Adaptive Bitrate Technology

Peter Chave
Cisco
chavep@cisco.com

Agenda

- How did we get to ABR?
- Today’s delivery model
- Differences in ABR technologies
- A strategy to handle divergence
- The CDN challenge

HTTP Progressive Download

Why HTTP?

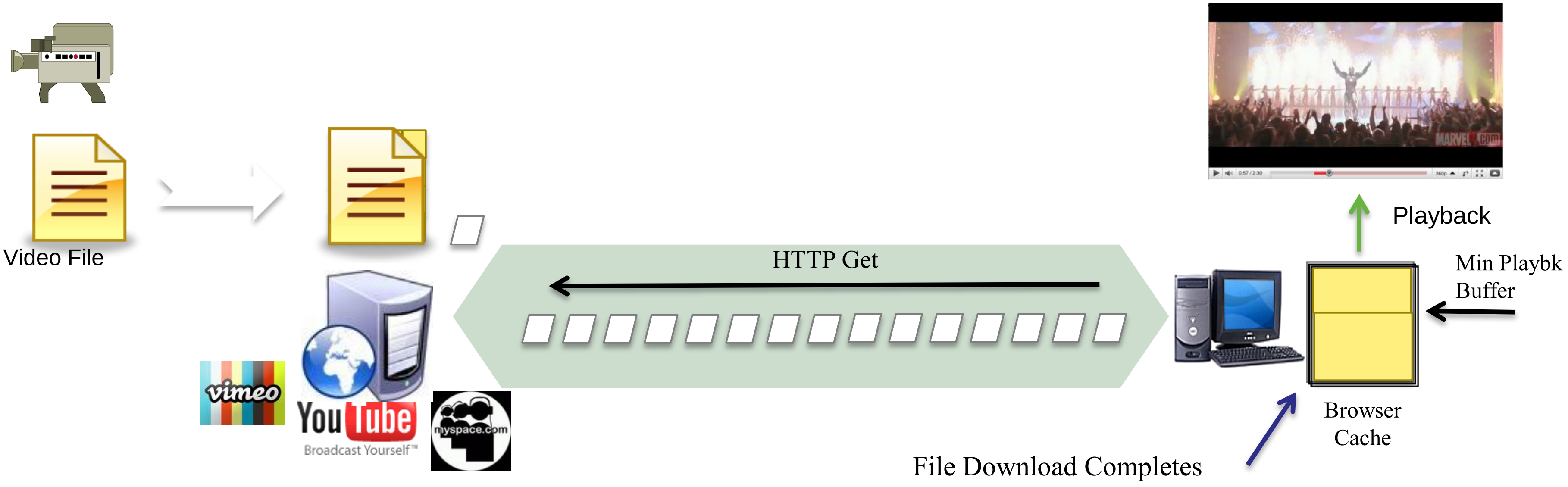
Source: VRT medialab

- Web download services have traditionally been less expensive than media streaming services offered by CDNs and hosting providers.
- HTTP-based media delivery has no issues traversing routers/NAT/firewalls because firewalls and routers know to pass HTTP downloads through port 80
- HTTP media delivery doesn't require special proxies or caches. A media file is just like any other file to a Web cache
- It's much easier and cheaper to move HTTP data to the edge of the network, closer to users through HTTP caches
- Key point: adapt video to Web rather than change the Web to allow video

GLOBECOMM®

HTTP Progressive Download

- Prevalent form of Web-based media delivery for Video Share Sites.
- ‘Ordinary’ File Download from HTTP Web Server
- ‘Progressive’ = Playback begins while download is in progress Byte Range Request Supported HTTP 1.1+



HTTP Progressive Download

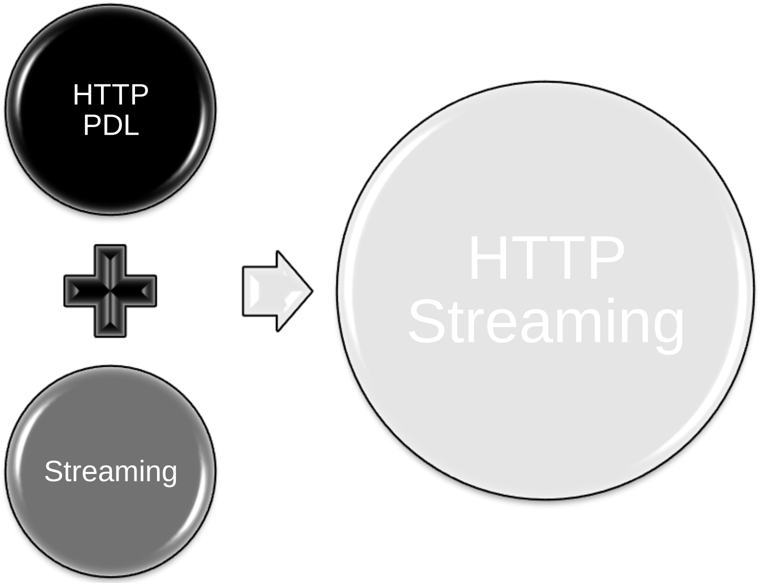
Progressive Download – Behavior

- Downside – Real-time viewing often suffers from poor quality unless network/bandwidth conditions are sufficient.
- Upside - media file is resident in browser cache. Subsequent playout is smooth.

…. Buffering….

HTTP Adaptive Bit Rate (ABR) Streaming

- A hybrid content delivery method which acts like traditional streaming but is in fact based on HTTP progressive download

