Contents

List of Abbreviations

VOD	Video On Demand
DRM	Digital Rights Management
OTT	Over The Top
QoE	Quality of Experience
AWS	Amazon Web Services
VM	Virtual Machine
GOP	Group Of Pictures
MB	Macroblock
MPEG	Moving Picture Experts Group
AVC	Advanced Video Coding
VP	Video Phone
HEVC	High Efficiency Video Coding
ISOBMFF	ISO Base Media File Format
HTTP	HyperText Transfer Protocol
RTMP	Real-Time Messaging Protocol
RTSP	The Real Time Streaming Protocol
HLS	HTTP Live Streaming
DASH	Dynamic Adaptive Streaming over HTTP
CDN	Content Delivery Networks
P2P	Peer-to-Peer
MCU	Multipoint Control Unit
DVD	Digital Versatile Disc
UDP	User Datagram Protocol
RTP	Real-Time Protocol
MPD	Media Presentation Description
XML	Extensible Markup Language
CMAF	Common Media Application Format
M2TS	MPEG-2 Transport Stream
M3U8	MP3 Playlist File (UTF-8)
SDTP	Stall Duration Tail Probability
CPU	Central Processing Unit
VRC	Video Cassette Recording
MAC	Message Authentication Code
CVSE	Cloud-based Video Streaming Engine
SLA	Service Level Agreement
SLO	Service Level Objectives

Chapter 1

A Survey on Cloud-Based Video Streaming Services

Xiangbo Li, Mahmoud Darwich, Magdy Bayoumi, Mohsen Amini Salehi

ABSTRACT

Video streaming, in various forms of video on demand (VOD), live, and 360 degree streaming, has grown dramatically during the past few years. In comparison to traditional cable broadcasters whose contents can only be watched on TVs, video streaming is ubiquitous and viewers can flexibly watch the video contents on various devices, ranging from smart-phones to laptops and large TV screens. Such ubiquity and flexibility are enabled by interweaving multiple technologies, such as video compression, cloud computing, content delivery networks, and several other technologies. As video streaming gains more popularity and dominates the Internet traffic, it is essential to understand the way it operates and the interplay of different technologies involved in it. Accordingly, the first goal of this paper is to unveil sophisticated processes to deliver a raw captured video to viewers' devices. In particular, we elaborate on the video encoding, transcoding, packaging, encryption, and delivery processes. We survey recent efforts in academia and industry to enhance these processes. As video streaming industry is increasingly becoming reliant on cloud computing, the second goal of this survey is to explore and survey the ways cloud services are utilized to enable video streaming services. The third goal of the study is to position the undertaken research works in cloud-based video streaming and identify challenges that need to be obviated in future to advance cloud-based video streaming industry to a more flexible and user-centric service.

KEYWORDS

Video Streaming; Video Transcoding; Video Packaging; Delivery Network; Digital Management Right (DRM); Cloud Computing.

1.1 INTRODUCTION

1.1.1 Overview

The idea of receiving a stream of video contents dates back to the invention of television in the early years of the 20th century. However, the medium on which people receive and watch video contents has substantially changed during the past decade—from conventional televisions to streaming on a wide variety of devices (*e.g.,* laptops, desktops, and tablets) via Internet. Adoption of the Internet-based video streaming is skyrocketing to the extent that it has dominated the whole Internet traffic. A report by Global Internet

Phenomena shows that video streaming has already accounted for more than 60% of the whole Internet traffic [109]. The number of Netflix[1] subscribers has already surpassed cable-TV subscribers in the U.S. [128].
Nowadays, many Internet-based applications function based on video streaming. Such applications include user-generated video contents (*e.g.,* those in YouTube[2], Vimeo[3]), live streaming and personal broadcasting through social networks (*e.g.,* UStream[4] and Facebook Live[5]), over the top (OTT) streaming (*e.g.,* Netflix and Amazon Prime[6]), e-learning systems [120] (*e.g.,* Udemy[7]), live game streaming platform (*e.g.,* Twitch[8]), video chat and conferencing systems [87], natural disaster management and security systems that operate based on video surveillance [30], and network-based broadcasting channels (*e.g.,* news and other TV channels) [67].
As video streaming services grow in popularity, they demand more computing services for streaming. The uprising popularity and adoption of streaming has coincided with the prevalence of cloud computing technology. Cloud providers offer a wide range of computing services and enable users to outsource their computing demands. Cloud providers relieve video streaming providers from the burden and implications of maintaining and upgrading expensive computing infrastructure [84]. Currently, video stream providers are extensively reliant on cloud services for most or all of their computing demands [53]. The marriage of video streaming and cloud services has given birth to a set of new challenges, techniques, and technologies in the computing industry.
Although numerous research works have been undertaken on cloud-based video streaming, to our knowledge, there is no comprehensive survey that shed lights on challenges, techniques, and technologies in cloud-based video streaming. As such, the essence of this study is to *first*, shed light on the sophisticated processes required for Internet-based video streaming; *second*, provide a holistic view on the ways cloud services can aid video stream providers; *third*, provide a comprehensive survey on the research studies that were undertaken in the intersection of video streaming and cloud computing; and *fourth* discuss the future of cloud-based video streaming technology and identify possible avenues that require further research efforts from industry and academia.
Accordingly, in this study, we first explain the way video streaming works and elaborate on each process involved in it. Then, in Section 1.3, we provide a holistic view of the challenges and demands of the current video streaming industry. Next, in Section 1.4, we discuss how cloud services can fulfill the demands of video streaming, and the survey the research works undertaken for that purpose. In the end, in Section 1.5, we discuss the emerging research areas in the intersection of video streaming and cloud computing.

1.1.2 Cloud Computing for Video Streaming

To provide a high Quality of Experience (QoE) for numerous viewers scattered worldwide with diverse display devices and network characteristics, video stream providers

1. https://www.netflix.com
2. https://www.youtube.com
3. https://www.vimeo.com
4. https://video.ibm.com
5. https://www.facebook.com
6. https://www.amazon.com
7. https://www.udemy.com
8. https://www.twitch.tv

commonly pre-process (*e.g.,* pre-transcode and pre-package) and store the video contents of multiple versions [83]. As such, viewers with different display devices can readily find the version matches their devices. An alternative approach to pre-processing videos is to process them in a lazy (*i.e.,* on-demand) manner [83, 84]. In particular, this approach can be useful for videos that are rarely accessed. Recent analytical studies on the statistical patterns of accessing video streams (*e.g.,* [28]) reveal that the videos of a repository are not uniformly accessed. In fact, streaming videos follows a long-tail pattern [97]. That means, many video streams are rarely accessed and only a small portion (approximately 5%) of the videos (generally referred to as *hot video streams*) are frequently accessed. Both pre-processing and on-demand approaches need the video streaming provider to provide an enormous computing facility.

Maintaining and upgrading in-house infrastructure to fulfill the computational, storage, and networking demands of video stream providers is costly. Besides, it is technically far from the mainstream business of stream providers, which is video content production and publishing. Alternatively, cloud service providers, such as Amazon Cloud (AWS), Google Cloud, and Microsoft Azure, can satisfy these demands by offering efficient services with a high availability [82]. Video stream providers have become extensively reliant on cloud services for most or all of their computing demands. For instance, Netflix has outsourced the entirety of its computational demands to Amazon cloud [101].

In spite of numerous advantages, deploying cloud services has presented new challenges to video stream providers. In particular, as cloud providers charge their users in a pay-as-you-go manner for their services [111], the challenge for stream providers is to minimize their cloud expenditure, while offering a certain level of QoE for their viewers.

Numerous research works have been conducted to overcome the challenges of video streaming using cloud services. For instance, researchers have studied the cost of deploying cloud services to transcode videos [34, 82], the cost-benefit of various video segmentation models [65, 83], applying customized scheduling methods for video streaming tasks [88, 34], and resource (Virtual Machine) provisioning methods for video streaming [83, 64]. Nonetheless, these studies mostly concentrate on one aspect of video streaming processing and how that aspect can be performed efficiently on the cloud. Further studies are required to position these works in a bigger picture and provide a higher level of vision on the efficient use of cloud services for the entire workflow of video streaming.

1.2 THE MYSTERY OF VIDEO STREAMING OPERATION

1.2.1 Structure of a Video Stream

As shown in Figure 1.1, a video stream consists of a sequence of multiple smaller segments. Each segment contains several *Group Of Pictures* (GOP) with a segment header at the beginning of each GOP. The segment header includes information such as the number of GOPs in that segment and the type of the GOPs. A GOP is composed of a sequence of frames. The first frame is an `I` (intra) frame, followed by several `P` (predicted) and `B` (bi-directional predicted) frames. A frame in a GOP is divided into multiple *slices* and each slice consists of several *macroblocks* (MB). The MBs are considered as the unit for video encoding and decoding operations.

Two types of GOP can exist, namely closed-GOP and open-GOP. In the former, GOPs are independent, *i.e.,* there is no relation between GOPs. Therefore, closed-GOPs can

be processed independently. Alternatively, in open-GOP, a GOP is dependent on another GOP, hence, cannot be processed independently.

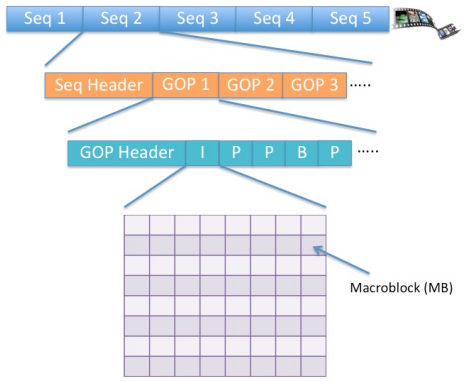


FIGURE 1.1 The structural overview of a video stream. A video stream contains multiple segments and each segment contains multiple GOPs. A frame in a GOP includes a number of macroblocks (MB).

To process video streams, they can be split at different levels, namely segment level, GOP level, frame level, slice level, and macroblock level. Sequence-level contains several GOPs that can be processed independently. However, due to the large size of a sequence, its transmission and processing time become a bottleneck [75]. Processing at the frame, slice, and macroblock levels implies dealing with the spatio-temporal dependencies that makes the processing complicated and slow [75]. In practice, video stream providers generally perform processing at the segment or GOP levels. That is, they define a segment or a GOP as a unit of processing (*i.e.,* a *task*) that can be processed independently [65].

1.2.2 Video Streaming Operation Workflow

In both Video On Demand (VOD) (*e.g.,* Hulu, YouTube, Netflix) and Live streaming (*e.g.,* Livestream[9]), the video contents generated by cameras have to go through a complex workflow of processes before being played on viewers' devices. In this section, we describe these processes.

Figure 1.2 provides a bird's-eye view of the main processes performed for streaming a video—from video production to playing the video on viewers' devices. These processes collectively enable raw and bulky videos, generated by cameras, be played on a wide variety of viewers' devices in a real-time manner and with the minimum delay. It is noteworthy that, in addition to these processes, there are generally other processes to enable features such as video content protection and cost-efficiency of video streaming. In the rest of this section, we elaborate on the main processes required for video streaming. Additional processes (*e.g.,* for video content protection and analysis of video access rates) are discussed in the later parts of the paper (Sections 1.3.6, 1.3.7, and 1.3.8, respectively).

9. https://livestream.com/

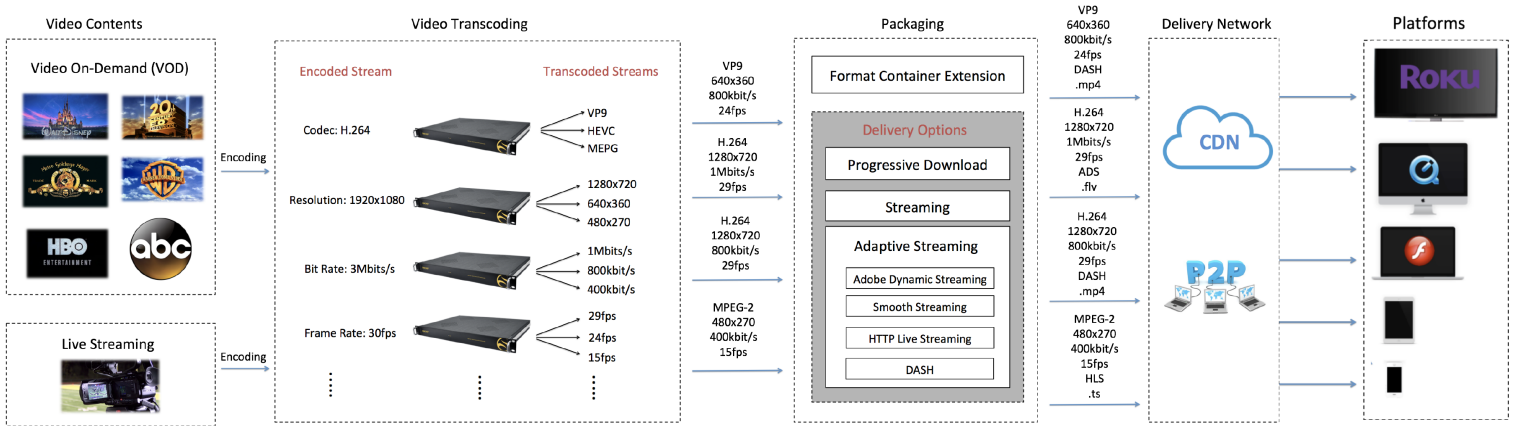


FIGURE 1.2 A bird's-eye view to the workflow of processes performed on a video stream from the source to the viewer's device. The workflow includes compression, transcoding, packaging, and delivery processes.

