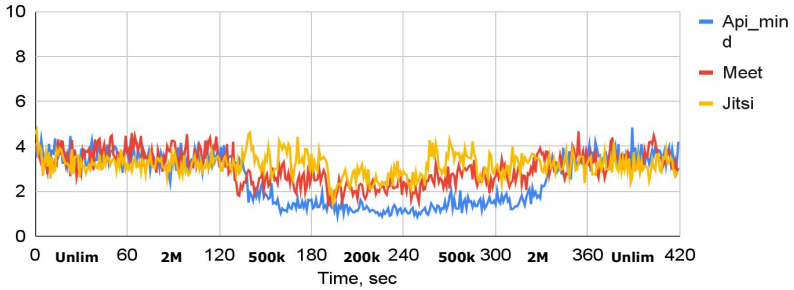
CPU Sender Overtime: App Comparison (Changing Bandwidth)



CPU Sender: App comparison (Changing Bandwidth)



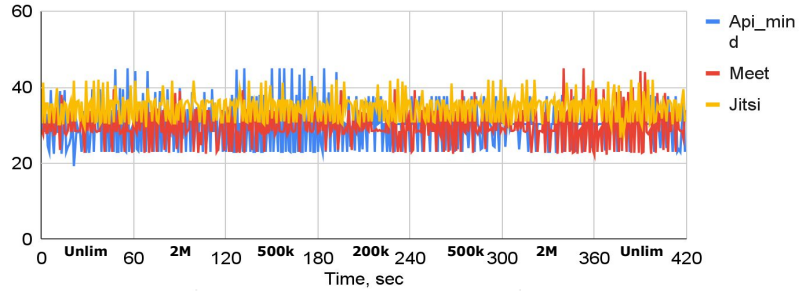
CPU Receiver Overtime: App Comparison (Changing Bandwidth)



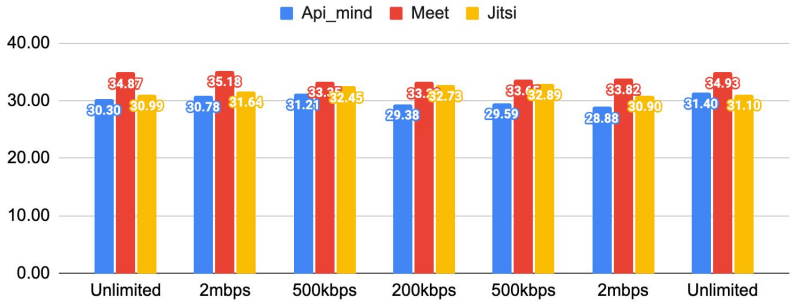
CPU Receiver: App comparison (Changing Bandwidth)

GPU comparison

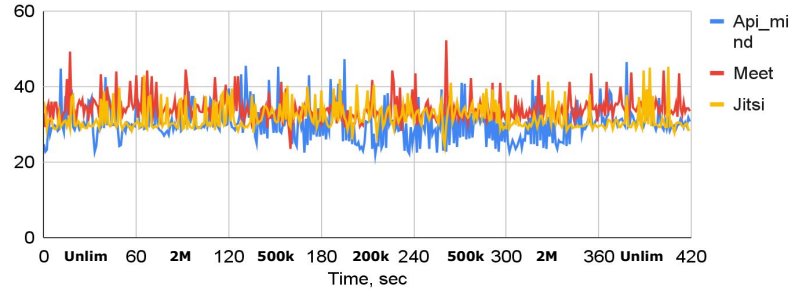
GPU Sender Overtime: App Comparison (Changing Bandwidth)



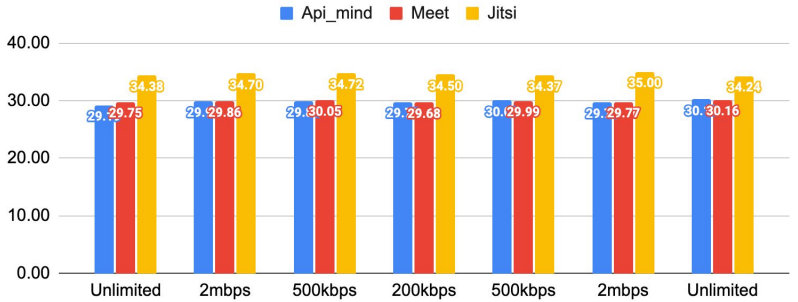
GPU Receiver: App comparison (Changing Bandwidth)



GPU Receiver Overtime: App Comparison (Changing Bandwidth)

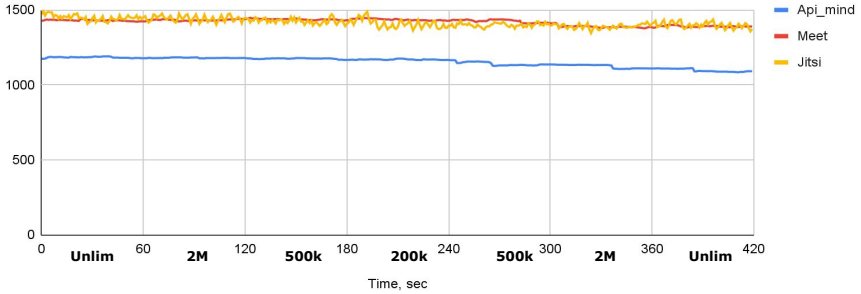


GPU Sender: App comparison (Changing Bandwidth)

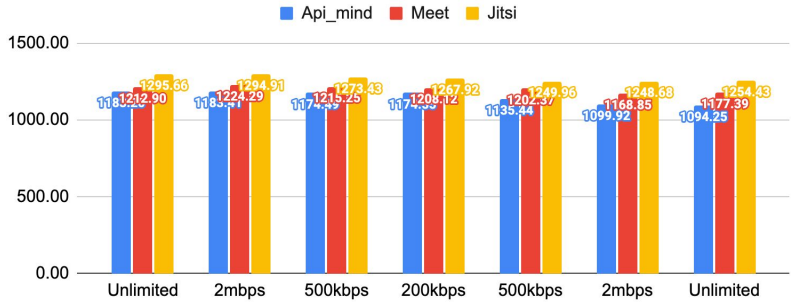


Memory comparison

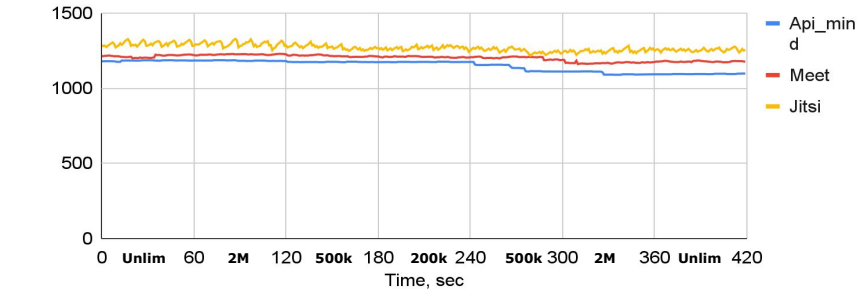
RAM Sender Overtime: App Comparison (Changing Bandwidth)



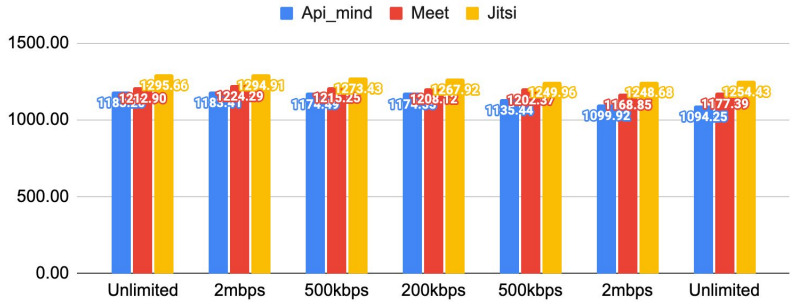
RAM Sender: App comparison (Changing Bandwidth)



RAM Receiver Overtime: App Comparison (Changing Bandwidth)

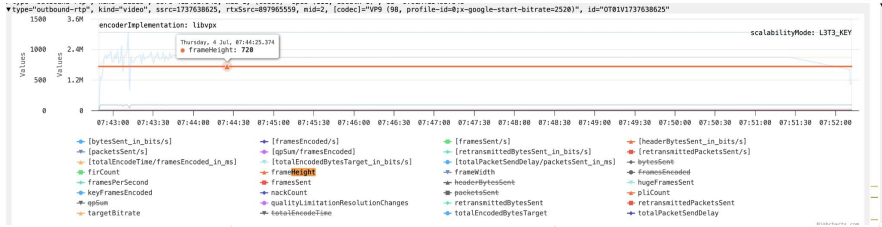


RAM Receiver: App comparison (Changing Bandwidth)

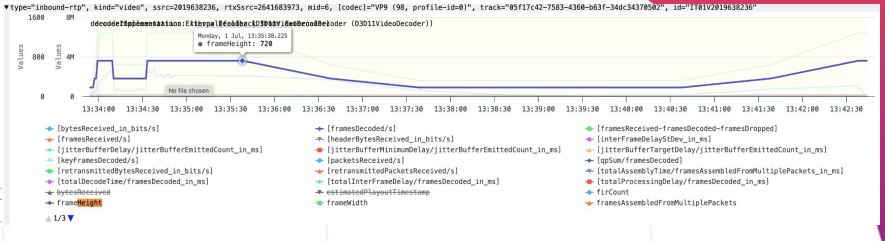


Resolution

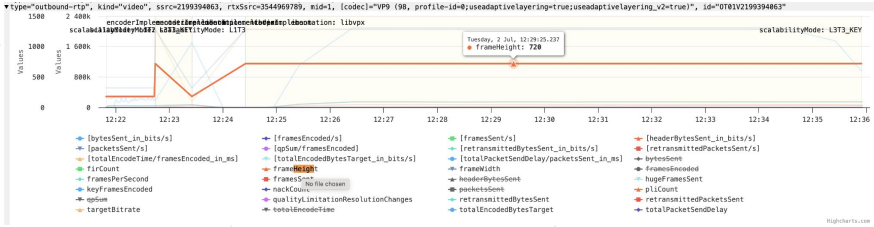
apiMind changing Bandwidth sender webrtc



apiMind changing Bandwidth receiver webrtc

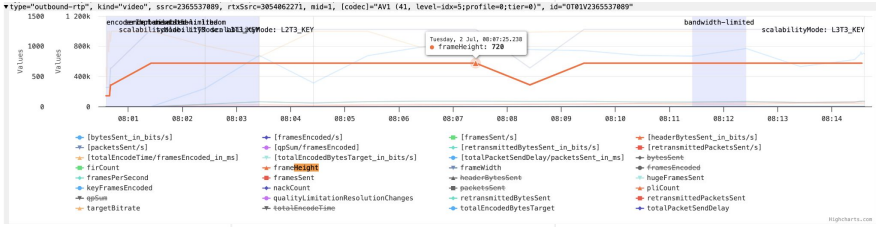


Meet changing Bandwidth sender webrtc



Meet changing Bandwidth receiver webrtc

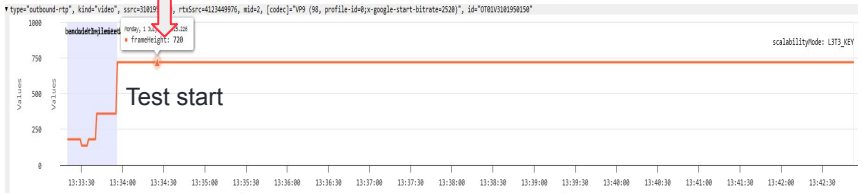
Jitsi changing Bandwidth sender webrtc



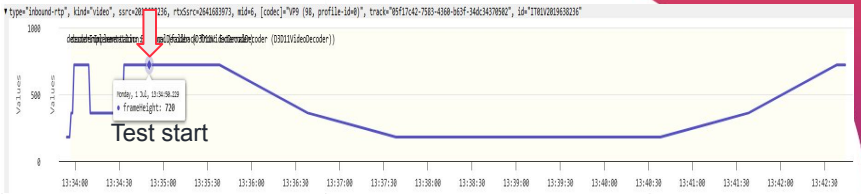
Jitsi changing Bandwidth receiver webrtc