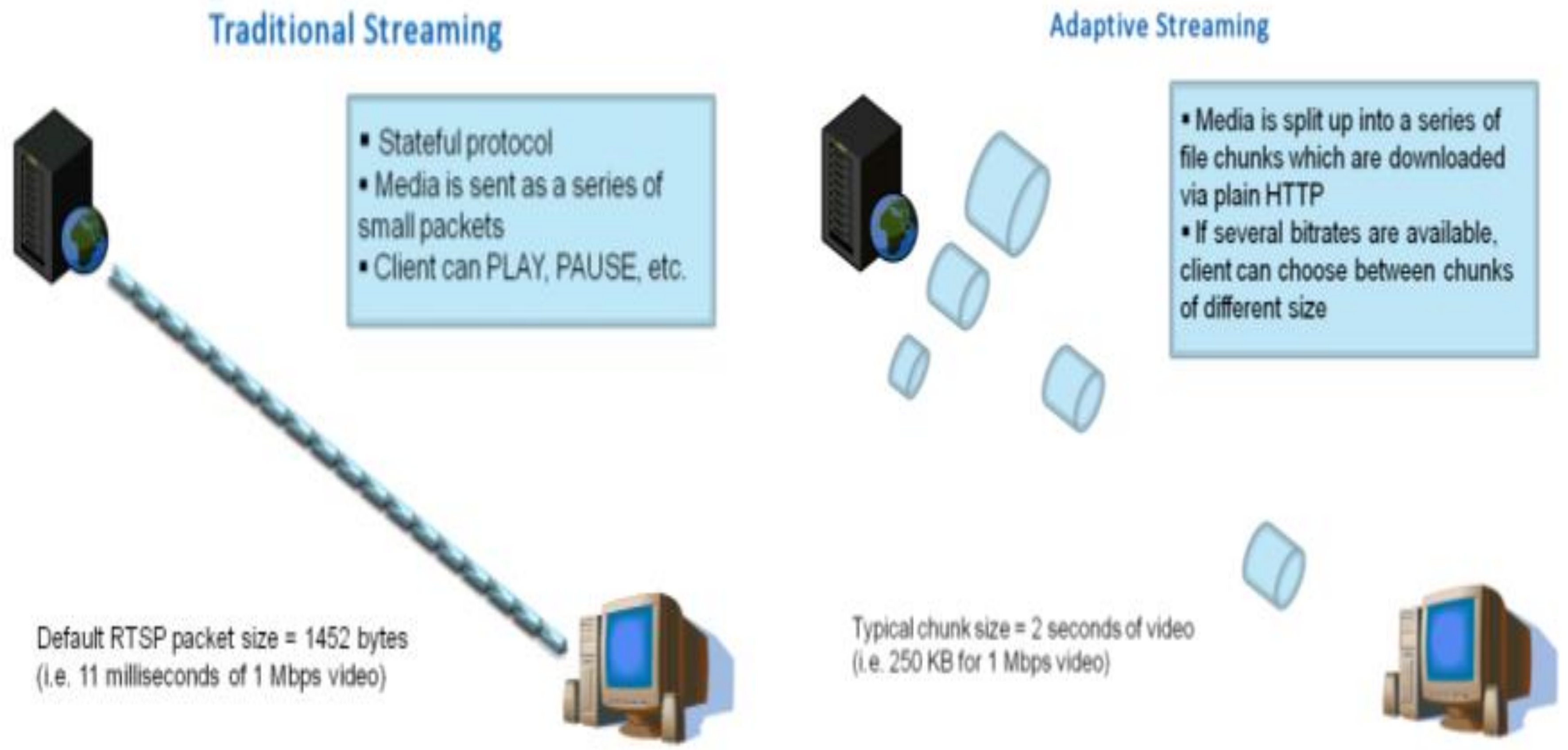


HTTP PDL

Streaming

HTTP Streaming

Traditional Streaming Vs HTTP ABR Streaming



Traditional Streaming

- Stateful protocol
- Media is sent as a series of small packets
- Client can PLAY, PAUSE, etc.

Default RTSP packet size = 1452 bytes
(i.e. 11 milliseconds of 1 Mbps video)

Adaptive Streaming

- Media is split up into a series of file chunks which are downloaded via plain HTTP
- If several bitrates are available, client can choose between chunks of different size

Typical chunk size = 2 seconds of video
(i.e. 250 KB for 1 Mbps video)

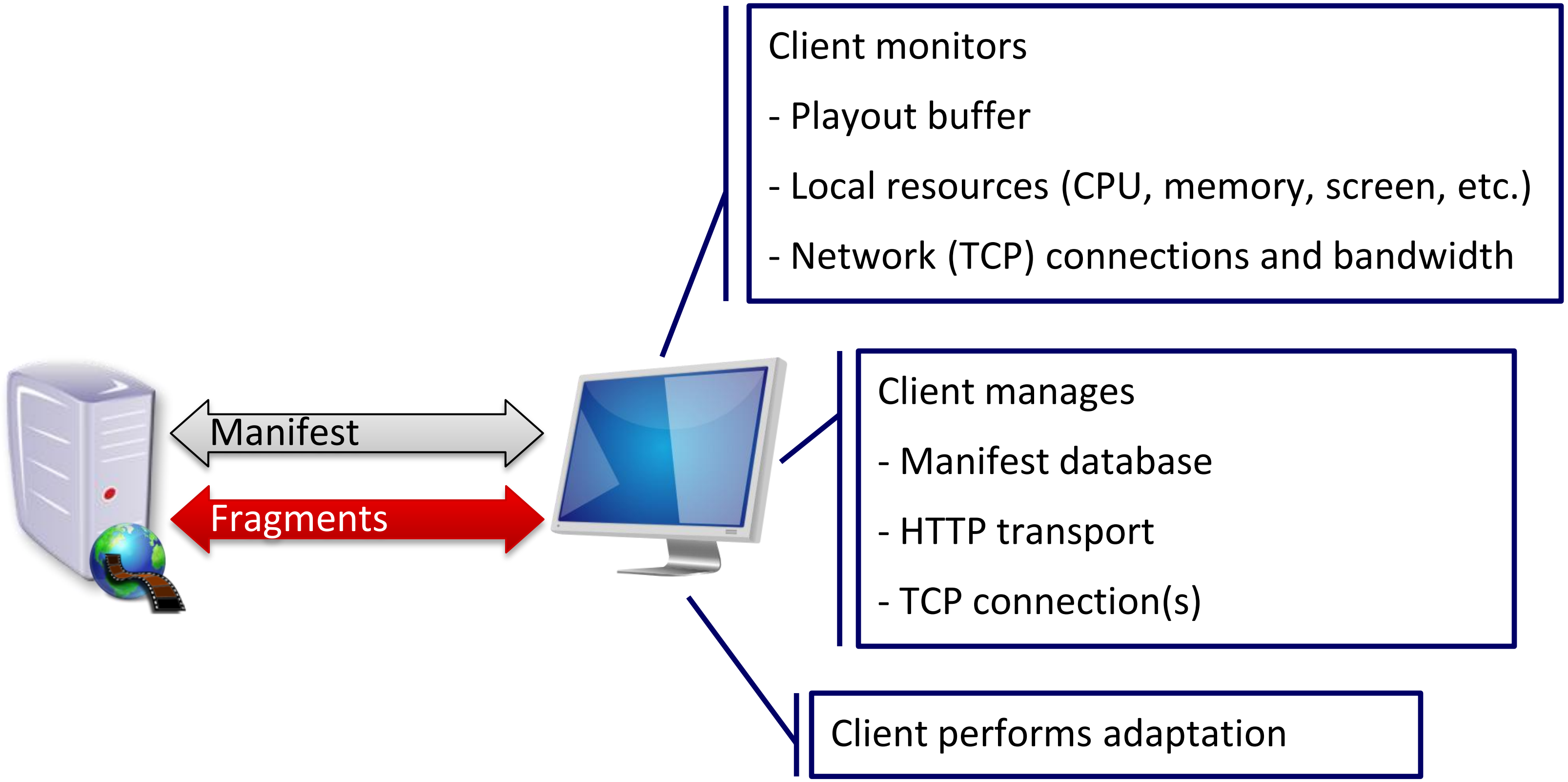
HTTP ABR Streaming

Advantages

- Fast start-up and seek times because start-up/seeking can be initiated on the lowest bit rate before moving to a higher bit rate
- No buffering, no disconnects, no playback stutter (as long as the user meets the minimum bit rate requirement)
- Allows client to adapt to the content, rather than requiring content providers to guess which bit rates are most likely to be accessible to their audience
- Seamless bit rate switching based on network conditions and CPU capabilities. A generally consistent, smooth playback experience
- Facilitates ‘any device, anywhere, anytime’ paradigm. Major step towards mobility
- Changing legacy SP service model. New business, services, revenue opportunities.

HTTP ABR Streaming

Client has a prominent role



Manifest

Fragments

Client monitors

- Playout buffer
- Local resources (CPU, memory, screen, etc.)
- Network (TCP) connections and bandwidth

