Streaming

Digitale AV Technik, MIB 5

Eine Einführung in Audio, Video und Live-Streaming

Lernziele

- **Verstehen:** Was ist Streaming und wie hat es sich entwickelt?
- **Erkennen:** Welche Komponenten und Protokolle sind beteiligt?
- **Analysieren:** Typische Herausforderungen (Latenz, Bandbreite, Synchronisation, QoS)
- **Bewerten:** Lösungsansätze (Adaptive Bitrate Streaming, CDNs, Codecs, Low-latency-Techniken)
- **Unterscheiden:** Unterschiede zwischen Videostreaming, Audiostreaming und Livestreaming

Kurz: Was ist Streaming?

- Streaming = kontinuierliche Übertragung von Medieninhalten über ein Netzwerk.
- Wiedergabe beginnt, bevor die gesamte Datei vollständig übertragen ist.
- Ziel: flüssige Wiedergabe bei begrenzten Netzwerkressourcen.

Geschichte & Entwicklung (Kurzüberblick)

- Frühe Konzepte: Progressive Downloads, RealAudio/RealVideo (1990er).
- Internetfernsehen & On-Demand-Angebote entwickelten sich ab den 2000er Jahren
  - YouTube wurde 2005 gegründet.
  - Netflix startete 2007 mit Streaming.
  - HTML5 wird von den meisten Browsern ab 2009 unterstützt; `<video>` und `<audio>` -Tags.
- Wichtige Technologien: Streaming-Server, Breitband-Verbreitung, adaptive Bitraten, CDNs.

Komponenten eines Streaming-Systems

- **Quelle / Encoder:** Wandelt Rohdaten in komprimierte Medienströme (z. B. H.264, H.265, AAC).
- **Container/Segmenter:** Verpackt in kleine Chunks (MP4/TS) oder andere Formate.
- **Origin-Server & Transcoder:** Liefert Original- und transkodierte Versionen (mehrere Bitraten).
- **CDN:** Verteilt Segmente nah zum Nutzer für Skalierung und geringere Latenz.
- **Transport & Protokolle:** HLS, MPEG-DASH, RTMP, RTSP, WebRTC, SRT.
- **Player / Client:** Pufferung, Decoding, Synchronisation.

Wichtige Protokolle & Formate

- **HTTP Live Streaming, HLS (Apple):** Segment-basierter HTTP-Streaming-Standard (TS/Fragmented MP4).
- **Dynamic Adaptive Streaming over HTTP, MPEG-DASH:** Offener Standard, .mpd (Media presentation description) + MP4-Fragmente.
- **Real-Time Messaging Protocol (RTMP):** Älter, niedrige Latenz, oft in *Ingest*-Pipelines (z. B. zu Streaming-Plattformen).
- **WebRTC:** Peer-to-peer, für sehr niedrige Latenz und Interaktivität.
- **SRT / RIST:** neu, Ziel: Robustheit trotz unsicherem Netzwerk

Herausforderungen beim Streaming

- **Bandbreitenbeschränkungen:** Schwankende Verbindungsgeschwindigkeiten beim Nutzer.
- **Netzwerk-Latenz:** Verzögerung zwischen Erzeugung und Wiedergabe.
- **Paketverlust / Jitter:** Führt zu Stottern oder Verbindungsabbrüchen.
- **Synchronisation:** A/V-Synchronität (Lip-Sync) zwischen Audio und Video.

