Streaming Video

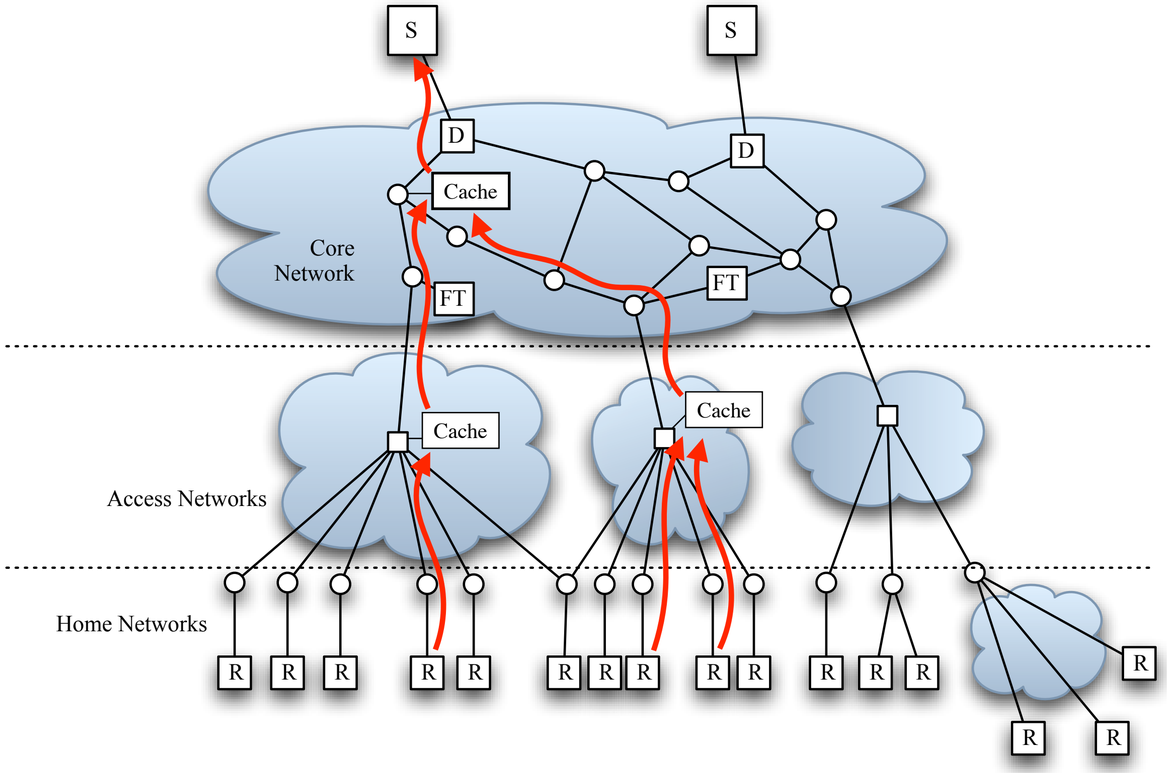
- HTTP Adaptive Streaming

Streaming Video Applications

- RTP **should** be usable for streaming video
  - Real-time Streaming Protocol (RTSP) developed by Real Networks for Internet TV in late 1990s
  - Uses unidirectional RTP for video delivery
  - Very low latency, loss tolerant

- **Most** video-on-demand, TV, and movies are delivered instead over HTTPS
  - Performance is significantly worse, very high latency, but startup latency is less critical
  - Integrates better with content distribution networks