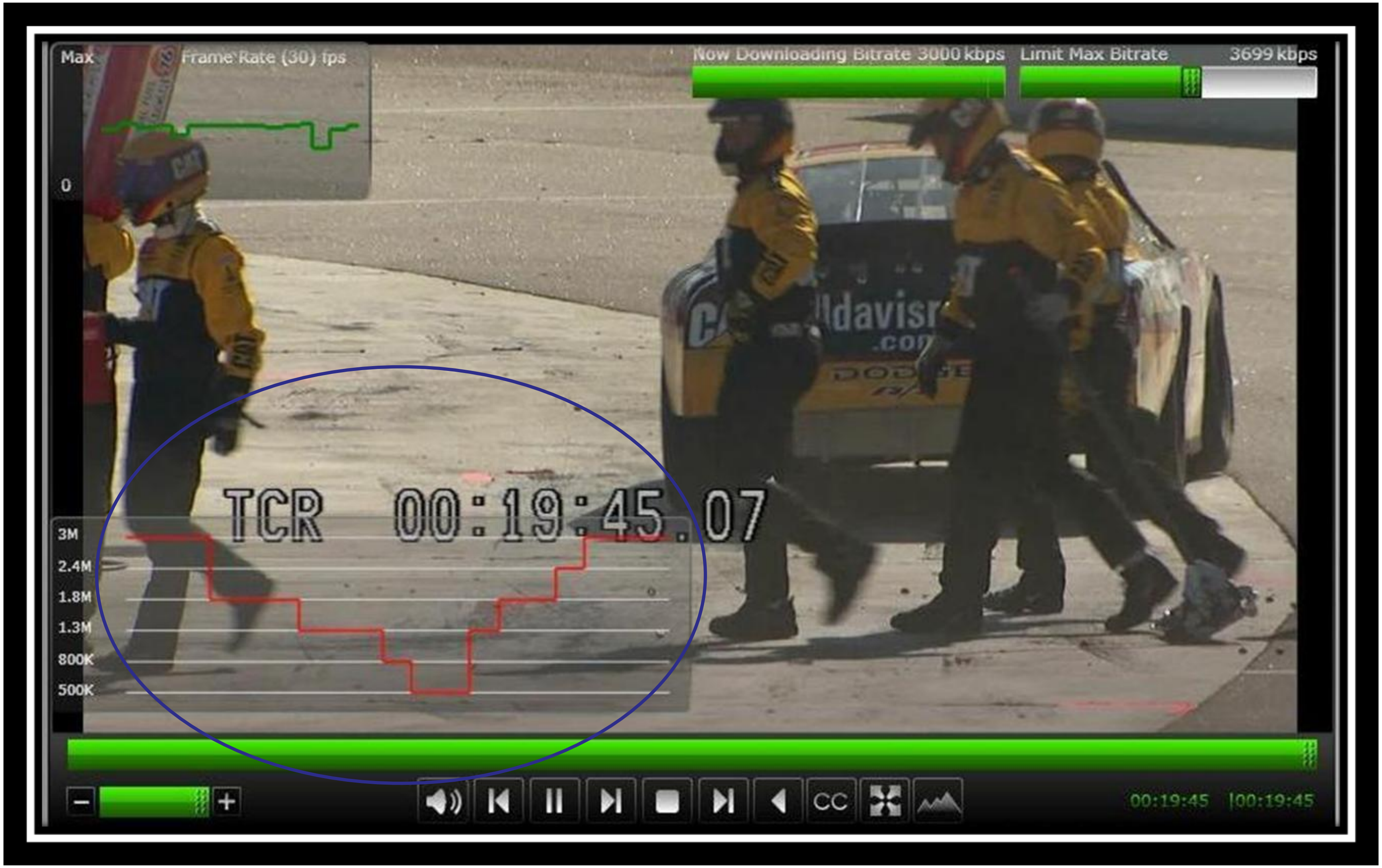
Client manages

- Manifest database
- HTTP transport
- TCP connection(s)