The first step in video streaming is video content production. A raw video content generated by cameras can consume enormous storage space, which is impossible to be transmitted via current Internet speed. For instance, one second of raw video with 4K resolution occupies approximately 1 GB storage. Therefore, the generated video, firstly, has to be compressed, which is also known as *video encoding*. The concept of the video is just continuously showing a large number of frames at a certain rate (aka frame rate) to create a moving delusion. This large number of frames usually contains spatial and temporal redundancies both inside a frame and between successive frames. In the video compression process, these redundancies are removed using a certain compression standard, *e.g.,* H.264/MPEG-4 AVC [115], VP9 [98], and H.265/HEVC [121]. The compression process encodes the raw video based on a specific compressing standard (known as *codec*), resolution, bit rate, and frame rate at a significantly smaller size.

A viewer's display device generally can decode videos with a certain compression format. Therefore, to support heterogeneous display devices, a video encoded in a certain format has to be converted (also called *transcoded*) to various formats. In the transcoding process, a video is first decoded and then encoded to another compression format. Thus, transcoding is generally a computationally intensive process. In Section 1.3.3, we elaborate further on the video transcoding process.

For streaming of a transcoded video to a viewer's device, the video file has to be structured to facilitate transferring and time-based presentation of the video content. Thus, the transcoded video file must be *packaged* based on a certain structure that is supported by the viewer's player. The packaging process is the basis for what is commonly known as the video file format (also, called as *container format*). The container format introduces a header that includes information about the supported streaming protocols and the rules to send video segments. ISO Base Media File Format (ISOBMFF) [126], which is the basis for *MP4* format, and 3GPP TS [104], which is the basis for *ts* format, are widely used for the packaging process.

To deliver the packaged (*i.e.,* formatted) video files over the network, they need to use an application layer network protocols (*e.g.,* HTTP [42], RTMP [78], and RTSP [114]). There are various *delivery techniques* that dictates the way a video stream is received and played. Progressive Download [26], HTTP Live Streaming (HLS) [106], and Dynamic Adaptive Streaming over HTTP (DASH) [118] are examples of delivery techniques. Each delivery technique is based on a particular network application layer protocol. For instance, HLS and DASH work based on HTTP and Adobe Flash streaming works based on RTMP. Further details about video packaging are discussed in Section 1.3.4.

Once the video is packaged, it is delivered to viewers around the world through a distribution network. However, due to the Internet transmission delay, viewers located far from video servers and repositories suffer from a long delay to begin streaming [70]. To reduce this delay, Content Delivery Networks (CDN) [21] are used to cache the frequently watched video contents at geographical locations close to viewers. Another option to reduce the startup delay is to use Peer-to-Peer (*i.e.,* serverless) approaches in which each receiver (*e.g.,* viewer's computer) acts as a server and sends video contents to another peer viewer [108]. More details of the delivery network will be discussed in Section 1.3.5

1.3 VIDEO STREAMING: CHALLENGES AND SOLUTIONS

1.3.1 Overview

In the previous section, we described the high-level workflow of processes for video streaming. However, there are various approaches and challenges to accomplish those processes. In this section, we elaborate on all issues, challenges and current solutions that are related to video streaming.

Figure 1.3 presents a taxonomy of all the issues that a video streaming system needs to deal with. The taxonomy shows different types of video streaming (*i.e.,* VOD and live streaming). It discusses variations of video transcoding and packaging processes along with important streaming protocols. The taxonomy shows possible ways of distributing video content, such as CDN [146] and P2P [92]. Video content protection is further detailed on video privacy for video streaming [144] and copyright issues, known as Digital Right s Management (DRM) [91]. The taxonomy covers analytical studies that have been conducted to understand the viewers' behaviors and discovering their access patterns to videos streams. Last, but not the least, the taxonomy covers the storage issue for video streaming and possible strategies for video streaming repository management, such as in-house storage versus outsourcing solutions via cloud storage services.

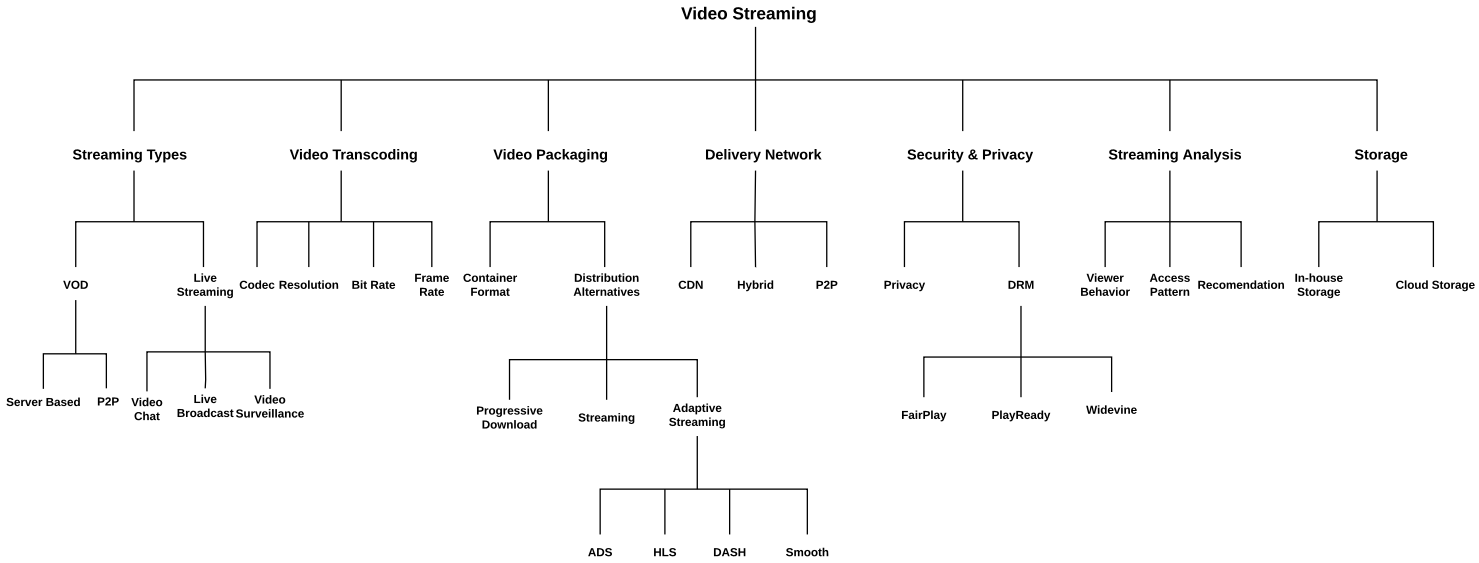


FIGURE 1.3 Taxonomy of all aspects we need to deal with in video streaming.

1.3.2 Video Streaming Types

The taxonomy shown in Figure 1.4 expresses possible ways a video streaming service can be provided to viewers. More Specifically, video streaming service can be offered in three main fashions: *Video On Demand* (VOD) streaming, *live* streaming, and *live-to-VOD* streaming.

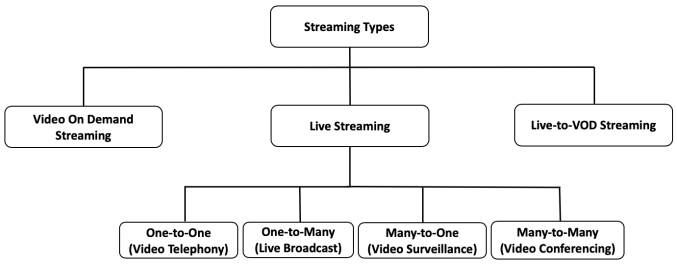


FIGURE 1.4 Taxonomy of different types of video streaming.

1.3.2.1 VOD Streaming

In VOD streaming, which is also known as Over-The-Top (OTT) streaming, the video contents (*i.e.,* video files) are already available in a video streaming repository and are streamed to viewers upon their requests. By far, VOD is the most common type of video streaming and is offered by major video streaming providers, such as YouTube, Netflix, and Amazon Prime Video.

Some VOD providers (*e.g.,* Netflix and Hulu) offer professionally made videos and movies that are subscription-based and viewers are generally required to pay a monthly fee to access their service. Alternatively, other VOD services (*e.g.,* YouTube) operate based on user-provided videos. Such services are generally advertisement-based and free of charge. VODs have also applications in e-learning systems [120], Internet television [107], and in-flight entertainment systems [40].

1.3.2.2 Live Streaming

In live video streaming, the video contents are streamed to the viewer(s), as they are captured by a camera. Live video streaming has numerous applications, such as event coverage and video calls. The live video streaming used in different applications have minor differences that are mainly attributed to the buffer size on the sender and receiver ends [76]. In general, a larger buffer size causes a more stable streaming experience but imposes more delay. This delay can be tolerated in live broadcasting applications, however, delay-sensitive applications (*e.g.,* video telephony) cannot bear the delay, thus need a shorter buffer size. Accordingly, live streaming can have four variations as follows:

(A) **One-to-one** (unicast) streaming is when a user streams video contents to another user. This type is primarily used in video chat and video call applications that require two live streams, one from each participant. This streaming type requires short delays to enable smooth conversation between participants. As such, these applications generally operate with a short buffer size and low picture quality to make the delay as

small as possible [76]. Skype[10] and video telephony applications [59] like FaceTime[11] are instances of this type of live streaming.

(B) **One-to-many** (multicast) streaming is when one source streams video to many viewers. A well-known example of this type is live broadcasting which is currently offered by many social network applications, such as Facebook and Instagram. Jo *et al.* [62] conducted one of the first studies on live streaming. They identified and addressed several challenges in multicast streaming regarding signaling protocols, network stability, and viewer variations.

(C) **Many-to-one** occurs when several cameras capture scenes and send them to one viewer. The most important application for this type of streaming is multi-camera video surveillance which is used for situational awareness for security purposes or natural disaster management [54]. In this type of streaming, the video contents are collected from multiple cameras and displayed on special multi-screen monitoring devices[49].

(D) **Many-to-many** streaming occurs when a group of users in different geographical locations holds a video conference. In this case, all users stream live to all others. For this streaming type, Multipoint Control Unit (MCU) [76] method can be used to combine individual participants into a single video stream and broadcast it. Most of video chat applications, *e.g.,* Skype and Google Hangouts, support many-to-many live streaming, in addition to one-to-many streaming.

1.3.2.3 Live-to-VOD Streaming

In addition to live and VOD streaming, we can also consider a combination of live and VOD streaming, known as Live-to-VOD [117], as another type of streaming. In this type of streaming, which is mostly used on one-to-many live streaming, the live video stream is recorded and can be readily used in form of VOD streaming.

Using live-to-VOD streaming, viewers who are not online during the live stream can watch the video at a later time. In addition, live-to-VOD can provide VOD-like services to live stream viewers. For instance, live stream viewers can have the rewind ability. Another application of live-to-VOD is to play live video contents in different time zones. For example, using live-to-VOD, the same TV program that is live streamed at 8:00 am in the Eastern Time Zone, can be played at 8:00 am in the Pacific Time Zone.

1.3.2.4 Differences in Processing Live and VOD Streaming

Although the workflow and processes that are applied to live video streams are the same as those for VOD, there are certain differences between them. Specifically, live and VOD streaming are different on the way they are processed [82].

First, in live streaming, the video segments are processed as they are generated. This has two implications:

- The video segments have to be processed (*e.g.,* transcoded) on-the-spot, whereas in VOD, it is possible to pre-process videos (*i.e.,* in an off-line manner).
- There is no historic execution (*i.e.,* processing) time information for live video segments [82]. In contrast, in VOD, each video segment is processed multiple times and

10.https://www.skype.com
11.https://support.apple.com/en-us/HT204380

the historic execution time information are available. The historic information are particularly important for the efficient scheduling of video streaming tasks.

The second difference between live and VOD streaming is the way they are treated for processing. In both live and VOD streaming, to ensure Quality of Experience (QoE), each video streaming task is assigned a deadline that must be respected. That is, the video task processing must be completed before the assigned deadline. The deadline for each video task is determined based on the presentation time of the pertinent video segment. If a task cannot meet its assigned deadline for any reason, then, in VOD, the task has to wait until it is processed [81]. In contrast, in live streaming, if a task misses its deadline, the task must be dropped (*i.e.,* discarded) to keep up with the live streaming [82]. In other words, there is no reason to process a task whose time has passed.

The third difference between VOD and live streaming is again related to the way deadlines are assigned to video streaming tasks. In fact, in video streaming, if a task misses its deadline, all the tasks behind that, *i.e.,* those process video segments later in the stream, should update their streaming deadlines (presentation times) accordingly. This is also known as the dynamic deadline, however, this is not the case in live streaming and the tasks' deadlines in live streaming cannot be changed.

1.3.3 Video Transcoding

Video contents are originally encoded with one specific spatial resolution, bit rate, frame rate, and compression standard (codec). In order to play the videos on different devices, streaming service providers usually have to adjust the original videos in terms of the viewer's device and network bandwidth. The process of this adjustment is called *video transcoding* [3, 131].

Video transcoding is a compute-intensive task and needs powerful computers to process it [80]. Therefore, video transcoding is generally carried out in an off-line manner, called pre-transcoding [84]. In the following subsections, we will elaborate different transcoding operations, respectively.