HTTP Content Distribution Networks

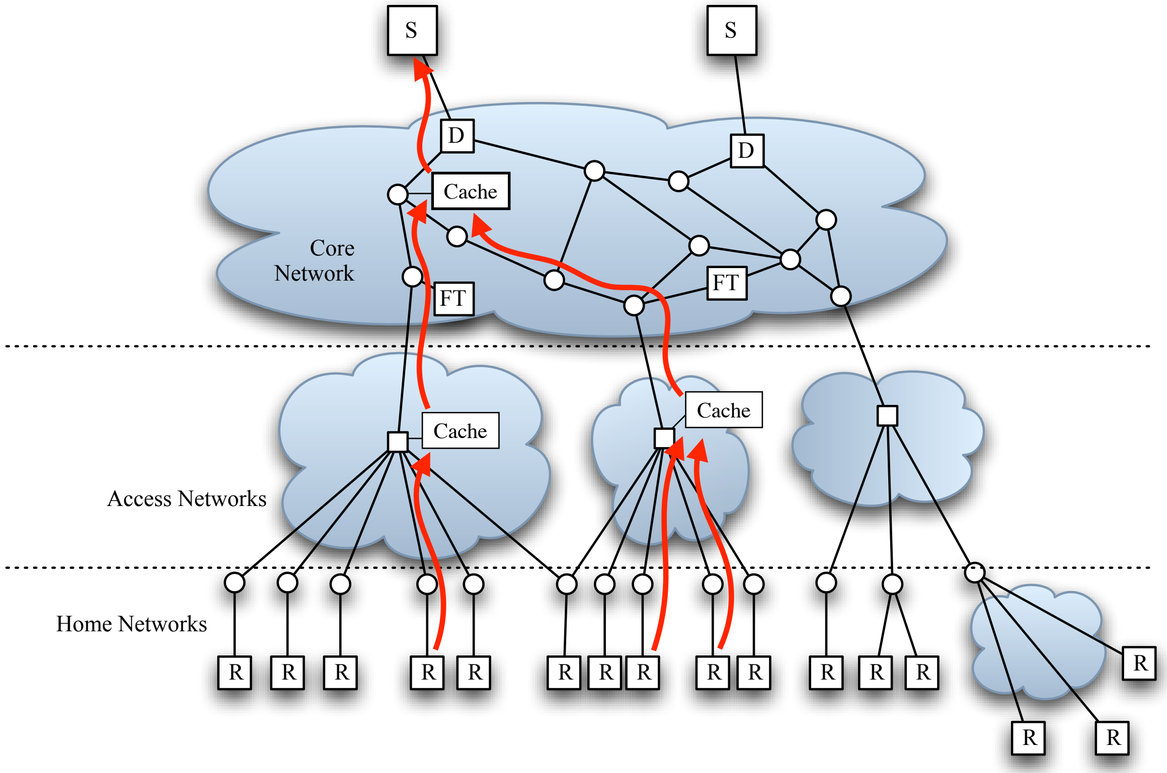


- There is an extensive, well-developed, content distribution infrastructure

Akamai CLOUDFLARE fastly

  - Highly effective for delivering and caching HTTP content
  - Global deployment – agreements with most large ISPs to host caching proxy servers
  - **Only works with HTTP-based content** – does not support RTP-based streaming

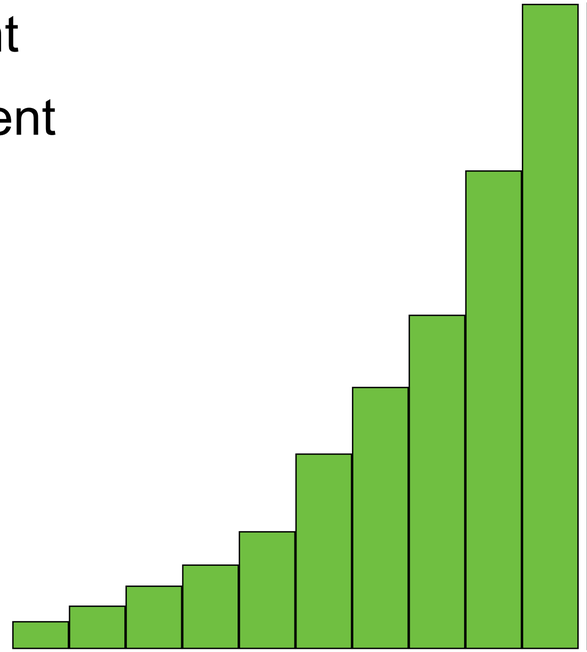
HTTP Adaptive Streaming (a.k.a. MPEG DASH)



- **Deliver video over HTTPS** to make use of content distribution networks
  - Video content encoded in multiple **chunks**
    - Each chunk encodes ~10 seconds of video
    - Each chunk independently decodable
    - Each chunk made available in many different versions at different encoding rates
  - A **manifest** file provides an index for what chunks are available
  - Clients fetch manifest, download chunks in turn
    - Standard HTTPS downloads
    - But monitor download rate, and choose what encoding rate to fetch next – adaptive bitrate
    - Playout chunks in turn, as they download

DASH: Chunked Media Delivery and Rate Adaptation

- Each chunk encodes ~10 seconds of video content
- Each chunk is compressed multiple times at different encoding rates (i.e., different sizes)
  - e.g., Netflix suggests encoding at ten different rates, as show on the right
- Receiver fetches manifest, containing the list of all versions of all each chunk
- Receiver fetches each chunk in turn
  - Measures download rate and compares to the encoding rate
  - If download rate slower then encoding rate, then switch to lower encoding rate for next chunk
  - If download rate faster than encoding rate, consider fetching higher rate for next chunk
- Chooses encoding rate to fetch based on TCP download rate

