University of Cincinnati

Date: 7/14/2021

I, Mohammad Nazmus Sadat, hereby submit this original work as part of the requirements for the degree of Doctor of Philosophy in Computer Science & Engineering.

It is entitled:

QoE-Aware Video Communication in Emerging Network Architectures

Student's name: **Mohammad Nazmus Sadat**

This work and its defense approved by:

Committee chair: Rui Dai, Ph.D.

Committee member: H. Howard Fan, Ph.D.

Committee member: Manish Kumar, Ph.D.

Committee member: Nan Niu, Ph.D.

Committee member: Carla Purdy, Ph.D.

38276

QoE-Aware Video Communication in Emerging Network Architectures

A dissertation submitted to the
Graduate School of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

in the Department of Electrical Engineering and Computer Science
of the College of Engineering and Applied Science
July 2021
by

Mohammad Nazmus Sadat

B.Sc., Khulna University of Engineering & Technology, Bangladesh, 2013

Dissertation advisor and committee chair:
Rui Dai, Ph.D.

Abstract

The demand for video content has skyrocketed in the past decade owing to the popularity of video streaming services such as YouTube and Netflix and the expanding usage of video analytics applications such as surveillance, telemedicine, and public safety. This tremendous demand for video has necessitated providing good quality of experience (QoE) to video application users. Although the concept of QoE is not new, the developments of new network architectures and computational paradigms have led to the need to design QoE-aware video communication frameworks that can handle the new challenges. The overall goal of this dissertation is to study video quality in emerging networks from the perspectives of both human users and video analytic tools and propose new QoE-aware video communication strategies to improve video quality in both cases. Content-Centric Networking (CCN) is a future Internet architecture that has been proposed to tackle the vast amount of global video traffic, provide better scalability, and allow more efficient bandwidth usage. A key feature of CCN is ubiquitous in-network caching, where popular contents are cached near the end-user for faster content fetching. However, this caching mechanism brings new challenges in maintaining QoE for video streaming. We investigated how in-network caching influences video content distribution and video streaming among CCN nodes. Then, we conducted human subjective tests to quantify the influence of the video stalling events on the overall QoE scores. After that, we proposed a new QoE-aware multi-source video streaming algorithm for CCN that aims to suppress the stalling resulted from switching between content sources.

The exchange of video among different wireless communication entities has seen an uprise due to the popularity of wireless imaging applications and the Internet of Things (IoT) technology. Software-defined radio (SDR) is a promising technology to communicate across different wireless networking standards because of its implementation flexibility, achieved

using software rather than hardware, for signal processing tasks. We designed and implemented (both simulation and hardware) an SDR video streaming platform with integrated QoE components, which can handle both real-time and stored videos. Then we studied how several parameters from the application and physical layers influence the QoE of the received videos. Lastly, we proposed a cross-layer QoE-aware video streaming solution to maximize viewers' QoE.

Video analytics applications often require fast response and high accuracy. Performing the video analysis at the network edge can provide significantly lower latency and bandwidth consumption. In our pursuit to study video quality from the perspective of video analytics tools, we designed and implemented a video analytics platform based on edge computing. Then, we investigated the factors that influence the quality of edge computing-based video analysis as well as the factors that contribute to the overall latency both in terms of computation and communication. Next, we studied the trade-off between the video analysis quality and the latency, i.e., computation benefits at the edge vs. remote server. Finally, we proposed a quality-aware edge computing-based video analytics framework to minimize latency while guaranteeing detection accuracy.

Acknowledgements

I am indebted to my advisor, Dr. Rui Dai, for her invaluable support and guidance throughout this dissertation research. Her substantial knowledge, pragmatic insights, and constructive feedback have been immensely helpful in developing my research skills and preparing myself for the next step in my career.

I would like to sincerely thank other members of my dissertation committee. Dr. H. Howard Fan, Dr. Nan Niu, Dr. Carla Purdy, and Dr. Manish Kumar have closely supervised my dissertation research, and I appreciate their time and guidance.

I also want to thank all my Cincinnati friends whom I met on and off-campus. My life in Cincinnati would not be nearly as good without their tremendous help and support. In addition, I highly value the research collaborations with my friends from the Multimedia Networking and Computing (MNC) Lab. Friends who have been instrumental in my journey at UC include Nesrin, Hiranya, Erwin, Aishvarya, and Madhu.

My parents (Mr. Quzi Mofazzel Huq and Mrs. Kamrun Naher) have made innumerable sacrifices to provide me with all the opportunities to reach this stage in my life, and I cannot express in words how grateful I am to them. They have always supported me and encouraged me to push forward. Likewise, I thank my wife, Fariha, for her unceasing understanding and support; life as the spouse of a Ph.D. student was not easy. Furthermore, my sister, Mou, has always been my biggest supporter, and I am grateful for her presence in my life.

Lastly, I am thankful to my one-year-old son, Zavian, who, albeit made this journey a little more challenging but gave me the strength and love to persist through.

Contents

List of Figures

List of Tables

List of Abbreviations

ASDC	Adaptive Streaming with Distributed Caching
AVC	Advanced Video Coding
BTW	Betweenness Centrality
BRISQUE	Blind/Referenceless Image Spatial Quality Evaluator
CCE	Caching Everything Everywhere
CCN	Content-Centric Networking
CNN	Convolutional Neural Network
CL-SDR	Cross-Layer Video Streaming Protocol for SDR
CLT	Content-Listing Table
CoA	Cost-Aware
CS	Content Store
CUDA	Compute Unified Device Architecture
DASH	Dynamic Adaptive Streaming over HTTP
DNN	Deep Neural Network
ER	Edge Router
FIB	Forwarding Information Table
FDT	File Distribution Table

GPU	Graphics Processing Unit
HEVC	High Efficiency Video Coding
ICN	Information-Centric Networking
IoT	Internet of Things
IP	Internet Protocol
IR	Intermediate Router
LCE	Leave Copy Everywhere
LRU	Least Recently Used
LTE	Long Term Evolution
M2M	Machine-to-Machine
MEC	Multi-Access Edge Computing
MOS	Mean Opinion Score
MSE	Mean Squared Error
NDN	Named Data Network
NTP	Network Time Protocol
PC	Principal Component
PCA	Principal Component Analysis
PIT	Pending Interest Table
QoE	Quality-of-Experience
QoS	Quality-of-Service
QP	Quantization Parameter
QVAF	Quality-Aware Video Analytics Framework
RMSE	Root-Mean-Squared Error

RTMP	Real Time Messaging Protocol
SDR	Software Defined Radio
SF	Stalling Frequency
SHF	Super High Frequency
SL	Stalling Length
SNR	Signal-to-Noise Ratio
SPR	Shortest Path Routing
SQI	Streaming QoE Index
SSIM	Structural Similarity Index
SwSF	Satisfaction with Stalling Frequency
SwSL	Satisfaction with Stalling Length
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicles
UHD	USRP Hardware Driver
UHF	Ultra High Frequency
URI	Uniform Resource Identifier
USRP	Universal Software Radio Peripheral
VHF	Very High Frequency
VQA	Video Quality Assessment
VR	Virtual Reality

Chapter 1

Introduction and Overview

The global Internet traffic has been increasing exponentially over the past two decades. This rapid increase is primarily due to the ever-growing number of networked devices, which is predicted to be more than three times the world population by 2023 [1]. Video devices, in particular, can have a multiplier effect on traffic. According to the Cisco Visual Networking Index [2], Internet video will represent 82% of all business Internet traffic by 2022. The majority of this video traffic comes from two categories: end-user devices running video streaming applications and Machine-To-Machine (M2M) connections. The first category includes video streaming applications such as YouTube, Netflix, cloud gaming, and virtual reality (VR) streaming. While traditionally, the M2M traffic, also known as the Internet of Things (IoT), has been less than that from the end-user devices, the amount of traffic is growing faster than the number of connections due to the increase of M2M video application deployment [1]. Moreover, many IoT applications such as surveillance, telemedicine, industrial automation, public safety, and autonomous vehicles have integrated automatic video analytics (e.g., facial recognition, smoke and flame detection, motion detection, etc.). This enormous demand for video communication has made it crucial to provide good video quality to both human viewers and video analytics tools.

Quality of Experience (QoE) is a measure of the satisfaction or annoyance of a customer's

experience with a service. The factors that influence human viewers' QoE include stalling or video freezing, video resolution, initial delay, bitrate, and quality variations. In contrast, video analytics quality depends on the quality of input video frames, video content, compression method, computing capacity, and detector algorithms. Although the concept of QoE is not new, the development of new network architectures and computational paradigms has led to the need to design quality-aware video communication frameworks that can handle the new challenges and take advantage of the new features.

The overall objective of this dissertation is to study video quality in emerging networks from the perspectives of both human users and video analytic tools and then design new QoE-aware video communication strategies to improve video quality in both cases.

To cope with the rapidly growing Internet traffic and provide a more bandwidth-efficient, flexible, and scalable network, several future Internet architectures have been proposed in the past decade. Content-Centric Networking (CCN) is one such approach that aims to address the Internet's modern-day need for secure content dissemination on a massive scale to a diverse set of end devices. Because of the new features and the core differences between the traditional IP network and CCN, it is essential to study how video streaming will be different in CCN, along with the new challenges that need to be tackled to guarantee good QoE. First, we investigated the impact of network-level caching, a key feature of CCN, on content distribution and video streaming. Next, we determined how different CCN caching mechanisms influence video content distribution across multiple sources and whether it leads to more frequent video stallings. Then, we conducted human subjective tests to find out the degree to which video stalling influences the mean opinion scores (MOS). Lastly, we proposed a new QoE-aware multi-source video streaming framework to maximize QoE in CCN video streaming. Our video streaming scheme utilizes a QoE prediction model with customized parameters based on our human subjective test results.

The expanding usage of the IoT and M2M video applications has primarily been over

wireless connections. These connections linking different types of devices often have varying standards and specifications. Hence, it is vital to provide satisfactory QoE for video communication across various wireless communication entities. Software-defined radio (SDR) is a radio communication technology that uses software to implement components that have been traditionally implemented using hardware. This flexibility in implementation allows SDR to be used for exchanging videos among various wireless technologies. We developed a QoE-aware video streaming framework over SDR. First, we designed and built a video streaming platform over SDR featuring integrated QoE measurement components. The SDR platform has been constructed using Universal Software Radio Peripheral (USRP), GNU Radio [3], and GStreamer [4]. Second, we observed how the different application layer settings and physical layer parameters influence the QoE of both real-time streaming and stored video transmission. Lastly, we proposed a cross-layer QoE-aware video streaming architecture that adjusts both physical and application layer parameters to maximize the received videos' QoE.

In the next phase of our research, we studied video quality from the perspective of video analytics tools. Video analytics applications are often time-sensitive and demand high accuracy. To meet both of these requirements, edge computing-based video analysis has been widely used since performing video analysis at the network edge can provide lower latency, reduced bandwidth consumption, and the availability of powerful cloud computing resources offers high accuracy when required. However, there is a trade-off between latency and analysis quality; faster response time means less time for computation, i.e., lower analysis quality and vice versa. We studied this trade-off and optimized the streaming process for the best overall quality and latency. First, we designed and implemented a quality-aware video analytics platform based on edge computing. Then, we investigated the factors that influence the quality of edge computing-based video analysis as well as the factors that contribute to the overall latency both in terms of computation and communication. Instead of focusing just on the video bitrate or file size, we dived deeper and studied how video properties such as resolution, frame rate, and quantization parameter influence detection accuracy. Lastly,

we proposed a quality-aware edge computing-based video analytics framework to guarantee required analysis accuracy while minimizing total latency.

The rest of this dissertation is organized as follows. Chapter 2 describes our study on the impact of CCN content distribution on QoE and also illustrates our QoE-aware multi-source video streaming solution for CCN. Chapter 3 presents our cross-layer video streaming architecture over SDR. Chapter 4 illustrates our research on quality-aware video analytics in edge computing environments. In Chapter 5, our developed experimental platforms are discussed. Chapter 6 delineates the overall contribution of our research and concludes the dissertation.

Chapter 2

QoE-Aware Multi-Source Video Streaming in CCN

2.1 Background

The traditional host-centric Internet architecture is no longer suitable for the Internet, as the vast majority of current Internet usage consists of data being distributed from a source to numerous clients. Information-Centric Networking (ICN) [5] has been proposed as a future Internet architecture aiming to handle the vast amount of data, to provide better scalability and efficiency in terms of bandwidth demand. ICN allows a user to focus on the data itself rather than the location of the data. ICN-based video streaming has been demonstrated to generate better average quality than traditional TCP/IP-based streaming, thanks to the load balancing capability of ICNs [6].

Content-Centric Networks (CCN) [7] is a popular implementation of the ICN paradigm, particularly because of its CCNx [8] platform. We will focus on CCN in this work. The in-network caching mechanism of CCNs brings new challenges in maintaining the quality of experience (QoE) for video streaming. In-network caching allows faster content-fetching from nodes that are geographically closer to the user. By default, CCN follows the leave-

copy-everywhere (LCE)[7] caching strategy, which means popular contents will be cached nearer to the end-user. So, there might be multiple copies of the same content in the nearby nodes. Contrary to the classic streaming assumption that a client downloads a video from a single source, a client in CCN may get the video from multiple sources. Videos being larger in file size than other file types increases the probability of this scenario. Moreover, all users do not watch the whole video. For instance, in the case of YouTube, 60% of viewers watch only the initial 20% of a video [9]. Hence, the whole video may not be cached in all the nodes; instead, each node may cache a different amount depending on the popularity of the video among the viewers connected to it. In addition, due to the limitation of the cache size [10], each node will only cache locally popular segments.

Getting data from multiple sources may cause stalling during the playback at the user side. Stalling, defined as the momentary disruption in the fluidity of video playback due to network impairments, is a prominent factor that could degrade user experience [11]. Stalling events during the video playback are caused by the rebuffering of the video due to an insufficient content delivery rate. When a client gets videos from multiple sources in CCNs, switching between sources could introduce additional delays. Moreover, the link quality between a user and each source could be different, and the associated delays will be different, causing intermittent stalling and lower QoE [12].

To date, little is known to what extent video streaming from multiple sources affects QoE in CCN. In our preliminary work [13], we investigated the impact of content source switching, which is brought by distributed in-network caching, on the QoE for video streaming in CCNs. We conducted human subjective tests using videos generated from a CCN emulation platform. Our test results indicate that switching between content sources could cause more stalls in video playback and thus decrease QoE scores.

Next, we proposed a new QoE-aware multi-source video streaming scheme for CCN. First, we investigated the characteristics of video content distribution under a variety of CCN caching methods. We then proposed a new adaptive video streaming with distributed

caching (ASDC) algorithm that aims to suppress the stalling resulting from switching between content sources to maintain satisfactory QoE during the switching process. The design of ASDC is based on the conclusions from many user experience studies[11, 14]: i) stalling or disruption of video fluidity has the largest effect on QoE, and ii) temporary reduction of the video quality to avoid stalling is preferable to most viewers. To achieve adaptive video quality, we used scalable video streaming in which scalable video coding (SVC) [15] is applied to the source videos. Our algorithm employs a QoE model that characterizes the effect of stalling, and the parameters of the model are adjusted by our human subjective test results. Based on the prediction of possible stalling and resulting QoE degradation, the proposed algorithm automatically adjusts the video quality to maintain playback fluidity.

2.2 Related Works

2.2.1 Stalling in CCN Video Streaming

Stalling or video freezing is the momentary disruption in the fluidity of video playback. Stalling is a major factor that degrades the QoE of end-users. A study [16] on stalling events caused by network bottlenecks showed that QoE is primarily influenced by the frequency and duration of stalling events. Moreover, stalling has a worse impact on QoE than frame rate reduction [17]. For conventional networks, network congestion, collision, and packet loss are the main contributing factors to stalling, and many techniques have been proposed to reduce stalling frequencies and lengths [18, 19, 20]. For CCN, however, little is known about how source switching introduced by distributed caching could impact the QoE for video streaming. A few recent studies aimed at improving the QoE for video streaming in CCNs using strategies such as adaptive bitrate streaming [21, 22] and proactive content caching [23], but none of them studied the relationship between source switching and QoE in depth.

2.2.2 Usage of DASH in CCN

Dynamic adaptive streaming over HTTP (DASH) is widely used in conventional networks. Traditionally, advanced video coding (H.264/AVC) is most commonly used in DASH to encode videos. We refer to H.264/AVC as AVC in the rest of this section. However, there are some challenges in the direct application of this technique in CCNs. In DASH-AVC, each quality layer is independent of all other quality layers, leading to a significant amount of redundancy, both in segment storage and delivery. In contrast, scalable video coding (SVC) creates a dependency between different quality layers, and the addition of each enhancement layer to the base layer upgrades the quality. This reduces the load on video servers [24], improves cache hit ratio [25], lowers cache redundancy [26], and helps to tackle the cache-size constraints in CCN [27].