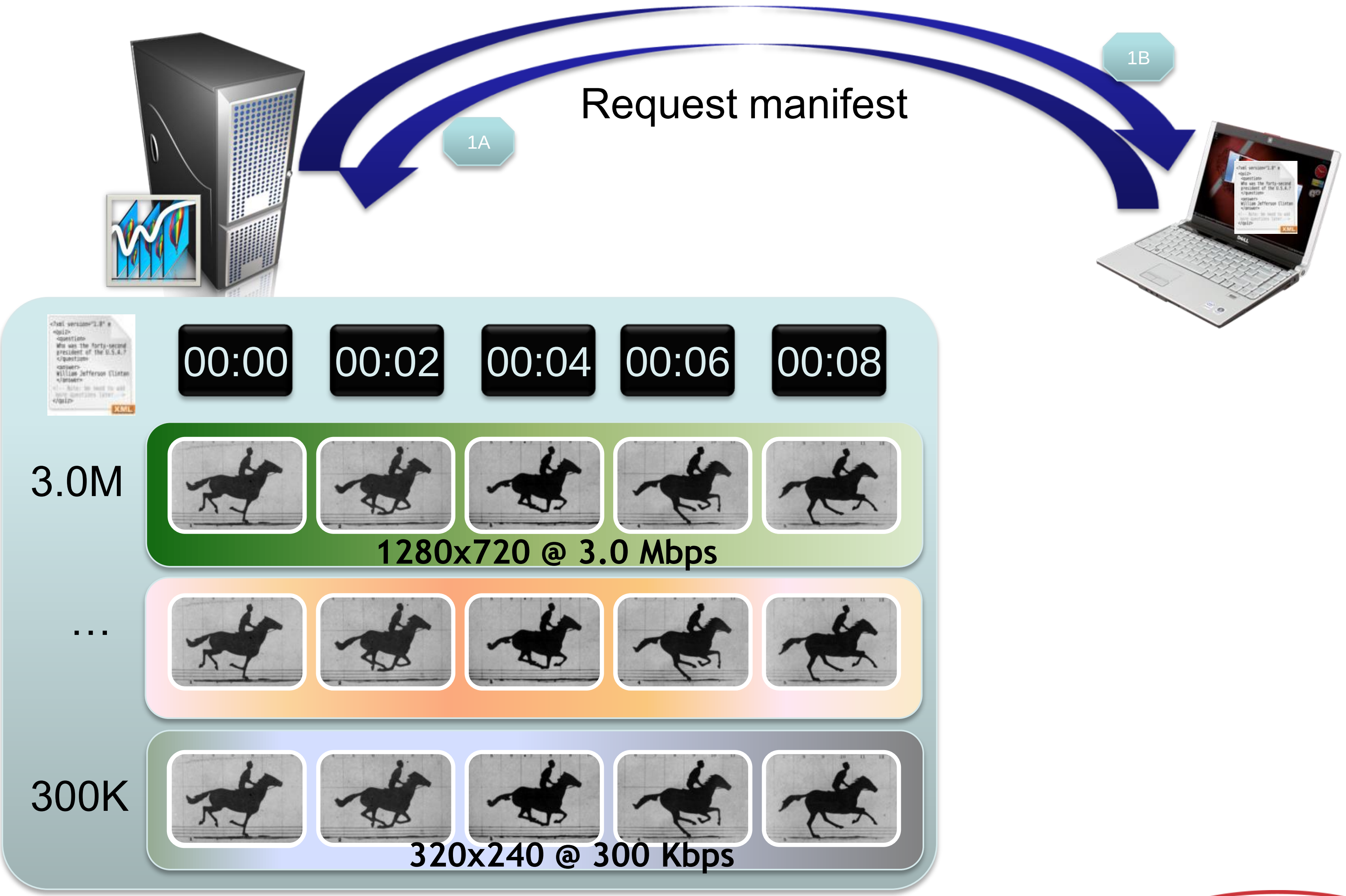
Client performs adaptation

HTTP ABR Streaming

A Client Application



Request Manifest



1B

Request manifest

1A

	00:00	00:02	00:04	00:06	00:08

3.0M

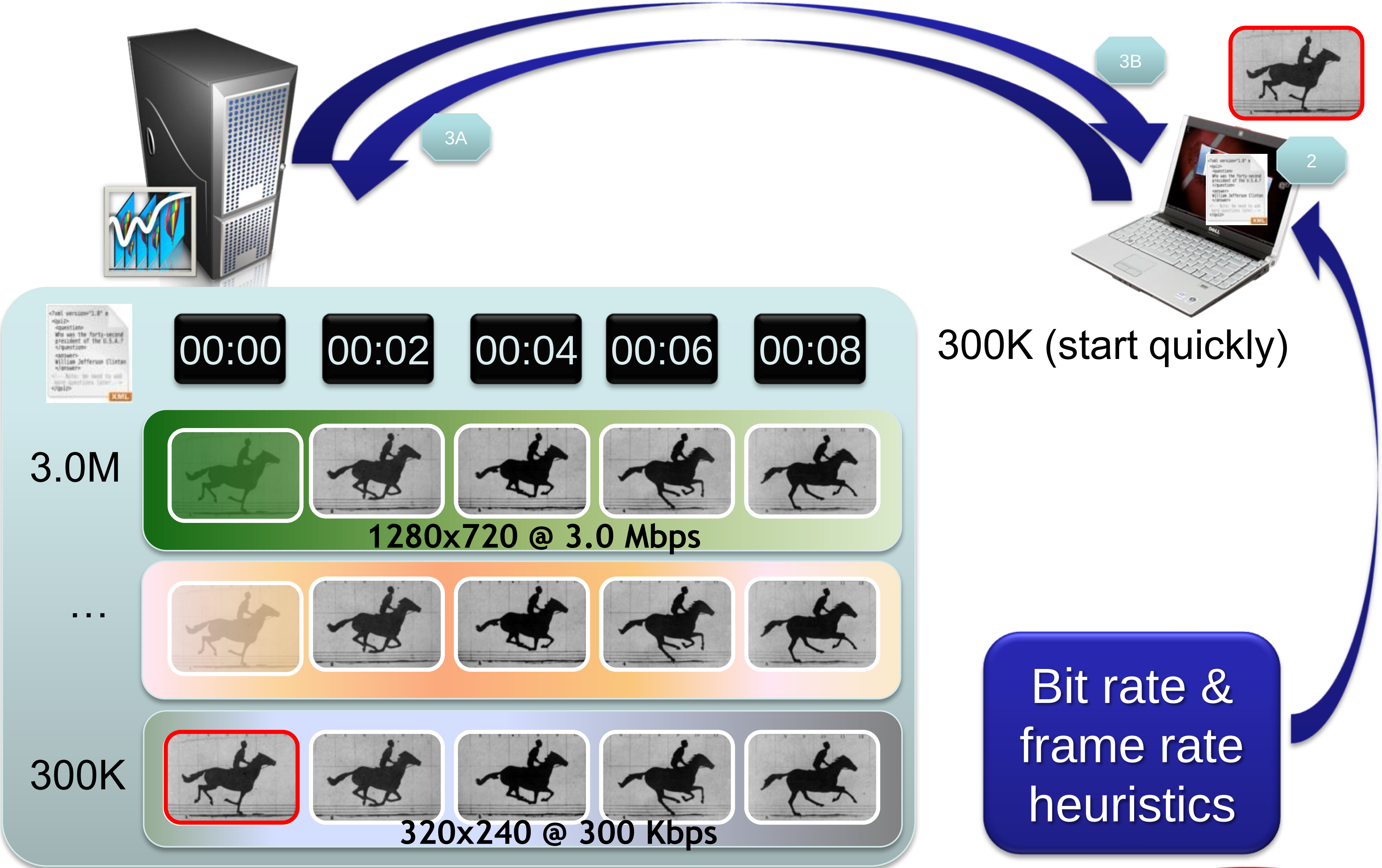
1280x720 @ 3.0 Mbps

…

300K

320x240 @ 300 Kbps

Quickstart Fragment Requests



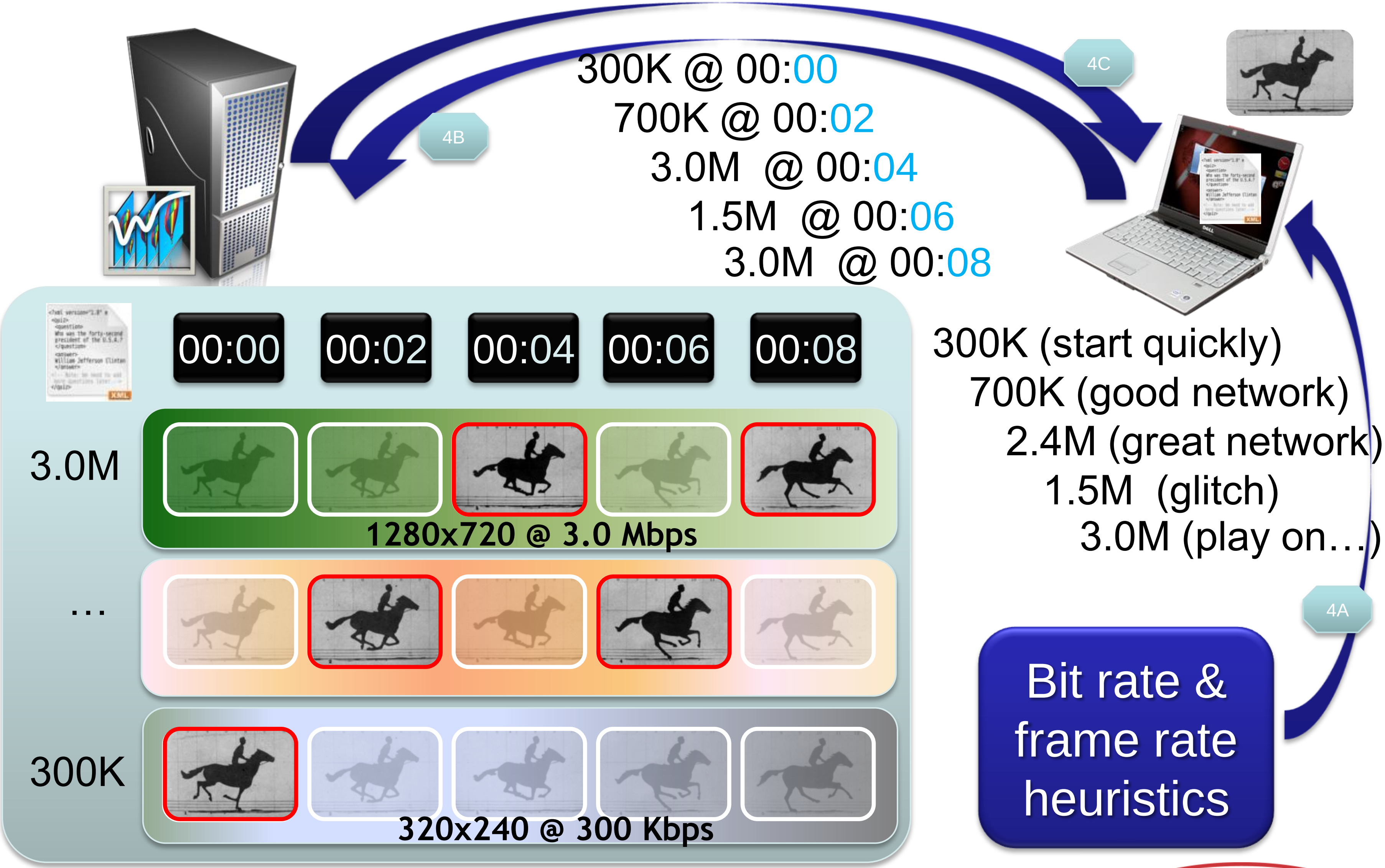
1280x720 @ 3.0 Mbps

320x240 @ 300 Kbps

300K (start quickly)