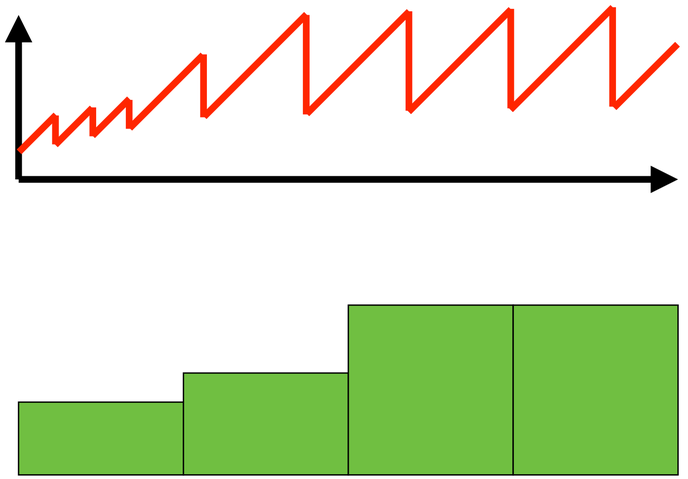


Bitrate (kbps)	Resolution
235	320x240
375	384x288
560	512x384
750	512x384
1050	640x480
1750	720x480
2350	1280x720
3000	1280x720
4300	1920x1080
5800	1920x1080

Source: Netflix

DASH and TCP Congestion Control

- DASH rate adaptation and TCP congestion control work at different time scales
  - TCP adjusts congestion window once per RTT
    - TCP Reno or Cubic
    - AIMD algorithm
  - DASH receiver measures **average download speed** of TCP connection over ~10 seconds
    - Selects size of next chunk to download based on average TCP download speed

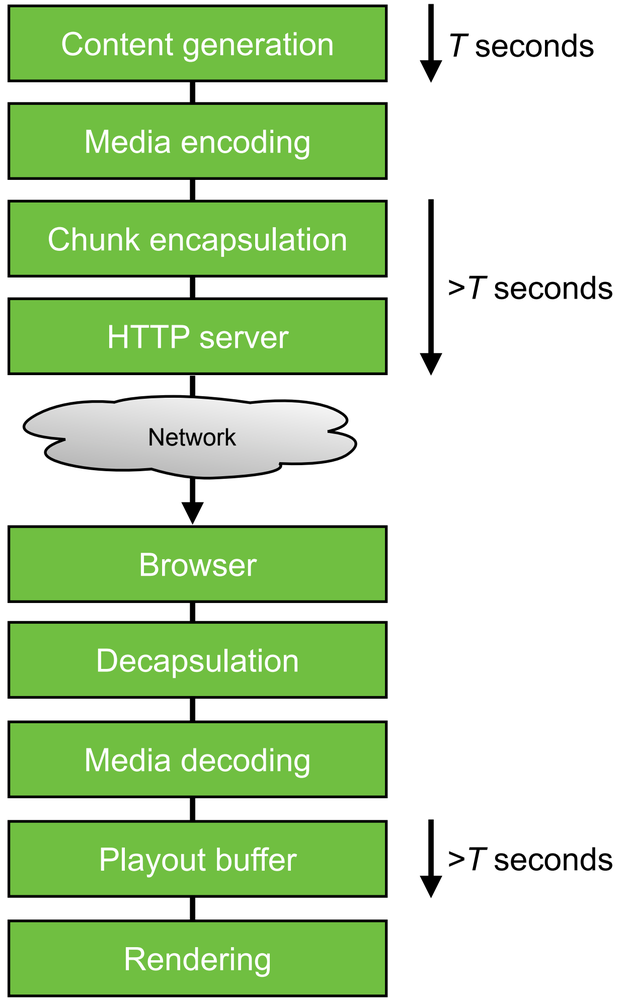


- Videos played using DASH often have relatively poor quality for first few seconds
  - Receiver picked a conservative download rate – relatively poor quality, small size – to start
  - Uses that to measure connection download speed, then switches to more appropriate rate