Moreover, AVC based DASH clients strongly depend on end-to-end throughput estimation to select a proper version of the content [28]; once a quality is chosen, the segment cannot be upgraded to a higher quality and a partially received segment cannot be decoded at a lower quality. This can be an issue in CCNs, where, unlike traditional networks, content is served through multiple caches with different throughput, and content sources are transparent, making the throughput assessment task difficult. Overestimation of available throughput can lead to buffer drainage, an increased number of quality switches and video freezes. SVC allows the client to gradually upgrade or downgrade the quality by adjusting the number of enhancement layers, leading to more efficient quality-switching and a higher average video rate compared to AVC [26]. In [22], the throughput estimation issue of DASH is identified, and a hop-by-hop adaptive video streaming scheme is proposed. SVC is utilized to enable the routers to proactively drop the less valuable SVC layer data packets, which is not possible in the case of AVC based DASH.

Additionally, in traditional DASH applications, a large playout buffer is required to guarantee a continuous playout. The incremental characteristic of SVC allows reducing the buffer size as the base layer guarantees an uninterrupted playout. This can be helpful in

CCN video streaming as the client may switch to a different cache during playback.

Furthermore, AVC does not permit to fully utilize one of the main innovations introduced by CCN, i.e., the possibility to simultaneously use all the available network interfaces of a device. This is especially beneficial in mobile networks, where the video streaming client has different antennas, e.g., LTE, Bluetooth, Wi-Fi, etc. In standard AVC, only the interface providing the highest bitrate is used to forward an interest. The layered structure of SVC provides a natural way to use multiple interfaces concurrently, thus increasing the aggregated throughput for the client and, hence, improving the QoE. In [29], a network coding-based DASH implementation over named data network (NDN) is proposed, where the client uses different network interfaces to receive video packets across multiple interfaces simultaneously, providing better cache-hit ratio, faster video quality adaptation, and reduced server load.

2.2.3 Novelty of Our Work

Our work is consistent with the recent CCN video streaming solutions [22, 27] in that we also utilize scalable video streams. The unique contribution of our work is two-fold. First, we investigate the problem of video streaming from multiple routers in CCNs, which has not been addressed in other CCN video streaming studies. Second, our QoE-aware video streaming solution is based on a statistical QoE model obtained from extensive experimental results as well as realistic caching mechanisms in CCNs.

2.3 Impact of Distributed Caching on QoE

Fig. 2.1 shows a typical CCN network topology and an example of content source switching during video streaming. A client is connected to an edge router ER, and ER is in turn connected to the content server through several intermediate routers, such as IR_i, IR_j, and IR_k. To get a particular video, the client will send requests to ER. Depending on the availability of the content at ER, two scenarios may arise. First, the whole video may be

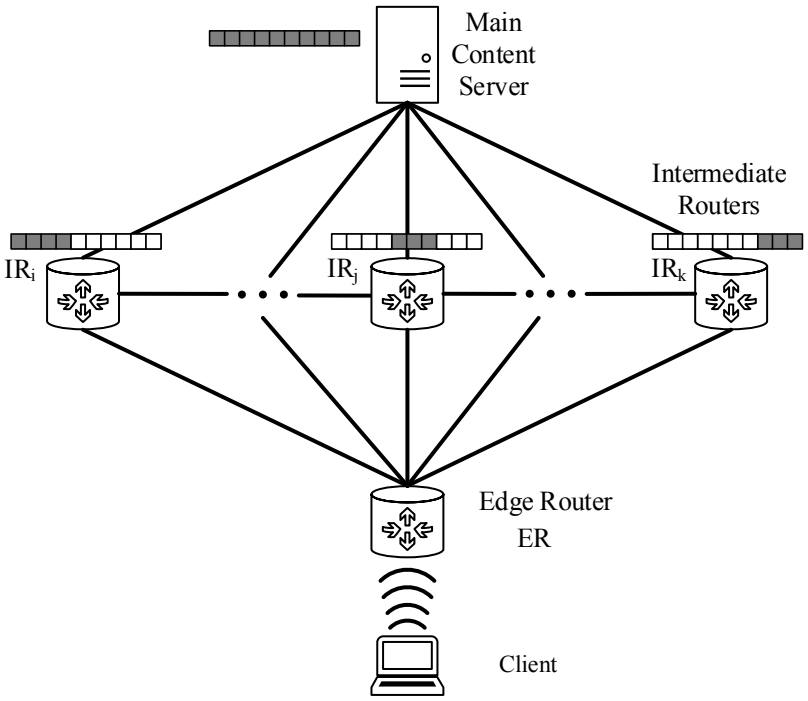
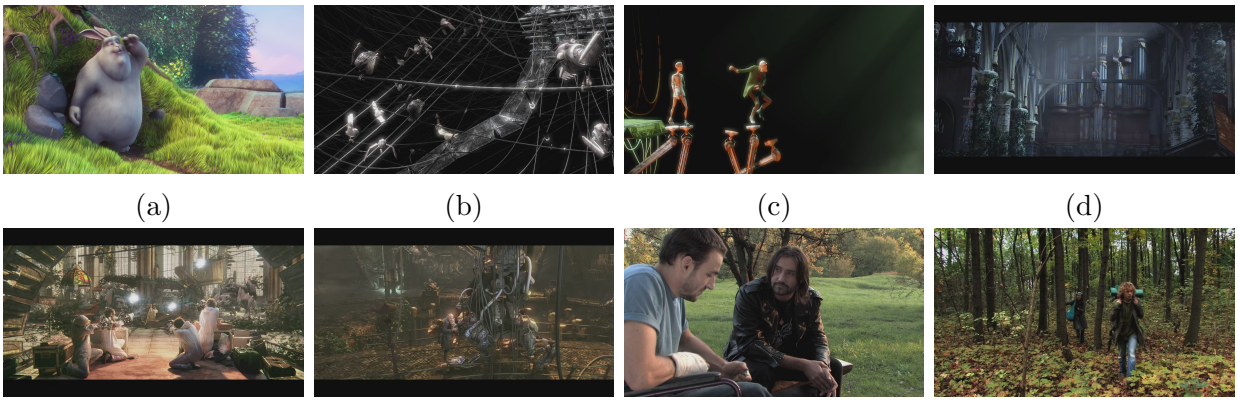


Figure 2.1: The diagram of CCN network topology

cached at the Content Store (CS) of ER, and thus ER will directly serve the video to the client. As discussed earlier, this is unlikely for most content. Second, the video may be partially cached at ER or not cached at all. In this case, ER will obtain the complete or parts of the video from the intermediate routers or the content server. In the example of Fig. 2.1, IR_i has cached the first 40% frames of a video. When IR_i finds that it does not have the rest of the video, it will switch to IR_j and IR_k, which can provide the middle 30% and the last 30% frames of the video. This results in a total number of two switches in the video streaming process. It should be noted that, even if the complete video is cached at one intermediate router, ER may still switch to another router with the same content if the link quality between ER and the first router deteriorates. If none of the routers contain a specific part of the video, the main content server will respond.

We installed CCNx 1.0 [8] (Distillery 2.0) on Ubuntu 14.04. All the routers and the main content server were connected through the Ethernet, while the client and the edge router were connected through 802.11n wireless connection. The main content server had



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2.2: Snapshots of video sequences

the complete video files, and the intermediate routers contained different parts of the same video. Different cache distributions were used for different video files with up to 20% cache overlap between routers. This content distribution simulates the in-network caching of the CCN routers with the default LCE [7] caching strategy. LCE, also known as CCE (caching everything everywhere), is an on-path, homogeneous, and non-cooperative caching technique.

Each video file was chunked into 1200 byte segments and stored in the CS of intermediate routers. This segment size was chosen to prevent IP fragmentation of CCNx content object messages. Each intermediate router had a server application that maintained the videos in the CS and parsed the incoming requests to serve appropriate segments to the client. A modified version of the CCN-VLC plugin [30] was used for the streaming purpose on the client side. The requests sent from the client to the routers had the following format: *ccnx:/ccnx/fetch/Videoname.mp4/Chunknumber*. The client determines the chunk number using the file size information obtained from the server.

2.4 Human Subjective Test

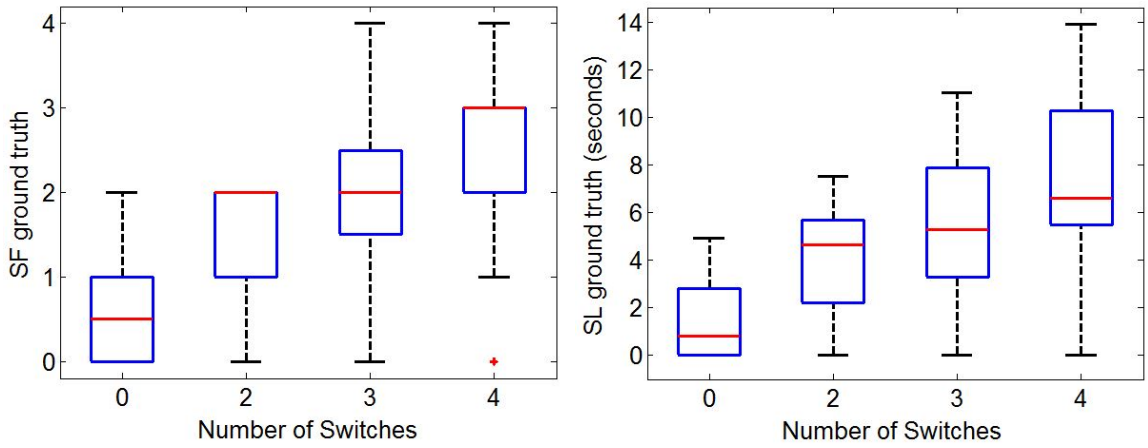
2.4.1 Overview and Test Procedure

We conducted human subjective tests using videos generated from the aforementioned platform. We selected 8 raw video clips from four movies: *Big Buck Bunny* and *Elephants Dream* in Xiph.org Video Test Media Collection [31], *Tears of Steel* [32], and *Valkaama* [33]. These video clips contain different motion characteristics: half classified as high motion and half as low motion, based on the temporal information index defined by ITU-T P.910 [34]. The genres of the 8 videos cover animation, talk show, human with nature, science fiction, and action movie, as shown in Fig. 2.2. Each video has 1280×720 pixels with a duration of 30 seconds. We encoded the eight raw videos in two bitrates (1500 kbps and 200 kbps) with the HM reference software (HEVC Standard) and generated 16 encoded videos. The choices of bitrates align with common bandwidth conditions in the network, and the two bitrates reflect different levels of clarity. Four file distribution cases for CCN caching were emulated: a video is distributed among 1, 3, 4, and 5 routers, resulting in 0, 2, 3, and 4 switches during the video streaming process. The 16 videos were transmitted over each of these cases, and finally, 64 video samples were obtained. The channel bandwidth between the edge router and the client was set to 2600 kbps for higher quality videos and 350 kbps for lower quality ones. These bandwidth values were carefully selected to ensure that the channel had enough capacity to transmit videos without stalling in the absence of source switching.

In the subjective test, 36 participants (21 males, 15 females with normal vision aged from 18 to 34) rated the video samples in an indoor office with ordinary illumination. The methodologies for the assessment of video quality are based on the recommendation of ITU-R BT.500-13, the single-stimulus method [35]. A graphical user interface was used to record individual ratings by playing 64 videos in random order. And the rating index with nine scales ranging from 1 (unbearable/poor) to 9 (satisfied/excellent) was used to evaluate the overall quality of the video, i.e., overall mean opinion score (MOS), satisfaction with stalling

length (SwSL), satisfaction with stalling frequency (SwSF), and video clarity, etc. The screening method recommended by BT.500-13 was used to screen the collected data [35]. The first four video ratings of each subject as "dummy presentations" were removed for stabilizing opinions [35]. After examining the outliers [35], no observers were rejected.

2.4.2 Analysis of Subjective Test Data

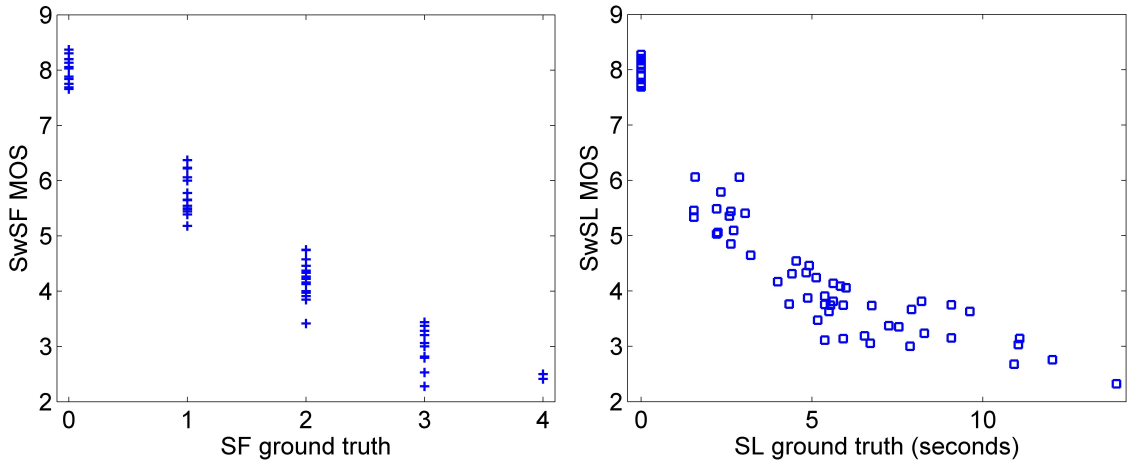


(a) Stalling Frequency (b) Stalling Length

Figure 2.3: Ground truths of stalling frequency and stalling length

To understand the effect of source switching on the stallings in received videos, we first check the distributions of the ground truths of stalling frequency (SF) and stalling length (SL) under a different number of switches (Fig. 2.3). The edges of the blue boxes are the 25th and 75th quartiles, and the central red marks are the medians. In Fig. 2.3(a), the SF ground truth is defined as the times that stalling appears in each video; in Fig. 2.3(b), the SL ground truth is defined as the total duration of each stalling event in a video. We find that as the number of switches increases, both the SF ground truth and the SL ground truth increase. An outlier is marked as a red dot in Fig. 2.3(a). It corresponds to the video *TearsofSteel2* encoded under 200 kbps, where no stalling occurred even though there were four switches between the caching sources.

We then compare the subjective scores of SF and SL with their ground truths. Fig.



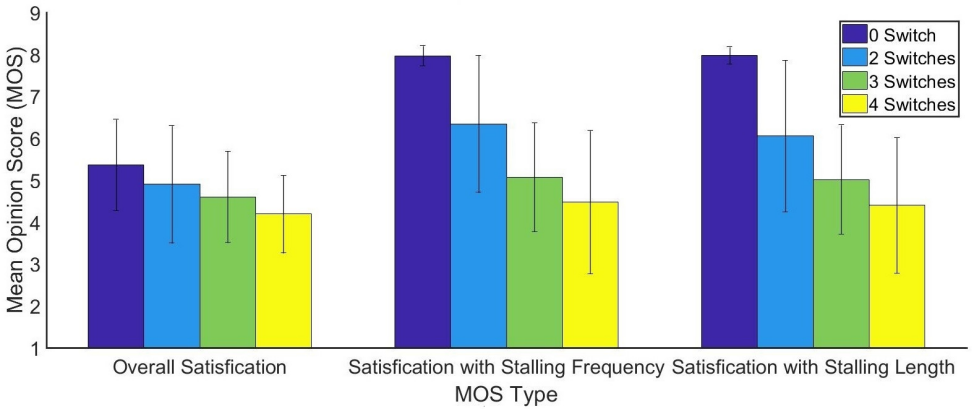
(a) Stalling Frequency (b) Stalling Length

Figure 2.4: Stalling ground truths vs. subjective scores

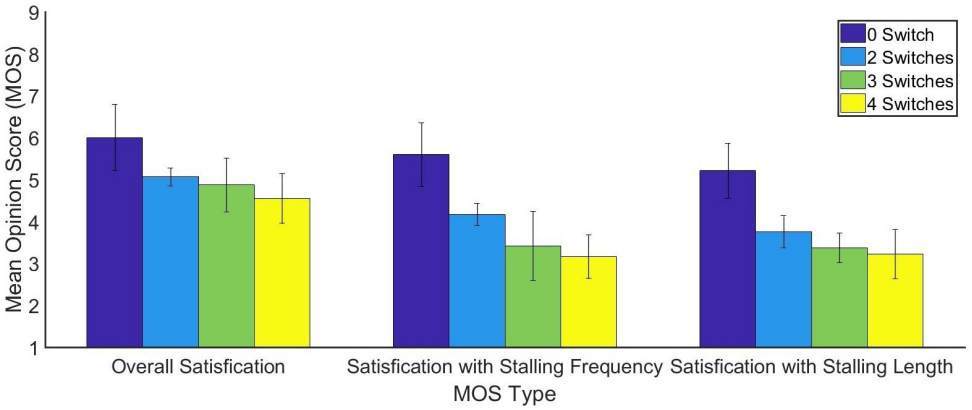
2.4 shows the relationships between the stalling ground truths of the 64 videos and their corresponding subjective scores. In Fig. 2.4(a), the SwSF MOS has a negative correlation with the SF ground truth. In Fig. 2.4(b), as the SL ground truth increases, the SwSF MOS decreases. These results indicate that the subjective evaluations of stalling align with the true stalling events in the videos.