Resolution - UPDATED

apiMind changing Bandwidth sender webrtc

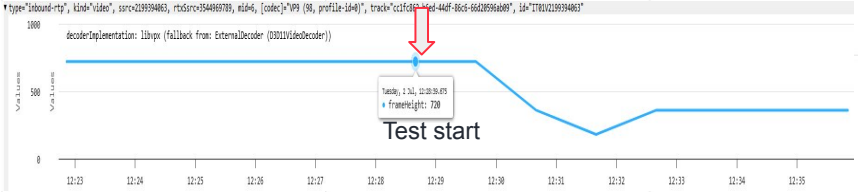


apiMind changing Bandwidth receiver webrtc

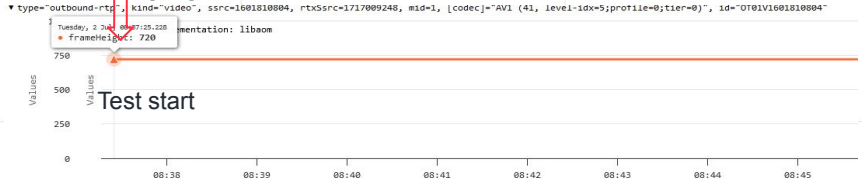


Meet changing Bandwidth sender webrtc

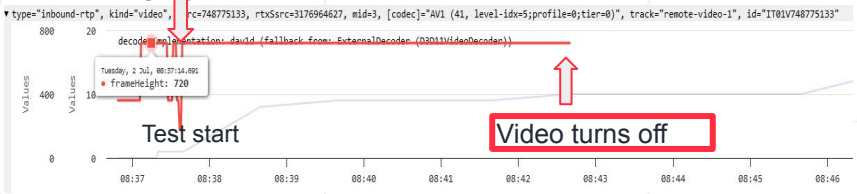
Meet changing Bandwidth receiver webrtc



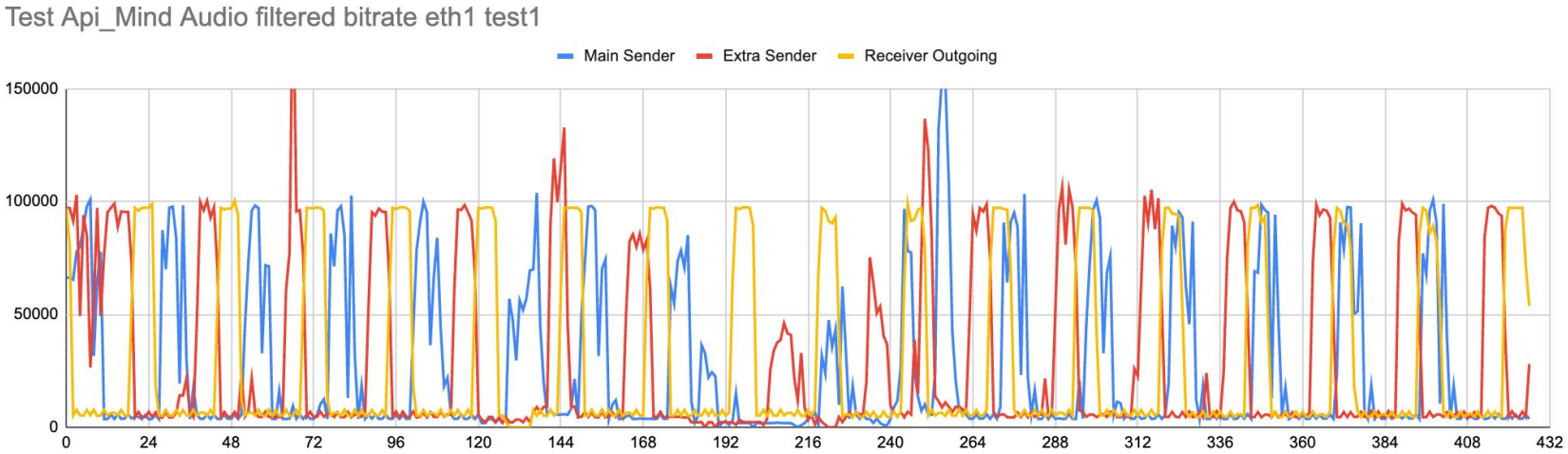
Jitsi changing Bandwidth sender webrtc



Jitsi changing Bandwidth receiver webrtc



Audio bitrate



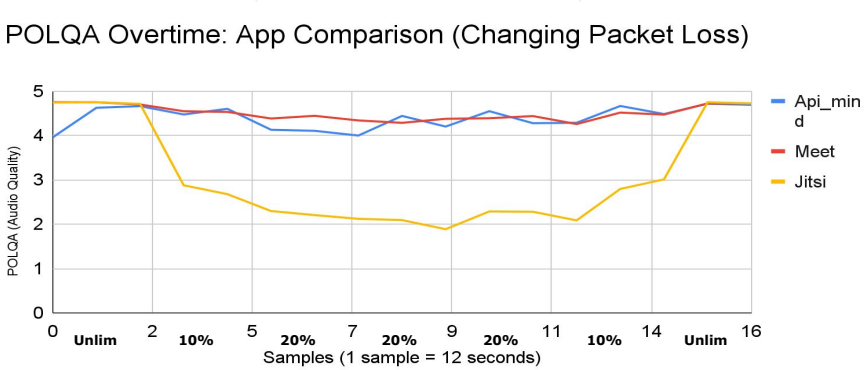
Audio bitrate chart of one test. All tests showed similar trend

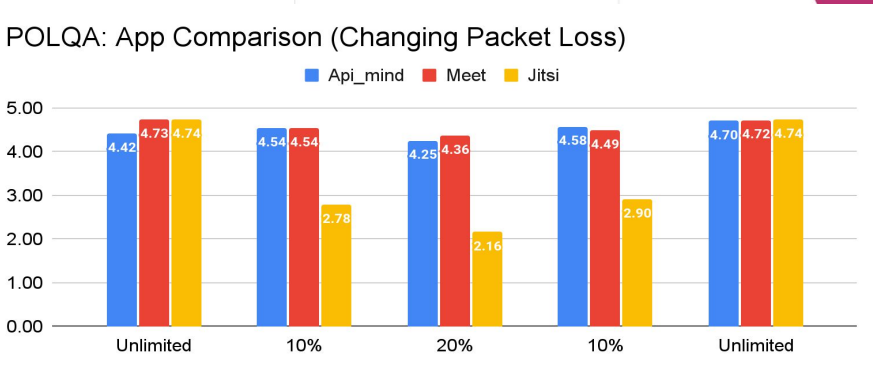
CHANGING PACKET LOSS

Changing Packet Loss Test Process

1. Sender creates a room
2. Receiver starts recording the screen and performance/delay data
3. Sender starts playing the video using OBS
4. Audio script along with network trace capture and “Changing Packet Loss” script are executed with conditions:
    1. Unlimited limitation enabled for 1 minute
    2. 10% limitation enabled for 1 minute
    3. 20% limitation enabled for 1 minute
    4. 20% limitation enabled for 1 minute
    5. 20% limitation enabled for 1 minute
    6. 10% limitation enabled for 1 minute
    7. Unlimited limitation enabled for 1 minute
5. Test ends when the sender video reaches white screen, delay video recording and network trace capturing is stopped
6. Receivers leave the room/call
7. Sender disconnects from the room/call and the chrome browser is restarted

POLQA comparison

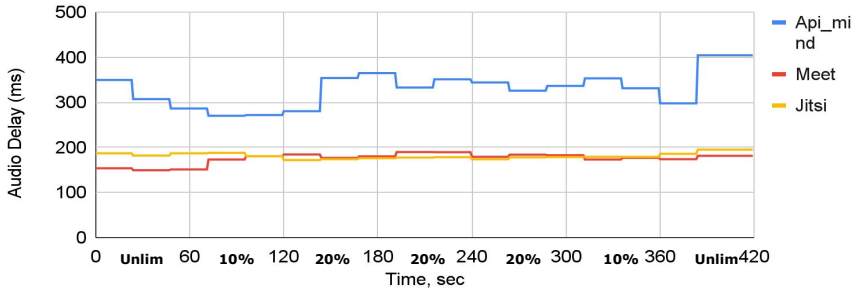




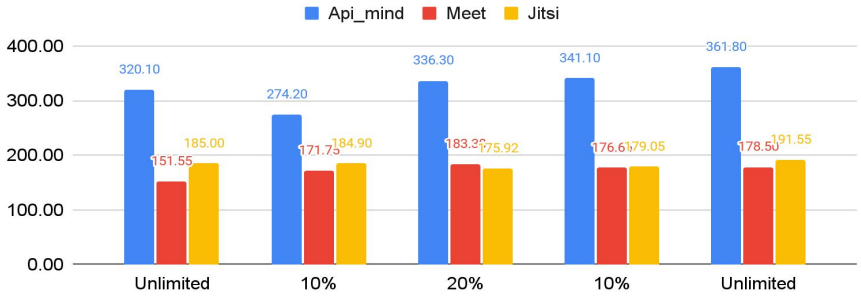
apiMind is on-par **Google Meet** in all scenarios.
Jitsi has fair score in 10% & 20% packet loss limitation.

Audio Delay comparison

Audio Delay Overtime: App Comparison (Changing Packet Loss)



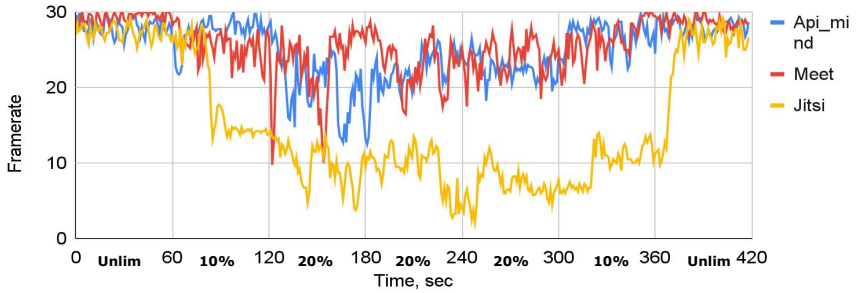
Audio Delay: App Comparison (Changing Packet Loss)



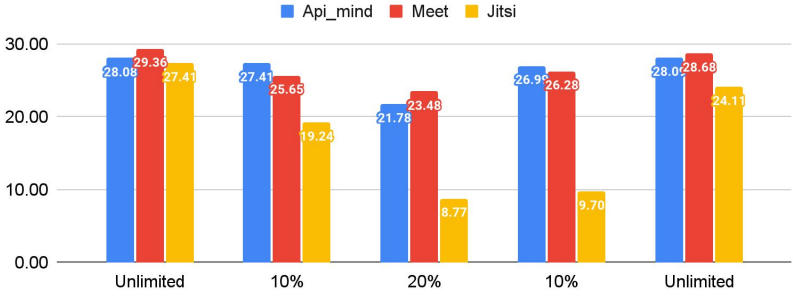
apiMind has higher Audio Delay than **Google Meet** & **Jitsi**.