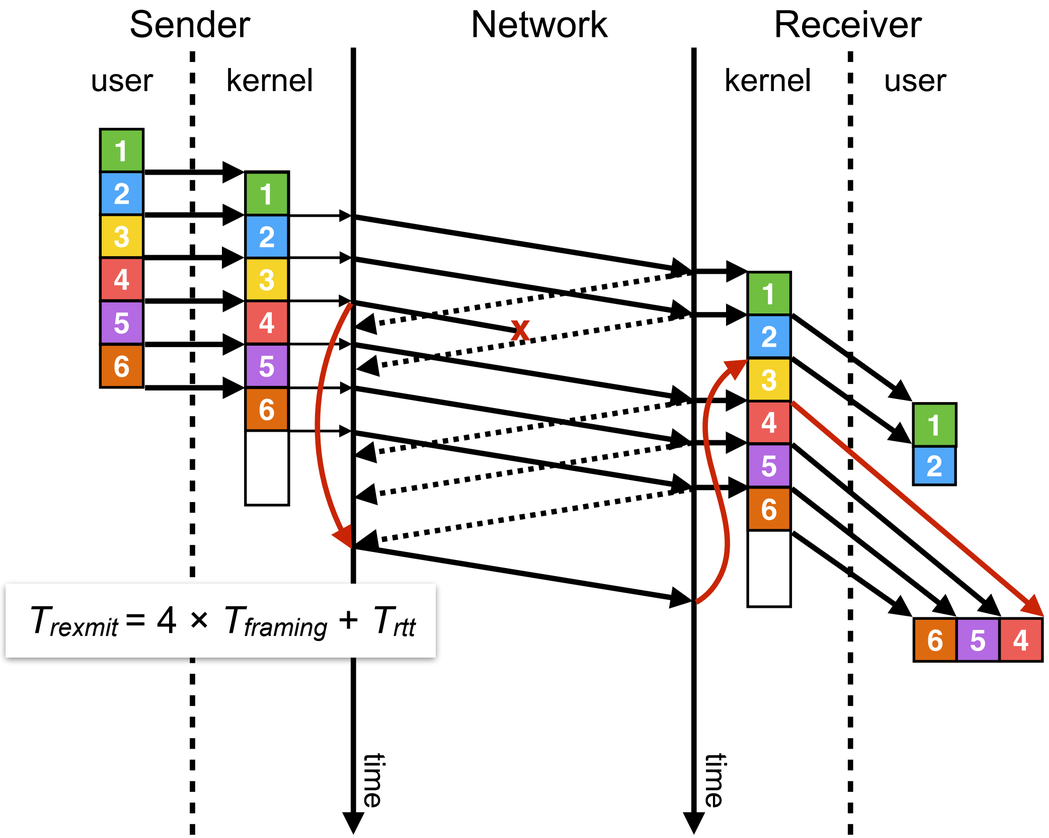
DASH Latency

- Multi-second playout latency common, due to:
  - Chunk duration
  - Playout buffering
  - Encoding delays for live streaming

- **Chunk duration is a key** – must download complete chunk before starting playout
  - Download and decompress chunk *n*
  - Start playing chunk *n* and while playing download chunk *n+1*
  - No packets lost → latency equals chunk duration



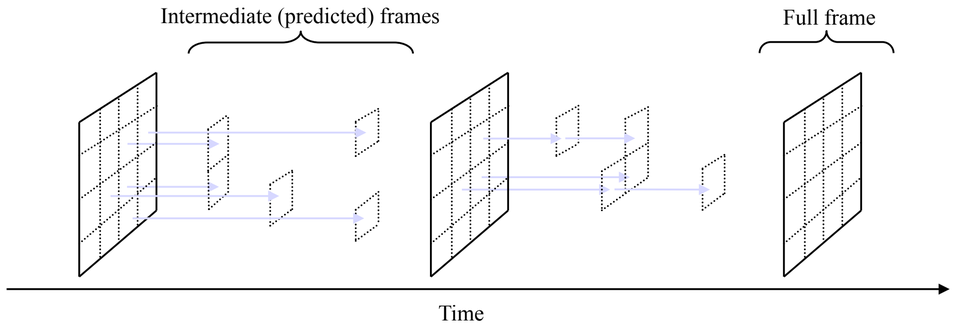
Sources of Latency: TCP



- If a packet is lost, TCP retransmits after triple duplicate ACK
  - This takes time, and delays download of the chunk:
    - 4× packet transmission latency (the lost packet, plus three following that generate duplicate ACKs)
    - 1× network RTT to retransmit packet
  - Each packet loss and retransmission will increase chunk download time – and causes TCP to reduce its window
- Both add latency if using TCP

Sources of Latency: Chunks and Video Compression

- Each chunk of video is independently decodable
  - Can’t compress based on previous chunk, because previous chunk depends on the network capacity → unpredictable
- Each chunk therefore starts with an I-frame, followed by P-frames
  - I-frames are large: often 20× size of P-frames
  - If chunks are smaller duration → contain fewer P-frames compared to number of I-frames; video compression ratio goes down
- Trade-off between chunk size and compression overhead
  - Large chunks require less data, but add latency
  - Small chunks reduce latency, but need more data
  - Overheads become excessive for chunk duration ≲2 seconds



Future Directions for Streaming Video

- Streaming video over WebRTC?
  - Web browsers now all support WebRTC
  - Intended for interactive conferencing, but entirely usable for RTP-based video streaming to browser
  - Much **lower latency** than DASH
  - Will this encourage CDNs to support RTP?

- Streaming video over QUIC?
  - HTTP/3 is a drop-in replacement for existing HTTP stack, and delivers requests over QUIC
  - YouTube already delivers video over QUIC this way – expect to see much more deployment and special purpose QUIC extensions for video

Real-time and Interactive Applications

- Real-time traffic and its constraints
- Interactive applications → WebRTC
- Streaming applications → DASH
- Custom infrastructure for low latency; CDN and HTTP if latency less critical