Fig. 2.5 presents the statistical results about the overall MOS, SwSF, and SwSL under a different number of switches. The overall MOS, SwSF, and SwSL all decline as the number of switches increases from 0 to 4. The overall MOS under 1500 kbps is better than the ones under 200 kbps, mostly due to the fact that higher bitrates correspond to higher clarity in the videos. On the other hand, the satisfaction scores with stalling (SwSF and SwSL) for videos encoded under 1500 kbps are lower than the ones encoded under 200 kbps, indicating that stallings are more likely to occur under higher bitrates.

We further analyze to what extent the stalling events could contribute to the overall QoE. Fig. 2.6 depicts the relationship between SwSF, video clarity, and overall MOS in the two bitrates, in which the video clarity is represented by the color bars. For the high bitrate scenario, the video clarity MOSs are relatively high, and the overall MOS increases when SwSF increases. That is to say, stalling has a greater impact on the overall MOS



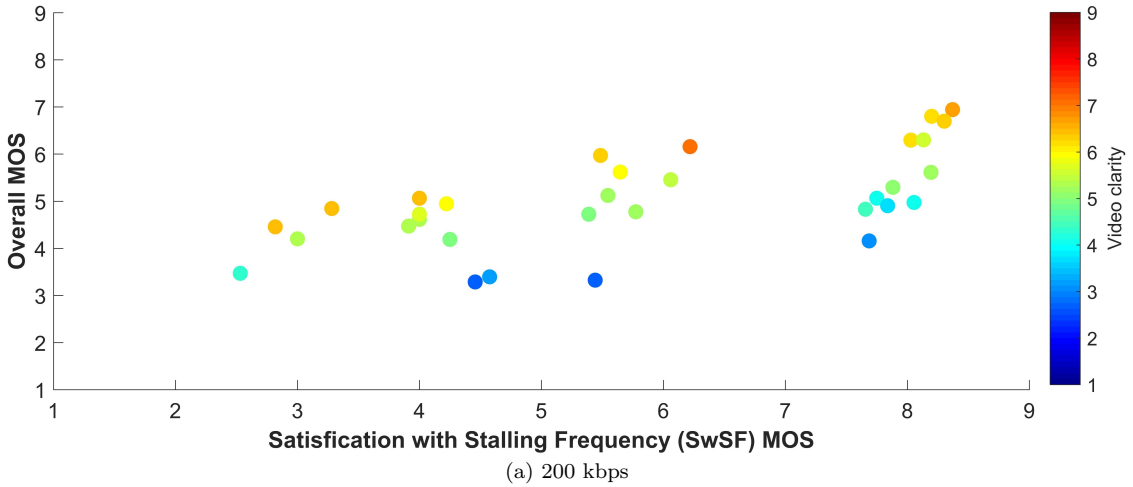
(a) 200 kbps



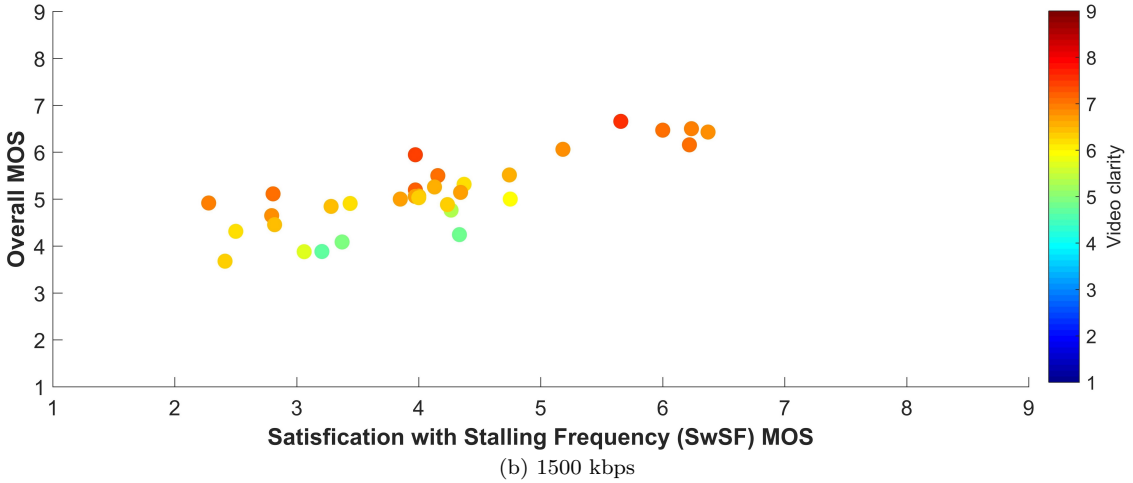
(b) 1500 kbps

Figure 2.5: Overall MOS, SwSF, and SwSL under four CCN conditions

when the videos have high clarity levels. For the low bitrate scenario, the data are relatively dispersed, and there is no significant linear correlation between the overall MOS and SwSF. Furthermore, we analyze the correlation between the overall MOS and SwSF, SwSL, and video clarity for the two bitrates, and the results are shown in Table 2.1. Similar conclusions could be drawn from this correlation analysis. In both the two bitrates, SwSF, SwSL are correlated with the overall MOS. Under 1500 kbps, SwSF and SwSL have more obvious linear correlations with the overall MOS, while under 200 kbps, video clarity also plays a significant role.



(a) 200 kbps



(b) 1500 kbps

Figure 2.6: Overall MOS vs. SwSF under different video clarity levels

Finally, we use Principal Component Analysis (PCA) method to quantify the relationships between the different quality-contributing factors. The results are shown in Table 2.2.

Table 2.1: Correlation coefficient between Overall MOS with SwSF, SwSL and clarity

Correlation Type	1500 kbps			200 kbps		
	SwSF	SwSL	Clarity	SwSF	SwSL	Clarity
Pearson	0.8279	0.8025	0.7043	0.6466	0.6551	0.8576
Kendall	0.5804	0.5131	0.5602	0.5576	0.5631	0.6276
Spearman	0.7467	0.6746	0.7502	0.7361	0.7539	0.8139

Table 2.2: The results of principal component analysis

Principal Component	1st	2nd	3rd	4th	5th
Eigenvalue	6.5327	2.9608	0.1883	0.0857	0.0529
Proportion (%)	66.52	30.15	1.92	0.87	0.54
Cumulative (%)	66.52	96.67	98.59	99.46	100.00

The 1st and 2nd principal components (PC) maintain overwhelming superiority with 66.52% and 30.15%. Since the cumulative contribution is larger than 95%, we can use the first two PCs to explain the total variability. The transform equation of PCs with original factors is $P_i = \sum_{j=1}^{j=5} c_{ij} \cdot X_j$, where P_i is the i principal component, c_{ij} is corresponding coefficient, X_j is the original factor. The five factors are movement consistency, video clarity, video quality constancy, SwSF, and SwSL, sequentially. The 1st PC transform vector is (0.0701, -0.1402, -0.0039, 0.7008, 0.6959), and the 2nd PC one is (0.4528, 0.7326, 0.5024, 0.0661, 0.0382). The transform vectors show that stalling satisfaction (SwSF and SwSL) dominate the 1st PC, and the video clarity contributes most to the 2nd PC.

2.5 QoE Model

We propose a model to accurately predict the QoE. To achieve this, we conducted two human subjective tests and collected mean opinion scores (MOS) from human subjects who rated videos streamed under various network conditions and streaming techniques. A total of 92 videos have been used in this study from the two subjective tests.

We used the results for higher bitrate (32 videos, encoded at 1500 kbps) from our first subjective test described in Section 2.4.1. In our second subjective test, we used 60 videos

encoded at bitrates ranging from 1500kbps to 4500kbps. Each video had 1280×720 pixels with a duration of 30 seconds. The test videos were obtained by streaming six videos under five different CCN caching methods (listed in section 2.6) and varying network bandwidths. In this subjective test, 28 participants (24 males and 4 females, with normal vision aged from 18 to 34) rated the video samples. We used the same test environment and methodologies in both subjective tests (described in Section 2.4.1). The total time including training and breaks was within 1.5 hours.

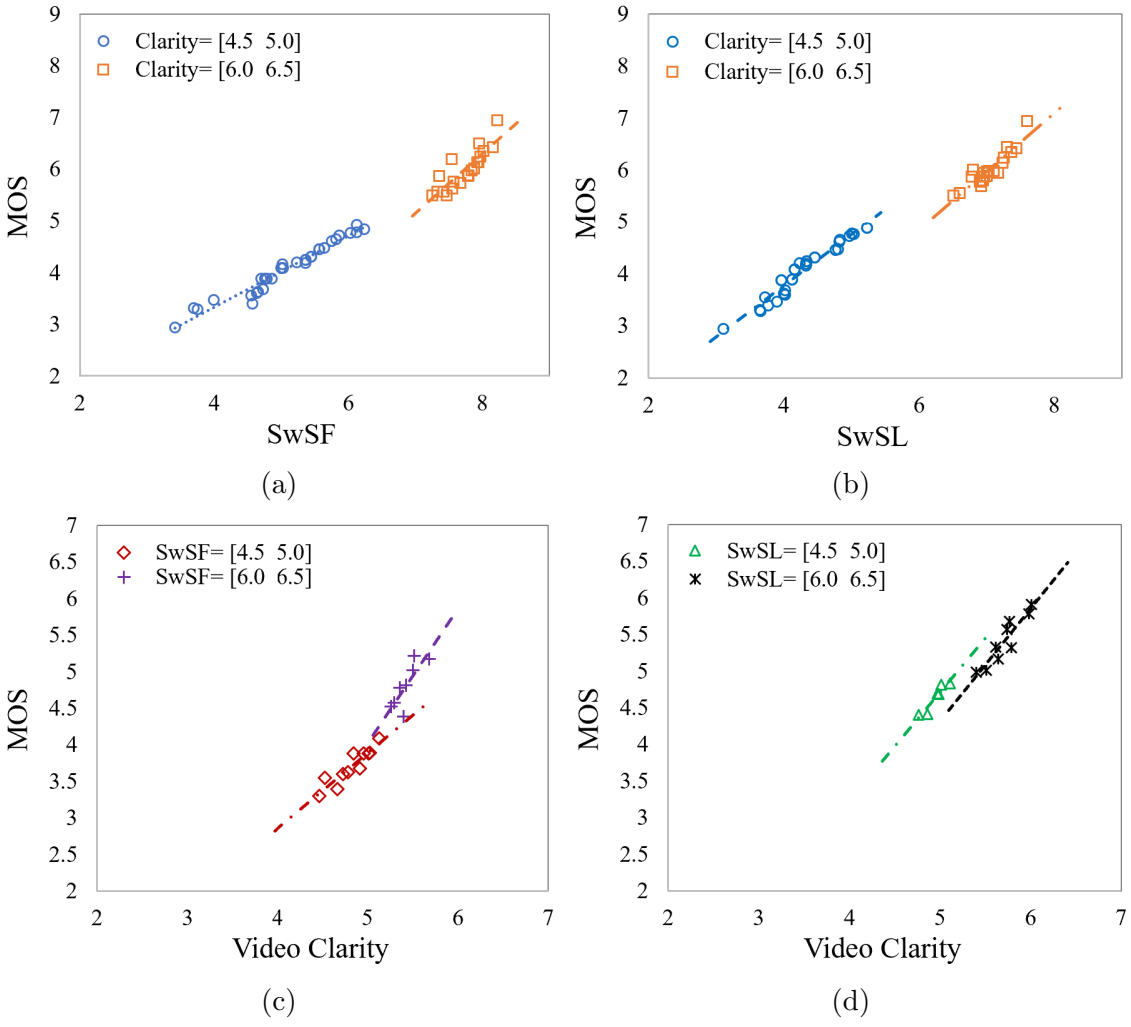


Figure 2.7: MOS from human subjective tests. (a) MOS vs. SwSF for different clarity levels, (b) MOS vs. SwSL for different clarity levels, (c) MOS vs. Clarity for different SwSF levels, and (d) MOS vs. Clarity for different SwSL levels

From our human subjective tests, we found that satisfaction with stalling frequency

(SwSF), satisfaction with stalling length (SwSL), and video clarity all had linear relationships with overall mean opinion scores (MOS), which is shown in Fig. 2.7. For the 92 videos rated by human subjects, the average value (each video was rated by multiple viewers as described above) of video clarity ranged from 4.364 to 6.897, SwSF ranged from 3.242 to 8.336, and SwSL ranged from 3.013 to 8.246. To demonstrate the linear relationships, we plotted data points from two ranges as examples: 4.5 to 5.0 and 6.0 to 6.5 for each case in Fig. 2.7(a), 2.7(b), 2.7(c), and 2.7(d).

We built a QoE prediction model featuring linear relationships between MOS and both stalling and video clarity based on Streaming QoE Index (SQI) [36] with customized parameters using results obtained from our human subjective tests. Linear QoE models have been used in several other studies on video streaming [37, 38, 39, 40].

The original SQI model takes the impact of stalling, initial buffering, and video presentation quality into consideration. The overall QoE is,

$$Q = P_n + S_n \tag{2.1}$$

P_n represents the instantaneous video presentation quality, which can be estimated at the server-side by a frame-level video quality assessment (VQA) model. We use the structural similarity index (SSIM) [41] as the VQA operator in our work. The model views each stalling event independently and sums all the events to calculate the overall effect, and S_n represents the aggregated QoE drop due to stalling and is computed by summing the drop caused by each stalling event:

$$S_n = \sum_{k=1}^{N} S^k(t) \tag{2.2}$$

where N is the total number of stalling events and S^k quantifies the experience of the k-th stalling event. Assume that the k-th stalling event locates at $[i_k, i_k + l_k]$, where l_k is the length of the stall. The following piecewise model estimates the impact of the k-th stalling

event at time t:

$$S^k(t) = \begin{cases} P_{i_k-1}(-1+\exp\{-(\frac{tf-i_k}{T_0})\}) & \frac{i_k}{f} \leq t \leq \frac{i_k+l_k}{f} \\ P_{i_k-1}(-1+\exp\{-(\frac{l_k}{T_0})\}). & \\ (\exp\{-(\frac{tf-i_k-l_k}{T_1})\}) & t > \frac{i_k+l_k}{f} \\ 0 & otherwise \end{cases} \tag{2.3}$$

where f is the frame rate in frames/second, P_{i_k-1} is the scaling coefficient of the decay function, and T_0 and T_1 represent the rate of dissatisfaction and the relative strength of memory, respectively.

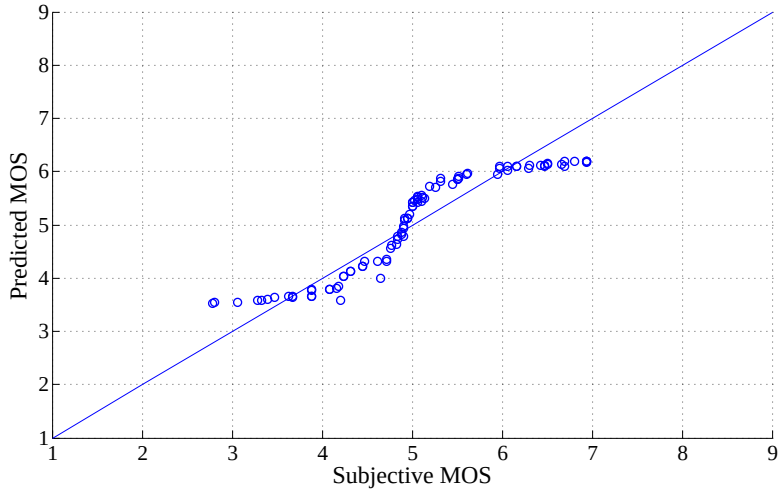


Figure 2.8: Mean opinion scores (MOS) from human subjective tests and QoE model for 92 videos

The original SQI model produces quality scores ranging from 15 to 90, according to the experimental results presented in [36]. However, in our subjective tests, human subjects rated video quality ranging from 1 to 9. To generate QoE prediction scores in the same range as the mean opinion scores (MOS) from our human subjective tests and improve the prediction quality, we propose to customize the parameters in the SQI model by applying curve-fitting on our test data. Based on these test results, the QoE model can be updated as,

$$Q = a + b \times P_n + c \times S_n \tag{2.4}$$

where, $a = 5.783$, $b = 0.4128$, and $c = -0.8593$. Fig. 2.8 shows the MOS from the human subjective tests and the overall QoE scores generated by the updated QoE model (Eq. 2.4) for the 92 videos. The R-square value for the prediction is 0.8674, the adjusted R-square value is 0.8644, and the root-mean-square error (RMSE) value is 0.3719.

2.6 Caching Mechanism and Content Distribution

In this section, we present preliminary results on how video contents are cached across different CCN nodes or routers for different caching methods. To produce a realistic content distribution in the content stores (CS) of the CCN routers, we used the ccnSim simulator [42]. We used five different caching mechanisms to observe the distribution of video content throughout the network. The caching strategies are: LCE, Betweenness Centrality (BTW) [43], 2-LRU [44], ProbCache [45], and Cost-Aware (CoA) [46]. We chose these caching methods because they are popular and they present enough variance in techniques for CCN.