FPS comparison

FPS Overtime: App Comparison (Changing Packet Loss)



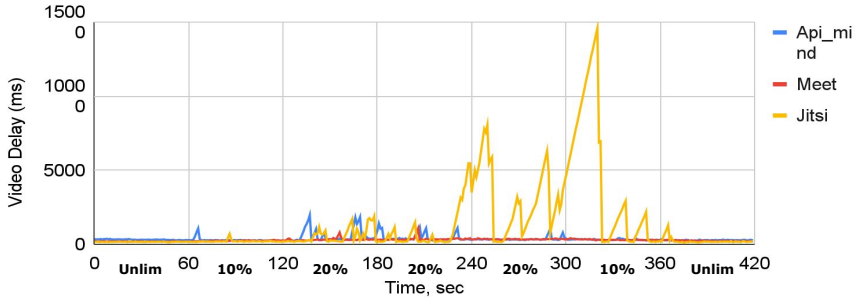
FPS: App Comparison (Changing Packet Loss)



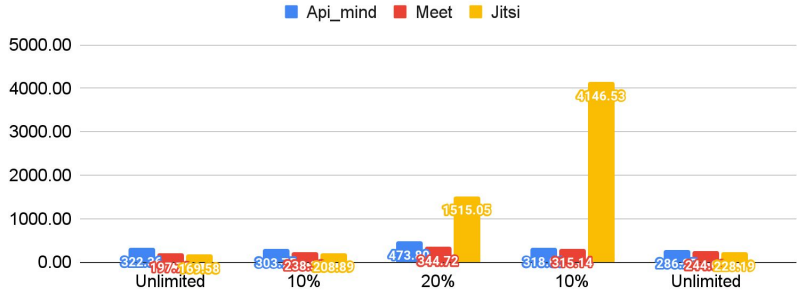
apiMind is on-par with **Google Meet** in all conditions.
Jitsi has issues in 10% & 20% packet loss conditions.

Video Delay comparison

Video Delay Overtime: App Comparison (Changing Packet Loss)



Video Delay: App Comparison (Changing Packet Loss)

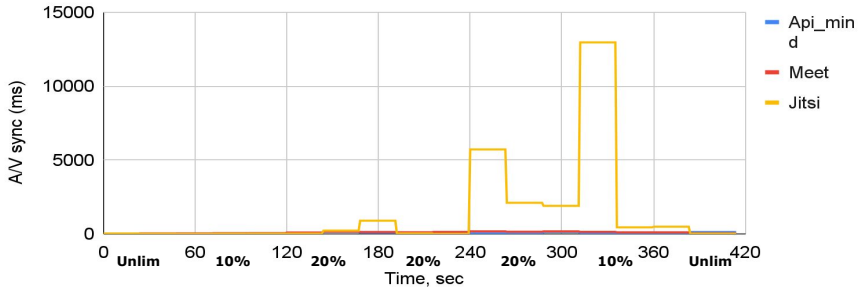


Google Meet has slightly lower Video Delay than **apiMind** in all conditions.

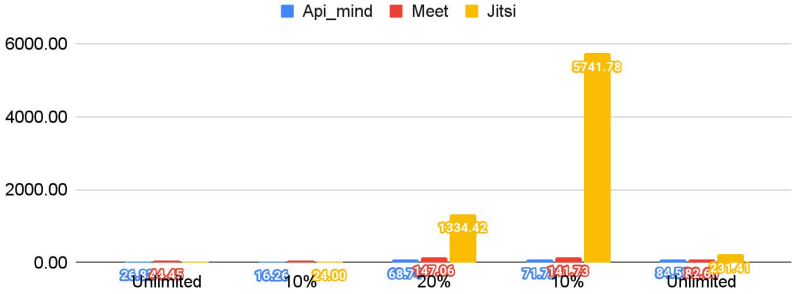
Video delay increases for all apps in 20% packet loss condition, but for it is especially noticeable for **Jitsi**.

Audio and Video synchronization comparison

AV Sync Overtime: App Comparison (Changing Packet Loss)



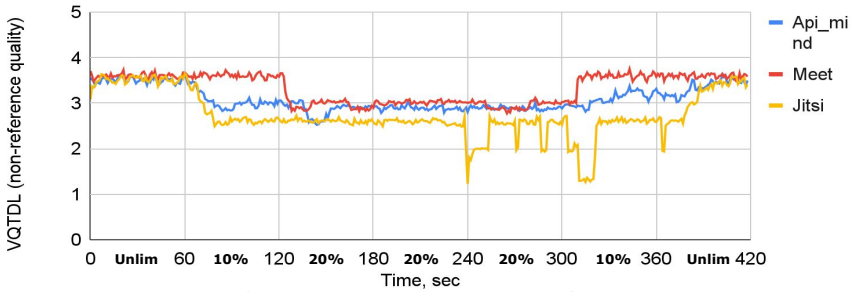
AV Sync: App comparison - (Changing Packet Loss)



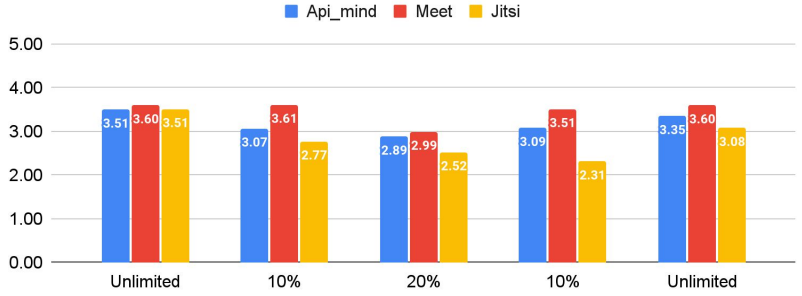
apiMind has better audio/video synchronization than **Google Meet** in all Packet loss conditions.
Jitsi scored poorly after 20% Packet loss condition with video falling behind audio.

VQTDL comparison

VQTDL Overtime: App Comparison (Changing Packet Loss)



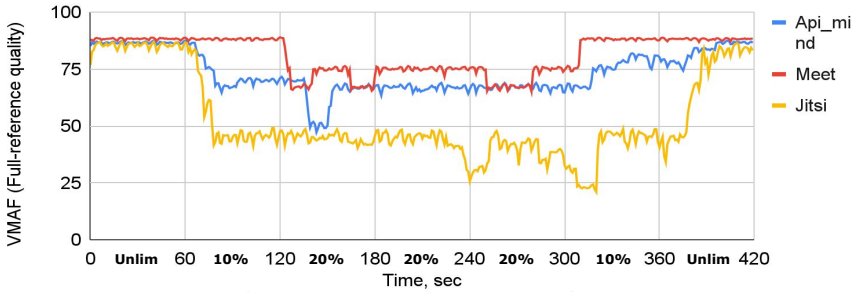
VQTDL: App Comparison (Changing Packet Loss)



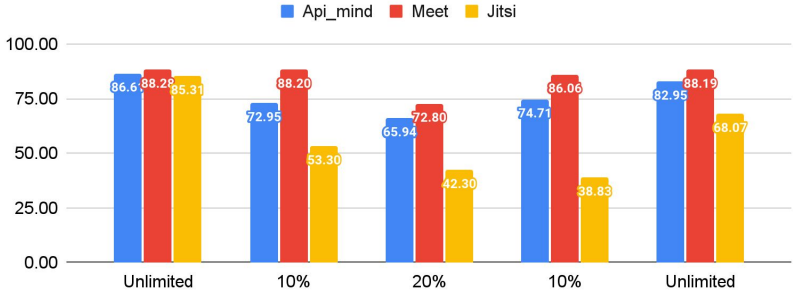
Google Meet has higher image quality than **apiMind**.
apiMind is on-par with **Jitsi**.
Google Meet slightly drops quality in 20% Packet loss limitation.
Jitsi switches the video on and off in 20% Packet loss limitation.

VMAF comparison

VMAF Overtime: App Comparison (Changing Packet Loss)



VMAF: App Comparison (Changing Packet Loss)

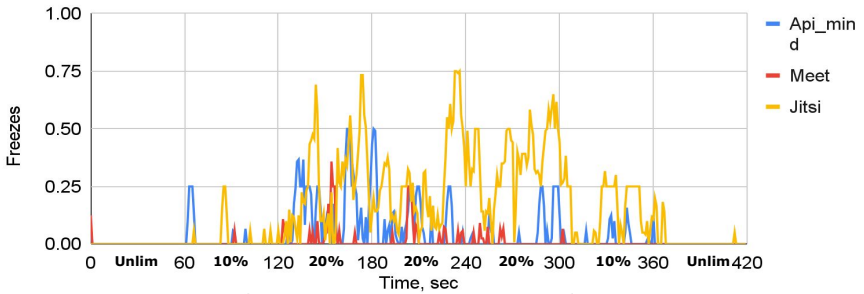


All three apps have similar score in beginning.
Google Meet scored the highest in all network conditions.
apiMind in the middle, **Jitsi** scored the lowest.

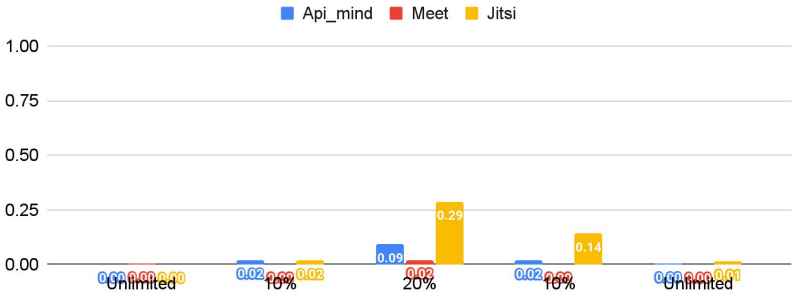
apiMind does not recover to original quality in the end.

Freeze count comparison

Freezes Count Overtime: App Comparison (Changing Packet Loss)



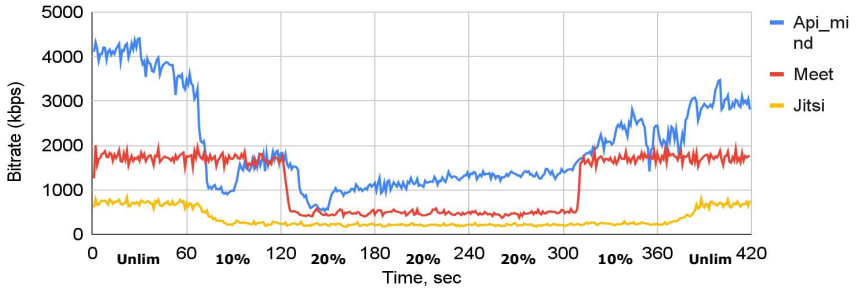
Freezes Count: App Comparison (Changing Packet Loss)



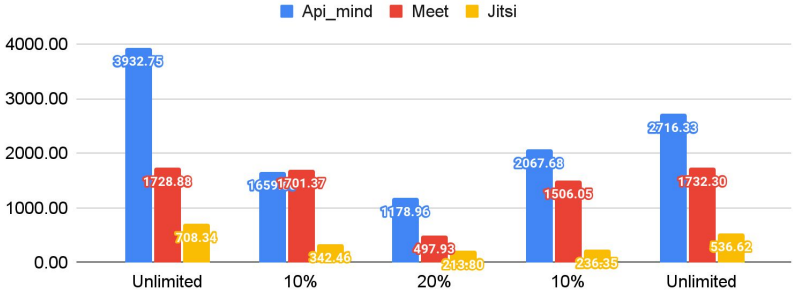
Jitsi has the most freezes in Changing Packet loss scenario.
apiMind has minimal freezes, but more frequent than **Google Meet**.