1.3.3.1 Bit Rate

Video bit rate is the number of bits used to encode a unit time of video. Bit rate directly impacts on video quality, as higher bit rate produces better quality, while higher bit rate consumes larger network bandwidth and storage. In order to stream videos to viewers with different network conditions, streaming service providers usually convert video with multiple bit rates [135].

1.3.3.2 Spatial Resolution

Resolution represents the dimensional size of a video, which indicates the number of pixels on each video frame. Therefore, higher resolution contains more pixels and details, as results in larger size. Video resolution usually needs to match with the screen size. Low resolution video plays on large screen will causes blurry after upsample, while high resolution plays on small screen is just a waste of bandwidth since viewer usually won't notice the difference due to the limited pixels on the screen. To adapt to the diverse screen size on the market, original videos have to be transcoded to multiple resolutions [18].

1.3.3.3 Frame Rate

When still video frames plays at a certain speed rate, human visual system will feel the object is moving. Frame rate indicates number of video frames shown per second. Videos or films are usually recorded at high frame rate to produce smooth movement, while devices may not support such high frame rate. Therefore, in some cases, videos have to be reduced to a lower frame by removing some frames. On the other hand, increasing frame rate is more complicated than reducing frame rate, since it have add non-existent frames. Overall, to be adaptive to larger scale device, video are transcoded to different frame rates [45].

1.3.3.4 Video Compression Standard

Video Compression standard is the key to compress a raw video, the encoding process mainly goes through four steps: prediction, transformation, quantization, and entropy coding, while decoding is a just reverted encoding process. With different codecs manufactured on different devices (*e.g.,* DVD player with MPEG-2 [52], BluRay player with H.264 [136], 4K TV with HEVC [121]), an encoded video may have to be converted to the supported codec on that device. Changing codec is the most compute-intensive type of transcoding [80] since it has to decode the bitstream with the old codec first and then encode it again with the new codec.

1.3.4 Video Packaging

Transmitting an encoded/transcoded video file from server to viewer involves multiple layers of network protocols, namely physical layer, data link layer, network layer, transport layer, session layer, presentation layer, and application layer [66]. The protocols of these layers dictate video packaging details, such as stream file header syntax, payload data, authorization, and error handling. Since video streaming protocols operate under the application layer, they potentially can use different protocols in the underlying layers to transmit data packets.

Choosing the right streaming technology requires understanding pros and cons of the streaming protocols and video packaging (aka container formats). In this section, we discuss three popular streaming technologies plus the streaming protocols and container formats required for each one of the technologies.

1.3.4.1 Progressive Download

Back in old days, when online video streaming was not practical, a video could not be viewed until it was completely downloaded on the device. This implies that viewers usually had to wait for a considerable amount of time (from minutes to even hours) before watching the video. Progressive download resolved this issue by allowing a video to be played as soon as the player's initial buffer is filled by segments of the video. This reduces the waiting time down to 3–10 seconds to begin watching a video [14].

Due to the downloading feature, progressive download can face three problems. First, since a video is downloaded linearly, if the viewer's network bandwidth is too low, the viewer cannot move forward a video until that part is fully downloaded. Second downside of progressive download is that if a video file is fully downloaded, but viewer stops watching in the middle, the rest bandwidths are wasted. Third, copyright is problematic

in progressive download because the whole video is downloaded on the viewer's storage device. Progressive Download utilizes HTTP protocol that itself operates based on the TCP protocol, which provides better reliability and error-resilience than UDP, but it incurs a high network latency [105]. These inherent drawbacks of HTTP-based progressive download raised the need to a dedicated technology for video streaming.

1.3.4.2 Dedicated Protocol for Video Streaming

To avoid problems of progressive download, a dedicated protocol for real-time streaming (known as RTP) [58] was created. This protocol delivers video contents from a separated streaming server. While traditional HTTP servers handle web requests, streaming servers only handle streaming. The connection is initiated between the player and the streaming server whenever a viewer clicks on a video in a web page. The connection persists until the video terminates or the viewer stops watching it. In comparison to stateless HTTP, RTP protocol is considered stateful because of this persistent connection.

Because of the persistent connection, dedicated streaming protocols allow random access (*e.g.,* fast forward) within the streamed video. In addition, they allow adaptive streaming, in which multiple encoded video streams could be delivered to different players based upon available bandwidth and processing characteristics. The streaming server can monitor the outbound flow, so if the viewer stops watching the video, it stops sending video packet to the viewer.

Video content in the streaming server is split into small chunks, whenever these chunks are sent to the viewers, they are cached at the local device and will be removed after they are played. This feature offers freedom to viewer to move back and forth within the streamed video. It also protects the copyright of the video content. Although streaming technology was attractive in the beginning, its drawback appeared after deployment. A streaming protocol, *e.g.,* RTMP [78] used by Adobe Flash, utilizes different port numbers from HTTP. As such, RTMP packets can be blocked with some firewalls. The persistent connection between the streaming server and viewer players increases the network usage and causes limited scalability for streaming servers.

1.3.4.3 Adaptive Streaming

To address the limitations of previous streaming technologies, HTTP-based streaming solutions came back to the forefront of streaming technology-adaptive streaming. All adaptive streams are broken into chunks and stream separate videos. There is no persistent connection between the server and the player. Instead of retrieving a single large video file in one request, adaptive streaming technology retrieves a sequence of short video files in an on-demand basis.

Adaptive streaming has the following benefits: First, like streaming server, there is no network wastage, because the video content is delivered on the go. Therefore, one HTTP server can efficiently serves several streams. Second, HTTP-based streaming is delivered through HTTP protocol, which avoids the firewall issue faced by RTMP. Third, it costs less than using streaming server. Fourth, it can scale quickly and effectively to serve more viewers. Fifth, seeking inside the stream is no more an issue. When the viewer moves the player forwarder, the player just retrieves the exact video segments as opposed to the entire video up to the requested point.

There are four main adaptive streaming protocols, namely MPEG Dynamic Adaptive Streaming over HTTP (DASH) [118], Apple HTTP Live Streaming (HLS) [123], and

Microsoft Smooth Streaming [16].
MPEG DASH delivers ISO Base Media File Format (ISOBMFF) [20] video segments. It defines a Media Presentation Description (MPD) XML document to provide the locations of video streams, so that players know where to download them. The media segments for DASH is delivered with formats either based on the ISOBMFF [126] or Common Media Application Format (CMAF) standards [43].
Apple's HLS is well-known and implemented on the Apple devices (and nowadays on Android devices too). It utilizes a M3U8 master manifest to include multiple media playlists, each playlist represents one stream version and it contains the location of all the video segments for this stream. HLS video segment uses either MPEG-2 Transport Stream (M2TS) [51] or CMAF [43] for H.264 encoded videos, and ISOBMFF for HEVC encoded videos.
Smooth Streaming has two separated manifest files, namely SMIL server manifest file and client manifest file. They are all defined in XML format documents. Smooth streaming also delivers video segment with a format (known as ISMV) based on ISOBMFF.
These four adaptive streaming technologies have empowered streaming service providers to deliver the video contents to viewers smoothly even under low bandwidth Internet connection. However, to support all viewers' platforms, stream providers have to deploy and maintain all these four streaming protocols that subsequently increases complexity and costs. The supported platforms of these four protocols are shown in Table 1.1

TABLE 1.1 Adaptive Streaming Supported Platforms

	Desktop Player	Mobile Device Support	OTT Support
MPEG-DASH	dash.js, dash.as, GPAC	Windows, Android Phone	Google TV, Roku, Xbox 360
Apple HTTP Live Streaming (HLS)	iOS, Mac OSX, Flash	iOS/Android3.0+	Apple TV, Boxee, Google TV, Roku
Smooth Streaming	Silverlight	Windows Phone	Google TV, Roku, Xbox 360

1.3.5 Video Streaming Delivery Networks

1.3.5.1 Content Delivery Networks (CDN)

The goal of CDN technology is to reduce the network latency of accessing web contents. CDNs replicate the contents to geographically distributed servers that are close to viewers [113, 129]. Considering that large size of video contents usually takes a long transmission time, using CDNs to cache video contents close to viewers reduces the latency dramatically. Netflix, as one of the largest stream providers use three different CDN providers (Akamai, LimeLight, and Level-3) to cover all viewers in different regions [1]. The transcoded and packaged video contents are replicated in all three CDNs.
Streaming video contents through CDN has been studied in earlier works [11, 31, 134]. Cranor *et al.* [31] proposed an architecture (called PRISM) for distributing, storing, and delivering high-quality streaming content over IP networks. They proposed a framework to support VOD streaming content via distribution networks services (*i.e.,* CDNs). Wee *et al.* [134] presented an architecture for mobile streaming CDN which was designed to fit the mobility and scaling requirements. Apostolopoulos *et al.* [11] proposed multiple paths between nearby edge servers and clients in order to reduce latency and deliver high-quality streaming. Benkacem *et al.* [17] provided an architecture to offer CDN slices

through multiple administrative cloud domains. Al-Abbasi *et al.* [5] proposed a model for video streaming systems, typically composed of a centralized origin server, several CDN sites, and edge-caches located closer to the end users. Their proposed approach focused on minimizing a performance metric, stall duration tail probability (SDTP), and present a novel and efficient algorithm accounting for the multiple design flexibilities. The authors demonstrated that the proposed algorithms can significantly improve the SDTP metric, compared to the baseline strategies.

Live streaming CDN is discussed in [69, 141, 93] to improve scalability, latency quality, and reliability of the service.

1.3.5.2 Peer to Peer (P2P) Networks

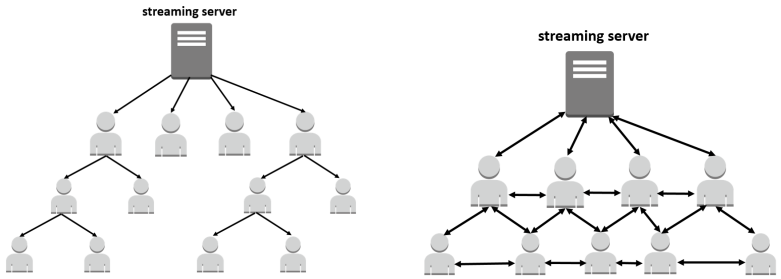
P2P networks enable direct sharing of computing resources (*e.g.,* CPU cycles, storage, and content) among peer nodes in a network [108]. P2P networks are designed for both clients and servers to act as peers. They can download data from same nodes and upload them to other nodes in the network. P2P networks can disseminate data files among users within a short period of time. P2P networks are extensively utilized for video streaming services as well.

P2P streaming is categorized into two types, namely tree-based and mesh-based. Tree-based P2P structure distributes video streams by sending data from a peer to its children peers. In mesh-based P2P structure, the peers do not follow a specific topology, instead, they function based on the content and bandwidth availability on peers [92]. One drawback of using tree-based P2P streaming is the vulnerability to peers' churn. The drawback of deploying mesh-based P2P streaming structure is video playback quality degradation, ranging from low video bit rates and long startup delays, to frequent playback freezes. Golchi *et al.* [46] proposed a method for streaming video in P2P networks. Their approach uses the algorithm of improved particle swarm optimization(IPSO) to determine the appropriate way for transmitting data that ensures service quality parameters. They showed that the proposed approach works efficiently more than the other related methods.

Figure 1.5 shows two popular variations of tree-based P2P structure, namely single tree-based and multi-tree based structures. In *Single Tree-based streaming*, shown in Figure 1.5a, video streaming is carried out at the application layer and the streaming tree is formed by participating users. Each viewer joins the tree at a certain level and receives the video content from its parent peer at the above level. Then, it forwards the received video to its children peers at the below level [29, 60].

In *Multi-Tree Streaming* (aka mesh-tree), shown in Figure 1.5b, a peer develops the peering connection with other close peers. Then, the peer can potentially download/upload video contents from/to multiple close peers simultaneously. If a peer loses the connection to the network, the receiving peer could still download video from remaining close peers. Meanwhile, the peer can locate new peers to maintain a required level of connectivity. The strong peering in mesh-based streaming systems supports them to be robust against peer churns [94, 130]. Ahmed *et al.* [4]proposed a multi-level multi-overlay hybrid peer-to-peer live video system that offers to the Online Games players the way of streaming the video simultaneously and enables to watch the game videos of other players. Their approach aimed to reduce the transmission rate while increasing the number of peers and the delivery and reliability of data are guaranteed on time.

Tree-based P2P VOD streaming shares the video stream on the network and achieves fast streaming. The video stream is divided into small size data block. The server disperses



(a) single tree-based streaming **(b)** mutli-tree streaming

FIGURE 1.5 Ways to achieve peer-to-peer (P2P) video streaming

the data blocks to different nodes. The nodes download their missing blocks from their neighboring peers [132]

Guo *et al.* [50] proposed an architecture that uses the P2P approach to cooperatively stream video by only relying on unicast connections among peers. Xu *et al.* [138] proposed an approach based on binary tree strategy to distribute VOD P2P networks. Their approach divides the videos into segments to be fetched from several peers. Yiu *et al.* [142] proposed VMesh distributed P2P VOD streaming approach. In their approach, videos are split into segments and then stored at the storage of the peers locally. Their proposed design presents to peers an ability to forward, backward, pause, and restart during the playback. Cheng *et al.* [27] proposed a topology enhancing video cassette recording (VCR) functions for VOD services in the networks. Their approach allows a peer to achieve fast seeks relocation by keeping close neighbors and remote them in a set of rings. Xu *et al.* [140] proposed a scheme based tree to make user interacting in the P2P streaming. The proposed scheme presents an advantage to support the users requests asynchronously while maintaining high resilience.