Weitere Herausforderungen beim Streaming

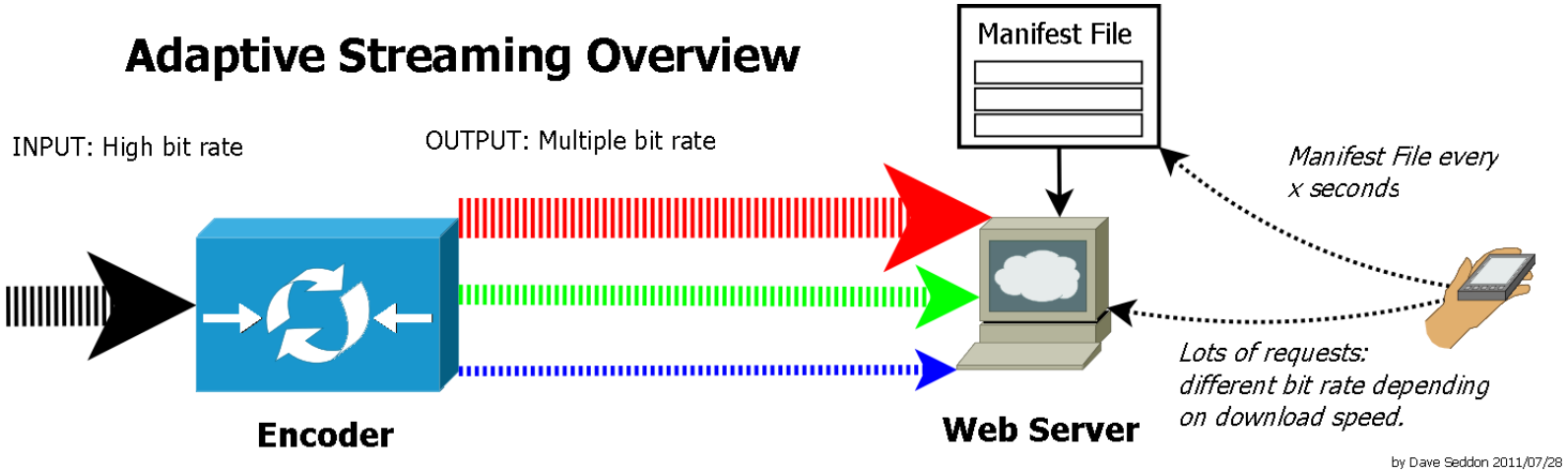
- **Skalierbarkeit:** Viele gleichzeitige Zuschauer erfordern CDNs/Transcoding-Pools.
- **Gerätevielfalt:** Verschiedene Bildschirmgrößen, Rechenleistung, Codecs.
- **Rechtliche/DRM-Anforderungen:** Kopierschutz, Geoblocking.

Typische Lösungsansätze (Übersicht)

- **Adaptive Bitrate Streaming:** Mehrere Qualitätsstufen; Player wählt dynamisch.
- **Puffer-Strategien:** Initiale Ladezeit anhand der Netzwerkbedingungen optimieren.
- **CDNs & Edge-Caching:** Inhalte näher an Nutzer bringen.
- **Transcoding & Packaging:** Mehrere Auflösungen/Bitraten, verschiedener Codecs/Container.
- **FEC & Retransmission / SRT:** Fehlerkorrektur und robuste Transportlayer bei Paketverlust.
- **Low-Latency-Techniken:** Chunked-Transfer, Low-latency HLS/DASH, WebRTC.

Adaptive Bitrate — Details

- Konzept: mehrere Versionen des Inhalts in unterschiedlichen Bitraten (i.d.R. die Auflösung).
- Manifest (HLS: Playlist, DASH: MPD) beschreibt verfügbare Repräsentationen.
- Player misst Durchsatz und wechselt Segment für Segment die Qualität.



Adaptive Bitrate — Vor- und Nachteile

- **Vorteil**: reduziert Pufferungen, optimiert QoE;
- **Nachteil**: erhöhte Komplexität und CDN-Last.

Latenz vs. Zuverlässigkeit — Tradeoffs

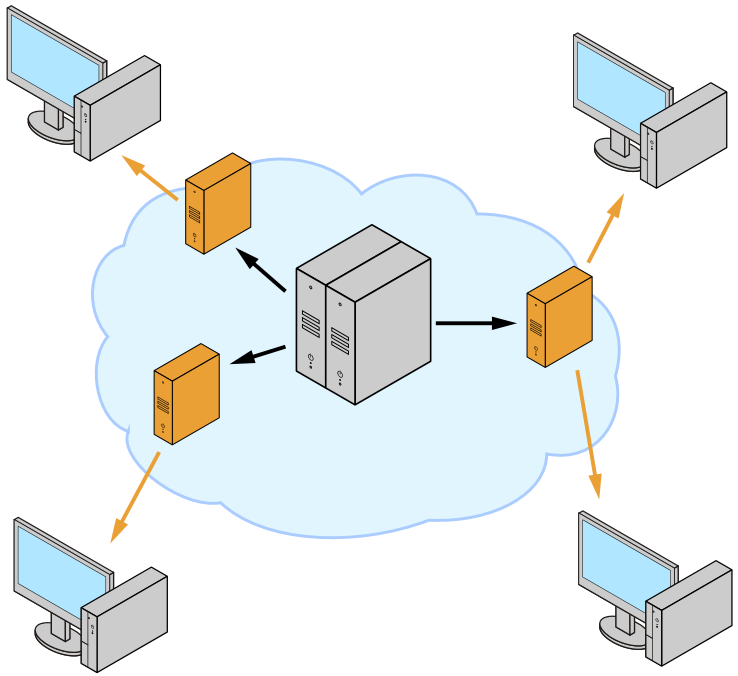
- Höhere Puffer → geringere Rebuffering-Wahrscheinlichkeit → höhere End-to-End-Latenz.
- Niedrige Latenz (Live) → weniger Puffer, höhere Anforderungen an Netzwerk & Protokolle.
- Wahl der Technik hängt vom Use-Case ab: Live-Interaktion vs. On-Demand-Qualität.