Bit rate & frame rate heuristics

Adapting Bit Rate in Real-Time



300K @ 00:00
700K @ 00:02
3.0M @ 00:04
1.5M @ 00:06
3.0M @ 00:08

00:00 00:02 00:04 00:06 00:08

3.0M

1280x720 @ 3.0 Mbps

…

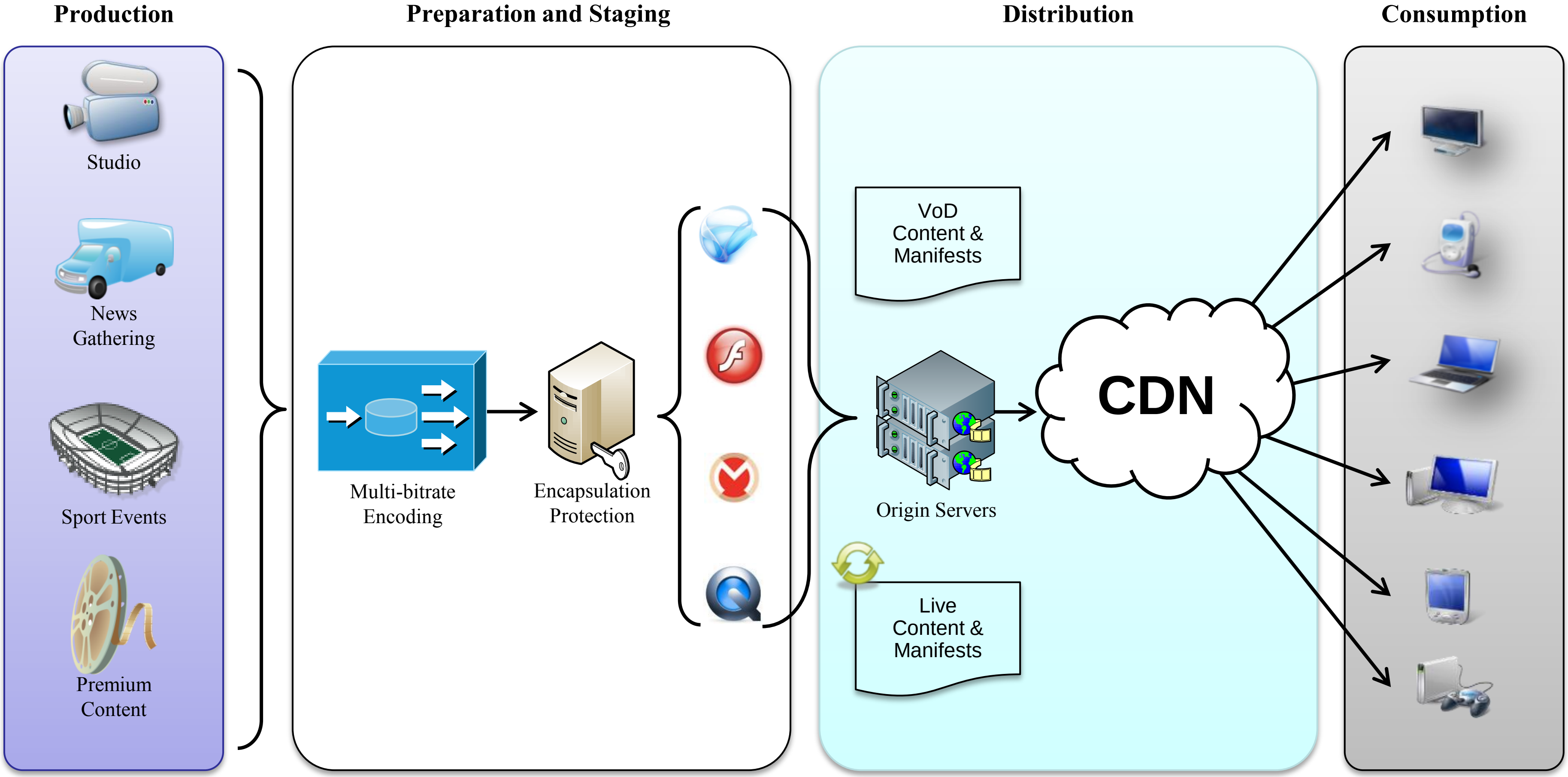
300K

320x240 @ 300 Kbps

300K (start quickly)
700K (good network)
2.4M (great network)
1.5M (glitch)
3.0M (play on…)

Bit rate & frame rate heuristics

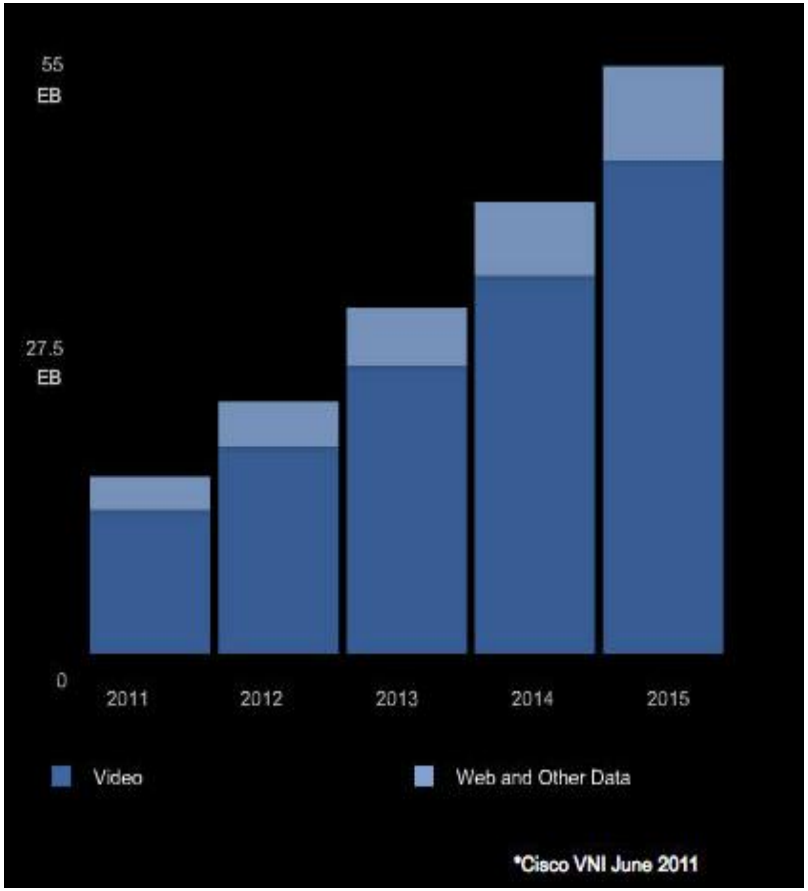
Today's Over-the-Top Adaptive Streaming Delivery



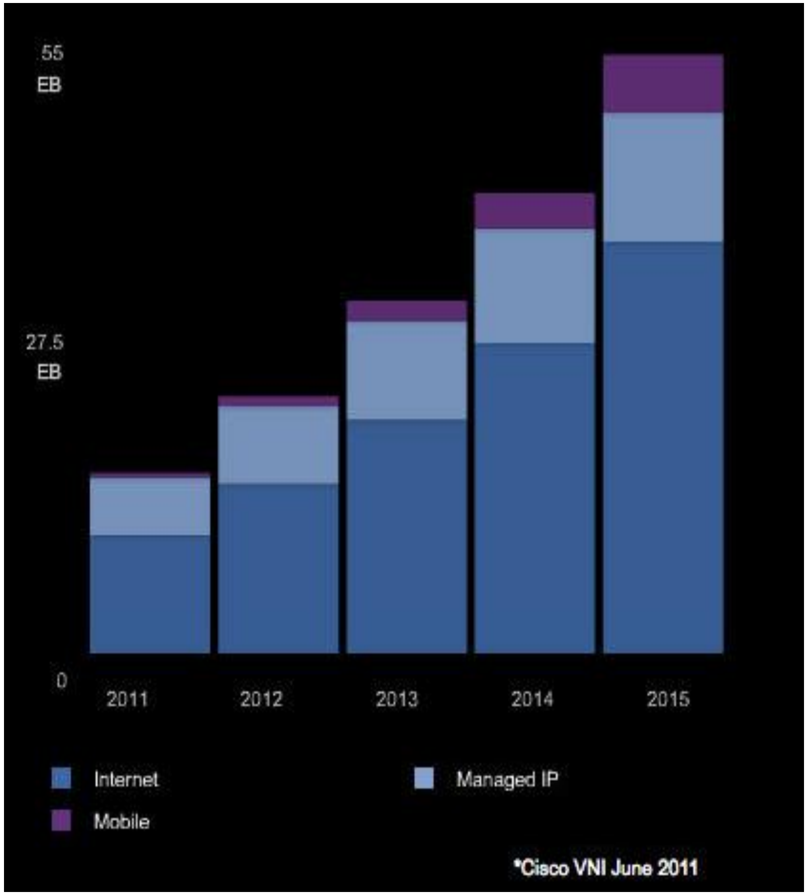
- •Service Providers have little control and visibility into OTT services
- •Content Providers have little control of the delivery of their content