Receiver bitrate comparison

Receiver Bitrate Overtime: App Comparison (Changing Packet Loss)



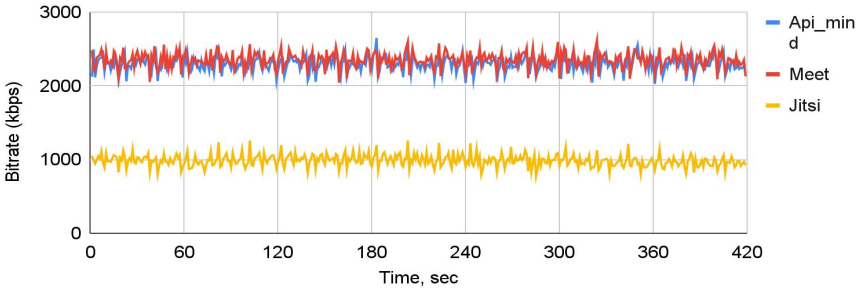
Receiver Bitrate: App Comparison (Changing Packet Loss)



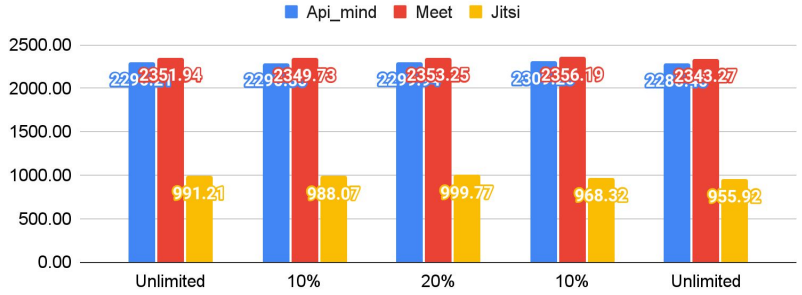
apiMind has higher average Receiver bitrate than **Google Meet** & **Jitsi**.

Sender bitrate comparison

Sender Bitrate Overtime: App Comparison (Changing Packet Loss)



Sender Bitrate: App Comparison (Changing Packet Loss)

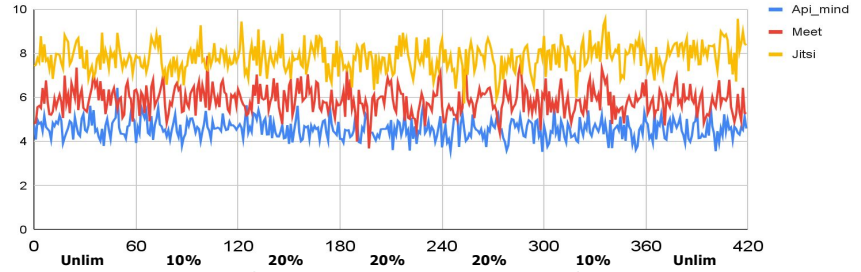


All app Sender bitrate does not adapt to Receiver bitrate in Changing Packet loss scenario, similar like in Changing Bitrate scenario.

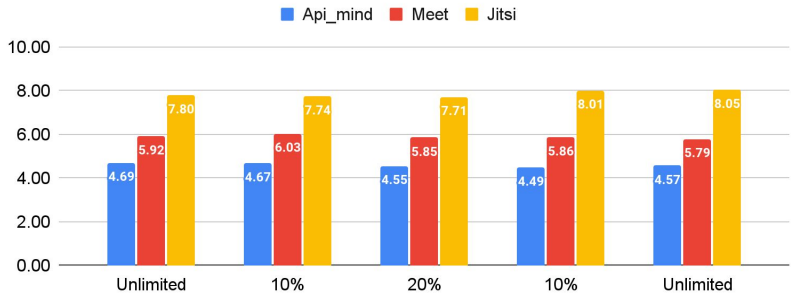
Google Meet has the highest sender bitrate, but is only slightly higher than **apiMind**.

CPU comparison

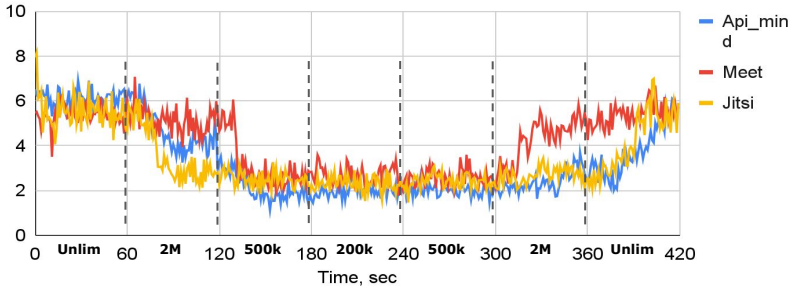
CPU Sender Overtime: App Comparison (Changing Packet Loss)



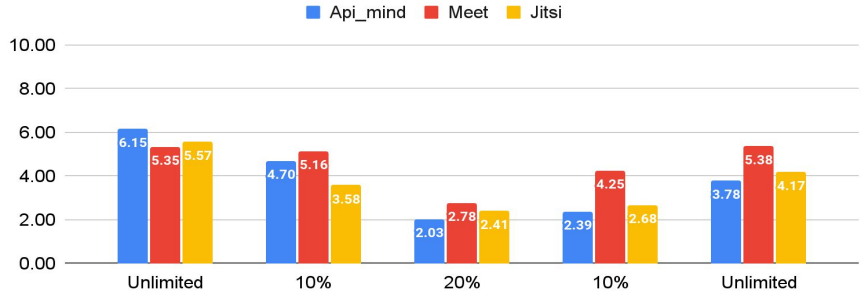
CPU Sender: App Comparison (Changing Packet Loss)



CPU Receiver Overtime: App Comparison (Changing Packet Loss)

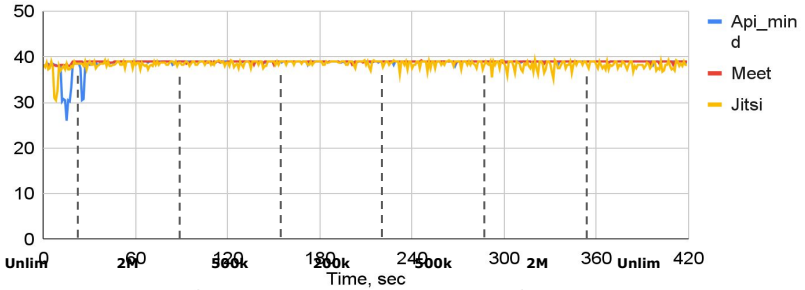


CPU Receiver: App Comparison (Changing Packet Loss)

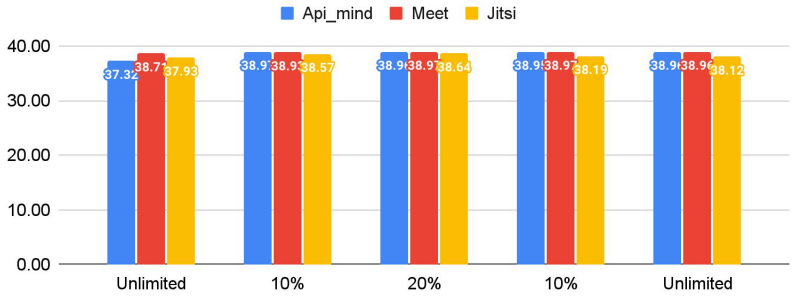


GPU comparison

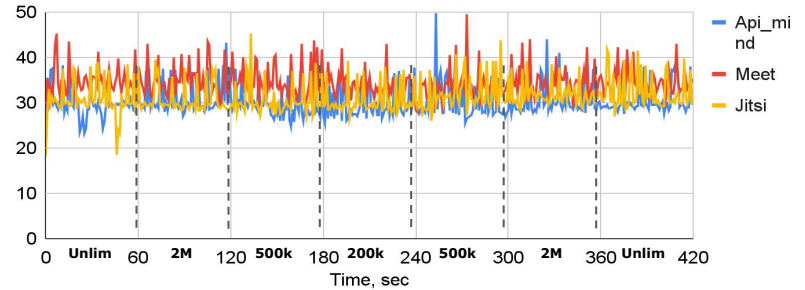
GPU Sender Overtime: App Comparison (Changing Packet Loss)



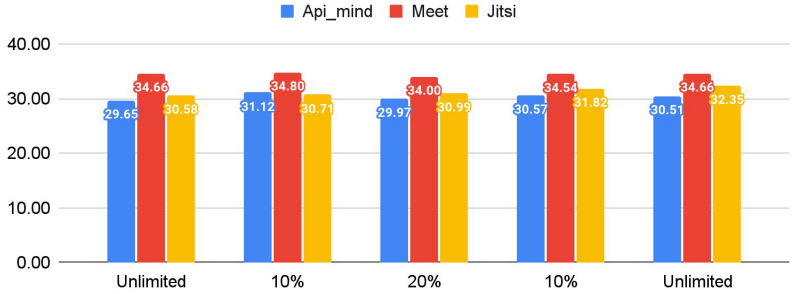
GPU Sender: App Comparison (Changing Packet Loss)



GPU Receiver Overtime: App Comparison (Changing Packet Loss)

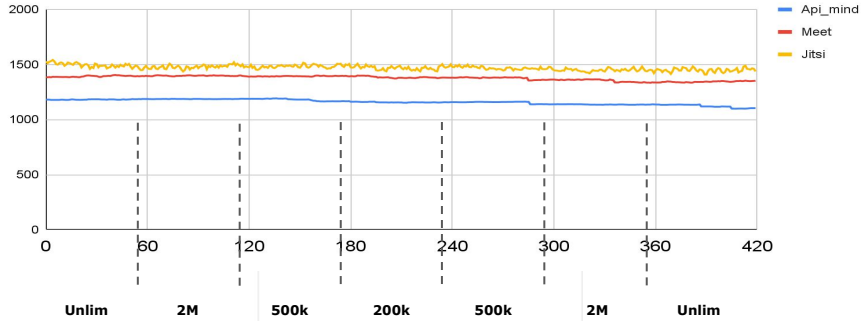


GPU Receiver: App Comparison (Changing Packet Loss)