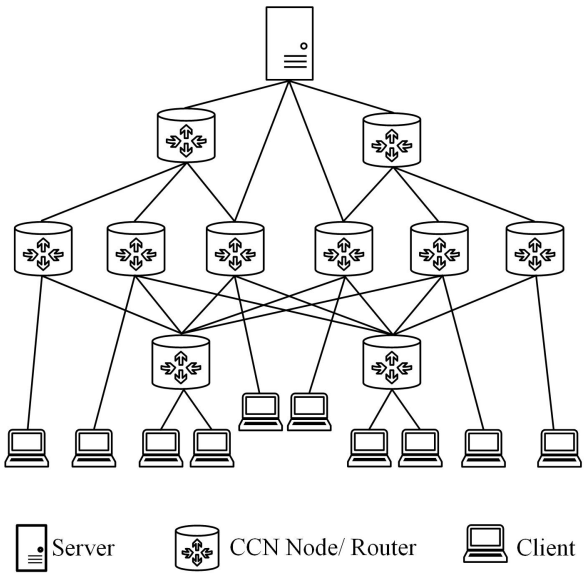


Figure 2.9: Network topology used in ccnSim

We used a custom topology inspired from [47] shown in Fig. 2.9, with the parameters shown in Table 2.3. The interest forwarding strategy was set to shortest path routing (SPR), and the cache replacement policy was least recently used (LRU). The popularity of the contents was varied among the clients, and similar viewer engagement as YouTube [9] was considered. We ran the simulation ten times (varying user interests) for each caching strategy. Table 2.4 shows the average distribution of video content from this simulation. For all the caching methods, the majority of the contents were retrieved from more than one source. Note that the amount of content fetched from four or more sources is comparatively low here because the topology used has fewer intermediate routers compared to real-world scenarios where the main content server may be located many hops away from the client.

Table 2.3: Parameters used for the simulation

Parameter	Values
Catalog size	2×10^{10}
Cache size	10^7
Aggregated request rate per client	15 req/s
Forwarding strategy	SPR
Replacement strategy	LRU
Meta caching	lce, two_lru, prob_cache, btw, costawareP
Initialization type	Cold
Link capacity (balanced)	2.5mbps

Table 2.4: Content distribution among CCN nodes

Caching method	Percentage of contents from			
	1 source	2 sources	3 sources	4≤ sources
LCE	14.4%	27%	47%	11.6%
BTW	28.2%	40.6%	27.2%	4%
2-LRU	21%	31.7%	38.1%	9.2%
ProbCache	23.3%	36%	32.5%	8.2%
CoA	20.1%	37.4%	32.4%	10.1%

In our previous study [13], we conducted experiments where the number of content sources was a controlled variable and observed the effect of content distribution on the number of

stalling and MOS through human subjective tests. Video transmission from multiple sources over CCN was emulated using the CCNx [8] platform, and then 36 human subjects rated each video. Details of the subjective test have been described at the beginning of section 2.5. We found that when the number of source switching increases, both the frequency and the length of the stalling events increase. The overall satisfaction declines when the number of switches increases. The results of correlation analysis and principal component analysis (PCA) indicate that the satisfaction with stalling has an obvious linear correlation with the overall MOS (Pearson coefficient for SwSF and SwSL are 0.8279 and 0.8025, respectively).

2.7 Proposed Streaming Algorithm

2.7.1 ASDC Overview

We propose an adaptive video streaming with distributed caching (ASDC) algorithm for CCNs to improve QoE by minimizing stalling caused by content source switching. To simplify the description of our algorithm, we use a sample network topology shown in Fig. 2.1. Client C_1 is connected to an edge router ER, and ER is in turn connected to the content server through several intermediate routers, such as IR_i, IR_j, and IR_k. To get a particular video, the client will send requests to the ER. Depending on the availability of the content at ER, two scenarios may arise. First, the whole video may be cached at the CS of ER. As discussed earlier, this is unlikely for most contents because of the small cache to video catalog ratio. Second, the video may be partially cached at the ER or not cached at all. In this case, the ER will have to get the complete video or parts of it from the intermediate routers or the content server itself. It should be noted that, even if the complete video is cached at one of the intermediate routers, ER may still switch to another router with the same content if the link quality between ER and that router deteriorates. Similarly, in the case of overlapping coverage, ER will select the source based on the link quality between the ER and those two sources. If ER finds there is a possibility of stalling, it reduces the number

of enhancement layers to avoid stalling, which is outlined in Fig. 2.10. Similarly, if the link quality is improved, the number of enhancement layers will be increased.

A CCN router consists of three main modules: content store (CS), forwarding information table (FIB), and pending interest table (PIT). We introduce a new component for the CS of the CCN routers: the content-listing table (CLT). At each router, the CLT will store the cached file names and the range of packets cached for each file. For example, for the file named $SampleVideo.mp4$, which has a total of 4000 chunks, one router may cache the first 800 chunks. In this case, the CLT in that router will have the following entry:

File name	Cached chunks
SampleVideo.mp4	1 to 800 (out of 4000)

When the router receives a request for the file $SampleVideo.mp4$, it will respond with the data that its CS has chunks 1 to 800. This allows the edge router ER to determine when to send requests for the rest of the chunks to other routers or the content server. In case the routers cache random chunks instead of consecutive ones, the number of switches will increase as ER will have to switch between different CS randomly.

Table 2.5: List of variables used in the model

Variable	Definition
L_i	Initial number of layers
L_j	Adjusted number of layers
B_i	Bit rate of L_i
B_j	Bit rate of L_j
B_w	Channel bandwidth between the client and the source
l_k	Stalling duration
θ	Duration for which quality is changed
t_0	Video beginning time
t_i	Time at which source is switched
t_n	Video ending time
δ	Extra content in buffer

2.7.2 Steps of ASDC

There are two algorithms in ASDC: Algorithm 1 delineates the streaming process and Algorithm 2 describes how the quality adjustment mechanism takes place inside Algorithm 1. Table 2.5 shows a list of the variables used in the algorithms.

Algorithm 1 Quality adaptation at the edge router

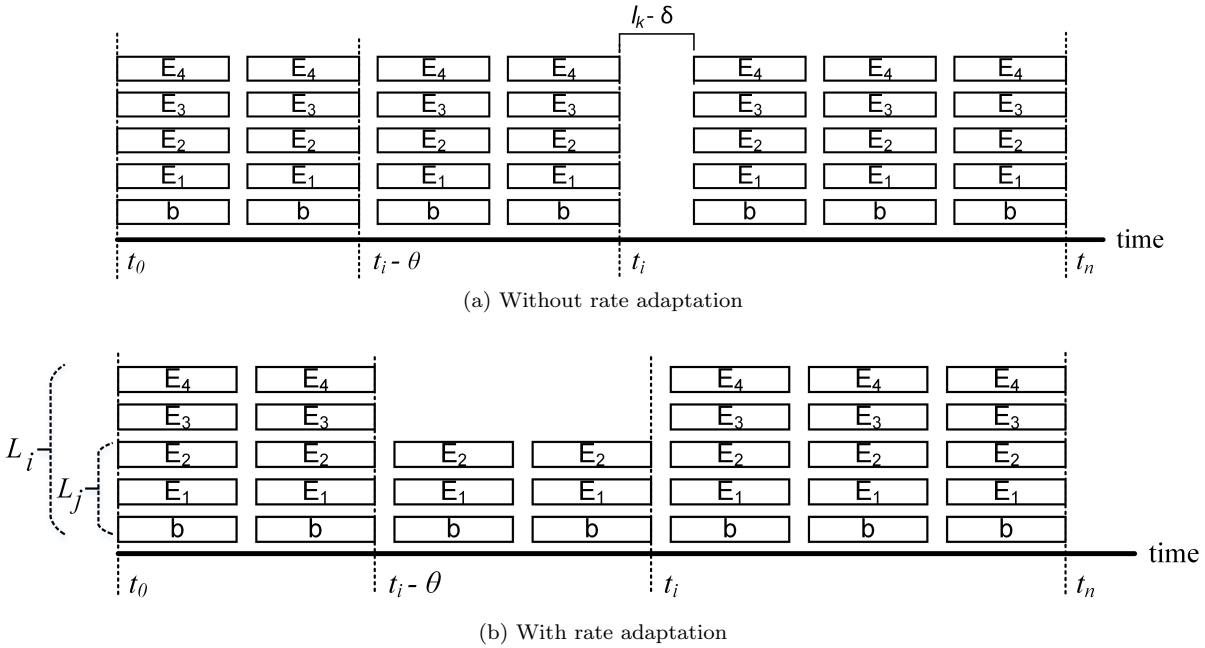
1: C_1 sends request to ER
2: **if** full video is available in the CS of ER **then**
3: ER starts streaming to C_1
4: **else**
5: ER sends request to neighboring routers through FIB
6: ER receives response from other nodes and generates FDT
7: ER starts streaming to C_1
8: **end if**
9: **while** $t < t_n$ **do**
10: ER determines l_k and gets δ from C_1
11: **if** l_k=0 OR $\delta > l_k$ **then**
12: C_1 gets L_i
13: **else**
14: ER determines θ and L_j using Algorithm 2
15: for duration $t = t_i - \theta : t_i$, C_1 gets L_j
16: **end if**
17: **end while**

Request for video (lines 1-8 of Algorithm 1): When C_1 sends a request for a video, ER will check its CS for the video and then add the request in its PIT. If the video is available at the CS of ER, it will be directly served to C_1. Otherwise, the request will be forwarded to the intermediate routers using the FIB. After the video is located and ER receives the response from one or more intermediate routers, it will generate a file distribution table (FDT) and then start streaming to C_1. The FDT will contain information about the distribution of the file among the content stores of different routers along with information such as the timing, the number of layers, bit rates, resolution, and file size.

Adapting to source switching (lines 9-17 of Algorithm 1): Suppose the video is stored in two of the intermediate routers and each one of them contains certain parts of the

whole video. The first source contains the parts for the duration $[t_0, t_i]$ and the second source contains $[t_i + 1, t_n]$. When ER starts to get packets from the first source, it will get the list of packets available at that router via CLT and then update FDT. So, ER will know when it has to send requests for the next batch of packets to the other source.

ER will then check if there is a possibility of stalling due to that source switching by checking l_k and δ (line 11). If no stalling is expected, ER calculates an initial quality of the video to transmit based on the current link quality and sends that information to C_1. C_1 then sends requests based on that information. The initial quality in the case of scalable video means a number of layers, and we denote it by $L_i(n_0, ..., n_i)$. However, if a stalling is expected, the number of layers will be reduced to $L_j(n_0, ..., n_j)$, where $L_j < L_i$ and this will be maintained for the interval $[t_i - \theta, t_i]$, which is depicted in Fig. 2.10. The variable θ is a measure of time, and it dictates the moment when to change the number of layers. The algorithm also works when there is no source switching and stalls are caused only by bandwidth fluctuation, in which case, t_i represents the possible time when the stall will take place.



(a) Without rate adaptation

(b) With rate adaptation

Figure 2.10: An example of rate adaptation considering switches

Determine θ and L_j (Algorithm 2): The values of θ and L_j depend on the length of the possible stalling event l_k, which is shown in Fig. 2.10 (a). In the case of source switching, l_k denotes the switch time. The value of l_k is predicted by ER. We assume the routers are aware of the throughput between them and their immediate neighbors. This is a reasonable assumption as routers continuously exchange packets and keep the routing table updated with connection status. Hence ER can reliably predict the possible delay in packets to be transmitted to and from its immediate neighbors, which allows the prediction of a possible stalling event's duration.

The effect of l_k will be offset by the extra content in the buffer. The extra content is cached in the buffer when the buffer fill rate x is more than the depletion or playout rate r. We denote the extra content in the buffer by δ, which can be computed by the client. If δ is larger than l_k (line 9 of Algorithm 1), quality reduction is not needed as the client can go through the switch maintaining the current quality without stalling. However, if δ is less than l_k, quality adjustment is necessary.

To avoid stalling, the total time required to transmit the video with adjusted number of layers plus the difference between the extra buffered content and stalling duration must be less than or equal to the time required to transmit the video with the initial number of layers:

$$\frac{B_i \times (t_i - t_0)}{B_w} \geq \frac{B_i \times (t_i - \theta - t_0) + B_j \times \theta}{B_w} + (l_k - \delta) \tag{2.5}$$

Eq. (2.5) can be reduced to

$$B_i\theta - B_w(l_k - \delta) \geq B_j\theta \tag{2.6}$$

ER computes the value of B_w from the amount of data transferred between the edge router and the content source per unit time. The number of available values of B_j or L_j depends on the number of video qualities available at the server. If the video is available at

four different qualities, B_j can be set to the bandwidth of any three of the other qualities apart from the current one. For each three of these values of B_j, to satisfy Eq.(2.6), the corresponding θ has to be determined.

Algorithm 2 Determine quality switch time and layer adjustment

```
1: Input: L_i, l_k, {L_j}
2: ER calculates {L_j}, {θ} that satisfies Eq. (2.6)
3: ER calculates L_j^*, θ^* for max Q using Eq. (2.4)
4: set min=0
5: if θ^* > ΔT then
6:     for (θ=θ^*; θ >0; θ=θ^*-Δθ) do
7:         calculate QoE value, Q using Eq. (2.4)
8:         if min <= Q then
9:             min = Q
10:            θ^* = θ
11:        else
12:            break
13:        end if
14:    end for
15: end if
16: Output: θ^* and L_j^*
```

Algorithm 2 shows the computation of θ and L_j. It first decides the candidate combinations of θ and L_j that reduce the stalling duration to zero, i.e., satisfying the condition presented in Eq. (2.5). However, they produce different overall QoE. To choose the best combination of L_j and θ, we use the QoE model described in section 2.5. We select the combination of L_j^* and θ^* that produces the maximum Q according to Eq. (2.4) (lines 2-3 of Algorithm 2). This ensures video streaming without any stalling and produces the best QoE for most cases.

However, reducing stalling to zero may not always be the optimal choice. If the video quality is reduced for too long, the overall QoE might decrease. To address this challenge, we introduce additional steps in Algorithm 2 (lines 4-15). If the video quality reduction period, i.e., θ^* becomes greater than a threshold ΔT, the algorithm considers the presence of a short stalling. So, instead of choosing the θ value that would have produced zero stalling (θ^*), the value is decreased by $\Delta\theta$ to make the quality switch at a later time. This new value of θ is

checked against the QoE model, and if needed, the same process is repeated until optimum QoE is found. This is done to reduce the computational complexity of the algorithm because combinations of θ and L_j that produce zero stalling almost always result in the highest QoE. Inclusion of combinations that do not suppress stalling results in a very large set of values for θ and L_j, which increases the complexity of the algorithm and wastes valuable computational power in real-time streaming. Therefore, we consider cases with stalling only if the quality reduction period is very long, and the impact of video presentation quality may be higher than a small stalling.

In our experiments, we observed that the values computed at line 3 almost always produce the best QoE. Steps in lines 4-15 are for extreme cases where the reduction duration is too long, and the presence of stalling might produce better QoE; we found this to be rare. In our human subjective tests, out of 180 streaming cases, the impact of the reduction in video quality was higher than the inclusion of a small stalling in only 2 cases where quality reduction times were 9.6 sec and 10.1 sec. In those cases, instead of a zero stalling (Sn=0), test cases including stallings (Case 1: Sn=0.98 and Case 2: Sn=1.2) produced higher QoE. Therefore, the value of ΔT has been set at 9 seconds, which is an empirical value based on the experiments conducted in our current work and previous work [13]. The value of $\Delta\theta$ depends on the time difference between two quality switch points, which is set during encoding.

2.8 Performance Analysis

We evaluated the performance of ASDC in comparison with DASH and SVC. First, we determined the content distribution using ccnSim as described in section 2.6. Then, we used the CCNx [8] platform to emulate the streaming process, in which we implemented all the algorithms.

We installed CCNx 1.0 (Distillery 2.0) on machines running Ubuntu 14.04. The devices

Figure 2.11: Snapshots of video sequences. (a) Big Buck Bunny, (b) Elephants Dream, (c) Of Forests and Men, (d) Tears of Steel, (e) The Swiss Account, (f) Valkaama

were connected following the topology shown in Fig. 2.9 and the content popularity among clients was based on Zipf distribution with exponent 0.8 [48]. The ten routers and the main content server were connected through Ethernet. The ten clients and the respective edge routers were connected through 802.11n wireless technology. The content distribution was kept the same as the results from ccnSim (Table 2.4). Each intermediate router had a server application that maintained the videos in the CS and parsed the incoming requests. A modified VLC module [30] with default buffer size 1.5s was used at the client-side to stream the videos.

We selected 6 video clips: *Big Buck Bunny* and *Elephants Dream* from Xiph.org Media Collection [31], *Of Forests and Men* [49], *Tears of Steel* [32], *The Swiss Account* [50], and *Valkaama* [33]. The genres of the six videos cover animation, talk show, human with nature, sports, science fiction, and action movie, as shown in Fig. 2.11. Each video has 1280×720 pixels with a duration of 30 seconds. For DASH, two second segments were used and five different representations (approximately 1500, 2000, 2500, 3500, and 4500kbps) taken from ITEC Database [51] were available on the server. The source switch was achieved by manipulating the MPD and the content listing table (CLT). The video files were pre-chunked

and stored at CCN routers following the content distribution similar to Table 2.4. In case of SVC, we used MainConcept H.264/SVC Encoder [52] to generate videos with SNR (signal-to-noise ratio) scalability and five different quality layers: 1500kbps (BL), 2000kbps (EL1), 2500kbps (EL2), 3500 kbps (EL3), 4500kbps (EL4).