Video consumption is exploding around the world

- In 2015, Video traffic will be 3 times larger than it is today
- And mobile traffic will be 12 times what it is today



*Cisco VNI June 2011



*Cisco VNI June 2011

HTTP ABR – Format Comparison

No clear common ground apart from H.264/AAC

	HSS (Microsoft)	HLS (Apple)	HDS (Adobe)
Transport Protocol	HTTP	HTTP	HTTP
Fragment Size (typical)	2 seconds	10 seconds	Variable
#TCP connections	1 or 2	1	Variable
# Content Files on Origin Server	#profiles	#profiles x 720/Hr	#profiles (VOD) #profiles x frag duration/Hr (Live)
Codec Support	VC-1, H.264,WMA	H.264	H.264
Wire/Xport Format	MP4 fragments	MP2TS fragments	MP4 fragments
Content File Format on Origin Server	.ismv Fragmented mp4	.ts Segmented TS	.f4f, .fmf Fragmented mp4
Byte Range Mechanism	No	No	Yes
Std HTTP Origin Server	No	Yes	No
Encryption/DRM	Windows DRM PlayReady	AES-128	Adobe Access
Client	Silverlight 2+ OSMF (OpenSource)	iPhone OS 3.0+ Quicktime X	Flash Player 10.1 with ZERI extensions
Manifest file	.ismc (.ism/Mfest or .isml/Mfest)	.m3u8	.fmf
Origin server	Helper integrated with IIS server	HTTP server	HTTP server with Helper module

Multi-Language Audio, Metadata Processing

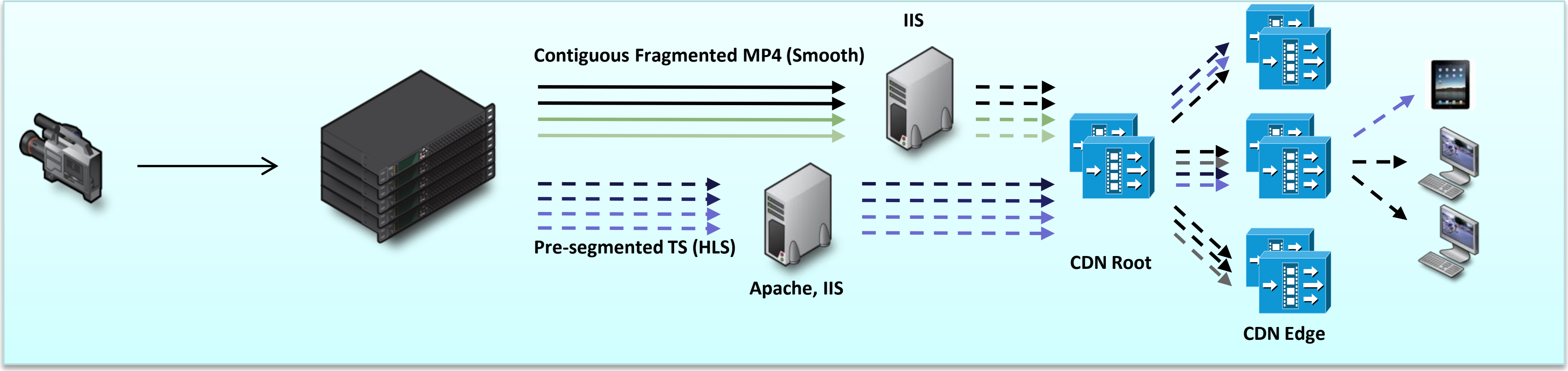
Still no convergence (actually worse)

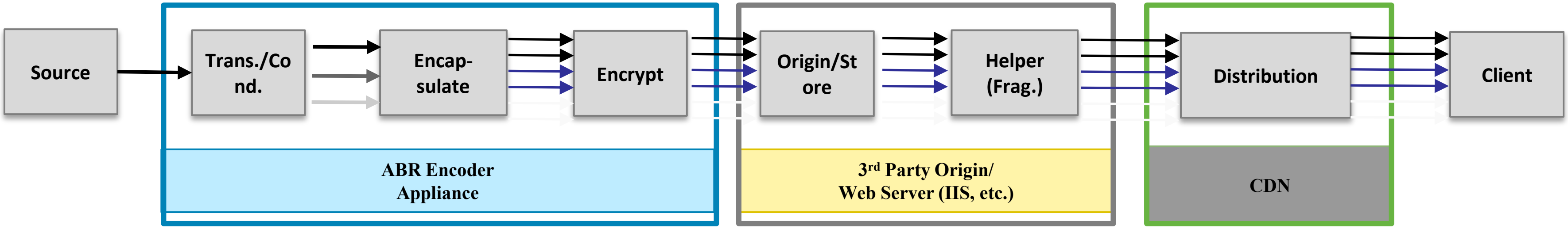
	HSS	HLS	HDS
Multi-Language Audio	• Single audio track per language • Track has language descriptor • URL fragment request contains descriptor	• HLS supports multiple audio tracks, but each segment contains all audio tracks (pre-iOS5) • iOS5 now allows for separable audio streams, TBD when non iOS devices will support (Roku, etc.) • Change result of Cisco working with Apple on requirements – Apple has tended to be very NA focused	• RTMP has no support for multiple audio tracks/IDs • HDS supports multiple audio tracks, but each segment contains video and all audio tracks • Cisco applying pressure on Adobe on both of these issues
Metadata Processing	• Data Tracks (Name, Language, Sub-type) • Sparse (has Parent Track) • Non-Sparse (always present)	• Timed metadata introduced earlier this year • Private TS stream • ES=ID3 tag payload	• Cue points • (Name, Multiple Parameters) • Each parameter is (tag,value) pair
Captions/Subtitles	• Source converted to TTML – natively supported by client • Different approach highly desired to support bitmap-based subtitles (DVB)	• 608 user data on AVC ES for Closed Captioning • No subtitle support • Apple unlikely to add support soon	• No formal support • Client specific customer implementations (BBC)
Ad Splicing**	• SCTE-35 like metadata in sparse track • Client based reaction to metadata • Dual timelines to track parent and child (ad) streams	• Cloud based manifest manipulation • Client unaware of ad splice, additional metadata can be used to control trickmodes, etc. • Scale, cacheability implications of supporting highly targeted – manifest file management	• Client based reaction to some form of metadata • Little effort to standardize this data

** Divergent views across providers on cloud-based only vs client-based only –based splicing, as well as combination of the two – implications on different ecosystems

So how do we address the divergence?

Look at a generic ABR Content Flow




Source

Trans./Cond.

Encap-sulate

Encrypt

Origin/Store

Helper (Frag.)

Distribution

Client

ABR Encoder Appliance

3rd Party Origin/ Web Server (IIS, etc.)