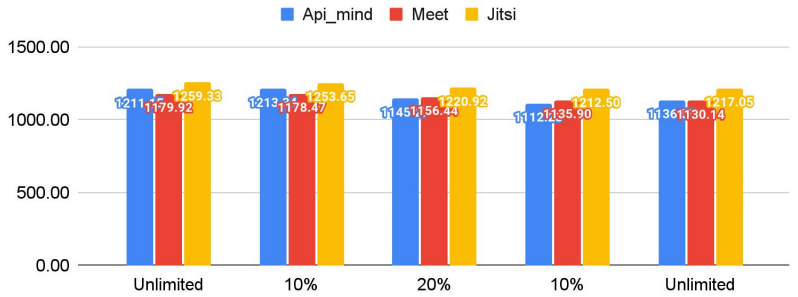


Memory comparison

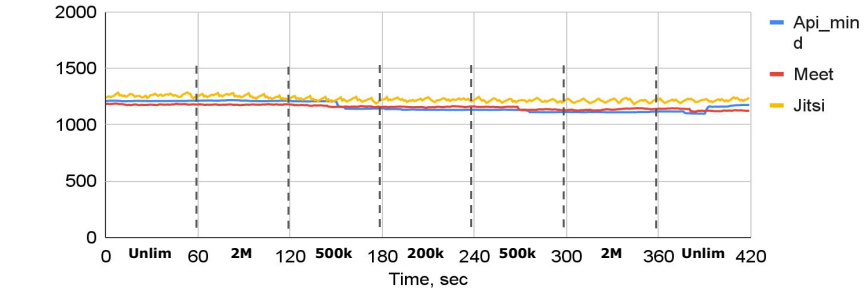
RAM Sender Overtime: App Comparison (Changing Packet Loss)



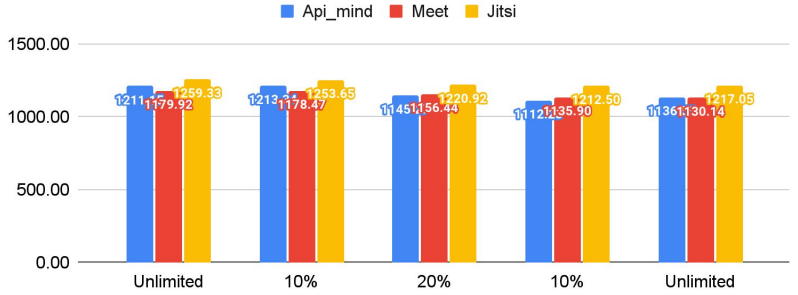
RAM Sender: App Comparison (Changing Packet Loss)



RAM Receiver Overtime: App Comparison (Changing Packet Loss)

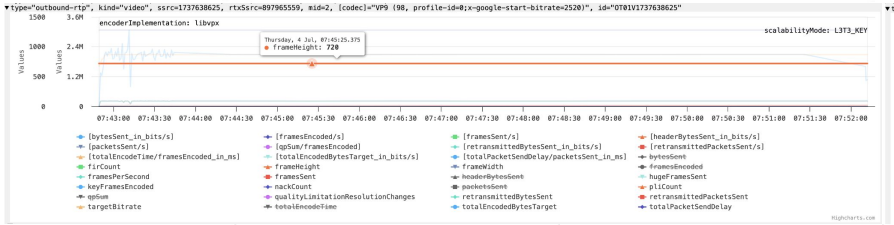


RAM Receiver: App Comparison (Changing Packet Loss)



Resolution

apiMind changing Packet Loss sender webrtc

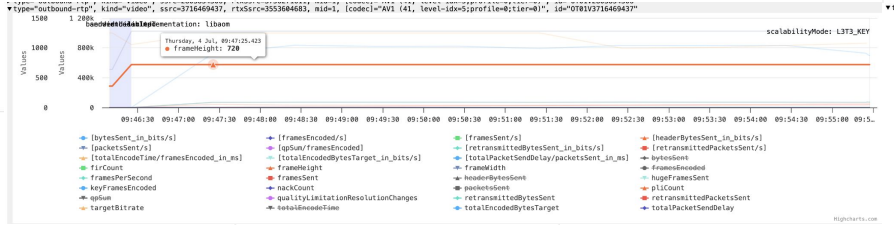


apiMind changing Packet Loss receiver webrtc

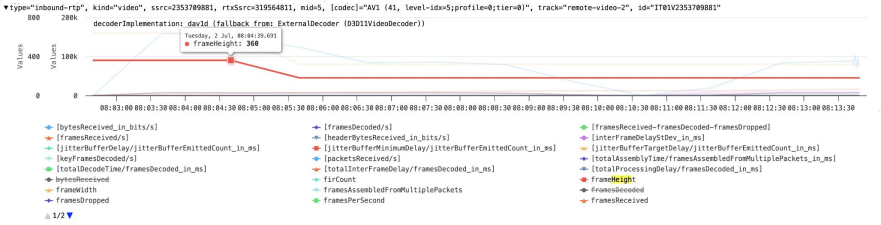
Meet changing Packet Loss sender webrtc

Meet changing Packet Loss receiver webrtc

Jitsi changing Packet Loss sender webrtc

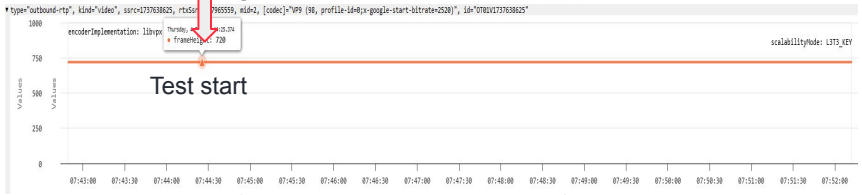


Jitsi changing Packet Loss receiver webrtc

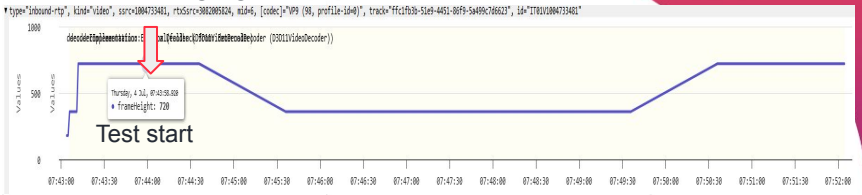


Resolution - UPDATED

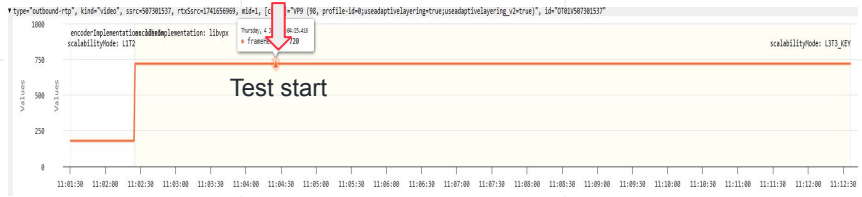
apiMind changing Packet Loss sender webrtc



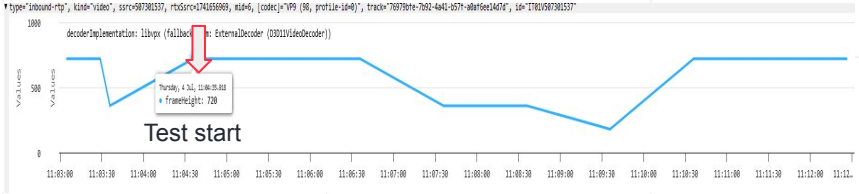
apiMind changing Packet Loss receiver webrtc



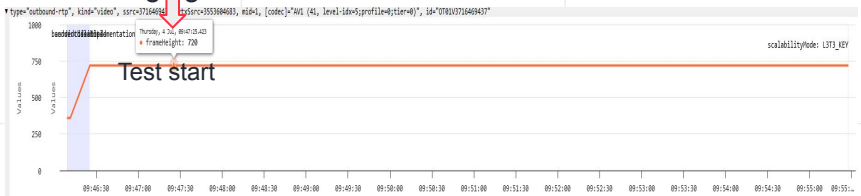
Meet changing Packet Loss sender webrtc



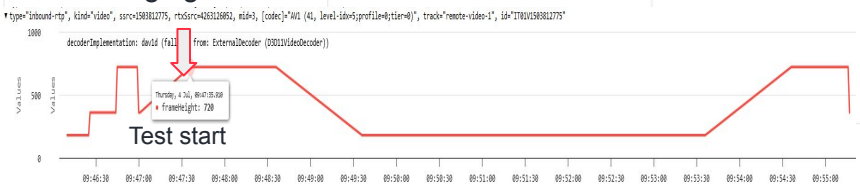
Meet changing Packet Loss receiver webrtc



Jitsi changing Packet Loss sender webrtc



Jitsi changing Packet Loss receiver webrtc



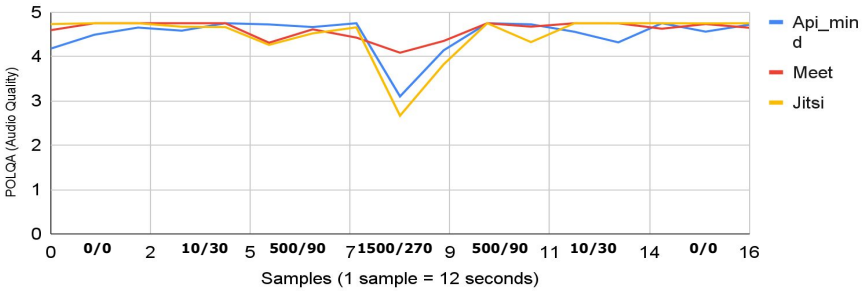
CHANGING LATENCY AND JITTER

Changing Latency and Jitter Test Process

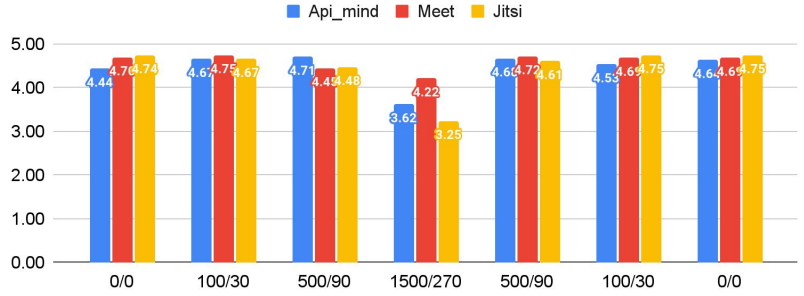
1. Sender creates a room
2. Receiver starts recording the screen and performance/delay data
3. Sender starts playing the video on OBS
4. Audio script along with network trace capture and “Changing Packet Loss” script are executed with conditions:
    1. 0/0 ms limitation enabled for 1 minute
    2. 10/30 ms limitation enabled for 1 minute
    3. 500/90 ms limitation enabled for 1 minute
    4. 1500/270 ms limitation enabled for 1 minute
    5. 500/90 ms limitation enabled for 1 minute
    6. 10/30 ms limitation enabled for 1 minute
    7. 0/0 ms limitation enabled for 1 minute
5. Test ends when the sender video reaches white screen, delay video recording and network trace capturing is stopped
6. Receivers leave the room/call
7. Sender disconnects from the room/call and the chrome browser is restarted

POLQA comparison

POLQA Overtime: App comparison (Changing Latency/Jitter)



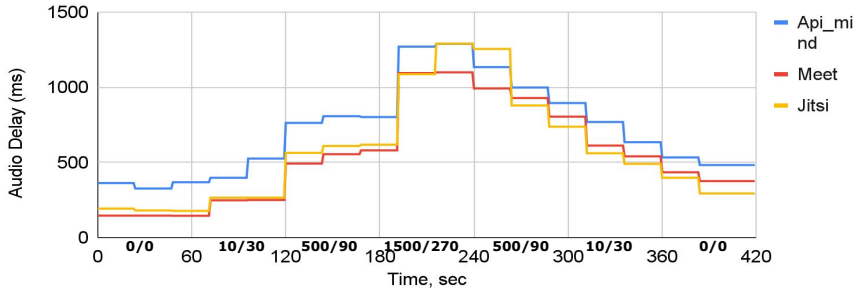
POLQA: App Comparison (Changing Latency/Jitter)



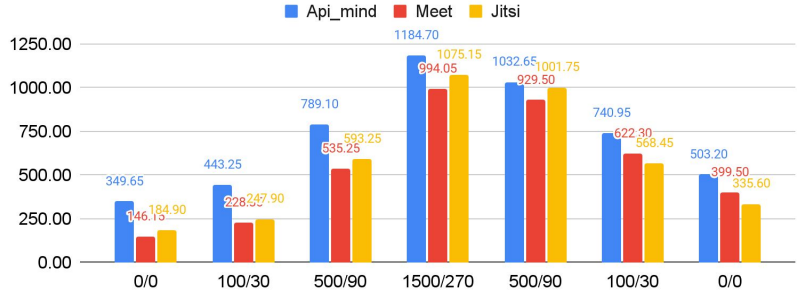
apiMind is on-part with **Google Meet** & **Jitsi** in Jitter/Latency limitation.
apiMind & **Jitsi** has POLQA drop in 1500 Jitter/270 Latency limitation period.