We first study the number of source switches for each of the videos streamed under different caching schemes when similar content distributions as Table 2.4 were used. The six videos are requested by ten clients following a Zipf distribution with exponent 0.8 [48]; the most popular video *Valkaama* was requested by ten clients while the least popular video *Of Forests and Men* was requested by two. The number of source switches for each video is an average from ten streaming cases, and the result is shown in Fig. 2.12, in which there have been three or more switches in most cases. The video *Valkaama* has a particularly low number of switches because of its high popularity and high viewer retention. Moreover, among different caching techniques, LCE is seen to produce the highest average number of source switches with CoA a close second. On the other hand, ProbCache and BTW have a comparatively lower average number of source switches. This pattern is consistent with the results found in Table II. Caching methods that obtained a higher percentage of contents from a higher number of sources also lead to a larger average number of source switches. It should be noted that the cache distribution in Table 2.4 is not a direct indicator of the number of source switches. If the same video is available at multiple nodes, a client may switch to a different node if the link quality is better or the current link deteriorates, e.g., a two-source distribution can result in two, three, or more switches depending on the network condition.

To compare the average CCN chunk arrival delays under DASH, SVC, and ASDC in the presence of a source switch, we streamed each of the six original videos five times with a double source switch. The video bitrate was 2000kbps, and the link capacity was 2500kbps. The chunk arrival time was obtained from the CCN client-side forwarder for a total of 1000 chunks. As shown in Fig. 2.13, between chucks 260 and 270 and again between chunks 675

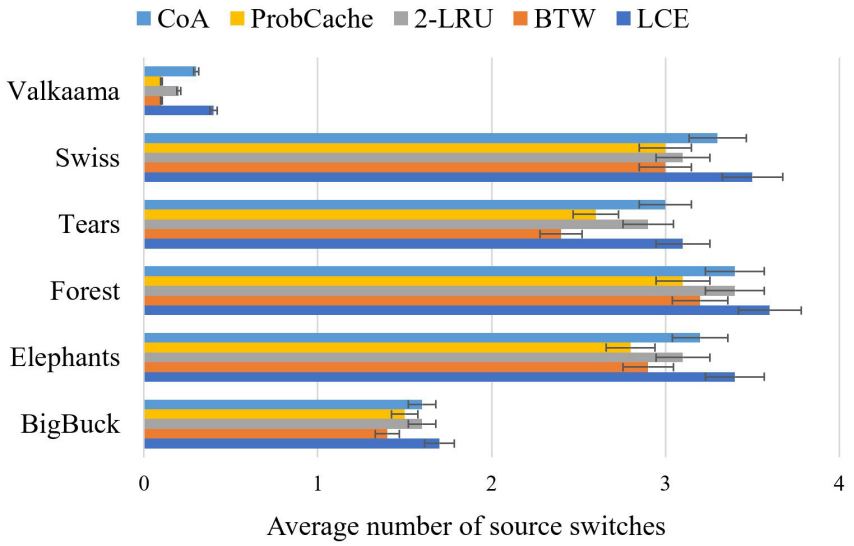


Figure 2.12: Avg. number of source switches for different caching methods

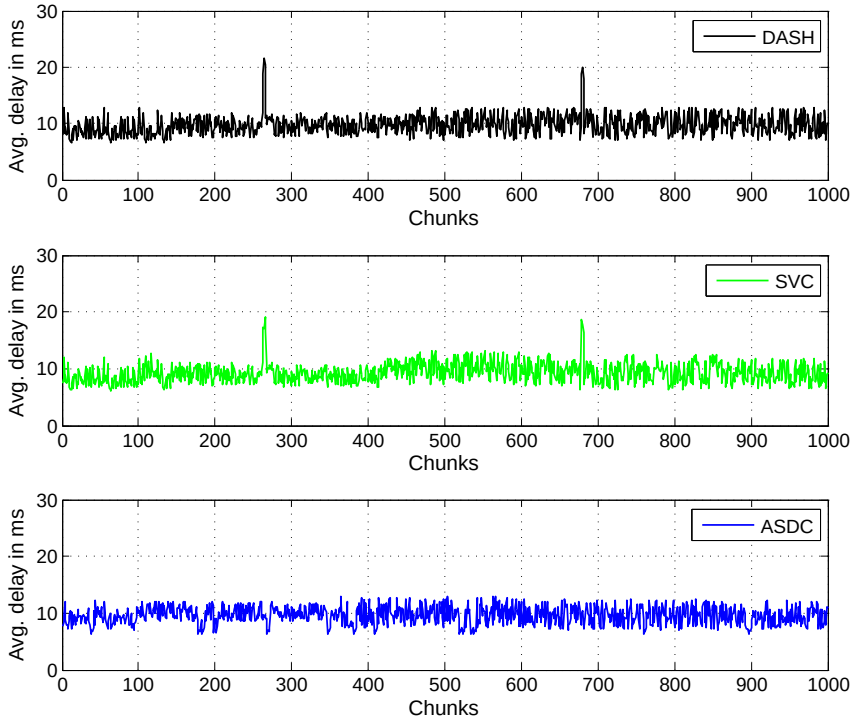


Figure 2.13: Delay variations of chunk arrival for DASH, SVC, and ASDC

and 685, there are radical increases in delays for DASH (peak at chunk 264 and 680, around 2.21 and 2.05 times the average, respectively) and for SVC (peak at chunk 266 and 679, around 2.06 and 2.02 times the average respectively), which took place due to the source switches. In the case of ASDC, there was no such increase in delay in the presence of the same switches. Although the average delays otherwise are similar, this sudden increase in delay for DASH and SVC raises the possibility of stalling.

We also study the average bitrate per request against throughput, which can be a measure of video quality from the consumer's side. Here the average bitrate per request is the sum of each received chunk's bitrate divided by the number of chunks requested by the client. High throughput means the client's link to the upstream is sufficient for requested content delivery while a lower throughput indicates a congested link. Fig. 2.14 shows the average results of streaming 25 videos with three source switches. DASH, SVC, and ASDC- all perform well for high throughput clients, while a difference is noticed in lower throughput. For clients with a throughput of more than 2500kbps, ASDC produces a 12.91% higher average bitrate compared to DASH and a 6.98% higher bitrate compared to SVC. The comparatively higher average bitrate results from the better quality adjustment ability of ASDC.

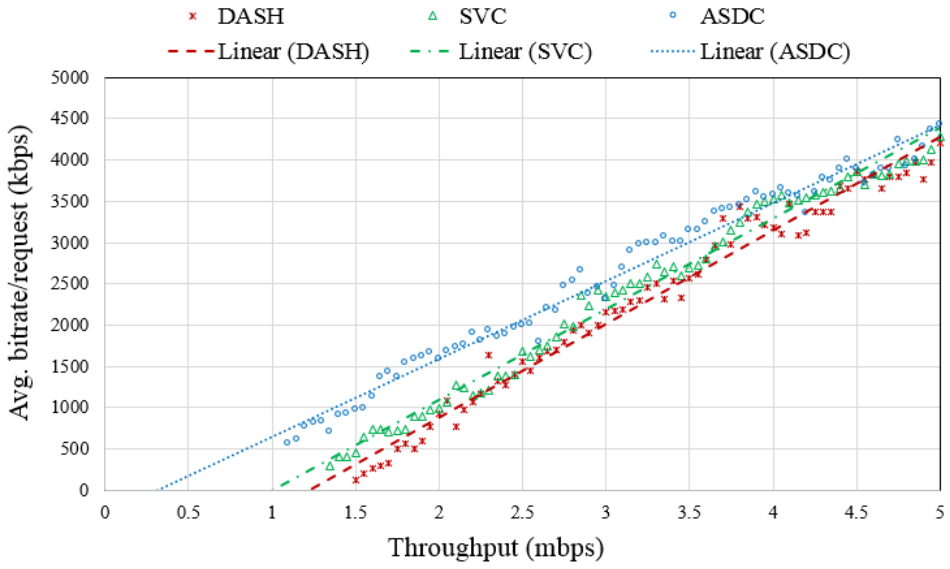


Figure 2.14: Average bitrate/request vs. throughput for DASH, SVC, and ASDC

To compare the estimated QoE for ASDC, DASH, and SVC in the presence of source switches, for each streaming method, we streamed each of the six videos three times for five different content distributions resulting from the five caching mechanisms (Table 2.4) producing a total of 270 samples. Fig. 2.15 shows the average estimated QoE for each video streamed under the five different cache distributions and the three streaming methods. Performance of ASDC, DASH, and SVC are similar for two videos where the number of source switches is smaller while the difference in the performance becomes more apparent for the four videos with a higher number of source switches (Fig. 2.12). For instance, in the case of *Valkaama* (less number of switches), estimated QoE from ASDC is 4.57% more than DASH and 2.09% more than SVC while in case of *Swiss* (more switches), estimated QoE from ASDC is 15.85% more than DASH and 10.38% more than SVC. On average, the overall estimated QoE improvement by ASDC is 12.42% more than DASH and 7.35% more than SVC. Between DASH and SVC, SVC performed better (average 4.71%) due to its ability to better adjust the video quality.

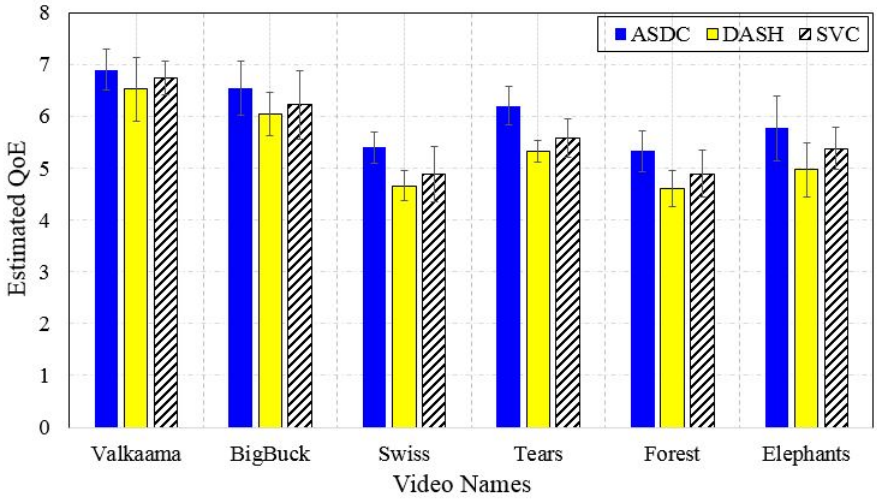


Figure 2.15: Estimated QoE for different videos under ASDC, DASH, and SVC

In Fig. 2.16, we further analyze how DASH, SVC, and ASDC compare against each other under the five caching methods in terms of estimated QoE. The QoE values in the figure are the average of eighteen video streamings (six videos streamed three times each) under each

caching technique and each streaming method. ASDC performs better than DASH and SVC for all the different caching methods in terms of estimated QoE. However, the performance gain is more pronounced in the case of CoA (10.68% for DASH, 8.10% for SVC) and LCE (11.40% for DASH, 6.65% for SVC), while it is less noticeable for BTW (2.65% for DASH, 2.03% for SVC) and ProbCache (3.62% for DASH, 2.09% for SVC). This difference arose from the higher number of source switches taken place for LCE and CoA while the opposite is true for BTW and ProbCache. We observe that with the increase in source switches, i.e., the possibility of stalling, the advantage ASDC becomes more apparent.

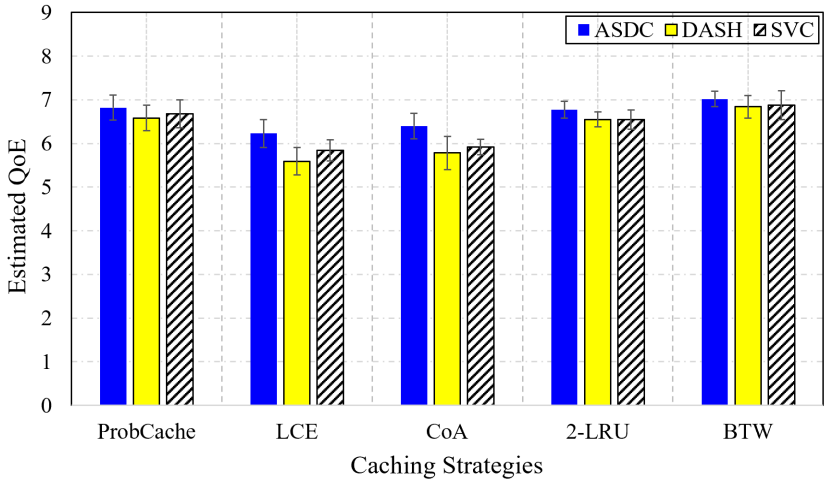


Figure 2.16: Comparison of estimated QoE between ASDC, DASH, and SVC under different caching methods

In our experiments, we noticed stallings ranging from 28 frames to 121 frames in length (1.12s to 4.84s for 25 frames/sec, average 65.328 frames or 2.613s) depending on the network configuration and the transmitted video. Comparison of the following buffer sizes used in various streaming models focused on ICN, 5s in [53] and [54], 1s in [55], and 3s in [56] with our observed stalling duration shows that source switching delay may cause stalling. As the client buffers are not always full, especially in cases where end-to-end bandwidth is similar to streaming bit rate, a larger client buffer would be required to mitigate the stalling resulted from source switching. However, although a larger buffer size can help improve the continuity, that may lead to higher startup and jumping latency.

Computational complexity plays an important role in algorithm selection. We compare and analyze the complexity of the proposed algorithm against DASH and SVC. Complexity is measured exclusively on a computer with an Intel Core i7-5820k (3.30GHz) processor, 8GB of RAM, and Ubuntu 14.04 operating system. Complexity is evaluated by the average computational time over 1000 CCN chunks of the *Big Buck Bunny* video under a different number of source switches. In Table 2.6, it can be noted that the proposed algorithm has a similar computation time as both DASH and SVC, and thus can be implemented in real-world video streaming. Moreover, the computation time increases with the increase in the number of source switches, which is expected as the source switching necessitates the quality adjustment calculation. Note that ASDC has a low computational overhead despite additional QoE calculations because the second algorithm of ASDC goes into effect only when there's a possibility of stalling; otherwise, the first algorithm handles the streaming. Another factor that impacts the computation time is the number of video representations available at the source. As we consider the same number (five) of representations for all the streaming techniques, this parameter is excluded from the comparison.

Table 2.6: Average computational complexity

Mechanism	No switch	2 switches	4 switches
DASH	7.6 ms (±0.13)	11.2 ms (±0.19)	14.2 ms (±0.28)
SVC	9.3 ms (±0.26)	13.3 ms (±0.30)	16.4 ms (±0.33)
ASDC	9.1 ms (±0.19)	12.8 ms (±0.23)	14.9 ms (±0.24)

2.9 Conclusion

We studied the impact of source switching, brought by distributed caching, on the QoE of video streaming in CCNs. We found that when the source switching increases, both the frequency and length of stalling events increase, which was detected by human subjects. The overall satisfaction with the received videos declines when the number of switches increases.

The results of correlation analysis and PCA indicate that the satisfaction with stalling is correlated with the overall MOS. Specifically, in high bitrate videos with high clarity, the satisfaction with stalling has an obvious linear correlation with the overall MOS; while in low bitrates with lower video clarity, both the satisfaction with stalling and the clarity play significant roles in the overall MOS.

We demonstrated that most videos were streamed from multiple sources in CCN because of in-network caching. Stalling plays a major role in QoE degradation, and we designed the ASDC algorithm for maintaining QoE during switches between multiple content sources. ASDC relies on quality adaptation based on scalable video streams and a QoE prediction model that characterizes stalling effects. Experimental results have shown that ASDC improves QoE performance over DASH for different caching methods and a variety of video contents in CCN.

Chapter 3

Cross-Layer QoE-Aware Video Streaming over SDR

3.1 Background

With the increasing deployment of camera sensors in various wireless imaging applications, such as surveillance, industrial process control, intelligent transportation, and telemedicine, there is an emerging need to support the efficient delivery of video traffic in wireless environments. In many human-centered wireless applications, human users rely on the received images or videos to make critical decisions, such as identifying intruders from wireless surveillance cameras or determining the quality of products for industrial applications. Maintaining good perceptual quality or Quality of Experience (QoE) of received videos is a major design challenge for these applications.

In the radio communication field, software-defined radio (SDR) uses software instead of dedicated hardware to implement signal processing components such as frequency mixers, filters, modulators and demodulators, synchronizers, amplifiers, etc. With the flexibility and the reconfiguration capability to serve a wide range of changing radio protocols in real-time, SDR has been applied in many domains such as cognitive radio [57], heterogeneous IoT

networking [58], and 5G networks [59]. SDR has the potential to satisfy the needs of video applications in dynamic wireless environments.

On the other hand, the unique QoE properties for visual information should be taken into consideration in video communication schemes. For example, based on many studies (such as the ones in [13, 16, 36, 60, 61]), the QoE for video streaming is directly related to application layer parameters such as initial delay, stalling, video clarity, disruption recency, etc. The perceptual quality of compressed videos under the same bit rate can vary with different video content characteristics, such as the level of spatial details (e.g., brightness, edges, and texture complexity) and temporal details (e.g., the extent of motion) [62, 63]. Furthermore, when a video is compressed under a bit rate constraint, the choices of video clarity level, spatial resolution, and frame rate could jointly contribute to perceptual quality [64]. The impact of packet loss on QoE is related to the locations and the patterns of lost packets in a video bitstream, and the visibility of packet loss also depends on the content of the video [65, 66].