CDN

Single Stream
Highest Bitrate/Quality

Multiple Bitrates
Target Quality (N)

Multiple Bitrates
Target encapsulation formats (NxM)

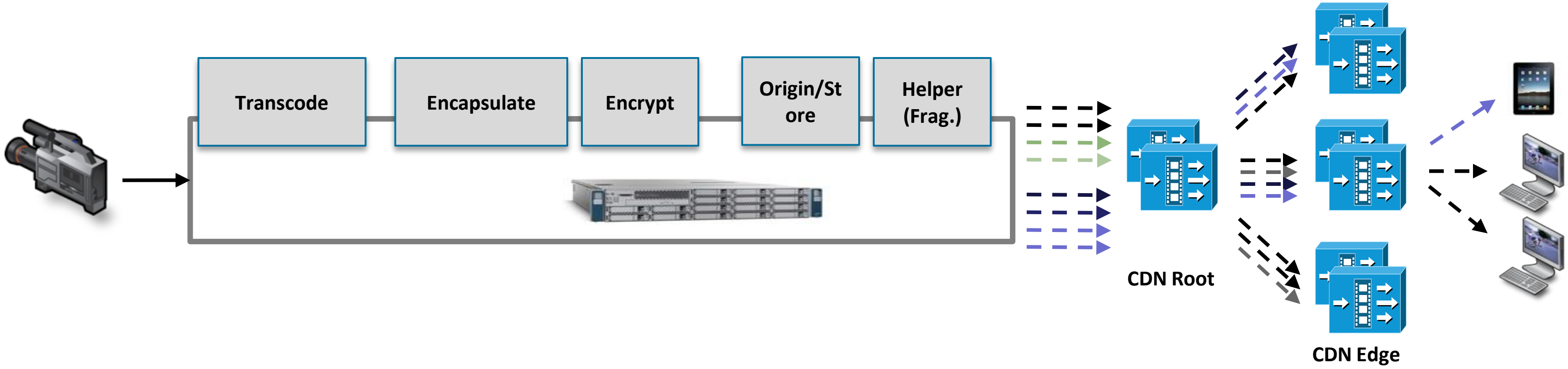
Multiple Bitrates
Target encapsulation formats with DRM

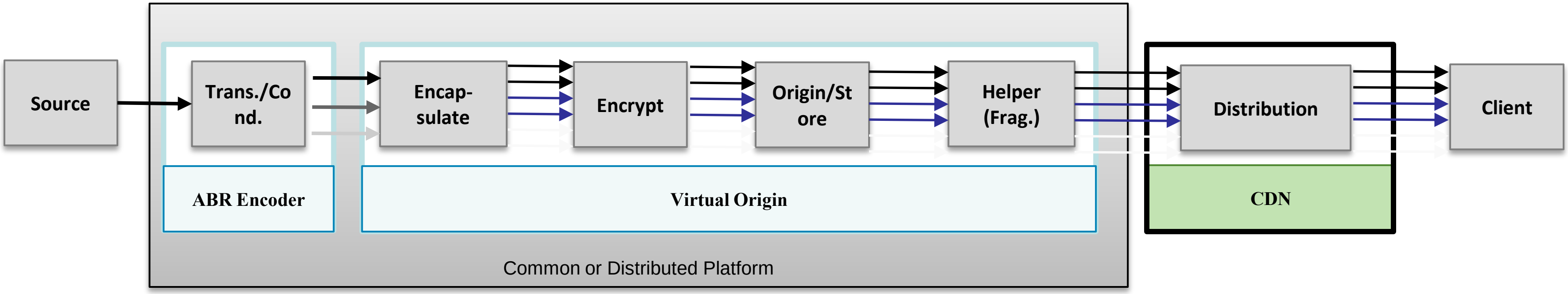
Large video files or virtual files and manifests

Player-specific fragments/segments

GLOBECOMM

Encoding, Encapsulation, & Origin on a single platform





Virtual Origin

- Separates the Encapsulation, Encryption, Storage, and Helper functions into flexible processes that can be instantiated in different locations of the architecture
- Provides a unified architecture for VOD, Linear, and Timeshifting (CloudDVR). Supports multiscreen deployments (Legacy STB & ABR clients)
- Proximity Routing, Load Balancing and Resiliency
- Supports External Origins as well as direct ingest from Transcoders
- Multi-vendor solution (Microsoft, Apple, Adobe).
  - For protocols with Helper functions (IIS & FMS), implements Helper functionality directly in VOS, eliminating the need for a layer of servers in the Data Center.
  - Removes a point of failure, increases ability to scale, deployment approaching the edge of the network
- **Adapts to evolving standards like DECE UV and DASH**

What is Just-in-Time Processing (JITP)?

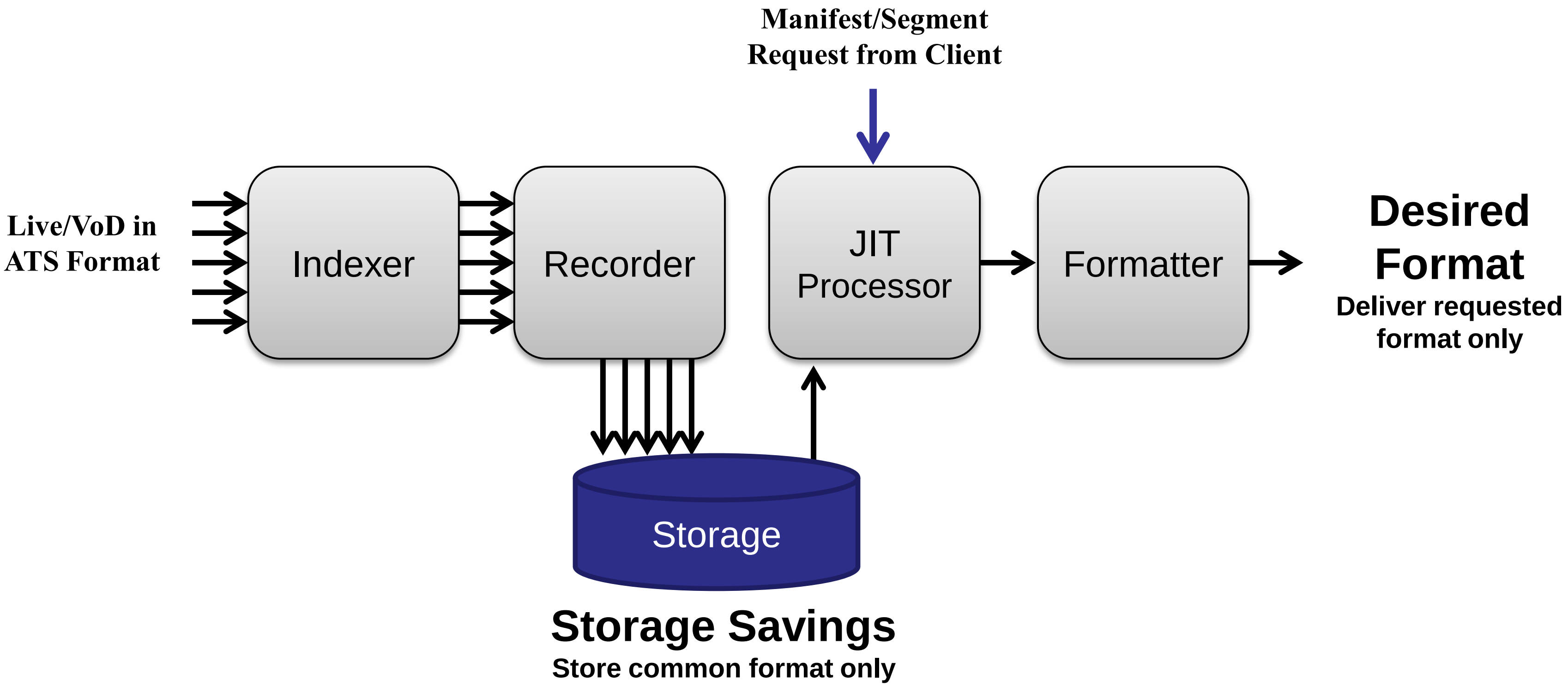
- Single flavor in storage (Intermediary ABR-conditioned Format)
  - Result of VoD Transcode or Linear Recording
  - Assets Index to assist JIT

- On-demand, JITP produces Target-specific Manifest
  - Complete VoD Manifest if source asset complete
  - Linear Manifest starting at beginning of assert if still recording

- Client makes requests against provided manifest
  - Fragments: Random seeks against known fragments
  - Updated Manifest in case of manifest updates (HLS)

- JITP continues to update Manifest if required
- JITP only produces fragments on-demand that are requested