Audio Delay comparison

Audio Delay Overtime: App comparison (Changing Latency/Jitter)



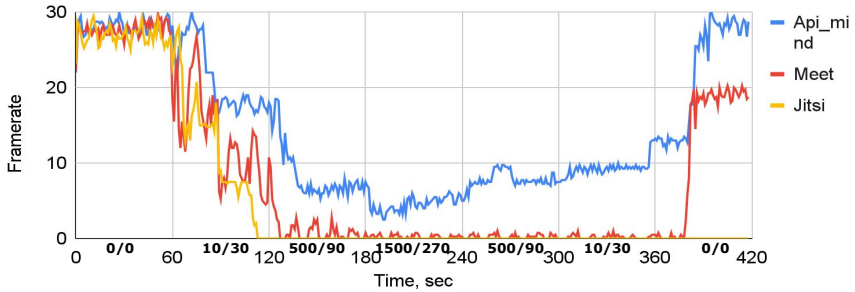
Audio Delay: App Comparison (Changing Latency/Jitter)



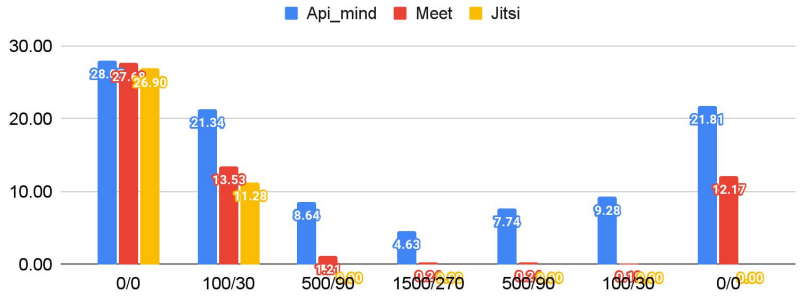
Google Meet has the lowest audio delay in all conditions.
apiMind has the highest.

FPS comparison

FPS Overtime: App comparison (Changing Latency/Jitter)



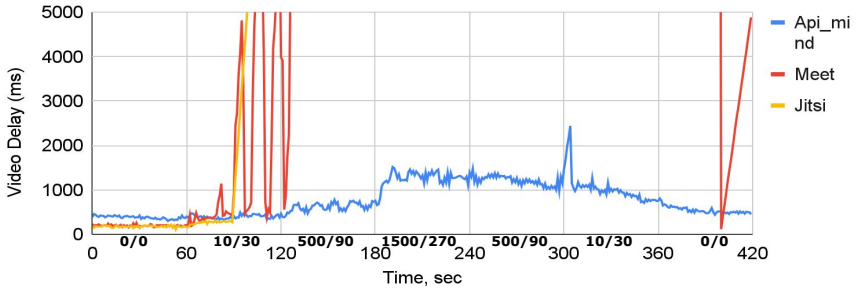
FPS: App Comparison (Changing Latency/Jitter)



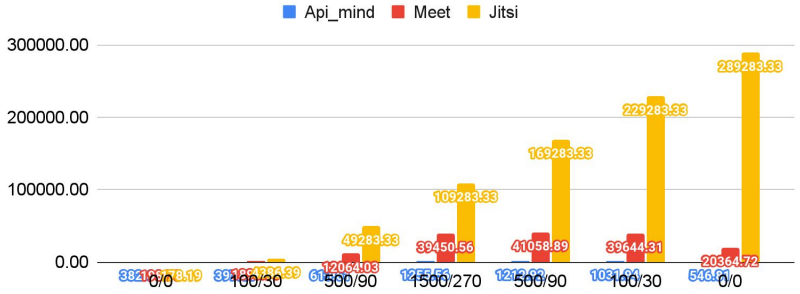
When Jitter/Latency limitation is applied, all apps have FPS drop and it stays low until limitation is removed.
apiMind FPS average value is higher than **Google Meet** & **Jitsi**.

Video Delay comparison

Video Delay Overtime: App comparison (Changing Latency/Jitter)



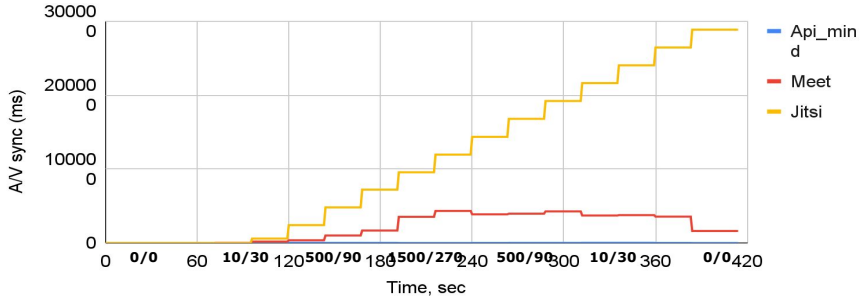
Video Delay: App Comparison (Changing Latency/Jitter)



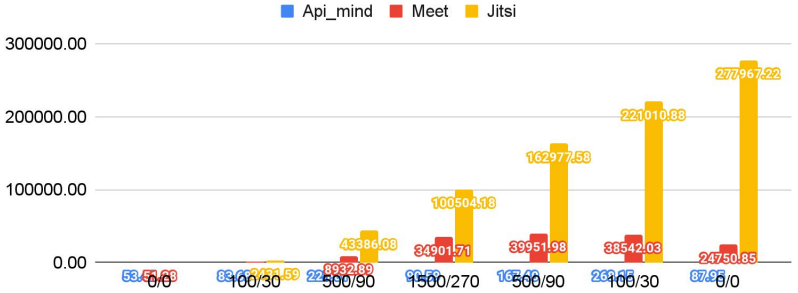
apiMind has the lowest video delay when Jitter/Latency limitation is applied in comparison with **Google Meet** & **Jitsi**.

Audio and Video synchronization comparison

AV Sync Overtime: App comparison (Changing Latency/Jitter)



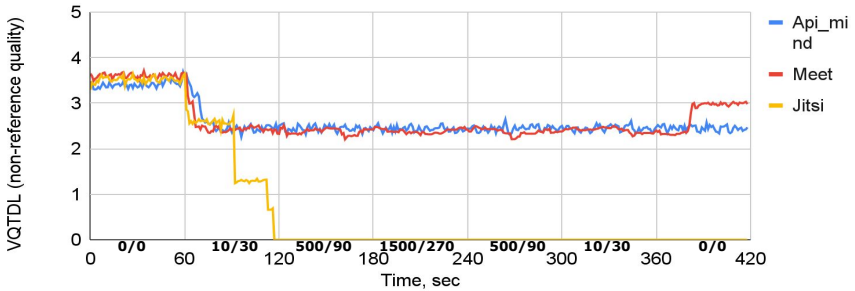
AV Sync: App Comparison (Changing Latency/Jitter)



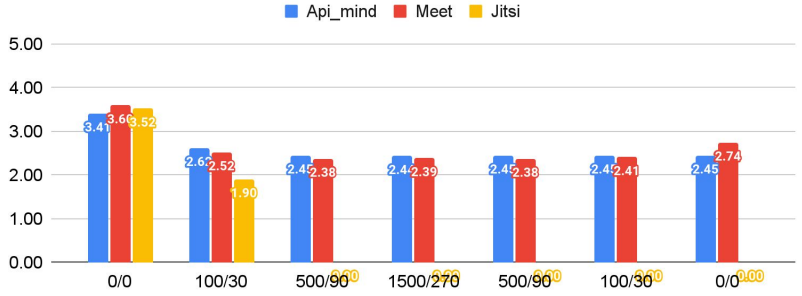
apiMind has the best audio/video synchronization when Jitter/Latency limitation is applied.

VQTDL comparison

VQTDL Overtime: App comparison (Changing Latency/Jitter)



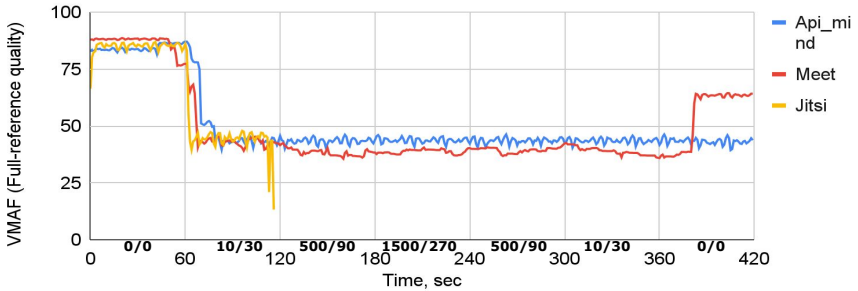
VQTDL: App Comparison (Changing Latency/Jitter)



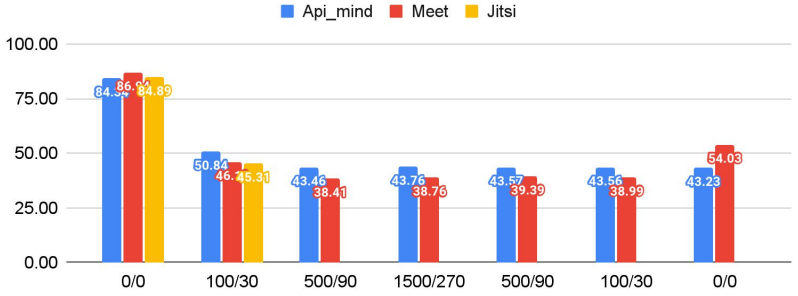
apiMind has higher VQTDL than **Google Meet** when Jitter/Latency limitation is applied.
Jitsi soon drops the video after limitation is applied.