The main advantages of P2P are low cost and flexibility for scalability, however, it suffers from instability in QoS. Therefore, researchers proposed to combine the advantages of both CDN and P2P in one system. Accordingly, several hybrid systems were developed to combine P2P and CDN for content streaming. Afergan *et al.* [2] proposed an approach which utilizes CDN to build CDN-P2P streaming. In their proposed design, they optimized dynamically the number and locations of replicas for P2P service. Xu *et al.* [139] presented a scheme that is formed by both CDN and P2P streaming. They showed the efficiency of their approach by reducing the cost of CDN without impacting the quality of the delivered videos.

1.3.6 Video Streaming Security

1.3.6.1 Privacy

With the ubiquity of video streaming on a wide range of display devices, the privacy of the video contents has become a major concern. In particular, live contents either in form of video surveillance or user-based live-streaming capture places and record many unwanted/unrelated contents. For instance, a person who live-streams from a street, unintentionally may capture plate number of vehicles passing that location. Therefore,

video streaming systems can compromise the privacy of people (*e.g.,* faces and vehicles tags). Various techniques have been developed to protect the privacy of live video contents.

Dufeaux *et al.* [38] introduced two techniques to obscure the regions of interests while video surveillance systems are running. Zhang *et al.* [144] came up with a framework to store the privacy information of a video surveillance system in form of a watermark. The proposed model embeds a signature into the header of the video. Moreover, it embeds the authorized personal information into the video that can be retrieved only with a secret key. Another research, conducted by Carrillo *et al.* [23], introduces a compression algorithm for video surveillance system, which is based on the encryption concept. The proposed algorithm protects privacy by hiding the identity revealing features of objects and human. Such objects and human identity could be decrypted with decryption keys when an investigation is requested.

Live video contents commonly are transmitted via wireless media which can be easily intercepted and altered. Alternatively, DOS attacks can be launched on the live video traffics [6]. As such, in [7, 41] algorithms are provided to distinguish between packets that were damaged because of noise or an attack. In the former case, the errors must be fixed while in the latter the packet must be resent. The algorithm counts the number of 1s or 0s in a packet before the packet is send and uses that count to generate a message authentication code (MAC). The MAC is appended to the end of the packet and sent over the network. When the packet is received, the MAC is calculated again, and the two codes are compared. If the differences in MACs is past a certain threshold, the past is marked as malicious and discarded. Because the MAC is also sent over the network, the algorithm will detect bit errors in both the packet and the MAC.

1.3.6.2 Digital Rights Management

Another security aspect in video streaming is the copyright issue. This is particularly prominent for subscription-based video streaming services (such as Netflix). Digital Rights Management (DRM) is the practice of securing digital contents, including video streams, to prevent unlawful copying and distribution of the protected material. Specifically, its goals are typically centered around controlling access to the content, enforcing rules about usage, and identifying and authenticating the content source. As such, contents protected with DRM are typically sold as a license to use the contents, rather than the content itself. DRM solutions to meet these goals often incorporate tools such as encryption, watermarking, and cryptographic hashing [124].

The process of DRM starts with encryption, video contents are encrypted and stored in the video repository and DRM providers keep the secret keys. Fig. 1.6 summarizes the steps to stream a video protected with DRM. Upon request to stream an encrypted video, after downloading it, the secret key is requested from the DRM provider server to decrypt the video.

Currently, there are three major DRM technologies, namely Apple's Faireplay [119], Microsoft's PlayReady [73] and Google's Widevine [73]. These technologies are able to serve most of the devices and platforms in the market. To cover all devices, streaming service providers have to encrypt every video content with three different DRM technologies and store them in the repository. The three DRM technologies supported platforms are shown in Table 1.2.

With the increasing demand to stream DRM-protected VOD, an offline application-driven

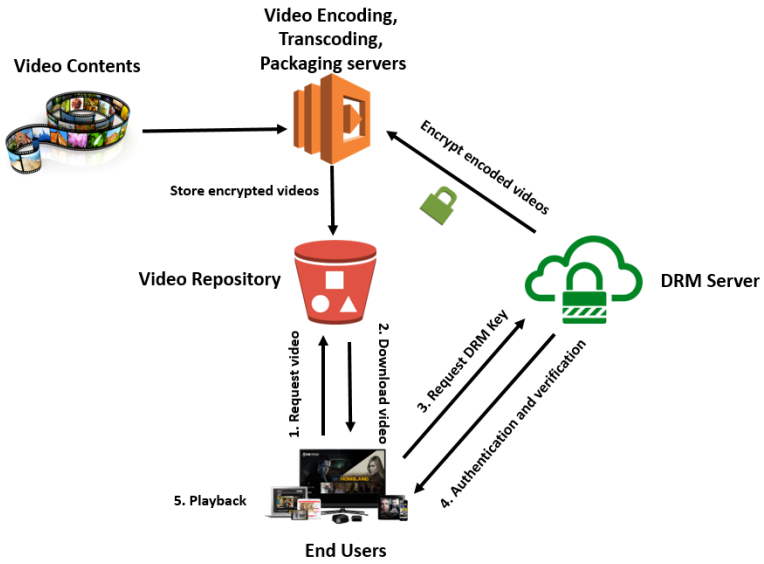


FIGURE 1.6 Workflow of Digital Right Management (DRM) to support of video streaming security

TABLE 1.2 DRM Supported Platforms

	Chrome	FireFox	IE 11	Safari	Android	iOS	Windows Phone	ChromeCast	Roku	Apple TV	Xbox
Fairplay				✓		✓				✓	
PlayReady			✓	✓			✓	✓	✓		✓
Widevine	✓	✓			✓			✓			

DRM interceptor has been proposed in [37] that is the awareness of the network connection status of the client device. In this case, if the interceptor application decides that the client device is offline, it requests the license/key for the protected content. The license/key is controlled by the interceptor application. Accordingly, the requests of license/key are handled by the interceptor application that retrieves them from a locally data-store, and then send the key/license to the DRM module.

1.3.7 Analysis of Video Streaming Statistics

1.3.7.1 Impact of quality on viewers' behavior

Previous studies show that the quality of video streaming impacts on viewers' reception to the video and the revenue of the stream provider [70]. According to the studies, starting video streaming with delay and interruption during video streaming significantly increases the possibility of abandoning watching the videos.

Florin *et al.* [36] addressed the impact of video quality on viewers' interest. They claimed that the percentage of time spent on buffering (*i.e.,* buffering ratio) has the largest impact on the user interest across all content types. The average bit-rate of live streaming has a significant impact on user abandonment of watching the video.Video streaming

providers should attempt to maximize viewer engagement by minimizing the buffering time and rate and increasing the average bit-rate.

1.3.7.2 Access Pattern to Video Streams

Analysis of access pattern to videos shows that the access pattern of videos streams in a repository follows a long-tail distribution [28]. That is, few popular videos, known as *hot* videos that construct around 5% of the repository, are accessed very often while a large portion of non-popular videos are accessed infrequently.

The studies also reveal that viewers are typically interested in recently posted videos. Moreover, for new videos, the popularity fluctuates significantly while the popularity of old videos does not fluctuate significantly [24].

Video access rate indicates how many times a video is accessed by a user, however, it does not implicate if the accessed video stream is played to end of it or not. In fact, recent studies (*e.g.,* [97]) showed that, the beginning segments of a video stream are played more frequently than the rest of it. Miranda et al. [97] revealed that in a video stream, the views are distributed following a long tail distribution. More specifically, the distribution of views of the segments (GOPs) in a video stream can be calculated by the Power-law [102] model.

The access rate of GOPs in all video streams in a repository does not necessarily follow long-tail pattern as stated earlier. There are video streams whose some GOPs in the middle or end of the video stream are accessed more frequently than other GOPs. An example of a soccer match streaming can show GOPs with a tremendously higher access rate where a player scores a goal. We define this type of video streams as those with non-long-tail access pattern [33].

1.3.7.3 Video Streaming based Recommendation

As mentioned above, provided the access pattern of video streams for a given viewer, stream providers are able to predict the video categories the viewer prefers and recommend them at the front page. The same strategy can be used for video recommendation to the viewers located in the same geographical area. To improve the accuracy of prediction and recommendation, machine learning [100] and deep learning [77] approaches have widely been applied for this purpose. The most successful example is Netflix and YouTube machine-learning-based recommendation systems as explained in [47].

1.3.8 Storage of Video Repositories

Video streaming repositories are growing in size, due to the increasing number of content creation sources, diversity of display devices, and the high qualities viewers desire. This rapid increase in the size of repositories implies several challenges for multimedia storage systems. In particular, video streaming storage challenges are threefold: capacity, throughput, and fault tolerance.

One of the main reasons to have a challenge in video streaming storage capacity is the diversity of viewers' devices. To cover increasingly diverse display devices, multiple (more than 90) versions of a single video should be generated and stored.However, storing several formats of the same video implies a large-scale storage system. Previous studies provide techniques to overcome storage issues of video streaming. Miao *et al.* [96] proposed techniques to store some frames of a video on the proxy cache. Their proposed

technique reduces the network bandwidth costs and increases the robustness of streaming video on poor network conditions.
When a video is stored on a disk, concurrent accesses to that video are limited by the throughput of the disk. This restricts the number of simultaneous viewers for a video. To address the throughput issue, several research studies have been undertaken. Provided that storing video streams on a single disk has a low throughput, multiple disk storage are configured to increase the throughput. Shenoy *et al.* [116] propose data stripping where a video is segmented and saved across multiple storage disks to increase the storage throughput. Videos streams are segmented into blocks before they are stored. The blocks can be stored one after another (*i.e.,* contiguously) or scattered over several storage disks. Although contiguous storage method is simple to implement, it suffers from the fragmentation problem. The scattered method, however, eliminates the fragmentation problem with the cost of more complicated implementation.
Scattering video streams across multiple disks and implementing data striping and data interleaving methods improves reliability and fault tolerance of video streaming storage systems [137].

1.4 CLOUD-BASED VIDEO STREAMING

1.4.1 Overview

In this section, we discuss how offered cloud services can be useful for different processes in video streaming. Broadly speaking, we discuss computational services, networking services, and storage services offered by cloud providers and are employed by video stream providers. We also elaborate on possible options within each one of the cloud services.

1.4.2 Computing Platforms for Cloud-based Video Streaming

Upon arrival of a streaming request, the requested video stream is fetched from the cloud storage servers and the workflow of actions explained in Section 1.2.2 are performed on them, before streaming them to the viewers. These processes are commonly implemented in form of independent services, known as micro-services [103], and are generally deployed in a loosely coupled manner on independent servers in cloud datacenters. Services like web server, video ingestion, encoding, transcoding, and packaging are examples of micro-services deployed in datacenters for video streaming. For the sake of reliability and fault tolerance, each of these micro-services is deployed on multiple servers and form a large distributed system. A load balancer is deployed for the servers of each Microservice. The load balancer assigns streaming tasks to an appropriate server with the goal of minimizing latency and to cover possible faults in the servers.
The aforementioned micro-services are commonly implemented via container technologies [125], possibly using serverless computing paradigm [10]. Docker containers scale up and down much faster than VMs and have faster startup (boot up) times that gives them an advantage to VMs in handling fluctuating video streaming demands. In addition, Docker containers are used in video packaging, handling arriving streaming requests (known as request ingestion), and inserting advertisements within/between video streams. Sprocket [10] is a serverless system implemented based on AWS Lambda [133] and enables developers to program a series of operations over video content in a modular

and extensible manner. Programmers implement custom operations, ranging from simple video transformations to more complex computer vision tasks, and construct custom video processing pipelines using a pipeline specification language. Sprocket then manages the underlying access, encoding and decoding, and processing of video and image content across operations in a parallel manner. Another advantage of deploying serverless (aka function-as-a-service) computing paradigm, such as AWS Lambda, for video streaming is to relieve stream providers from scheduling, load balancing, resource provisioning, and resource monitoring challenges.

Apart from the type of computing platform (*e.g.,* VM-based and container-based) for video streaming, the type of machines provisioned to process streaming tasks are also influential in the latency and incurred cost of streaming. For instance, clouds offer various VM types with diverse configurations (*e.g.,* GPU base, CPU base, IO base, and Memory base), or various reservation types (*e.g.,* on-demand, spot, and advance reservation).

1.4.3 Cloud-based Video Transcoding

To cover viewers with heterogeneous display devices that work with diverse codecs, resolutions, bit-rates, and frame rates, video contents usually have to be transcoded to several formats. Video transcoding process takes major computing power and time in the whole video streaming workflow [80]. The Methods and challenges for video transcoding have been studied by Vetro *et al.* [131] and Ahmad *et al.* [3]. In the past, streaming service providers had to maintain large computing systems (*i.e.,* in-house datacenters) to achieve video transcoding. However, due to the update and maintenance costs of in-house datacenters, many streaming service providers have chosen to outsource their transcoding operations to cloud servers [80, 84, 83, 82, 32]. Extensive computational demand of video transcoding can potentially impose a significant cost to stream providers. As such, it is important that stream providers apply proactive methods to transcode videos in their repositories.

A taxonomy of various cloud-based video transcoding is shown in Figure 1.7. Cloud-based solutions for transcoding VODs are either based on creating several versions of the original video in advance (aka pre-transcoding [68, 65, 12, 64, 88]) or transcoding videos upon viewer's request (aka on-demand transcoding [83, 81, 86]). In considering the long-tail access pattern to video streams on the Internet has made the on-demand approach an attractive option for stream providers. However, pure on-demand transcoding approach can increase latency and even the incurred cost for stream providers [32]. Therefore, approaches have been introduced to perform pre-transcoding in a more granular level. That is, pre-transcoding only parts of a video streams (*e.g.,* few important GOPs). This approach is known as partial pre-transcoding [33].