Considering the properties of QoE, it is natural to adopt a cross-layer design approach, which can perform optimization based on parameters from different layers that could influence QoE. It would also be beneficial to leverage the capabilities of SDR systems for cross-layer design. To this end, we propose a new cross-layer approach for video communication over SDR, with the objective of providing satisfactory QoE for end-users. Our contributions are summarized as follows.

1. First, we designed and implemented an SDR platform that can achieve real-time video communication in both simulation and hardware. Our platform is built upon GNU radio (an open-source SDR platform), Universal Software Radio Peripheral (USRP) B210 hardware, and GStreamer (a comprehensive library for video processing). Our platform has integrated QoE measurement components for monitoring the QoE of received videos in real-time, utilizing the latest findings in the field of QoE for videos. The platform provides the capability of jointly adjusting physical layer parameters and

video application settings to maximize QoE in changing wireless conditions.

2. Second, we measured QoE under a variety of hardware settings and video use cases, and based on these measurements, we analyzed how the parameters in the physical layer and the application layer contribute to the QoE of videos presented to application users.

3. Accordingly, we designed a cross-layer solution that jointly adjusts the physical layer parameters and the application settings to maximize the QoE for video communication over SDR. We integrated the proposed cross-layer solution into our platform and evaluated its performance in different test conditions. Our solution advances existing cross-layer studies by: 1) considering the latest findings in QoE research for videos; and 2) adjusting communication parameters based on measurement results from a practical SDR platform.

3.2 Related Works

3.2.1 SDR-Based Platforms for Video Communication

A few studies have designed and implemented SDR systems for video communication via simulation or hardware implementation. A real-time video transceiver SDR testbed was implemented based on USRP 2943R and LabVIEW in [67]. In this testbed, high packet efficiency was achieved by optimizing transmission and reception during real-time video transmission. Additionally, four directional antennas were used to attain a high frame rate (30 fps). A real-time video streaming simulation platform was proposed in [68], comprising of a web camera, GNU Radio [3], GStreamer [4], Universal Software Radio Peripheral (USRP), and RTL-SDR. At the receiver side, buffering is done to store a live video stream, and then the video is played with Mplayer. Different from the simulation-based platform in [68], a real-time video streaming system was developed using USRP B200 hardware and 850 Mhz-

6.5 GHz antenna in [69]. The system in [69] has tested video transmission under different modulation schemes, including BPSK, QPSK, 8-PSK, 16-QAM, 64-QAM, and OFDM.

SDR has also been utilized for mobile sensing systems. In [70], a UAV radio telemetry (UAV-RT) system was proposed that integrates VHF radio telemetry equipment with an unmanned aerial system. The system uses a modified hexacopter in conjunction with an SDR receiver to track radio-tagged wildlife. In [71], an application of UAVs in video surveillance and search and rescue operations was designed through an ad-hoc GSM network in areas without available infrastructure. Each UAV was fitted with a GSM base station built using SDR that was implemented via LimeSDR [72]. A similar SDR-based ad-hoc GSM network based on USRP was demonstrated in [73], where applications of UAVs during natural disasters and surveillance were discussed.

3.2.2 Cross-layer Video Communication

We highlight cross-layer solutions that provide QoS or QoE support for wireless video communication, which are most related to our work.

Many cross-layer studies aimed at improving QoS parameters, such as the PSNR (peak signal to noise ratio), transmission delay, and packet loss rate, based on the information available at different layers. In [74], a cross-layer video streaming algorithm over mobile ad hoc networks was designed to minimize end-to-end delay and packet loss rate, and it selected the most efficient PHY mode of IEEE 802.11 multi-rate service based on the information available at the application, data link, and physical layers. The cross-layer optimization strategy in [75] jointly optimized the application, data link, and physical layers. The following parameters were adjusted to maximize the PSNR of the received videos: video source rate, time slot allocation, and modulation scheme. Cross-layer design techniques for video streaming over cooperative networks were studied in [76]. Here, the problem of the joint control of video encoding rate, relay selection, and power allocation was formulated as a mixed-integer nonlinear problem with the objective of maximizing PSNR. In [77] and [78],

the PSNR of scalable video streaming was maximized through cross-layer adaptation, based on parameters such as video coding rate, radio resource scheduling policy, and modulation technique.

Several cross-layer studies also addressed the problem of improving the QoE or perceptual quality of videos. A structural similarity (SSIM)-based cross-layer optimization scheme for wireless video streaming was proposed in [79], considering H.264/AVC encoding parameters in the application layer and modulation and channel coding modes in the physical layer. This scheme was shown to achieve good perceptual video quality in terms of mean opinion scores (MOS). In the cross-layer video streaming scheme over wireless ad hoc networks in [80], video packets were scheduled according to the states of frame buffers at the destination nodes to increase playback continuity. In the quality-aware rate control and adaptation scheme for real-time video communications over Long Term Evolution (LTE) networks in [81], the transmission bit rate was automatically adjusted according to the estimated packet loss, based on both packet queueing delay and transmission delay. This feedback-free system was found to provide satisfactory QoE while maximizing the number of supported users.

3.2.3 Novelty of Our Work

While the aforementioned systems in Section 3.2.1 have demonstrated the feasibility and benefit of SDR for video communication and mobile sensing applications, our proposed SDR platform advances them by integrating the specific QoE properties for video applications in an SDR platform.

Our proposed cross-layer video communication solution jointly adjusts parameters in the application and physical layers. Different from the aforementioned studies in Section 3.2.2, the proposed cross-layer solution leverages the specific properties of QoE for video, and it is designed based on the QoE measurement results from a practical SDR platform.

3.3 SDR Platform Design

We designed and implemented a half-duplex video transmission system supported by USRP devices. The solution features a hardware platform that emulates a digital communication system via software radio. With the capability to deliver real-time video packets, the system integrates interfaces for cross-layer adjustment of transmission parameters, and it also supports the measurement of QoE metrics at the receiver side.

3.3.1 Components for Video Communication over USRP

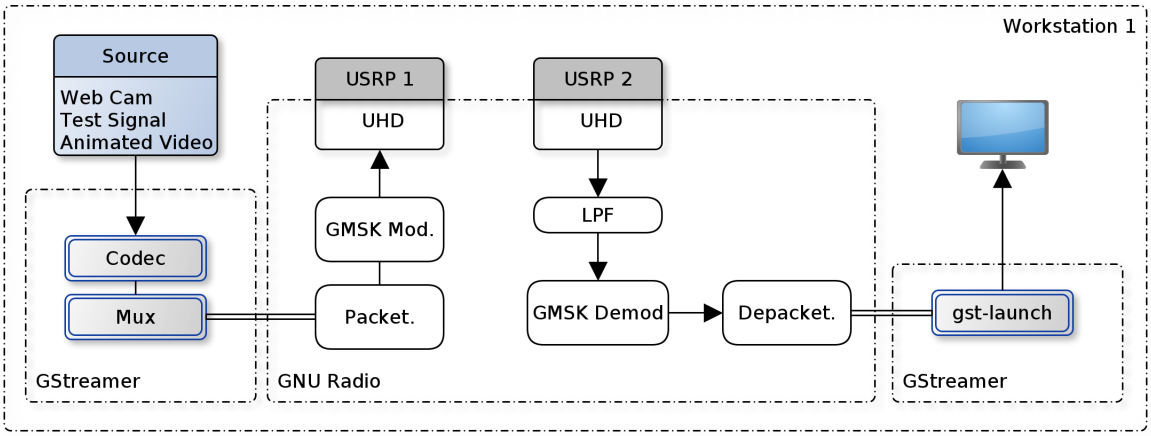


Figure 3.1: System overview

The overview of our transceiver design is shown in Fig. 3.1 and the configuration parameters for our experiments are noted in Table 3.1. The main components of the system are described below.

- *Video Sources:* Our system accepts video signal from three different sources: real-time video from webcam, stored video file, and test video signal from GStreamer's *videotestsrc* element [4].

- *GStreamer:* Gstreamer [4] is a comprehensive video processing library. It can format the source video frames, add text overlay, re-scale, compress, and multiplex into the transport stream. A Unix pipe (*mkfifo videotx.ts*) is created to connect the GStreamer

Table 3.1: List of configuration parameters for video streaming

Parameter	Value
Environment	Indoors
Distance between Tx and Rx	3 ft.
Frequency	2.68 GHz
Bandwidth	3 MHz
Modulation scheme	GMSK
Transmitter gain	40 dB
Receiver gain	35 dB
Source video resolution	1280×720 pixels
Frame rate	24 fps
Streaming encapsulation	MPEG-TS

output with the input (*filesink*) of the GNU Radio Companion (GRC), which is shown in Fig. 3.2.

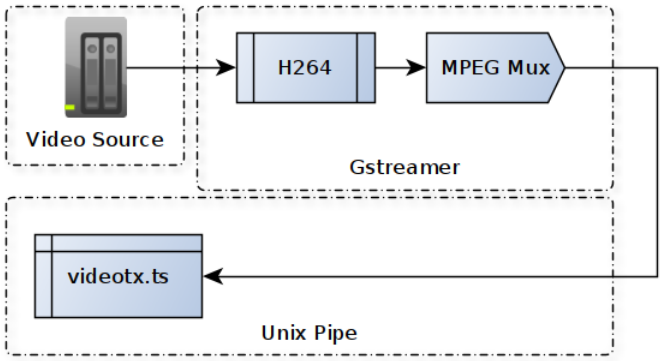


Figure 3.2: A Unix Pipe connecting GStreamer to GRC

- *GNU Radio:* SDR transmitters and receivers are designed using GRC. The GNU Radio transmitter captures the video frames from GStreamer and processes them for transmission over the RF channel, i.e., conducts necessary signal processing. GMSK modulation scheme is used for improved spectral efficiency and lower power consumption at the receiver [82].

- *USRP:* The platform uses Universal Software Radio Peripherals (USRP) B210 with an operational frequency range of 70 MHz to 6 GHz. The USRPs have USRP Hardware Driver (UHD) driver support, an open design schematic, and seamless integration features with the GNU Radio platform for signal processing. As shown in Fig. 3.1,

one USRP is used as the transmitter and another USRP as the receiver.

- *Antennas:* Ettus VERT2450 antennas (dual Band 2.4 to 2.48 GHz and 4.9 to 5.9 GHz) are used for transmission and reception.

- *Computer:* Each USRP is connected to a computer that runs the GNU Radio and GStreamer programs.

- *Video player:* Most of the results are visualized using the *playbin* property from Gstreamer. However, VLC Player and MPlayer may work as alternatives to play the videos for both real-time streams and stored videos.

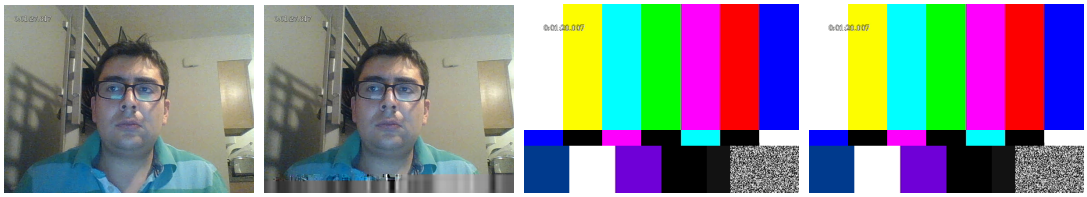
At the transmitter side, the source video is first encoded in H.264 and then passed to a packet encoder. Here we use 1 bit/symbol and 2 samples/symbol. Then the packetized data is modulated using the GMSK modulation and sent out to the USRP connected to GNU Radio.

The receiver USRP captures the transmitted data and sends it to the GNU Radio. The data is then demodulated and decoded to generate a bitstream. This bitstream is then sent to either a file sink to be either stored or played in real-time. Snapshots of transmitted and received frames for different videos are shown in Fig. 3.3. More details about the platform components, including GNU Radio flowgraph, can be found in Section 5.2.

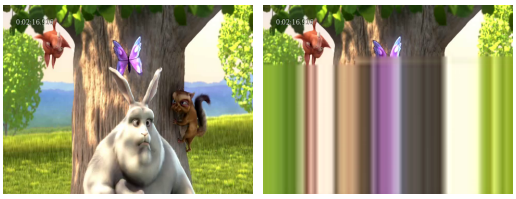
3.3.2 Cross-layer Parameters

In our system, to support the design of cross-layer video communication, we provide the capability to adjust several input parameters from the application (APP) and physical (PHY) layers based on the bandwidth (PHY) constraints. The adjustable parameters are as follows: video resolution (APP), encoding rate (APP), frame rate (APP), transmission frequency (PHY), transmission gain (PHY), and reception gain (PHY).

The value of available bandwidth directly controls the video encoding rate. For good video quality, the recommended encoding rate should be 64% of the bandwidth [83]. Therefore,



(a) Webcam - Tx (b) Webcam - Rx (c) Color Bars - Tx (d) Color Bars - Rx



(e) Stored video - Tx (f) Stored video - Rx

Figure 3.3: Snapshots of transmitted and received video frames

the minimum bandwidth required for a particular encoding rate can be found by multiplying the encoding rate by 1.5625. For instance, if the encoding rate is 128kbps, the recommended minimum bandwidth will be (128*1.5625) or 200kbps.

Due to the varying conditions of wireless operations, several energy-detection and efficiency-based methods mitigate situations that deteriorate the performance of the system [84, 85, 86]. In our case, we included the SNR estimation block from the GNU Radio library, which is aimed to maintain an optimal SNR. This adaptive procedure employs the M2M4 algorithm [85] and operates over the PHY layer by adjusting the gain at the receiver end.

3.3.3 QoE components

Based on the latest results in QoE research, frame-level video presentation quality and QoE drop due to video stalling are among the most prominent factors affecting overall viewers' QoE [60], and they are linearly correlated to mean opinion scores (MOS) collected from human subjective tests [13, 36, 61].

We propose to measure the following three metrics to evaluate the received video quality: (1) video presentation quality (P_n), (2) QoE drop due to stalling (S_n), and (3) frame loss rate (F_L). These metrics are integrated into our system; more specifically, they are computed

at the receiver side in real-time. This provides the capability to assess the QoE after each video session and to adjust the input configurations accordingly.

The instantaneous video presentation quality (P_n) or video clarity can be estimated by a frame-level video quality assessment (VQA) model. Both full-reference and no-reference VQA models can be used to measure P_n. Examples of full-reference models include Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) [41] index. On the other hand, examples of no-reference models include Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [87] and BLIINDS-II [88]. Any of the aforementioned models can be used with our platform. The choice of the VQA model depends on the specific application case.

Stalling refers to the momentary disruption in the fluidity of video playback due to network impairments. The QoE drop due to stalling (S_n) can be computed using the Streaming QoE Index (SQI) [36] model described in Section 2.5. Prediction accuracy of this model in comparison with Mean Opinion Scores (MOS) from human subjective tests can be found in [13, 61]. The following input metrics are used in the calculation of S_n: video frame rate, initial delay, stalling length, stalling frequency, and the location of the stall during the video playback. Stalling and initial delay have different impacts on QoE, and therefore they are computed with appropriate weights as specified in the SQI framework [36].

The third QoE metric, frame loss rate (F_L), can be computed by comparing the number of transmitted frames (N_{T_x}) and the number of received frames (N_{R_x}) for a particular video:

$$F_L = \frac{N_{T_x} - N_{R_x}}{N_{T_x}} \tag{3.1}$$

To extract the frames from the video clips, *FFmpeg* [89], a popular open-source multimedia framework, can be used.

3.4 Measurement Results: Which Parameters Influence QoE?

To determine how each of the input parameters influences the viewers' QoE, we studied the values of the QoE metrics for a wide range of input parameter configurations as shown in Table 3.2. We streamed stored videos (*Big Buck Bunny*, an animated movie downloadable from the Peach open movie project [90]) with resolutions up to 1920×1080, and real-time webcam videos up to 1280×720. The frequency of the operational band was varied among the UHF Band (917.5 MHz), the S-Band (2.68 GHz), and the C-Band (SHF, 5.656 GHz). For the UHF and SHF bands, there is a high level of occupancy due to a significant number of wireless technologies being allocated there. In the majority of our experiments, we achieved the best results by transmitting on the S-Band. Due to the absence of an active frequency sensing hardware component in our platform, we were unable to detect interference-free channels and were limited to these three bands. For all three bands, the RF-channel bandwidth was set at 3 MHz.

Utilizing the M2M4 estimator, we measured different SNR levels for multiple transmission gains in the absence of additive noise. We observed that values before 35dB produced an estimated SNR around 9dB at the receiver, and it led to signals from which the video stream could not be decoded. Given these results, channel gain values ranging from 40dB to 70dB have been selected for the following experiments.