VMAF comparison

VMAF Overtime: App comparison (Changing Latency/Jitter)



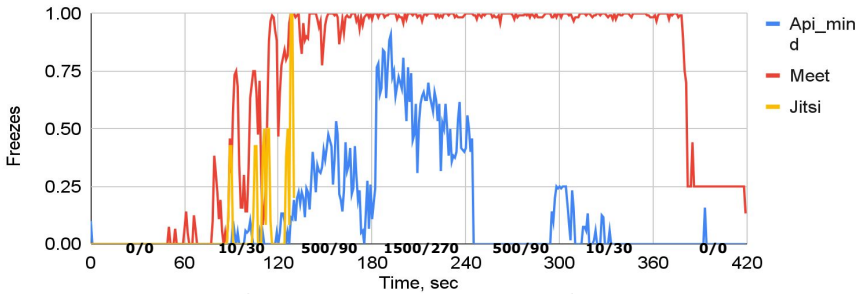
VMAF: App Comparison (Changing Latency/Jitter)



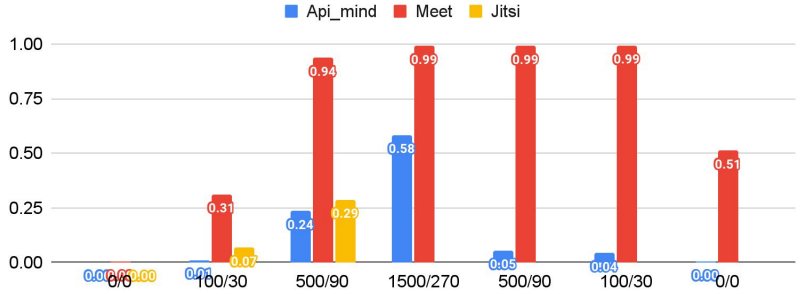
All competitors drop VMAF when limitation applies.
apiMind has higher VMAF than **Google Meet** & **Jitsi**.

Freeze count comparison

Freezes Count Overtime: App comparison (Changing Latency/Jitter)



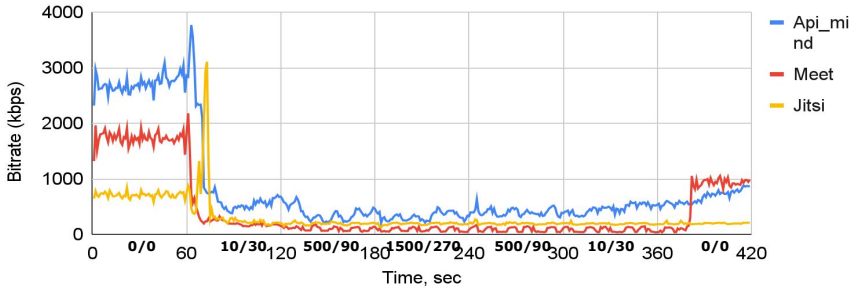
Freeze count: App Comparison (Changing Latency/Jitter)



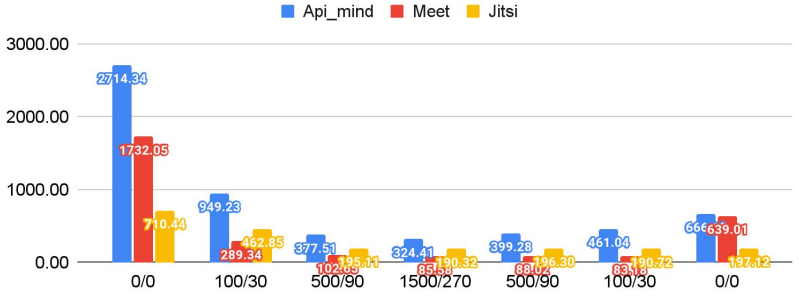
When Jitter/Latency limitations are applied, **apiMind** has the least freezes.
Google Meet has the most freezes once limitation is applied.
Jitsi drops the video, so no freezes are detected.

Receiver bitrate comparison

Receiver Bitrate Overtime: App comparison (Changing Latency/Jitter)



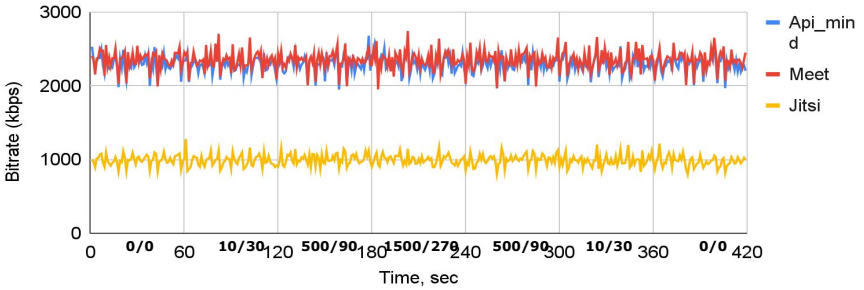
Receiver bitrate: App Comparison (Changing Latency/Jitter)



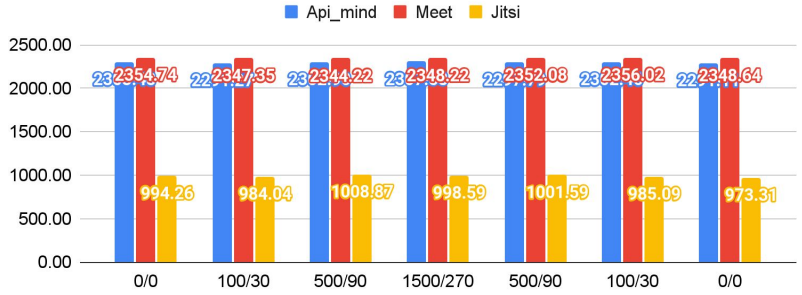
Participants drop their Receiver bitrate when limitation is applied.
apiMind has higher Receiver bitrate than **Google Meet** & **Jitsi** in all limitation conditions.

Sender bitrate comparison

Sender Bitrate Overtime: App comparison (Changing Latency/Jitter)



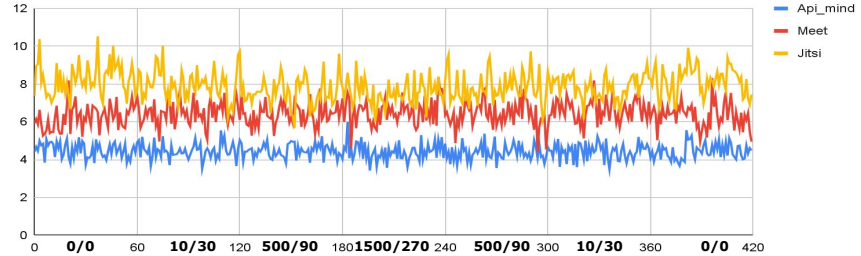
Sender bitrate: App Comparison (Changing Latency/Jitter)



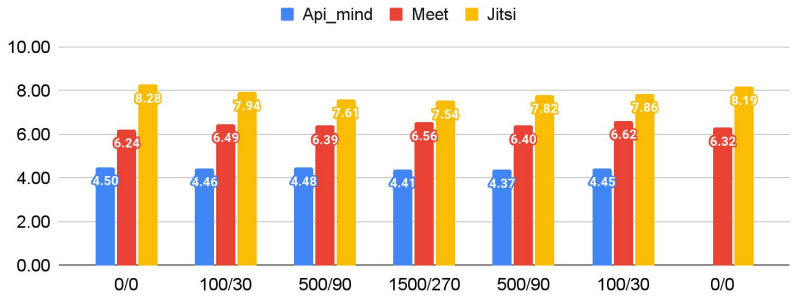
For all apps Sender bitrate is not affected by Receiver network limitation.
Google Meet has slightly higher Sender Bitrate than **apiMind**.

CPU comparison

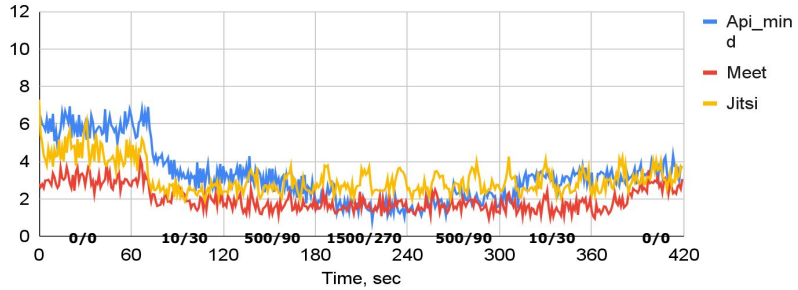
CPU Sender Overtime: App comparison (Changing Latency/Jitter)



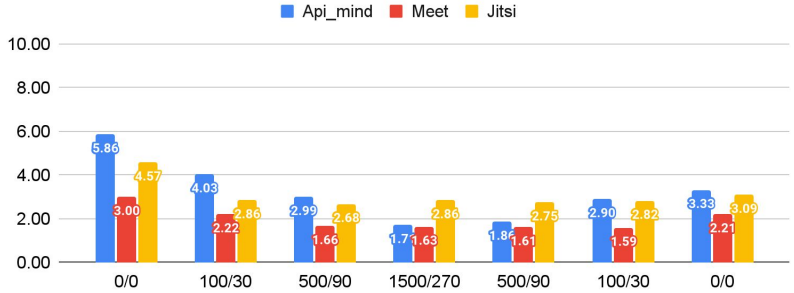
CPU Sender: App Comparison (Changing Latency/Jitter)



CPU Receiver Overtime: App comparison (Changing Latency/Jitter)

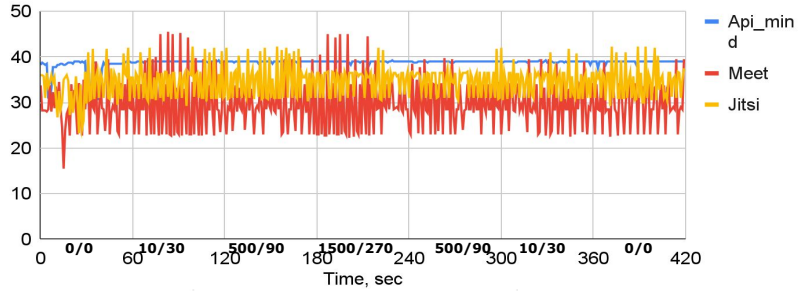


CPU Receiver: App Comparison (Changing Latency/Jitter)

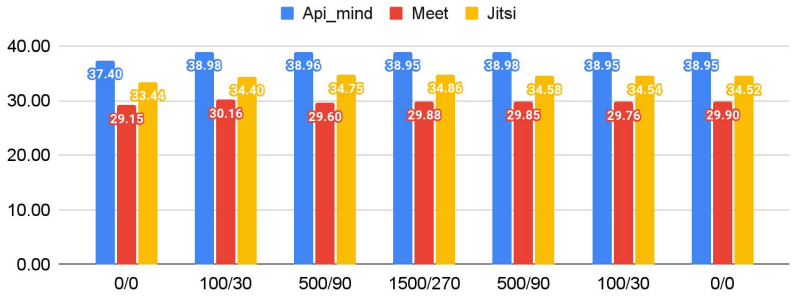


GPU comparison

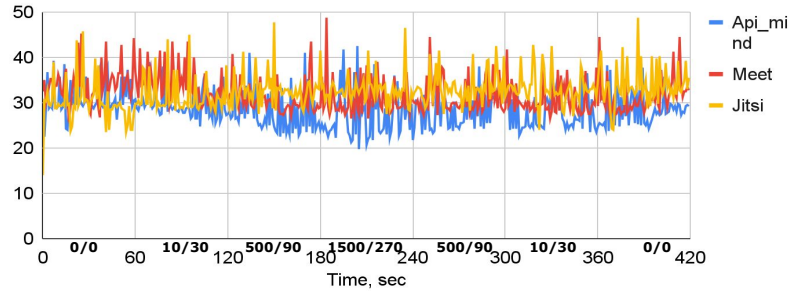
GPU Sender Overtime: App comparison (Changing Latency/Jitter)