According to Darwich *et al.* [33], partial pre-transcoding can be carried out either in a deterministic or non-deterministic manner. Considering the fact that the beginning of video streams are generally watched more often [97], in the deterministic approach, a number of GOPs from the beginning of the video are pre-transcoded, and transcode the rest in an on-demand manner [32]. Alternatively, in the non-deterministic approach [33], pre-transcoding is not limited to the beginning of video streams and can be performed on any popular GOP, disregarding its position in the stream. Although the non-deterministic approach is proven to be more efficient, it imposes the overhead of maintaining and processing view metadata for each GOP.

Procuring various video formats in live streaming can be achieved by camera, upon

video production (known as source transcoding). However, due to the high latency and inefficiency of source transcoding, live videos are commonly transcoded in the run time [82, 74, 122].

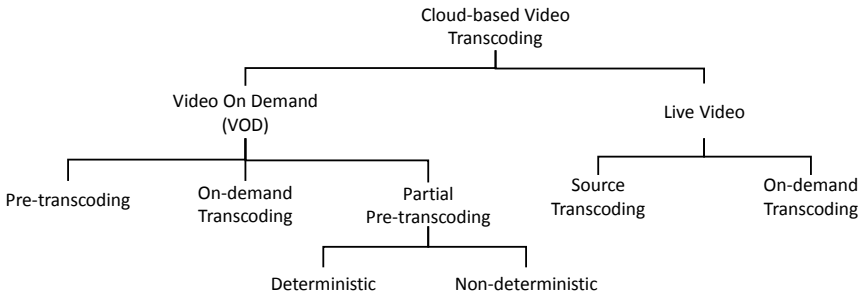


FIGURE 1.7 Approaches to perform video transcoding using cloud services.

Research have been focused on video segmentation [68, 65], load balancing [12, 88], and resource provisioning [12, 64], with the goal of maximizing throughput of the pre-transcoding operation. Alternatively, in on-demand transcoding, the goal is to respect QoE of viewers in form of minimizing the response time of the transcoding operation. More specifically, each segment (GOP) of the video needs to be transcoded before its presentation time (aka deadline).

To use cloud services efficiently, it is crucial to realize the performance difference among video transcoding operations on different VM types. Li *et al.* [80] have evaluated the performance of various transcoding operations on heterogeneous VMs to investigate the key factors on transcoding processing time. They identified the affinity (*i.e.,* match) of various VM types (*e.g.,* GPU versus CPU and Memory-based) with different transcoding operations. They show that GOP frame numbers is the most influential factor on transcoding time. In fact, frame numbers in a GOP can imply its content type. Videos with fast-motion content include numerous GOPs with few frames and vice versa. The execution time of each small GOP is usually low, and therefore it can be transcoded on cost-efficient VM types without compromising performance. In addition, Li *et al.* provide a suitability model of heterogeneous VM types for various transcoding operations in consideration of both performance and cost factors. Transcoding time estimation is needed to improve the quality of scheduling and VM provisioning on the cloud. Deneke *et al.* [34] propose a machine learning model to predict the transcoding time based on the video characteristics (*e.g.,* resolution, frame rate, and bit rate).

To further cut the cost of utilizing cloud services for on-demand video transcoding, Li *et al.* [83, 84] propose Cloud-based Video Streaming Engine (CVSE) that operates at the video repository level and pre-transcodes only hot videos while re-transcodes rarely accessed videos upon request. Li *et al.* define desired QoE of streaming as minimizing video streaming startup delay via prioritizing the beginning part of each video. The engine is able to provide low startup delay and playback jitter with proper scheduling and resource provisioning policy.

Transcoding video in an on-demand way does reduce the storage cost, however, it incurs computing cost. How to balance these two operations to minimize the incurred cost is another challenge. Jokhio *et al.* [63] presents a trade-off method to balance the

computation and storage cost for cloud-based video transcoding. The method is mainly based on the time and frequency for a given video to be stored and re-transcoded, respectively. Compared to Jokhio's work, Zhao *et al.* [145] also take the popularity of the video into consideration. Darwich *et al.* [33] propose a method to partially pre-transcode video streams depending on their degree of hotness. For that purpose, they define a method based on the hotness of each GOP within the video.

Barais *et al.* [15] propose a Microservice architecture to transcode videos at a lower cost. They treat each module (*e.g.,* splitting, scheduling, transcoding, and merging) of transcoding as a separate service, and running them on different dockers. To reduce the computational cost of on-demand transcoding, Denninnart *et al.* [35] propose to aggregate identical or similar streaming micro-services. Identical streaming micro-services appear when two or more viewers stream the same video with the same configurations. Alternatively, similar streaming micro-services appear when viewers stream the same video but with different configurations or even different operations. An example of identical micro-services can be when two or more viewers stream the same video for the same type of device (*e.g.,* smart-phone). However, when the viewers stream the same video on distinct devices (*e.g.,* smart-phone versus TV) that have different resolution characteristics, the video has to be processed to create two different resolutions, which creates micro-services with various configurations. Interestingly, during peak times the method becomes more efficient because it is more likely to find similarity between micro-services.

Cloud-based video transcoding has also been widely used in live streaming [82, 74]. Lai *et al.* [74] propose a cloud-assisted real-time transcoding mechanism based on HLS protocol, it can analyze the online quality between client and server without changing the HLS server architecture, and provides the good media quality. Timmerer *et al.* [127] present a live transcoding and streaming-as-a-service architecture by utilizing cloud infrastructure, it is able to take the live video streams as input and output multiple stream versions based on the MPEG-DASH [122] standard.

1.4.4 Cloud-based Video Packaging

Video packaging is computationally lighter than video transcoding, therefore, it is beneficial and more feasible to process them in an on-the-fly by utilizing cloud services (VMs or containers) [99]. For VOD streaming, video contents are usually pre-transcoded to different renditions (formats), then each rendition is packaged into various versions to meet different streaming protocol requirements (*e.g.,* DASH, HLS, Smooth).

Instead of statically packaging transcoded videos into different protocol renditions and store them in the repository (*i.e.,* pre-packaging), dynamic video packaging packages video segments based on the device's supported protocols. The whole process only takes milliseconds [48], and viewers do not generally notice it, however, it saves a significant storage cost for video streaming providers.

Due to the lightweight nature of the packaging process, container services are generally used for their implementation in cloud data centers.

1.4.5 Video Streaming Latency and Cloud-based Delivery Networks

Clouds are known for their centrality and high latency communication to users [57]. To reduce the network latency and to reduce the load of requests from the servers, video

contents are normally cached in the Content Delivery Networks (CDN) such as Akamai. The CDNs are distributed and located geographically close to viewers.

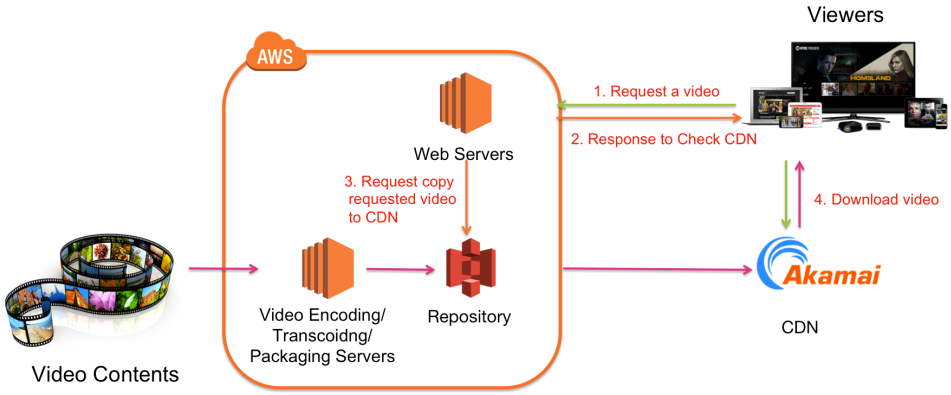


FIGURE 1.8 Workflow of actions taken place for low latency video streaming

A workflow of the whole video streaming process is shown in Figure 1.8. Upon a viewer's request to stream a video, the request is first received by a web server in the cloud datacenter. If the requested video is already cached in the CDN, the web server will send back a manifest file which informs the viewer's computer about the CDN that holds the video files. Then, the viewer sends another request to the CDN and stream the video. However, if the requested video is not cached in the CDN, the web server has to process and send the content from cloud storage to the CDN. Then, the server sends a copy of the file that includes the CDN address to the viewer to start streaming from the CDN.

Content delivery networks (CDNs) that are offered in form of a cloud service are known as cloud CDN. Compared to traditional CDNs, cloud CDNs are cost-effective and offer low latency services to content providers without having their own infrastructure. The users are generally charged based on the bandwidth consumption and storage cost [25].

Hu *et al.* [55] presented an approach using the cloud CDN to minimize the cost of system operation while maintaining the service latency. Based on viewing behavior prediction, Hu *et al.* [56] investigated the community-driven video distribution problem under cloud CDN and proposed a dynamic algorithm to trade-off between the incurred monetary cost and QoS. Their results came with less operational cost while satisfying the QoS requirement.

Jin *et al.* [61] proposed a scheme that offers the service of on-demand virtual content delivery for user-generated content providers to deliver their contents to viewers. The proposed approach was developed using a hybrid cloud. Their scheme offered elasticity and privacy by using virtual content delivery service with keeping the QoE to user-generated content providers.

Li *et al.* [79] proposed an approach for partial migration of VoD using the hybrid cloud deployment. Their proposed solution allows the requests of user to be partly served based on the self-owned servers and partly used the cloud. their Proposed migration approach (active, reactive, and smart strategies) helps the hybrid cloud to save up to 30% bandwidth expenses compared to the client/server mode. Researchers at Microsoft conducted an

experiment within Microsoft public cloud CDN, Windows Azure, to demonstrate the benefits of CDN integration with the cloud. The results show a significant gain in large data download by utilizing CDN in Cloud Computing [85].

1.4.6 Cloud Storage for Video Streaming

TABLE 1.3 Comparison of different technologies for storing video stream

	Accessibility	Capacity	Scalability	Reliability	Cost	QoE
In House Storage	limited	large	no	no	high	high
Cloud Storage (Outsourcing)	high	large	yes	yes	low	low
Content Delivery Network (CDN)	high	small	no	yes	high	high
Peer 2 Peer	high	large	yes	yes	low	low
Hybrid CDN-P2P	high	small	yes	yes	low	high

The rapid growth of video streaming usage in various applications, such as e-learning, video surveillance and situational awareness, and on various forms of mobile devices (*e.g.,* smart-phones, tablets, laptops) has created the problem of *big video data* [39]. The fast growth of video contents on the Internet requires massive storage facilities. However, the current storage servers face scalability and reliability issues, in addition to the high maintenance and administration cost for storage hardware.The cloud storage services provide a solution for scalability, reliability, and fault tolerance [110]. As such, major streaming service providers (*e.g.,* Netflix and Hulu) have relied entirely on cloud storage services for their video storage demands.

Table 1.3 provides a comparison of different storage solutions for video streaming in terms of accessibility of viewers to the same video stream, available capacity (storage space), scalability, reliability, incurred cost, and QoS. It is worth noting that although CDN is not a storage solution, but it can be used to reduce the storage cost by caching temporally hot video streams. Therefore, we consider it in our comparison table.

On-demand processing of video streaming is one effective method to reduce the storage volume and cost. This is particularly important when the long-tail access pattern to video streams is considered [33]. That is, except for a small portion of video streams that are hot the rest of videos are rarely accessed. Gao *et al.* [44] propose an approach that partially pre-transcodes video contents in the Cloud. Their approach pre-transcodes the beginning segments of video stream and which are more frequently accessed, while transcoding the remaining contents video stream upon request, this results to a reduction of storage cost.They demonstrated that their approach reduces 30% of the cost compared to pre-transcoding the whole contents of video stream.

Darwich *et al.* [33] proposed a storage and cost-efficient method for cloud-based video streaming repositories based on the long-tail access patterns. They proposed both repository and video level solutions. In the video level, they consider access patterns in two cases, (A) when it follows a long-tail distribution; and (B) when the video has random (*i.e.,* non-long-tail) access pattern. They define the cost-benefit of pre-transcoding for each GOP and determine the GOPs that need to be pre-processed and the ones that should be processed in an on-demand manner.

Krishnappa *et al.* [71] proposed strategies to transcode the segments of a video stream

requested by users. To keep a minimized startup delay of video streams when applying online strategies on video, they came up with an approach to predict the next video segment that is requested by the user. They carried out their prediction approach by implementing Markov theory. Their proposed strategy results a significant reduction in the cost of using the cloud storage with high accuracy.