Content Delivery Networks (CDNs)

Verteilte Servernetzwerke, die Inhalte näher zum Endnutzer bringen.

- **Beispiele**:

Akamai, Cloudflare, AWS CloudFront, Google Cloud CDN.



Avelludo, CC BY-SA 4.0, via Wikimedia Commons

Unterschiede: Video- vs. Audiostreaming

- **Datengröße & Bitrate:** Video deutlich größer → stärkere Bandbreitenanforderungen.
- **Wahrnehmungsanforderungen:** hängt auch stark vom Inhalt und Abspielgerät ab.
- **Synchronisation:** Video+Audio benötigen strikte A/V-Sync; reines Audio braucht das nicht.
- **Codec-Auswahl:** Video: H.264/H.265/AV1; Audio: AAC/Opus/MP3 — Opus ideal für Low-latency/VoIP.

Unterschiede: On-Demand vs. Livestreaming

- **On-Demand (VOD):** Inhalte sind vorproduziert → Transcoding/Optimierung möglich.
- **Livestreaming:** Echtzeit-Erzeugung → geringere Zeit für Transcoding, Fokus auf Latenz.
- **Skalierung:** Beide brauchen CDNs; Live benötigt häufig Ingest-Server und Echtzeit-Transcoding.
- **Interaktivität:** Live: Chat, Reaktionen, Rückkanal, bei On-Demand nur Selektion.

Wir entwickeln einen typischen Live-Streaming-Workflow an der Tafel.

Praxisbeispiele / Einsatzfälle

- OTT-VOD: Netflix, YouTube (VOD + Live)
- Live-Sport: sehr niedrige Latenz gewünscht, viele Zuschauer ⇒ große CDN-Infrastruktur
- Interaktive Video (Gaming, Videokonferenzen): WebRTC / specialized low-latency stacks
- Radio / Podcasts (Streaming-Audio): konstante Bitraten, geringer CPU-Aufwand beim Decoding für tragbare Geräte

Qualitätsmetriken und Messgrößen

- **Startup Time:** Zeit bis zur Wiedergabe beginnt.
- **Rebuffering Rate / Duration:** Häufigkeit und Dauer von Unterbrechungen.
- **Average Bitrate / Resolution:** tatsächlich gelieferte Qualität.
- **End-to-End Latenz:** wichtig für Live-Interaktion.
- **MOS / QoE Indices:** subjektive Nutzerbewertung.

Tipps für Implementierende

- Verwende adaptive Bitraten und wähle für den Empfänger passende Optionen.
- Nutze CDNs oder Edge Caching und geografisches Routing.
- Wähle Codec/Container mit Blick auf Zielgeräte und deren Rechenleistung.
- Für Live: teste Latenzpfade, optimiere Ingest und setze ein entsprechendes Low-Latency-Protokoll ein.

Kurze Zusammenfassung

- Streaming ist ein Set aus Technologien zur kontinuierlichen Medienübertragung.
- Hauptprobleme: Bandbreite, Latenz, Fehlerresilienz und Skalierbarkeit.
- Lösungen: Adaptive Bitraten, CDNs, passende Protokolle (HLS/DASH/WebRTC) und Codecs.
- Livestreaming unterscheidet sich durch geringe Latenzanforderungen und Echtzeit-Charakter.

Weiterführende Links / Quellen

- Wikipedia: Streaming Media
- Wikipedia: Geschichte und Entwicklung des Streaming Media
- Wikipedia: Livestreaming
- Wikipedia: Internetfernsehen
- oder die jeweiligen Artikel auf Englisch (in den Folien verlinkt)