Real-Time Audio Bitrate Control - Cheat Sheet

Fora Soft Learn

How the network's spare capacity is estimated, how audio is protected, and the two knobs to set.

Two clues the network leaves behind

Delay	Inter-arrival gap grows	Earliest warning - eases rate down before any loss
Loss	Packets discarded	Late but unambiguous - cut hard above ~10%, probe below ~2%

Who computes the estimate

REMB (older)	Receiver computes the estimate, sends one 'max bitrate = N' RTCP message back. Needs absolute-send-time.
transport-cc (default)	Sender numbers every packet; receiver reports raw arrival times; sender computes the estimate. Reacts faster.
RFC 8888 (spec-track)	Standardised congestion-control feedback (CCFB, FMT 11) - the IETF successor to the transport-cc draft.

Why video breaks before audio

The allocator serves audio first with a small protected floor, then hands the rest to video.
When the estimate falls, video is squeezed across its wide range; audio's floor stays untouched.
Only when the estimate nears audio's floor does audio narrow its band and lean on PLC.
Result: video turns blocky long before the voice drops - and that is the correct priority.

Opus numbers that matter

Opus runs 6-510 kbps and changes rate every 20 ms with NO glitch and NO renegotiation (RFC 6716).
Clear voice lives at ~16-40 kbps; video can usefully absorb hundreds-to-thousands of kbps.
Default to VBR for calls; use CBR only for fixed-size-frame transports or certain encryption (RFC 6716).
Worked split at a 250 kbps estimate: audio floor 24 kbps protected; video gets 250 - 24 = 226 kbps.

Set the two knobs (min / max)

Voice / telehealth: min 16 kbps, max 32-40 kbps - intelligible, leaves room for video and resilience.
Webinar / one-to-many: min 12-16 kbps, max 32 kbps - protect the floor, cap the ceiling.
Music / high-fidelity: min 24 kbps, max 64-128 kbps - a good link should sound excellent.
The MINIMUM is the floor the allocator guards; the MAXIMUM caps quality when the network is generous.

Remember

- [ ] Bandwidth is invisible - software estimates it from packet delay and loss, then sets bitrate beneath it.
- [ ] Google Congestion Control combines a delay estimator and a loss estimator and takes the lower of the two.
- [ ] transport-cc moved the math from receiver (REMB) to sender; it is the modern WebRTC default.
- [ ] Audio gets a small protected floor and is starved last - that is why video breaks first.
- [ ] Fix congestion BEFORE adding FEC: more redundancy into a full pipe makes the loss worse.