1.5 SUMMARY AND FUTURE RESEARCH DIRECTIONS

1.5.1 Summary

Video streaming is one of the most prominent Internet-based services and is increasingly dominating the Internet traffic. Offering high quality and uninterrupted video streaming service is a complicated process and involves divergent technologies—from video streaming production technologies to techniques for playing them on a widely variety of display devices. With the emergence of cloud services over the past decade, video stream providers predominantly have become reliant on various services offered by the clouds. In this study, *first*, we explained the workflow and all the technologies involved in streaming a video over the Internet. We provided a bird's-eye-view of how these technologies interact with each other to achieve a high quality of experience for viewers. *Second*, we provided details on each of those technologies and challenges the video streaming researchers and practitioners are encountering in those areas. *Third*, we reviewed the ways various cloud services can be leveraged to cope with the demands of video streaming services. We surveyed and categorized different research works undertaken in this area. The main application of cloud services for video streaming can be summarized as: **(A)** Cloud computational services via VM, containers, or function (*i.e.,* serverless computing) paradigms. Computational services can be used for processing video transcoding, video packaging, video encryption, and stream analytics; **(B)** Cloud network services via Cloud-based Content Delivery Networks (CDN) to reduce video streaming latency, regardless of viewers' geographical location; and **(C)** Cloud storage services to store video streaming repositories and enable persisting multiple versions of each video to support a wide range of display devices.

1.5.2 Future of Cloud-based Video Streaming Research

Although cloud services have been useful in resolving many technical challenges of video streaming, there are still areas that either remain intact or require further exploration by researchers and practitioners. In the rest of this section, we discuss several of these areas that we believe addressing them will be impactful in the future of video streaming industry. A summary of the future directions are provided in Figure 1.9.

1. **Interactive video streaming using clouds.**
   Interactive streaming is defined as to provide the ability for video stream providers to offer any form of processing, enabled by video stream providers, on the videos being streamed [54, 9]. For instance, a video stream provider in e-learning domain may need to offer video stream summarization service to its viewers. Another example can be providing a service that enables viewers to choose sub-titles of their own languages. Although some of these interactions are currently provided by video

FIGURE 1.9 Summary of the future research directions for cloud-based video streaming.

stream providers[12], there is still not a generic video streaming engine that can be extended dynamically by the video provider to offer new streaming services.
Since there is a wide variety of possible interactions that can be defined on a video stream, it is not feasible to process and store video streams in advance. Instead, they have to be processed upon the viewer's request and in a real-time manner. It is also not possible to process the video streams on viewers' thin-clients (*e.g.,* smart-phones), due to energy and compute limitations [89]. Providing such a streaming engine entails answering several researches and implementation problems including:

- How to come up with a video streaming engine that is extensible and a high-level language for users to extend the engine with their desired services and without any major programming effort?
- How does the streaming engine can accommodate a new service? That implies answering how the streaming engine can learn the execution time distribution of tasks for a user-defined interaction? How can the streaming engine provision cloud resources (*e.g.,* VMs or containers) and schedule video streaming tasks on the

12.For instance, YouTube allows viewers to choose their preferred spatial resolution

allocated resources so that the QoE for viewers is guaranteed and the minimum cost is imposed to the stream provider?

2. **Harnessing heterogeneity in cloud services to process video streams.**
   Current cloud-based video stream providers predominantly utilize homogeneous cloud services (*e.g.,* homogeneous VMs) for video stream processing [83, 82]. However, cloud providers own heterogeneous machines and offer heterogeneous cloud service [84]. For instance, they provide heterogeneous VM types, such as GPU, CPU-Optimized, Memory-Optimized, IO-Optimized in Amazon cloud [84]. The same is with heterogeneous storage services (*e.g.,* SSD and HDD storage services). Heterogeneity may also refer to the reliability of the provisioned services. For instance, cloud providers offer *spot* and *reserved* compute services that cost remarkably lower than the normal compute services [72].
   Traditionally, elastic resource provisioning methods for cloud determine *when* and *how many* resources to be provisioned. By considering heterogeneity, a third dimension emerges: *what type* of resources should be provisioned? The more specific questions that must be addressed are: how can we harness cloud services to offer video streaming with QoE considerations and with the minimum incurred the cost for the video stream provider? How can we learn the affinity of a user-defined interaction with the heterogeneous services? How can we strike a balance between the incurred cost of heterogeneous services and the performance offered? and finally, how should the heterogeneity of the allocated resources be modified (*i.e.,* reconfigured) with respect to the type and rate arriving video streaming requests?
3. **Specialized Clouds for Video Streaming.**
   So far, the idea of cloud has been in the form a centralized facility that can be used for diverse purposes. However, as clouds are evolving to be more service-oriented, their business model is shifting to specific purpose service providers. The reason being service consumers tend to purchase high-level services with certain QoE characteristics (*e.g.,* watching video and paying in a pay-as-you-watch manner) as opposed to basic services such as dealing with VMs [22] and pay on an hourly basis. Another example can be purchasing transcoding or transmuxing service with certain startup delay guaranteed.
   In addition, viewers need flexible and diverse subscription services for video streaming services they receive that is not offered under current general purpose cloud services. For instance, some viewers may prefer to pay for video content in a pay-as-you-watch manner and some others may prefer monthly flat rate subscriptions. Offering such detailed facilities entails creating cloud providers dedicated to video streaming services. Creation of such clouds introduces multiple challenges, such as possible connections with basic clouds, specific purpose VM and containers, heterogeneity of underlying hardware among many other challenges.
4. **Single streaming engine for both live- and VOD-streaming.**
   Live-streams and VODs are structurally similar and also have similar computational demands. However, processing them is not entirely the same. In particular, live-streaming tasks have a hard deadline and have to be dropped if they miss their deadline [82] whereas VOD tasks have a soft deadline and can tolerate deadline violations. Accordingly, in VOD, upon violating deadlines for a video segment, the deadlines of all segments behind it in the same stream should be updated. That is, VOD tasks have dynamic deadlines which is not the case for live streaming. In

addition, there is no historic computational information for live-streaming tasks to predict the execution time of new arriving tasks. This is not the case for VOD tasks. Based on these differences, video stream providers utilize different video streaming engines and even different resources for processing and providing the services [81, 82]. The question is that how can we have an integrated streaming engine that can accommodate both of the streaming types simultaneously on a single set of allocated resources? How will this affect the scheduling and processing time estimation of the streaming engine?

5. **Blockchain technology for video streaming.**
   The idea of blockchain is to create a secure peer-to-peer communication through a distributed ledger (database). In this system, every network node owns a copy of the blockchain, and every copy includes the full history of all recorded transactions. All nodes must agree on a transaction before it is placed in the blockchain.
   The idea has rapidly got adopted and is being extensively developed in various domains to improve traceability and auditability [22]. We envisage that this technology will have a great impact in the video streaming industry. Some of the applications can be as follows: White-listing, which means keeping a list of legitimate publishers or distributors; Identity Management: the ability to perform identity management in a decentralized manner.
   In addition, blockchain technology will grant more control and opportunity to video producers/publishers. In fact, the current video streaming industry is driven by quality expected and algorithms embedded in stream service providers. For instance, ordinary people cannot be publishers on Netflix. Even in the case of YouTube that enables ordinary users to publish content, the search and prioritization algorithms are driven by YouTube and not by the publisher. These limitations are removed in blockchain and publishers have more freedom in exposing their generated content. The secure distribution network of blockchain can be also useful for current streaming service providers (*e.g.,* Netflix). They can use the network to securely maintain multiple versions of videos near viewers and distribute them with low latency and at the meantime reduce their cloud storage costs.
6. **Reuse and approximate computing for video streaming.**
   Several video streaming viewers may request the same video under the same or dissimilar configurations (*i.e.,* processing requirements). Current scheduling methods treat each video streaming request separately. That is, for each video streaming request, they have to be processed separately, even though the same video is being streamed simultaneously for two different viewers.
   To make video streaming cost-efficient, for similar streaming requests (*e.g.,* streaming the same video for two users with different resolutions) instead of repeatedly decoding the original video and then re-encoding it, we can reuse processing and decreases the total processing time. For instance, the decoding of a video segment can be done once across all similar requests and then re-encoding is performed separately. In addition to saving cost, this can potentially reduce congestion in scheduling queues by reducing the number of tasks waiting for processing and shortening the overall execution time. While this approach is interesting and there are preliminary efforts to address that (*e.g.,* [35]), it can be challenging as integrating processing of the same video can jeopardize other video streaming tasks to be starved and viewers' QoE is impacted.
   Research studies are needed to be undertaken and address this challenge. One question can be on how to integrate video streaming tasks from different viewers to make a

more efficient use of cloud services? How does this approach impact other video streaming tasks? Also, some video streaming tasks might be semantically similar and with some approximation, the result of processing for one request can be used for another request. For instance, two resolutions can be close and compatible. Then, the question is how to identify semantically similar streaming tasks in the system?

7. **Machine learning for video processing.**
   Video encoding/decoding requires multiple predictions, *e.g.,* intra prediction and inter prediction [143]. Machine learning can play an important role to keep improving these predictions to produce smaller size video with the identical quality.
   An accurate task execution time estimation can significantly benefit task scheduling and resource provisioning in cloud [80, 111]. However, predicting the time is not effortless. It is proven that there is an affinity between GOP size and certain video stream processing tasks (*e.g.,* transcoding time). However, better estimation can be achieved by using machine learning approaches.
   Deneke *et al.* [34] utilize a machine learning technique to predict each video segment's transcoding time before scheduling. It shows significantly better load balancing than classical methods. Accordingly, one future research direction in cloud-based video streaming is to use the machine learning techniques to enable unsupervised learning of video streaming tasks for user-defined interactions. The estimation can be different for heterogeneous cloud VMs because various video processing tasks have a different affinity with the heterogeneous VMs.
8. **Reliable cloud-based video stream processing**
   Reliability of a video streaming service is based on its tolerance against possible failures. Streaming service providers receive services under a certain Service Level Agreement (SLA) with the cloud service provider. SLA explains the availability of services and the latency of accessing them. A video streaming engine translates the SLA terms to its Service Level Objectives (SLO) [90] and attempts to respect them even in the presence of failures. Failures can be of two types: cloud service (*e.g.,* VM or container) failures, and video streaming tasks failures.

   - **Cloud service fault tolerance.** Cloud service (*e.g.,* VM) availability is vital for streaming service providers. To maintain good availability, when one server fails, its workloads need to be migrated to another server to keep the streaming service uninterrupted. Service fault tolerance has been widely studied in cloud computing and solutions for that mainly include redundancy of cloud services and data checkpointing [95, 13].
   - **Video processing fault tolerance.** Some video streaming tasks can fail during processing. Video streaming engines should include policies to cope with the failure of video streaming tasks dispatching to the scheduling queue. The policies can re-dispatch the failed task for VOD streams or ignore it for live-streaming.

   Currently, there is no failure-aware solution tailored for video streaming processing. Given the specific characteristics of video streaming services, in terms of large data-size, expensive computation, and unique QoE expectations, it will be appropriate to investigate failure-aware solutions for reliable video streaming service.
9. **Federation of edge clouds for low-latency video streaming.**
   To efficiently serve customer demands around the world, cloud service providers setup datacenters in various geographical locations [112, 8]. For example, Netflix utilizes Open Connect [19] in numerous geographical locations to minimize latency of video

streaming.
However, existing cloud-based video streaming systems do not fully take advantage of this large distributed system to improve quality and cost of streaming. Mechanisms and policies are required to dynamically coordinate load distribution between the geographically distributed data centers and determine the optimal datacenter to provide streaming service for each video (*e.g.,* for storage, processing, or delivery).
To address this problem, Buyya *et al.* [112] advocate the idea of creating the federation of cloud environments. In the context of video streaming, a cost-efficient and low latency streaming can be achieved by federating edge datacenters and take advantage of cached contents or processing power of neighboring edge datacenters. Specifically, solutions are required to stream a video not only from the nearest datacenter (the way CDNs conventionally operate), but also from neighboring edge datacenters. Such solutions should consider the trade-off between the cost of processing a requested video on a local edge datacenter, and getting that from a nearby edge.

Bibliography

[1] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *Proceedings of the IEEE International Conference on Computer Communications*, INFOCOM '12, pages 1620–1628, 2012.

[2] Michael M Afergan, F Thomson Leighton, and Jay G Parikh. Hybrid content delivery network (cdn) and peer-to-peer (p2p) network, Dec. 2012. US Patent 8,332,484.

[3] Ishfaq Ahmad, Xiaohui Wei, Yu Sun, and Ya-Qin Zhang. Video transcoding: an overview of various techniques and research issues. *IEEE Transactions on Multimedia*, 7(5):793–804, Oct. 2005.

[4] Shakeel Ahmad, Christos Bouras, Eliya Buyukkaya, Muneeb Dawood, Raouf Hamzaoui, Vaggelis Kapoulas, Andreas Papazois, and Gwendal Simon. Peer-to-peer live video streaming with rateless codes for massively multiplayer online games. *Peer-to-Peer Networking and Applications*, 11(1):44–62, 2018.

[5] Abubakr O Al-Abbasi and Vaneet Aggarwal. Edgecache: An optimized algorithm for cdn-based over-the-top video streaming services. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 202–207. IEEE, 2018.

[6] Ekbal M. Al-Hurani and Hussein R. Al-Zoubi. Mitigation of dos attacks on video trafficin wireless networks for better qos. In *Proceedings of the 8th International Conference on Computer Modeling and Simulation*, ICCMS '17, pages 166–169, 2017.

[7] Mohammad A. Alsmirat, Islam Obaidat, Yaser Jararweh, and Mohammed Al-Saleh. A security framework for cloud-based video surveillance system. *Multimedia Tools and Applications*, 76(21):22787–22802, Nov. 2017.

[8] Mohsen Amini Salehi, Bahman Javadi, and Rajkumar Buyya. Preemption-aware admission control in a virtualized grid federation. In *Proceedings of the 26th IEEE*

International Conference on Advanced Information Networking and Applications, AINA '12, pages 854–861, 2012.