Table 3.2: Range of input parameters used in our experiments

Input Parameter	Values
Video Resolution (px)	176×144, 352×240, 480×360, 1280×720, 1920×1080
Frame Rate (fps)	10, 12, 15, 16, 20, 24, 30
Encoding Rate (kbps)	128, 256, 512, 1024
Tx Gain (dB)	40 – 70
Frequency (Hz)	917.5M, 2.68G, 5.656G

Although we tested our system with many different input parameter values as shown in Table 3.2, some of the input parameter combinations led to extreme frame loss rates, and the

received video files could not be played by the player. We used a total of 34 combinations in our analysis, which are shown in Figs. 3.4, 3.5, and 3.6. These are the valid combinations we found that generated playable video files on the receiver end. Furthermore, among these 34 samples, a few combinations produced video files from which only the individual received frames could be extracted but too many frames were missing to play the video or compute the presentation quality reliably. As a result, we could not calculate P_n for 2 videos and S_n for 5 videos. Therefore, Fig. 3.4, Fig. 3.5, and Fig. 3.6 shows results from 34, 32, and 29 input combinations respectively.

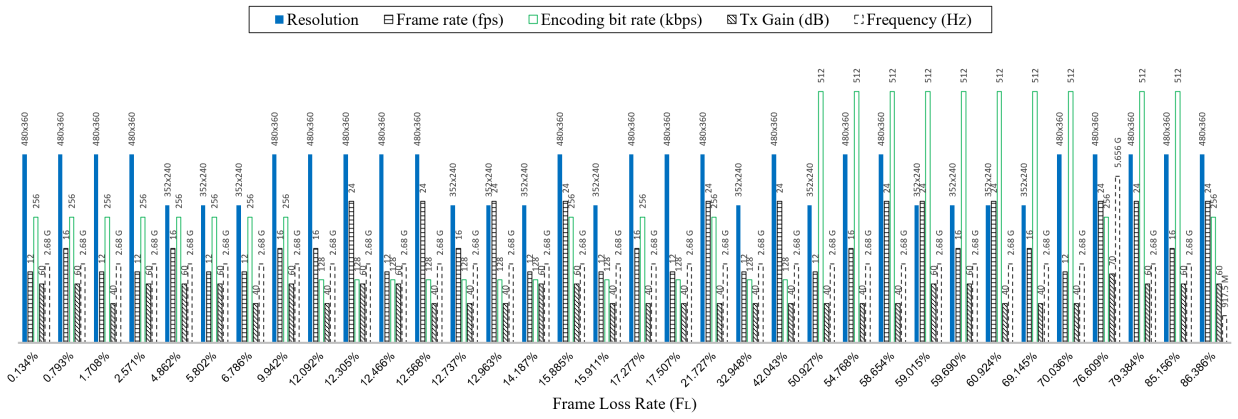


Figure 3.4: Relationship between the input parameters and F_L

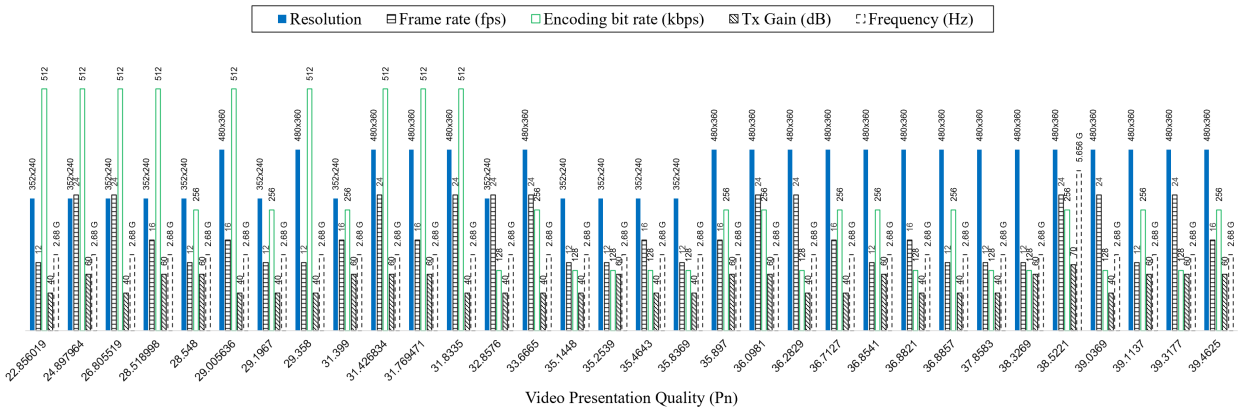


Figure 3.5: Relationship between the input parameters and P_n

Fig. 3.4 shows the relationship between the input parameters (encoding rate, Tx Gain, video frame rate, resolution, and frequency) and output frame loss rate (F_L). We computed

F_L using Eq. 3.1. The values of F_L ranged from 0.13% to 86.39%. High encoding rate and high frame rate led to higher frame loss. For example, when the encoding rate was raised from 256 kbps to 512 kbps keeping all other parameters constant, the frame loss rate increased from 1.71% to 70.04%. Similarly, when the video frame rate was changed from 16 fps to 24 fps, the frame loss rate went up from 0.79% to 15.89%. Frequency had a large impact on the frame loss as well; however, we did not see a direct or inverse relationship between frequency and frame loss. For instance, both increasing (from 2.68 GHz to 5.656 GHz) and decreasing (from 2.68 GHz to 917.5 MHz) the transmission frequency resulted in a higher frame loss rate (76.61% and 86.39% respectively). This result can be explained by the high interference present on those two channels, as described at the beginning of this section.

Fig. 3.5 represents the relationships between the input parameters and the video presentation quality (P_n). For computing P_n, we used the BRISQUE [87] framework utilizing the default feature model in Matlab. The BRISQUE score has a range of [0, 100] and a lower score indicates better perceptual quality. The values of P_n were between 22.856 and 39.463. Video encoding rate had the largest impact on P_n and a higher encoding rate usually led to a lower P_n value, e.g., 36.886 vs. 29.358 for increasing the encoding rate from 256 kbps to 512 kbps. Resolution had the next biggest effect on P_n, and a higher resolution almost always resulted in a higher P_n. For instance, with all other parameters unchanged, enhancing the video resolution from 352×240 to 480×360 increased P_n from 28.548 to 39.114.

Fig. 3.6 demonstrates the QoE drops associated with playback stalling (S_n). We calculated S_n using Eq. (2.2). The values of S_n stretched between -13.528 (highest quality drop caused by stalling) and 0 (lowest quality drop caused by stalling). Encoding rate had the highest impact on S_n followed by Tx gain. Higher encoding rates (e.g., 512kbps) consistently resulted in larger quality drops caused by stalling compared to videos with a lower encoding rate (e.g., 128kbps or 256kbps). For example, reducing the encoding rate from 512kbps to 256kbps, with all other parameters kept constant, also lowered the quality drop due to

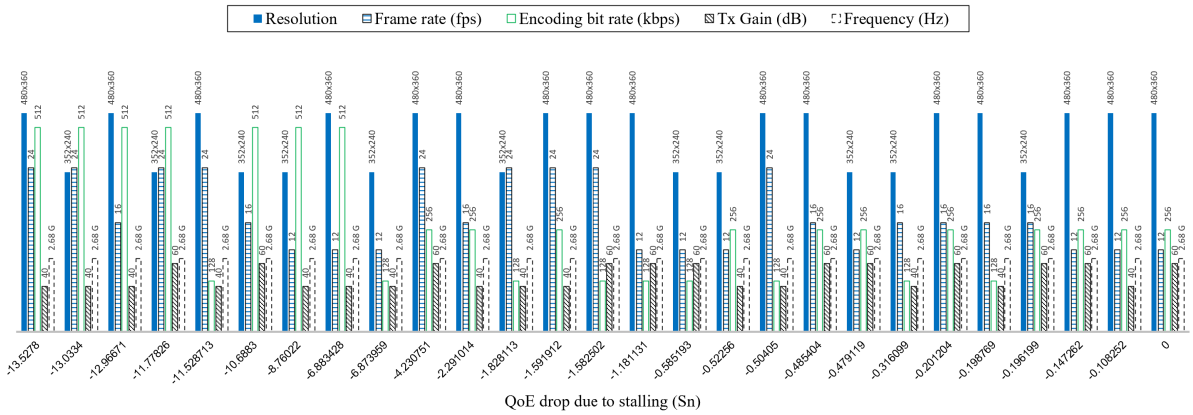


Figure 3.6: Relationship between the input parameters and S_n

stalling (value of S_n went up from -13.528 to -1.592).

3.5 QoE-Driven Cross-Layer Video Communication

We designed a cross-layer video communication solution based on our observations of the preliminary measurement results. The cross-layer model works on two layers: application layer and physical layer. Our goal was to adjust the following input parameters to maximize the QoE of received videos: video resolution (APP), encoding rate (APP), frame rate (APP), transmission power (PHY), and transmission frequency (PHY).

To quantify the degree of influence that each input parameter has on the video quality, we calculate the standard deviations of the output metrics. A larger standard deviation in the output metric due to changing an input parameter means a higher impact and vice versa.

Table 3.3: Impact of cross-layer parameters on video quality: Standard Deviation

Cross-Layer Parameter	Frame Loss Rate (%)	P_n	S_n
Encoding Rate	0.4629	6.053	7.923
Frame Rate	0.1421	2.79	2.45
Resolution	0.1801	5.56	6.019
Tx Gain	0.21487	1.95	6.07
Frequency	0.4985	N.A.	N.A.

For each input parameter, we find the highest variation (standard deviation) it caused in the output metrics over the whole set (a total of 34, as described in Section 3.4) of available

input combinations. This maximum standard deviation value denotes the input parameter's impact on the output video quality metric. Table 3.3 lists the highest standard deviations of the quality metrics (F_L, P_n, and S_n) for each of the five input parameters (encoding rate, frame rate, video resolution, transmission gain, and frequency).

To create a priority list for input parameter adjustment, the standard deviation values are proportionally translated into a relevance score that represents the overall impact of a single input parameter concerning a specific output metric. Table 3.4 shows the relevance scores on a scale of 1 to 5, where 5 denotes the highest impact or relevance. The rightmost column is the total overall score equated by adding the three individual relevance scores.

It should be noted that frequency is found to be a parameter of high relevance. However, due to the limitations of our testbed described in Section 3.4, this parameter is not taken into consideration in our current implementation. However, the frequency parameter should be considered in other wireless environments if necessary.

Table 3.4: Relevance Scores of the cross-layer parameters: Priority list

Cross-Layer Parameter	Frame Loss Rate (%)	P_n	S_n
Encoding Rate	4	4	4
Resolution	2	3	2
Tx Gain	3	1	3
Frame Rate	1	2	1
Frequency	5	N/A	N/A

We design our cross-layer protocol (CL-SDR) as a predictive scheme that will provide an optimal configuration of the cross-layer input parameters based on the measured QoE metrics of the previous streamings. The algorithm works in the following steps:

- **Step 1:** Copy the relevance scores from the Table 3.3 and 3.4, both of which are based on existing results.
- **Step 2:** Stream video with the input parameters (encoding rate, frame rate, video resolution, transmission gain, and frequency) set to their default values.
- **Step 3:** Measure both the signal SNR and the QoE metrics (F_L, P_n, and S_n) from the

received video. Estimate SNR and set the corresponding gain at the receiver in order to mitigate channel fading and degradation.

- **Step 4:** Adjust the cross-layer parameters following the relevance scores, i.e., priority list.
  - The parameter with the highest priority will be adjusted first. The adjustment direction (increase or decrease) and amount (how much) will be determined from the previous streaming data.
  - Stream video with the updated input parameters. Measure the QoE metrics after a set interval (δt).
  - The parameter with the second-highest priority will be adjusted next and so on.
  - Repeat until all combinations are explored and stop when the best QoE score is found.
- **Step 5:** Update Table 3.3 and Table 3.4 if the value of an input parameter falls out of the range of the pre-existing data.

As we discussed in Section 3.3.2, the video encoding rate is directly controlled by the available bandwidth. Therefore, we consider available bandwidth when selecting the encoding rate for a certain input parameter combination. For a given bandwidth value, we calculate the maximum possible encoding rate that could provide good video quality using the procedure described in Section 3.3.2. Then, the upper limit of the encoding rate is set at that number during the input parameter adjustments.

To illustrate how the CL-SDR algorithm works, let us consider a video stream where we want to maximize the video presentation quality (P_n). Suppose the priority list has similar relevance values as shown in Table 3.4. In that case, the parameter with the highest relevance score is the encoding rate (score= 4) followed by resolution (score= 3). Therefore, the algorithm will adjust the video encoding rate first and compute the QoE. After a certain

period δt, the resolution will be changed, and then similarly, the other parameters will follow. The combination producing the best P_n will be selected for streaming. The values of δt can be set to decide how frequently the streaming parameters are adjusted. We set δt to 60 seconds in our experiments.

3.6 Performance Analysis

3.6.1 General Performance

We measured three QoE metrics of the received video files: video presentation quality (P_n), QoE drop due to stalling (S_n), and frame loss rate (F_L). Our platform can stream videos with acceptable QoE for video resolutions up to 1280×720, encoding rates up to 1024kbps, and frame rates up to 30fps. The corresponding QoE scores are listed in Table 3.5. We observed that videos with higher resolutions (≥1920×1080) and encoding rates (≥1024kbps) were more demanding in terms of computational power and enhanced graphics hardware (e.g., dedicated graphics card) will provide better performance.

Table 3.5: General performance of the platform: QoE Scores

QoE Metrics		
P_n	S_n	F_L
39.463 to 12.649	-17.641 to 0	86.386% to 0.134%

We also observed our platform's performance under various distances between the transmitter and receiver. We streamed both stored and live webcam videos with fixed input parameter configurations (*Stored:* 352×240, 12fps, 256kbps, 40dB, 2.68 GHz; *Webcam:* 352×240, 15fps, 512kbps, 40dB, 2.68 GHz) and varied the Tx-Rx distance from 30cm to 210cm. For stored videos, the ranges of the resulting QoE metrics are as follows: P_n (27.978 to 32.302), S_n (-6.886 to -2.084), and F_L (25.46% to 51.79%). In case of webcam video: P_n (24.18 to 27.819), S_n (-0.799 to -3.024), and F_L (5.46% to 17.76%). We found that the QoE scores of the received video files were stable within that distance range (30-210cm).

We inserted an additive noise source in GNU radio in order to evaluate the performance for different noise magnitudes. Gaussian noise is added to the modulated signal before the USRP. After manually adjusting the receiver gain, we managed to maintain an SNR equal to 11, which exhibited a suitable performance regardless of the disruptive noise with amplitudes that range from 0 to 1.40×10^{-4}. Results are shown in Table 3.6.

Table 3.6: Estimated SNR levels for different Tx Power

Add. Noise	Rx Gain	SNR
0	35 dB	11 (10.41 dB)
7.20×10^{-4}	40 dB	
8.70×10^{-4}	45 dB	
1.00×10^{-4}	50 dB	
1.25×10^{-4}	55 dB	
1.40×10^{-4}	60 dB	

3.6.2 Cross-layer Adjustments

To evaluate the performance of our proposed cross-layer design, we streamed videos under a similar hardware setup as described in Section 3.3. We used both stored (*Big Buck Bunny*) and live webcam videos for this evaluation. The ranges of the cross-layer input parameters (video resolution, frame rate, encoding rate, Tx gain, and frequency) utilized in the evaluation are shown in Table 3.2. We conducted our evaluations under three different bandwidth constraints: 200kbps, 400kbps, and 800kbps.

We compared our proposed algorithm (CL-SDR) against the following two benchmarks. First, baseline QoE scores were produced by the platform with default input parameter configurations (no adjustment). Second, a customized video quality adaptation model based on [64], which optimizes video quality using three application layer parameters: quantization parameter (QP), spatial resolution, and frame rate (QP_R_F). When controlling the bit rate of a video stream, this model prioritizes QP over spatial resolution and frame rate (smaller QP with lower resolution and frame rate is preferred to larger QP combined with higher resolution and frame rate). Additionally, for the same QP value, the QP_R_F model will pick

the input configuration with a higher frame rate value as the videos used in our experiment have fast motion, and therefore, the frame rate should be kept close to the original file [64].

