Streaming Pipeline Cheat Sheet

Five jobs, the protocols across each hop, and the latency budgets

The pipeline, stage by stage

1. Capture	Camera or screen — raw pixels + audio	**5–20 ms**
2. Encode	Codec compresses (H.264 / H.265 / AV1)	**50–500 ms**
3. Package	Cut into segments + manifest (CMAF / fMP4)	**200 ms – 4 s**
4. CDN	Origin to shield to edge, cached close to viewer	**5–80 ms**
5. Player	Buffer, decode, draw (hls.js / Shaka / native)	**0.5 – 30 s**

Protocol families on each hop

Contribution (ingest)
RTMP, SRT, RIST, WHIP, Zixi — stages 1–3

Distribution (delivery)
HLS, LL-HLS, DASH, CMAF, WebRTC, WHEP, HESP, MoQ — stages 3–5

Adaptive bitrate (ABR)
Player picks rungs from a ladder — keeps quality stable

Typical glass-to-glass latency by stack

Traditional HLS / DASH	**20–30 s**
Low-Latency HLS (LL-HLS)	**2–5 s**
Low-Latency DASH (CMAF chunked)	**2–5 s**
HESP	**0.4–1 s**
WebRTC delivery	**0.2–0.5 s**
Media over QUIC (MoQ)	**0.3–1 s**

Where each stage usually fails first

- Capture — unplugged cable, dropped HDMI, mic gain too low
- Encode — encoder CPU saturates, bitrate target too high for content
- Package — segment duration mismatch, manifest cache TTL too long
- CDN — cache misses, hot-segment thundering herd, edge fill spikes
- Player — buffer underrun on bad Wi-Fi, ABR oscillation, codec unsupported

Rule of thumb: the player buffer dominates total latency. Shrinking it is the biggest lever for low latency, and the fastest way to introduce stalls if you misjudge the network.

Companion to: 'What is video delivery, and why is it harder than serving a JPEG?'
www.forasoft.com/learn/video-streaming/articles/what-is-video-delivery