[9] Mohsen Amini Salehi and Xiangbo Li. HLSaaS: High-Level Live Video Streaming as a Service. In *Stream2016 Workshop organized by Department of Energy*, Mar. 2016.

[10] Lixiang Ao, Liz Izhikevich, Geoffrey M. Voelker, and George Porter. Sprocket: A serverless video processing framework. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, pages 263–274, 2018.

[11] John Apostolopoulos, Tina Wong, Wai-tian Tan, and Susie Wee. On multiple description streaming with content delivery networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1736–1745, 2002.

[12] Adnan Ashraf, Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, Ivan Porres, and Johan Lilius. Stream-based admission control and scheduling for video transcoding in cloud computing. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid '13, pages 482–489, May 2013.

[13] Anju Bala and Inderveer Chana. Fault tolerance-challenges, techniques and implementation in cloud computing. *International Journal of Scientific and Research Publications (IJSRP)*, 9(1):1694–0814, June 2012.

[14] Selim Balcisoy, Marta Karczewicz, and Tolga Capin. Progressive downloading of timed multimedia content, June 9 2004. US Patent App. 10/865,670.

[15] Olivier Barais, Johann Bourcier, Yérom-David Bromberg, and Christophe Dion. Towards microservices architecture to transcode videos in the large at low costs. In *Proceedings of the International Conference on Telecommunications and Multimedia*, TEMU '16, pages 1–6, 2016.

[16] Ali Begen, Tankut Akgul, and Mark Baugher. Watching video over the web: Part 1: Streaming protocols. *IEEE Internet Computing*, 15(2):54–63, 2011.

[17] Ilias Benkacem, Tarik Taleb, Miloud Bagaa, and Hannu Flinck. Performance benchmark of transcoding as a virtual network function in cdn as a service slicing. In *Proceedings of the IEEE Conference on Wireless Communications and Networking (WCNC).*, 2018.

[18] Niklas Bjork and Charilaos Christopoulos. Transcoder architectures for video coding. *IEEE Transactions on Consumer Electronics*, 44(1):88–98, Feb. 1998.

[19] Timm Böttger, Felix Cuadrado, Gareth Tyson, Ignacio Castro, and Steve Uhlig. Open connect everywhere: A glimpse at the internet ecosystem through the lens of the netflix cdn. *ACM SIGCOMM Computer Communication Review*, 48(1):28–34, Apr. 2018.

[20] Nassima Bouzakaria, Cyril Concolato, and Jean Le Feuvre. Overhead and performance of low latency live streaming using mpeg-dash. In *Proceedings of the 5th International Conference on Information, Intelligence, Systems and Applications*, pages 92–97, 2014.

[21] Rajkumar Buyya, Mukaddim Pathan, and Athena Vakali. *Content Delivery Networks*, volume 9. Springer Science & Business Media, 2008.

[22] Rajkumar Buyya, Satish Narayana Srirama, Giuliano Casale, Rodrigo Calheiros, Yogesh Simmhan, Blesson Varghese, Erol Gelenbe, Bahman Javadi, Luis Miguel Vaquero, Marco AS Netto, et al. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Computing Survey*, Aug. 2018.

[23] Paula Carrillo, Hari Kalva, and Spyros Magliveras. Compression independent object encryption for ensuring privacy in video surveillance. In *proceedings of the IEEE International Conference on Multimedia and Expo, 2008*, pages 273–276, 2008.

[24] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[25] Fangfei Chen, Katherine Guo, John Lin, and Thomas La Porta. Intra-cloud lightning: Building cdns in the cloud. In *Proceedings of the IEEE Conference INFOCOM*, pages 433–441. IEEE, 2012.

[26] Hao-Nong Chen, Michael Rutman, Charles Duncan MacLean, Edward Charles Hiar, and Glenn A Morten. Progressive download or streaming of digital media securely through a localized container and communication protocol proxy, August 14 2012. US Patent 8,243,924.

[27] Bin Cheng, Hai Jin, and Xiaofei Liao. Supporting vcr functions in p2p vod services using ring-assisted overlays. In *Proceedings of the IEEE International Conference on Communications*, ICC '07, pages 1698–1703, 2007.

[28] Xu Cheng, Jiangchuan Liu, and Cameron Dale. Understanding the characteristics of internet short video sharing: A youtube-based measurement study. *IEEE Transactions on Multimedia*, 15(5):1184–1194, Aug. 2013.

[29] Yang-hua Chu, Sanjay G Rao, and Hui Zhang. A case for end system multicast (keynote address). In *ACM SIGMETRICS Performance Evaluation Review*, volume 28, pages 1–12. ACM, 2000.

[30] Robert T Collins, Alan J Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, et al. A system for video surveillance and monitoring. *VSAM final report*, pages 1–68, 2000.

[31] Charles D Cranor, Matthew Green, Chuck Kalmanek, David Shur, Sandeep Sibal, Jacobus E Van der Merwe, and Cormac J Sreenan. Enhanced streaming services in a content distribution network. *IEEE Internet Computing*, 5(4):66–75, 2001.

[32] M. Darwich, E. Beyazit, M. A. Salehi, and M. Bayoumi. Cost efficient repository management for cloud-based on-demand video streaming. In *Proceedings of 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 39–44, April 2017.

[33] Mahmoud Darwich, Mohsen Amini Salehi, Ege Beyazit, and Magdy Bayoumi. Cost efficient cloud-based video streaming based on quantifying video stream hotness. *The Computer Journal*, pages 1085–1092, June 2018.

[34] Tewodors Deneke, Habtegebreil Haile, Sébastien Lafond, and Johan Lilius. Video transcoding time prediction for proactive load balancing. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, ICME '14, pages 1–6.

[35] Chavit Denninnart, Mohsen Amini Salehi, Adel N. Toosi, and Xiangbo Li. Leveraging computational reuse for cost- and qos-efficient task scheduling in clouds. In *Proceedings of the 16th International Conference on Service-Oriented Computing*, ICSOC '18, Nov. 2018.

[36] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM Computer Communication Review*, volume 41, pages 362–373. ACM, 2011.

[37] David Kimbal Dorwin. Application-driven playback of offline encrypted content with unaware drm module, August 18 2015. US Patent 9,110,902.

[38] Frederic Dufaux and Touradj Ebrahimi. Scrambling for privacy protection in video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1168–1174, 2008.

[39] J. Edstrom, D. Chen, Y. Gong, J. Wang, and N. Gong. Data-pattern enabled self-recovery low-power storage system for big video data. *IEEE Transactions on Big Data*, Sep. 2018.

[40] Gokhan Erdemir, Osman Selvi, Veysi Ertekin, and Gökhan Eşgi. Project PISCES: Developing an in-flight entertainment system for smart devices. 2017.

[41] Gábor Fehér. Enhancing wireless video streaming using lightweight approximate authentication. In *Proceedings of the 2Nd ACM International Workshop on Quality of Service & Security for Wireless and Mobile Networks*, Q2SWinet '06, pages 9–16, 2006.

[42] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

[43] Common Media Application Format. https://mpeg.chiariglione.org/standards/mpeg-a/common-media-application-format/text-isoiec-cd-23000-19-common-media-application. Jun. 2016.

[44] G. Gao, W. Zhang, Y. Wen, Z. Wang, and W. Zhu. Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns. *IEEE Transactions on Multimedia*, 17(8):1286–1296, Aug 2015.

[45] Sumeer Goel, Yasser Ismail, and Magdy Bayoumi. High-speed motion estimation architecture for real-time video transmission. *The Computer Journal*, 55(1):35–46, Apr. 2012.

[46] Mahya Mohammadi Golchi and Homayun Motameni. Evaluation of the improved particle swarm optimization algorithm efficiency inward peer to peer video streaming. *Computer Networks*, 2018.

[47] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13, 2016.

[48] Josu Gorostegui, Angel Martin, Mikel Zorrilla, Iñaki Alvaro, and Jon Montalban. Broadcast delivery system for broadband media content. In *Proceedings of the IEEE International Symposium onBroadband Multimedia Systems and Broadcasting*, BMSB '17, pages 1–9, 2017.

[49] Giovanni Gualdi, Andrea Prati, and Rita Cucchiara. Video streaming for mobile video surveillance. *IEEE Transactions on Multimedia*, 10(6):1142–1154, 2008.

[50] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2cast: peer-to-peer patching scheme for vod service. In *Proceedings of the 12th international conference on World Wide Web*, pages 301–309. ACM, 2003.

[51] C Hanna, D Gillies, E Cochon, A Dorner, J Alred, and M Hinkle. Demultiplexer ic for mpeg2 transport streams. *IEEE Transactions on Consumer Electronics*, 41(3):699–706, 1995.

[52] Barry G Haskell, Atul Puri, and Arun N Netravali. *Digital video: an introduction to MPEG-2*. Springer Science & Business Media, 1996.

[53] Jian He, Di Wu, Yupeng Zeng, Xiaojun Hei, and Yonggang Wen. Toward optimal deployment of cloud-assisted video distribution services. *IEEE transactions on circuits and systems for video technology*, 23(10):1717–1728, 2013.

[54] Matin Hosseini, Mohsen Amini Salehi, and Raju Gottumukkala. Enabling interactive video stream prioritization for public safety monitoring through effective batch scheduling. In *Proceedings of the 19th IEEE International Conference on High Performance Computing and Communications*, HPCC '17, Dec. 2017.

[55] Han Hu, Yonggang Wen, Tat-Seng Chua, Jian Huang, Wenwu Zhu, and Xuelong Li. Joint content replication and request routing for social video distribution over cloud cdn: A community clustering method. *IEEE transactions on circuits and systems for video technology*, 26(7):1320–1333, 2016.

[56] Han Hu, Yonggang Wen, Tat-Seng Chua, Zhi Wang, Jian Huang, Wenwu Zhu, and Di Wu. Community based effective social video contents placement in cloud centric cdn network. In *Proceeding of the IEEE International Conference on Multimedia and Expo*, ICME '14, pages 1–6, 2014.

[57] Razin Hussain, Mohsen Amini Salehi, Anna Kovalenko, Omid Semiari, and Saeed Salehi. Robust resource allocation using edge computing for smart oil field. In *Proceedings of the 24th International Conference on Parallel and Distributed Processing Techniques and Applications*, PDPTA '18, pages 495–503, July 2018.

[58] Van Jacobson, Ron Frederick, Steve Casner, and H Schulzrinne. Rtp: A transport protocol for real-time applications. 2003.

[59] Shraboni Jana, Eilwoo Baik, Amit Pande, and Prasant Mohapatra. Improving mobile video telephony. In *Proceedings of the 11th Annual IEEE International Conference on Sensing, Communication, and Networking*, SECON '14, pages 495–503, 2014.

[60] John Jannotti, David K Gifford, Kirk L Johnson, M Frans Kaashoek, et al. Overcast: reliable multicasting with on overlay network. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation-Volume 4*, page 14. USENIX Association, 2000.

[61] Yichao Jin, Yonggang Wen, Guangyu Shi, Guoqiang Wang, and Athanasios V Vasilakos. Codaas: An experimental cloud-centric content delivery platform for user-generated contents. In *Proceedings of the International Conference on Computing, Networking and Communications*, ICNC '12, pages 934–938. IEEE, 2012.

[62] Jinyong Jo and JongWon Kim. Synchronized one-to-many media streaming with adaptive playout control. In *proceceeding of the International Society for Optics and Photonics on Multimedia Systems and Applications V*, volume 4861, pages 71–83, 2002.

[63] Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, and Johan Lilius. A computation and storage trade-off strategy for cost-efficient video transcoding in the cloud. In *Proceedings of the 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 365–372, Sept. 2013.

[64] Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Prediction-based dynamic resource allocation for video transcoding in cloud computing. In *Proceedings of the 21st IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 254–261, Feb. 2013.

[65] Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, and Johan Lilius. Analysis of video segmentation for spatial resolution reduction video transcoding. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, pages 1–6, Dec. 2011.

[66] Christine E Jones, Krishna M Sivalingam, Prathima Agrawal, and Jyh Cheng Chen. A survey of energy efficient network protocols for wireless networks. *wireless networks*, 7(4):343–358, 2001.

[67] Hyunchul Joo, Hwangjun Song, Dai-Boong Lee, and Inkyu Lee. An effective iptv channel control algorithm considering channel zapping time and network utilization. *IEEE Transactions on broadcasting*, 54(2):208–216, 2008.

[68] Myoungjin Kim, Yun Cui, Seungho Han, and Hanku Lee. Towards efficient design and implementation of a hadoop-based distributed video transcoding system in cloud computing environment. *International Journal of Multimedia and Ubiquitous Engineering*, 8(2):213–224, Mar. 2013.

[69] Leonidas Kontothanassis, Ramesh Sitaraman, Joel Wein, Duke Hong, Robert Kleinberg, Brian Mancuso, David Shaw, and Daniel Stodolsky. A transport layer for live streaming in a content delivery network. *Proceedings of the IEEE*, 92(9):1408–1419, 2004.

[70] S Shunmuga Krishnan and Ramesh K Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking*, 21(6):2001–2014, 2013.

[71] Dilip Kumar Krishnappa, Michael Zink, and Ramesh K Sitaraman. Optimizing the video transcoding workflow in content delivery networks. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 37–48. ACM, 2015.

[72] Dinesh Kumar, Gaurav Baranwal, Zahid Raza, and Deo Prakash Vidyarthi. A survey on spot pricing in cloud computing. *Journal of Network and Systems Management*, 26(4):809–856, Oct 2018.