JIT Processing Flow

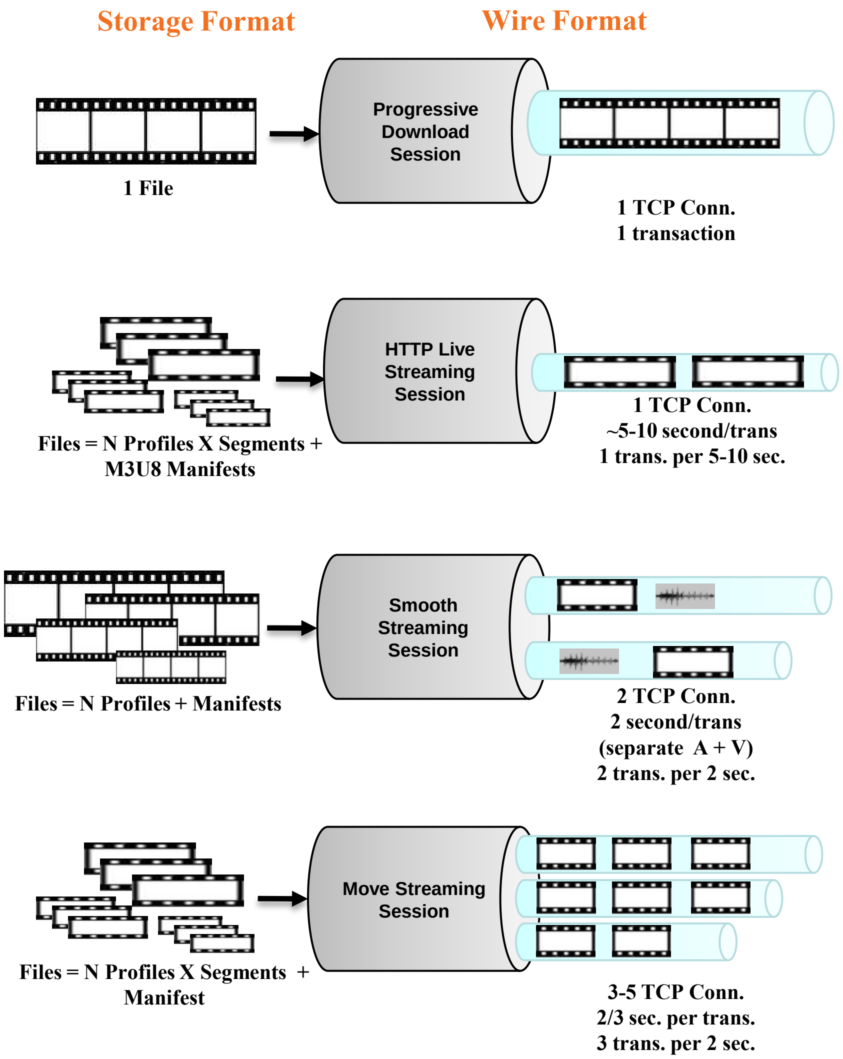


- Stored Indexed Intermediary Format
- Dynamic Manifest, Encapsulation and DRM based on requests
- Storage savings (only store common, ABR-independent format)
- Network savings (only deliver requested fragments, not full ABR set)

GLOBECOMM®

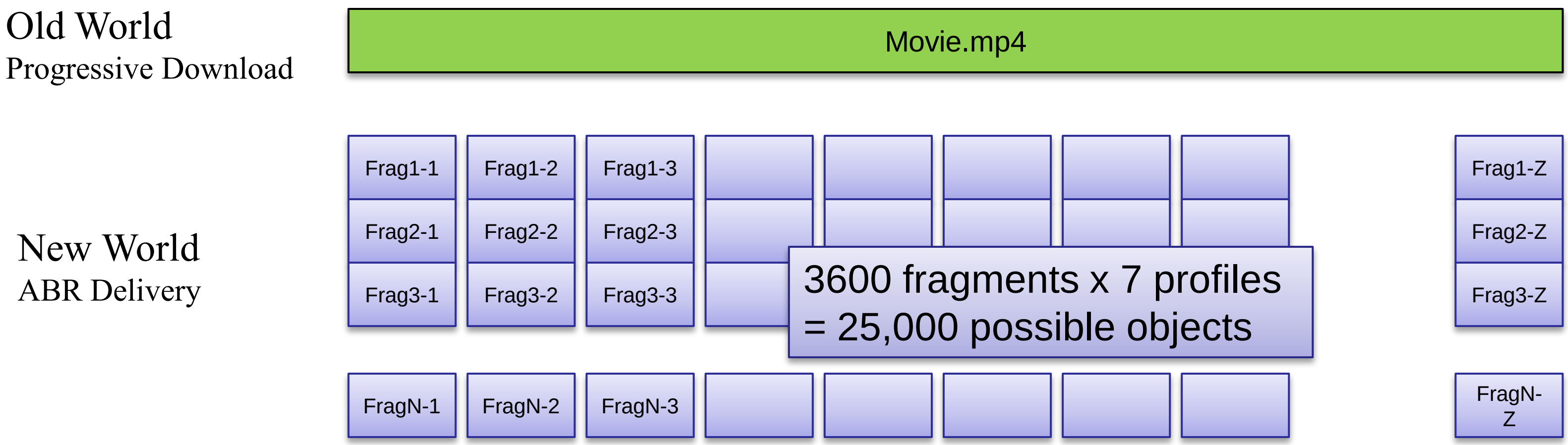
HTTP ABR – CDN Challenges

- ABR = Adaptive Bit Rate
  - Unicast HTTP-based delivery (and hence TCP congestion control)
  - Client-driven adaptation to available BW and CPU
- Large number of (relatively) small objects
  - File Storage vs. Wire Formats
- Transaction Load, File System Load
- Challenges to Reporting and Analytics
- No Inherent Server Side Session State
- Variability in client delivery implementations
- Lack of standard Content Access Protection methods
  - Prevent deep URL linking (including ABR fragments)
  - Prevent certain types of DoS attacks (e.g. Origin Server overload, cache poisoning")



GLOBECOMM

The Challenges with Distributing ABR Objects



3600 fragments x 7 profiles = 25,000 possible objects

- Short fragment sizes translate to very high request TPS
- TCP connections can be short-lived (client and network conditions)
- Different standard fragment sizes (HLS v. Smooth) mean object sizes are different for each Delivery Service. Object handling can be configured on a per-DS basis

Transaction Rates

	Obj Length(sec)	Client Request TPS	TPS for 2000 clients	Objects/Hour/Asset	Obj/Hr 200 channels
Smooth	2	0.500	1,000	1800	360,000
HLS	10	0.100	200	360	72,000
PDL	3600	0.000	0.56	1	200

Object Size (MB)

	3000 kbps	1500 kbps	500 kbps
Smooth	0.75	0.38	0.13
HLS	3.8	1.9	0.6
PDL	1,350	675	225

CDN Features to address the ABR challenge

- Optimized TCP connection handling
  - Scaling to support the large # of connections for ABR
- Optimized HTTP transaction handling
  - Scaling to support the high transaction rate of ABR. CDNs designed for ordinary HTTP transaction loads will not meet the high transactional demands of ABR
- Request Bundling
  - For live streaming, aggregates multiple cache-fill requests for same content into a single request from next cache-tier or Origin Server
- Small Object Cache Throughput Optimizations
  - Small objects written to memory, delayed write to disk
  - Large objects continue to be cached on disk
  - Customized object size caching behavior per Delivery Service
- Content Access Protection
  - URL signing
  - Access authentication
- Live ABR and Client Request Optimizations
  - Request Bundling – Multiple near-time requests result in single requests upstream
  - Range Request Caching (HLS clients, Progressive DL clients)
- Service Visibility
  - Reporting and Analytics optimizations for ABR
  - Streamer performance metrics associated with delivery transactions for overall system behavior views
  - Exposure of service metrics and transaction logs for 3rd party monitoring/reporting systems.

Thank you.