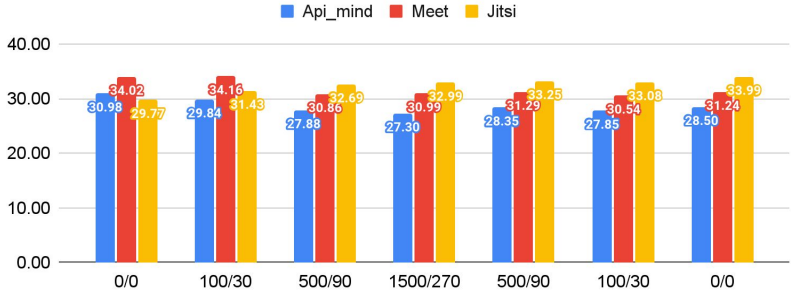
GPU Sender: App Comparison (Changing Latency/Jitter)



GPU Receiver Overtime: App comparison (Changing Latency/Jitter)

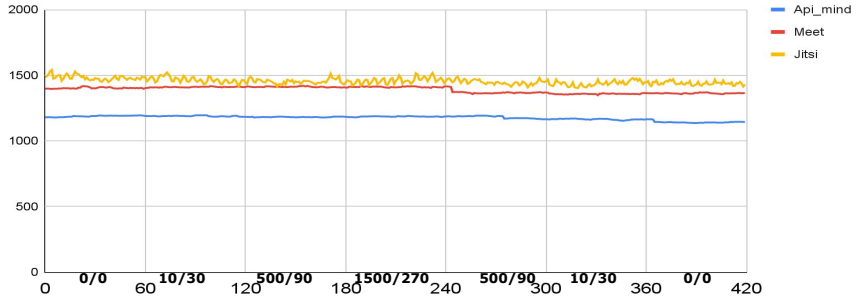


GPU Receiver: App Comparison (Changing Latency/Jitter)

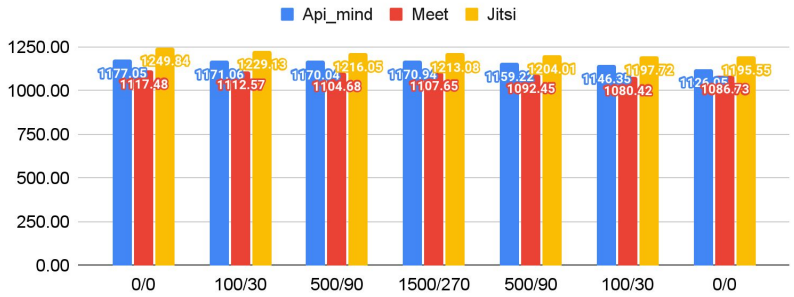


Memory comparison

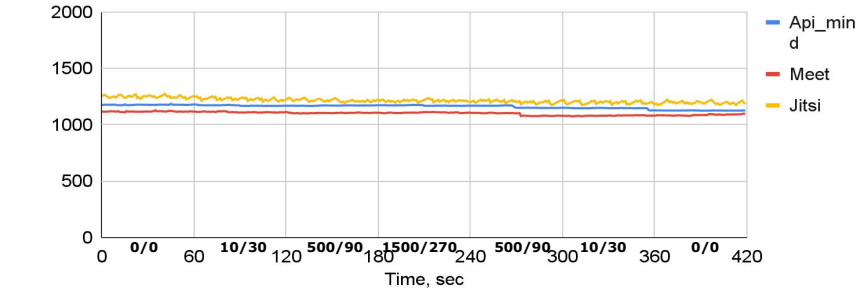
RAM Sender Overtime: App comparison (Changing Latency/Jitter)



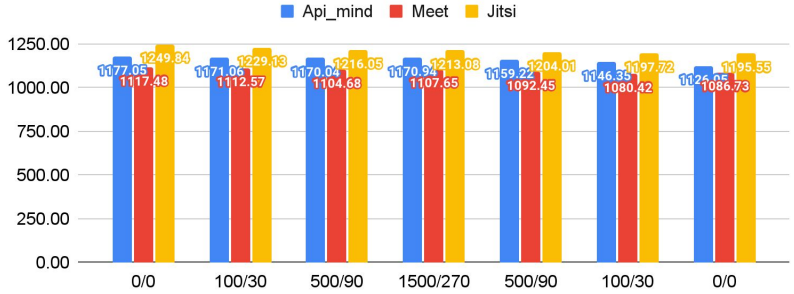
RAM Sender: App Comparison (Changing Latency/Jitter)



RAM Receiver Overtime: App comparison (Changing Latency/Jitter)

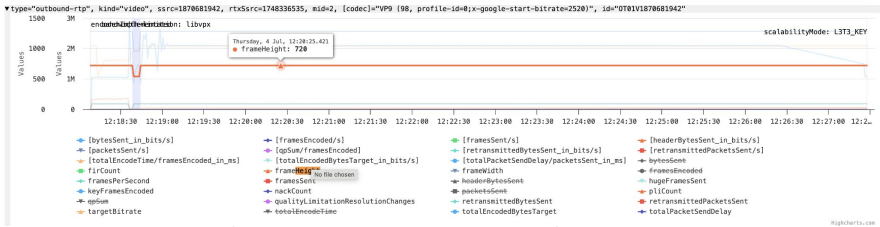


RAM Receiver: App Comparison (Changing Latency/Jitter)

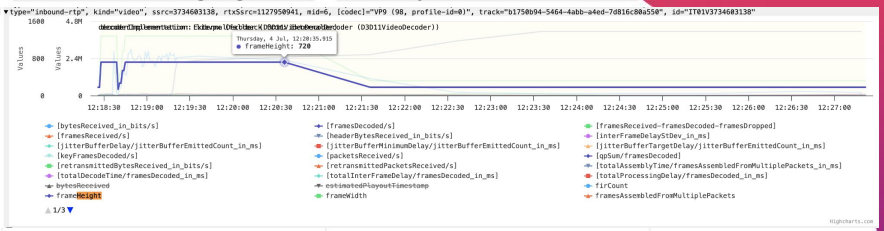


Resolution

apiMind changing Jitter/Latency sender webrtc

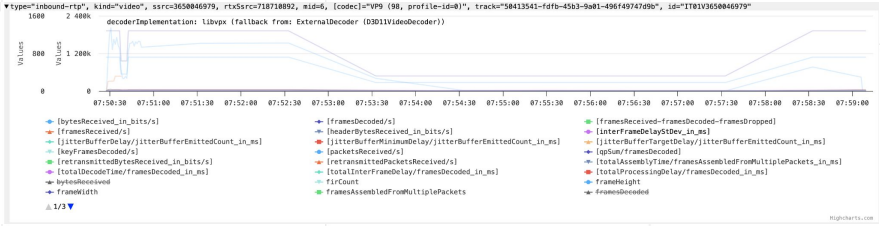


apiMind changing Jitter/Latency receiver webrtc



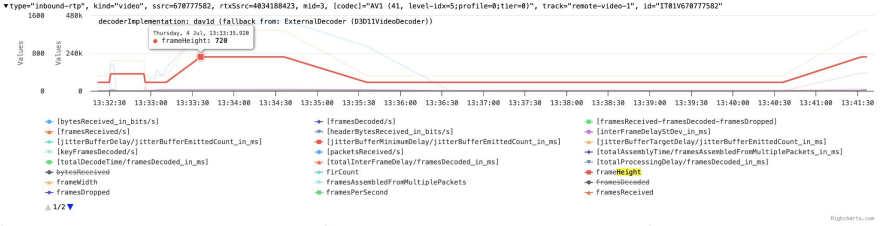
Meet changing Jitter/Latency sender webrtc

Meet changing Jitter/Latency receiver webrtc



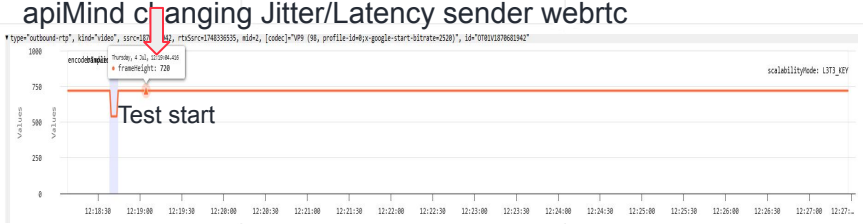
Jitsi changing Jitter/Latency sender webrtc

Jitsi changing Jitter/Latency receiver webrtc

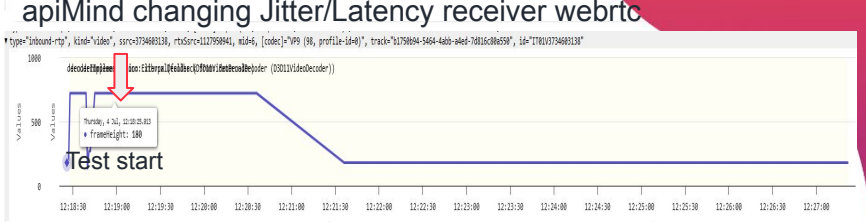


Resolution - UPDATED

apiMind changing Jitter/Latency sender webrtc

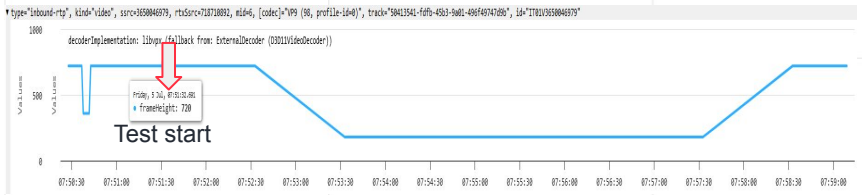


apiMind changing Jitter/Latency receiver webrtc

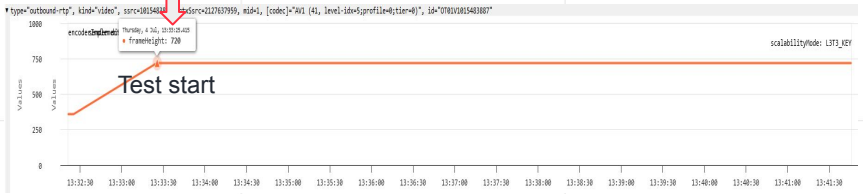


Meet changing Jitter/Latency sender webrtc

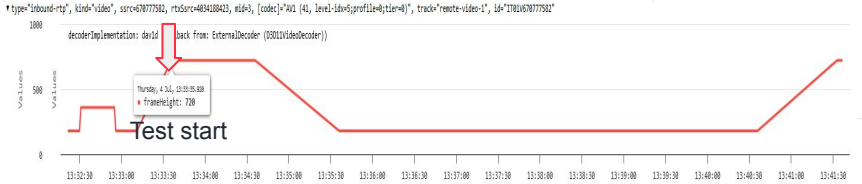
Meet changing Jitter/Latency receiver webrtc



Jitsi changing Jitter/Latency sender webrtc



Jitsi changing Jitter/Latency receiver webrtc



HEATMAPS

Link

https://docs.google.com/spreadsheets/d/1dLzZkDCYoV-hBQ-Uumyf5WnYI4DOGZCACWkBbIJNvzU/edit?gid=0#gid=0