[73] Kapil Kumar. Drm on android. In *Proceedings of the Annual IEEE India Conference*, INDICON '17, pages 1–6, 2013.

[74] Chin-Feng Lai, Han-Chieh Chao, Ying-Xun Lai, and Jiafu Wan. Cloud-assisted real-time transrating for http live streaming. *IEEE Wireless Communications*, 20(3):62–70, 2013.

[75] Feng Lao, Xinggong Zhang, and Zongming Guo. Parallelizing video transcoding using map-reduce-based cloud computing. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 2905–2908, May 2012.

[76] Eetu Latja. *Parallel Acceleration of H.265 Video Processing*. PhD thesis, Aalto University, 2017.

[77] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[78] Brian Lesser. *Programming flash communication server*. " O'Reilly Media, Inc.", 2005.

[79] Haitao Li, Lili Zhong, Jiangchuan Liu, Bo Li, and Ke Xu. Cost-effective partial migration of vod services to content clouds. In *Proceedings of the 2011 IEEE International Conference on Cloud Computing (CLOUD)*, pages 203–210, 2011.

[80] Xiangbo Li, Mohsen Amini Salehi, Yamini Joshi, Mahmoud Darwich, Brad Landreneau, and Magdi Bayoumi. Performance analysis and modelling of video stream transcoding using heterogeneous cloud services. *Accepted in IEEE Transactions on Parallel and Distributed Systems (TPDS)*, Sep. 2018.

[81] Xiangbo Li, Mohsen Amini Salehi, and Magdy Bayoumi. High perform on-demand video transcoding using cloud services. In *Proceedings of the 16th ACM/IEEE International Conference on Cluster Cloud and Grid Computing*, CCGrid '16, May.

[82] Xiangbo Li, Mohsen Amini Salehi, and Magdy Bayoumi. VLSC:Video Live Streaming Using Cloud Services. In *Proceedings of the 6th IEEE International Conference on Big Data and Cloud Computing Conference*, BDCloud '16, Oct. 2016.

[83] Xiangbo Li, Mohsen Amini Salehi, Magdy Bayoumi, and Rajkumar Buyya. CVSS: A Cost-Efficient and QoS-Aware Video Streaming Using Cloud Services. In *Proceedings of the 16th ACM/IEEE International Conference on Cluster Cloud and Grid Computing*, CCGrid '16, May 2016.

[84] Xiangbo Li, Mohsen Amini Salehi, Magdy Bayoumi, Nian-Feng Tzeng, and Rajkumar Buyya. Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 29(3):556–571, 2018.

[85] Yale Li, Yushi Shen, and Yudong Liu. Utilizing content delivery network in cloud computing. In *Proceeding of the IEEE International Conference on Computational Problem-Solving (ICCP)*, pages 137–143, 2012.

[86] Zhenhua Li, Yan Huang, Gang Liu, Fuchen Wang, Zhi-Li Zhang, and Yafei Dai. Cloud transcoder: Bridging the format and resolution gap between internet videos and mobile devices. In *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, pages 33–38, June 2012.

[87] Chia-Wen Lin, Yung-Chang Chen, and Ming-Ting Sun. Dynamic region of interest transcoding for multipoint video conferencing. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(10):982–992, 2003.

[88] Song Lin, Xinfeng Zhang, Qin Yu, Honggang Qi, and Siwei Ma. Parallelizing video transcoding with load balancing on cloud computing. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2864–2867, May 2013.

[89] Yuhua Lin and Haiying Shen. Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of service. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 28(2):431–445, Feb. 2017.

[90] Guoxin Liu, Haiying Shen, and Haoyu Wang. An economical and slo-guaranteed cloud storage service across multiple cloud service providers. *IEEE Transactions on Parallel and Distributed Systems*, 28(9):2440–2453, 2017.

[91] Qiong Liu, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. Digital rights management for content distribution. In *Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003 - Volume 21*, ACSW Frontiers '03, pages 49–58, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[92] Yong Liu, Yang Guo, and Chao Liang. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications*, 1(1):18–28, 2008.

[93] Zhi Hui Lu, Xiao Hong Gao, Si Jia Huang, and Yi Huang. Scalable and reliable live streaming service through coordinating cdn and p2p. In *Proceedings of the IEEE 17th International Conference on Parallel and Distributed Systems*, ICPADS '11, pages 581–588, 2011.

[94] Nazanin Magharei, Reza Rejaie, and Yang Guo. Mesh or multiple-tree: A comparative study of live p2p streaming approaches. In *Proceedings of the 26th IEEE International Conference on Computer Communications*, INFOCOM '07, pages 1424–1432, 2007.

[95] Sheheryar Malik and Fabrice Huet. Adaptive fault tolerance in real time cloud computing. In *Proceedings of the 2011 IEEE World Congress on Services*, SERVICES '11, pages 280–287, Jul. 2011.

[96] Zhourong Miao and A. Ortega. Scalable proxy caching of video under storage constraints. *IEEE Journal on Selected Areas in Communications*, 20(7):1315–1327, Sep 2002.

[97] Lucas CO Miranda, Rodrygo LT Santos, and Alberto HF Laender. Characterizing video access patterns in mainstream media portals. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1085–1092. ACM, 2013.

[98] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. The latest open-source video codec vp9-an overview and preliminary results. In *Picture Coding Symposium (PCS), 2013*, pages 390–393. IEEE, 2013.

[99] Robert Linwood Myers, Parasuram Ranganathan, Ivan Chvets, and Krzysztof Pakulski. Methods and systems for real-time transmuxing of streaming media content, July 18 2017. US Patent 9,712,887.

[100] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[101] Netflix and AWS. https://aws.amazon.com/solutions/case-studies/netflix/. Accessed May URL:.

[102] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

[103] Sam Newman. *Building microservices: designing fine-grained systems*. O'Reilly Media, Inc., 2015.

[104] MPEG-2 Transport of Compressed Motion Imagery and Metadata. http://www.gwg.nga.mil/misb/docs/standards/st1402.pdf. Feb. 2014.

[105] Venkata N Padmanabhan and Jeffrey C Mogul. Improving http latency. *Computer Networks and ISDN Systems*, 28(1-2):25–35, 1995.

[106] Roger Pantos and William May. Http live streaming. 2017.

[107] F. F. Al Quayed and S. S. Zaghloul. Analysis and evaluation of internet protocol television (iptv). In *Proceedings of the Third International Conference on e-Technologies and Networks for Development (ICeND2014)*, pages 162–164, April 2014.

[108] Naeem Ramzan, Hyunggon Park, and Ebroul Izquierdo. Video streaming over p2p networks: Challenges and opportunities. *Signal Processing: Image Communication*, 27(5):401–411, 2012.

[109] Global Internet Phenomena Report. https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf. Oct. 2018.

[110] D. A. Rodriguez-Silva, L. Adkinson-Orellana, F. J. Gonz'lez-Castao, I. Armio-Franco, and D. Gonz'lez-Martnez. Video surveillance based on cloud storage. pages 991–992, June 2012.

[111] Mohsen Salehi and Rajkumar Buyya. Adapting market-oriented scheduling policies for cloud computing. In *Proceedings of the Algorithms and Architectures for Parallel Processing*, volume 6081, pages 351–362. Jan. 2010.

[112] Mohsen Amini Salehi, Bahman Javadi, and Rajkumar Buyya. Qos and preemption aware scheduling in federated and virtualized grid computing environments. *Journal of Parallel and Distributed Computing (JPDC)'*, 72(2):231 – 245, 2012.

[113] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. *SIGOPS Oper. Syst. Rev.*, 36(SI):315–327, December 2002.

[114] Henning Schulzrinne. Real time streaming protocol (rtsp). 1998.

[115] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *IEEE Transactions on circuits and systems for video technology*, 17(9):1103–1120, 2007.

[116] Prashant J Shenoy and Harrick M Vin. Efficient striping techniques for variable bit rate continuous media file servers. *Performance Evaluation*, 38(3):175–199, 1999.

[117] Martin Smole. Live-to-VoD Streaming. https://bitmovin.com/bitmovins-live-vod-service/, Feb. 2017.

[118] Thomas Stockhammer. Dynamic adaptive streaming over http–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011.

[119] FairPlay Streaming. https://developer.apple.com/streaming/fps/. Accessed May URL:.

[120] George Suciu, Muneeb Anwar, and Roxana Mihalcioiu. Virtualized video and cloud computing for efficient elearning. In *Proceedings of The International Scientific Conference eLearning and Software for Education*, volume 2, page 205, 2017.

[121] Gary J Sullivan, J-R Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012.

[122] Truong Cong Thang, Quang-Dung Ho, Jung Won Kang, and Anh T Pham. Adaptive streaming of audiovisual content using mpeg dash. *IEEE Transactions on Consumer Electronics*, 58(1):78–85, Mar. 2012.

[123] Truong Cong Thang, Hung T Le, Anh T Pham, and Yong Man Ro. An evaluation of bitrate adaptation methods for http live streaming. *IEEE Journal on Selected Areas in Communications*, 32(4):693–705, 2014.

[124] T. Thomas, S. Emmanuel, A. V. Subramanyam, and M. S. Kankanhalli. Joint watermarking scheme for multiparty multilevel drm architecture. *IEEE Transactions on Information Forensics and Security*, 4(4):758–767, Dec 2009.

[125] Johannes Thönes. Microservices. *IEEE software*, 32(1):116–116, 2015.

[126] Christian Timmerer and Christopher Müller. Http streaming of mpeg media. *Streaming Day*, 2010.

[127] Christian Timmerer, Daniel Weinberger, Martin Smole, Reinhard Grandl, Christopher Müller, and Stefan Lederer. Live transcoding and streaming-as-a-service with mpeg-dash. In *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–4, June 2015. IEEE.

[128] Netflix Is Now Bigger Than Cable TV. https://www.forbes.com/sites/ianmorris/2017/06/13/netflix-is-now-bigger-than-cable-tv. Jun. 2017.

[129] A. Vakali and G. Pallis. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6):68–74, Nov 2003.

[130] Vidhyashankar Venkataraman, Kaouru Yoshida, and Paul Francis. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Proceedings of the 2006 14th IEEE International Conference on Network Protocols*, ICNP '06, pages 2–11, 2006.

[131] Anthony Vetro, Charilaos Christopoulos, and Huifang Sun. Video transcoding architectures and techniques: an overview. *IEEE on Signal Processing Magazine*, 20(2):18–29, Mar. 2003.

[132] Aggelos Vlavianos, Marios Iliofotou, and Michalis Faloutsos. BiToS: Enhancing BitTorrent for supporting streaming applications. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, INFOCOM '06, pages 1–6, 2006.

[133] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. Peeking behind the curtains of serverless platforms. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 133–146. USENIX Association, 2018.

[134] Susie Wee, John Apostolopoulos, Wai-tian Tan, and Sumit Roy. Research and design of a mobile streaming media content delivery network. In *Proceedings of the International Conference on Multimedia and Expo*, volume 1 of *ICME '03*, pages I–5, 2003.

[135] Oliver Werner. Requantization for transcoding of mpeg-2 intraframes. *IEEE Transactions on Image Processing*, 8:179–191, Feb. 1999.

[136] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[137] Dapeng Wu, Y. T. Hou, Wenwu Zhu, Ya-Qin Zhang, and J. M. Peha. Streaming video over the internet: approaches and directions. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):282–300, Mar 2001.

[138] Changqiao Xu, G-M Muntean, Enda Fallon, and Austin Hanley. A balanced tree-based strategy for unstructured media distribution in p2p networks. In *Proceedings of the IEEE International Conference on Communications*, ICC '08, pages 1797–1801, 2008.

[139] Dongyan Xu, Sunil Suresh Kulkarni, Catherine Rosenberg, and Heung-Keung Chai. Analysis of a cdn–p2p hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems*, 11(4):383–399, 2006.

[140] Tianyin Xu, Jianzhong Chen, Wenzhong Li, Sanglu Lu, Yang Guo, and Mounir Hamdi. Supporting vcr-like operations in derivative tree-based p2p streaming systems. In *Proceedings of the IEEE International Conference on Communications*, ICC '09, pages 1–5, 2009.

[141] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 25–34. ACM, 2009.

[142] W-P Ken Yiu, Xing Jin, and S-H Gary Chan. Vmesh: Distributed segment storage for peer-to-peer interactive video streaming. *IEEE journal on selected areas in communications*, 25(9), 2007.

[143] Rui Zhang, Shankar L Regunathan, and Kenneth Rose. Video coding with optimal inter/intra-mode switching for packet loss resilience. *IEEE Journal on Selected Areas in Communications*, 18(6):966–976, 2000.

[144] Wei Zhang, SS Cheung, and Minghua Chen. Hiding privacy information in video surveillance system. In *proceedings of the IEEE International Conference on Image Processing, ICIP 2005.*, volume 3, pages II–868, 2005.

[145] Hui Zhao, Qinghua Zheng, Weizhan Zhang, Biao Du, and Yuxuan Chen. A version-aware computation and storage trade-off strategy for multi-version VoD systems in the cloud. In *Proceedings of IEEE Symposium on Computers and Communication (ISCC)*, pages 943–948, July 2015.

[146] Zhenyun Zhuang and Chun Guo. Building cloud-ready video transcoding system for content delivery networks (cdns). In *Proceedings of the Global Communications Conference*, GLOBECOM '12, pages 2048–2053, 2012.