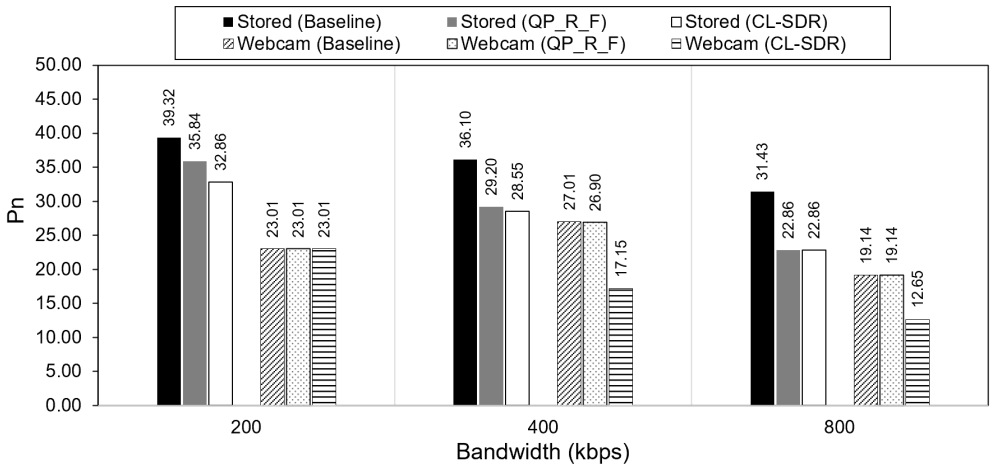


Figure 3.7: P_n comparison

Table 3.7: Cross-layer parameter config. to maximize P_n

Bandwidth (kbps)	Video Source	Parameter Configuration: P_n Scores Video resolution (px), Frame rate (fps), Encoding rate (kbps), Tx gain (dB), Frequency (GHz)		
		Baseline	QP_R_F	CL-SDR
200	Stored video	480×360, 24, 128, 60, 2.68	352×240, 12, 128, 40, 2.68	352×240, 24, 128, 40, 2.68
	Webcam	176×144, 15, 128, 60, 2.68	176×144, 15, 128, 60, 2.68	176×144, 15, 128, 60, 2.68
400	Stored video	480×360, 24, 256, 60, 2.68	352×240, 12, 256, 40, 2.68	352×240, 12, 256, 60, 2.68
	Webcam	176×144, 15, 256, 60, 2.68	176×144, 25, 256, 40, 2.68	480×360, 15, 256, 40, 2.68
800	Stored video	480×360, 24, 512, 60, 2.68	352×240, 12, 512, 40, 2.68	352×240, 12, 512, 40, 2.68
	Webcam	176×144, 15, 512, 60, 2.68	176×144, 15, 512, 60, 2.68	480×360, 15, 512, 60, 2.68

Fig. 3.7 and Table 3.7 show the comparison of P_n scores generated by the baseline configuration, the QP_R_F model, and the CL-SDR algorithm. In some cases (e.g., 200kbps and 800kbps Webcam), more than one model came up with the same configuration, and hence the scores were similar. In general, both QP_R_F and CL-SDR performed notably better than the baseline configuration. For stored videos, CL-SDR improved average P_n scores by 21.54% and 3.52% compared to the baseline and QP_R_F model, respectively. For webcam videos, the average improvements were 23.47% and 23.39%, respectively.

Fig. 3.8 and Table 3.8 show the comparison of S_n values for the three models. Both QP_R_F and CL-SDR performed better than the baseline configuration in case of stored videos. However, for live webcam videos, the baseline configuration and the QP_R_F model

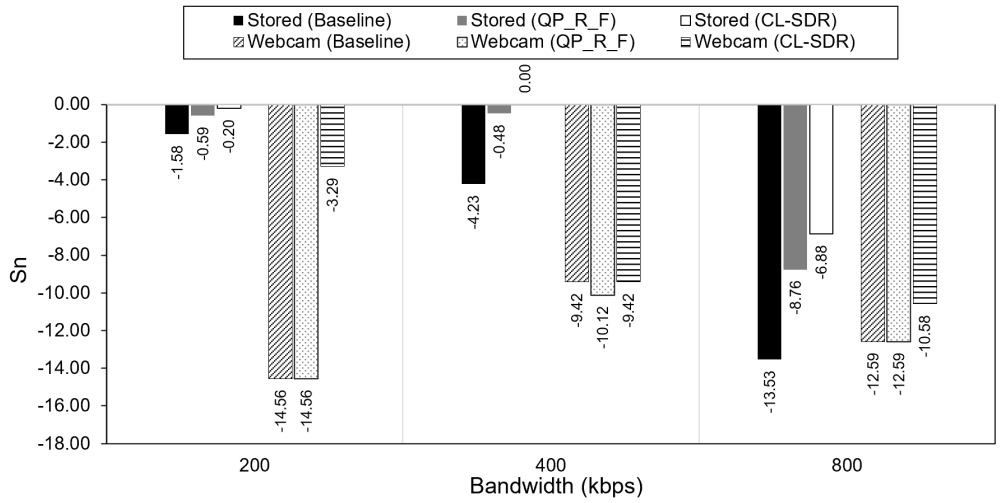


Figure 3.8: S_n comparison

Table 3.8: Cross-layer parameter config. to maximize S_n

Bandwidth (kbps)	Video Source	Parameter Configuration: S_n Scores Video resolution (px), Frame rate (fps), Encoding rate (kbps), Tx gain (dB), Frequency (GHz)		
		Baseline	QP_R_F	CL-SDR
200	Stored video	480×360, 24, 128, 60, 2.68	352×240, 12, 128, 60, 2.68	480×360, 16, 128, 40, 2.68
	Webcam	176×144, 15, 128, 60, 2.68	176×144, 15, 128, 60, 2.68	352×240, 15, 128, 60, 2.68
400	Stored video	480×360, 24, 256, 60, 2.68	352×240, 12, 256, 60, 2.68	480×360, 12, 256, 60, 2.68
	Webcam	176×144, 15, 256, 60, 2.68	176×144, 15, 256, 40, 2.68	176×144, 15, 256, 40, 2.68
800	Stored video	480×360, 24, 512, 40, 2.68	352×240, 12, 512, 40, 2.68	480×360, 12, 512, 40, 2.68
	Webcam	176×144, 15, 512, 60, 2.68	176×144, 15, 512, 60, 2.68	352×240, 15, 512, 40, 2.68

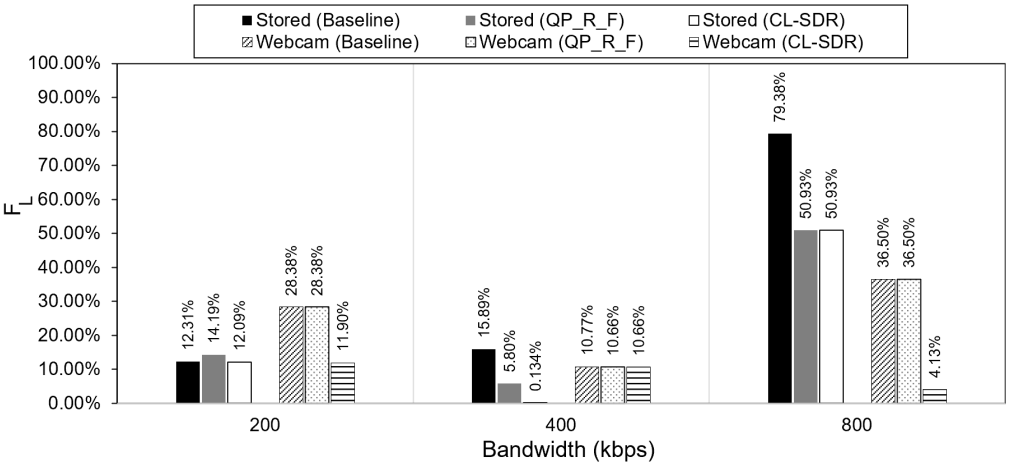


Figure 3.9: F_L comparison

Table 3.9: Cross-layer parameter config. to minimize F_L

Bandwidth (kbps)	Video Source	Parameter Configuration: F_L Scores Video resolution (px), Frame rate (fps), Encoding rate (kbps), Tx gain (dB), Frequency (GHz)		
		Baseline	QP_R_F	CL-SDR
200	Stored video	480×360, 24, 128, 60, 2.68	352×240, 12, 128, 60, 2.68	480×360, 16, 128, 40, 2.68
	Webcam	176×144, 15, 128, 60, 2.68	176×144, 15, 128, 60, 2.68	352×240, 15, 128, 60, 2.68
400	Stored video	480×360, 24, 256, 60, 2.68	352×240, 12, 256, 60, 2.68	480×360, 12, 256, 60, 2.68
	Webcam	176×144, 15, 256, 60, 2.68	176×144, 25, 256, 40, 2.68	176×144, 25, 256, 40, 2.68
800	Stored video	480×360, 24, 512, 60, 2.68	352×240, 12, 512, 40, 2.68	352×240, 12, 512, 40, 2.68
	Webcam	176×144, 15, 512, 60, 2.68	176×144, 15, 512, 60, 2.68	176×144, 20, 512, 60, 2.68

produced similar results. For stored videos, the average improvement in S_n was 78.85% and 62.49% compared to the baseline and QP_R_F model, respectively. In the case of live webcam videos, the corresponding average increases were 31.10% and 33.38%.

Fig. 3.9 and Table 3.9 show the comparison of F_L values for the three models. In most cases, the QP_R_F model generated frame losses similar to the baseline settings. For stored videos, the average reduction in F_L was 45.58% and 37.49% compared to the baseline and QP_R_F model, respectively. In case of live webcam videos, the corresponding average reductions were 49.26% and 48.91%.

3.7 Conclusion

We introduced a QoE-Driven cross-layer video communication scheme over SDR. Our solution is based on a practical SDR-based video communication platform built upon GNU radio, USRP B210, and GStreamer. This platform could monitor the QoE of video communication in real-time, which utilizes the latest research results in the field of QoE for video communication. We obtained measurements from this platform to understand how different application and physical layer parameters influence the QoE of received videos, and then based on these measurement results, we proposed a cross-layer solution that jointly adjusts video application settings and physical layer parameters to maximize QoE. Experimental results have shown that our solution could boost the quality of received videos in terms of frame loss rate, video clarity, and playback stalling.

Chapter 4

Quality-aware Video Analytics in Edge Computing Environments

4.1 Introduction

4.1.1 Overview

Edge computing is a distributed computing paradigm that uses decentralized processing power to process data near the source and thus reduces latency and saves bandwidth. In edge computing-based networks, data is processed by the source device itself or by a local computer rather than being sent to a data center. This reduction in the physical distance that data must travel leads to a significant reduction in delay, which is beneficial for mobile computing, the Internet of Things (IoT), and real-time video analytics applications.

Video surveillance and public safety are some of the major fields where edge computing-based video analysis has gained popularity. The prerequisites for intelligent video surveillance include the following stages: moving objects detection, object tracking, understanding and description of behaviors, and object recognition and identification [91].

Moving object detection segments the areas corresponding to the moving objects from the rest of an image. The process of moving object detection usually involves environment

modeling, motion segmentation, and object classification, which overlap each other during processing. An example of object detection is shown in Figure 4.1 [92]. The ground truths are highlighted in solid lines, three detected objects in dashed lines, and confidence levels in white text, respectively. Motion segmentation in image sequences aims at detecting regions corresponding to moving objects such as vehicles and humans. Currently, most segmentation methods use either temporal or spatial information in the image sequence [92]. The next step in tracking objects and analyzing their behaviors is to classify the moving objects accurately.

Figure 4.1: Object detection sample

After moving object detection, surveillance systems usually track the moving objects from one frame to the next. Well-known models for tracking include the hidden Markov models [93], Condensation algorithm [94] and Particle filter[95]. The next step, behavior understanding, involves the analysis and recognition of motion patterns and the generation of a high-level description of actions and interactions. Examples of behavior analyzing methods include dynamic time warping [96], finite-state machine [97], and hidden Markov model [98]. The final step, recognition and identification of objects such as human beings, vehicles, or animals, can be thought of as a special behavior-understanding problem. Human face and gait are now regarded as the main biometric features that can be used for personal identification in visual surveillance systems [92]. The bags-of-keypoints model [99] and the greedy generative deep learning model [100] are some of the most popular recognition methods.

Deep learning is a type of machine learning method that teaches computers to do what humans do naturally: learn by example. Deep learning models learn to perform classification

tasks directly from raw data, e.g., text, image, or audio. Learning models are trained using a large set of data and complex multi-layered artificial neural networks. Deep learning is a key component behind numerous applications such as automatic video analytics, speech recognition and virtual assistants, natural language processing, self-driving cars, bioinformatics, and social media filtering.

For computer vision-related tasks such as image recognition and classification, object detection, and image analysis, a widely used deep learning technique is the convolutional neural network (CNN). Convolutional networks feature a connectivity pattern between neurons similar to the human visual cortex, i.e., the part of the human brain that is in charge of interpreting and processing visual data received from the eyes. CNNs require comparatively less data pre-processing compared to other traditional image classification algorithms, which is a big advantage. Some well-known CNN architectures include VGGNet [101], ResNet [102], GoogLeNet [103], and LeNet [104].

Deep learning can achieve accuracy at higher levels than ever before and can even surpass humans at tasks like object classification in images or videos. However, two of the key requirements that must be met for efficient deep learning are huge datasets and very high computing power. High-performance graphics processing units (GPUs) with parallel architectures are usually used for deep learning. Consequently, using deep learning for object detection and tracking tasks at devices with constrained computing power can be prohibitive. Many edge devices such as smartphones, smart surveillance cameras, and Raspberry Pi's have limited computing power, and for those devices, alternative low-complexity object recognition and tracking methods should be considered. Traditional object detection schemes such as Histograms of Oriented Gradients (HOG) [105], Gaussian mixture models, Discriminatively Part Models (DPM) [106], and Locally Decorrelated Channel Features (LDCF) [107] can be useful for fast local detection tasks at edge devices. However, the low complexity and lower requirement of computing resources come at the price of lower accuracy compared to deep learning techniques. We employed a deep learning-based activity

detection algorithm in our experiments to achieve higher detection accuracy.

4.1.2 Proposed Work

We designed and built an edge computing-based video analytics platform that supports real-time video transmission and analysis. Our platform features a CNN-based activity detection model. Employing our testbed, we studied what factors contribute to the overall latency and detection accuracy in real-time video analytics. By examining the results from our video analytics experiments, we quantified the trade-off between latency and accuracy for video processing at the network edge vs. remote server. Finally, we proposed a quality-aware video analytics framework that optimizes the source video quality to meet accuracy requirements and maximizes response time.

4.2 Related Works

4.2.1 Edge Computing-based Video Analytics

Due to the benefits offered by edge computing, a number of research works have studied edge computing-based video analytics and related applications. Video analytics-based indoor positioning system using edge computing has been studied in [108] and [109]. Both works considered the IoT use cases of real-time video analytics and implemented their positioning system inside a conference room.

An edge-cloud coordination framework for live video analytics is proposed in [110]. This system utilizes the location data of video queries to configure the associated end-cameras to generate video frames that meet quality requirements. In [111], an intelligent task scheduling problem has been presented. An adaptive deep neural network (DNN) model selection method is employed to select the most effective DNN model for each task. They also present a graph searching approach outlining the trade-off between network delay and computing delay. A latency-aware video analytics platform has been presented in [112], where collaboration

between the nearby edge nodes is harnessed to minimize the response time. The authors also compare different task placement schemes for inter-edge collaboration.

A mobile edge computing-based video analytics platform is illustrated in [113]. It uses a DNN to partition an analysis task considering available resources on the mobile edge node and then upload the unprocessed data to the cloud for further analysis. Partitioning video analysis tasks across devices and servers is discussed in [114]. A black-box optimization scheme is implemented to ensure the optimal usage of computing resources.

4.2.2 QoE-aware Video Streaming in Edge Computing

A multi-access edge computing (MEC)-based mobile video streaming app is presented in [115]. The client app provides the general functions of the YouTube video streaming app. The authors have conducted simulations to investigate the effectiveness of the MEC architecture for improving the QoE. A similar MEC-based QoE maximization framework has been proposed in [116]. The QoE for dynamic adaptive video streaming is optimized through the coordination among MEC servers. The QoE optimization problem is cast as a Mixed-Integer Nonlinear Program (MINLP) that jointly calculates the video resolution levels and transmission rates.

In [117], an edge computing-based real-time QoE estimation system is deployed at a real LTE-A network. The system does not require any feedback from the clients to estimate the DASH clients' QoE. Another quality adaptation technique for MEC-based mobile video streaming is presented in [118]. This study aims to address the limitations of a purely client-based adaptation mechanism. The system performs better than client-based DASH when the available throughput is relatively low. However, the performance gain is insignificant under moderately high throughput or when the mobile clients have similar link quality.

4.2.3 Novelty of Our Work

The existing works have predominantly focused on task scheduling and latency minimization (e.g., [111, 112, 113]) while the impact of source video quality on network delay and video analysis quality for real-time video applications remain largely unexplored. Notably, videos with the same file size but different input settings (e.g., different resolution, frame rate, or quantization parameter) can potentially produce different detection accuracy while taking the same time to be transmitted over the network. Therefore, minimizing the latency without considering the specific video properties could result in sub-optimal detection accuracy. Furthermore, although some studies [115, 116] have considered the QoE of human users, the quality from the viewpoint of video analytics tools has not been rigorously investigated. We investigated the factors that impact video analysis quality with the primary concentration on analytics tools.

The unique contribution of our work is that we studied the performance of real-time edge-based video analytics considering the interrelation between video quality metrics such as video resolution, bitrate, quantization parameter (QP), and frame rate. We also studied the variation of total analysis time under different channel bandwidths and processing locations (server vs. edge). Finally, our quality-aware video analytics framework jointly optimizes source video quality and computation location to meet detection accuracy requirements with minimum latency.

4.3 Experimental Platform for Video Analytics based on Edge Computing

4.3.1 Testbed Components

Our edge computing-based video analytics platform (Fig. 4.2) comprises multiple edge devices and a high-performance server. The edge devices and the high-performance server can

be connected using either wireless or wired connections. The platform supports edge devices with different computing capacities, i.e., high and low, such as laptops, cellphones, and Raspberry Pi's. We found that although it is possible to run CNN models on devices with constrained computing power, the detection speed is extremely slow and not suitable for applications demanding real-time results. As we focus on real-time video analytics in this study, we used laptops with Linux OS (Ubuntu 18.04) as edge devices. The high-performance server has Nvidia Compute Unified Device Architecture (CUDA)-enabled graphics processing unit (GPU) to perform complex video analysis tasks.

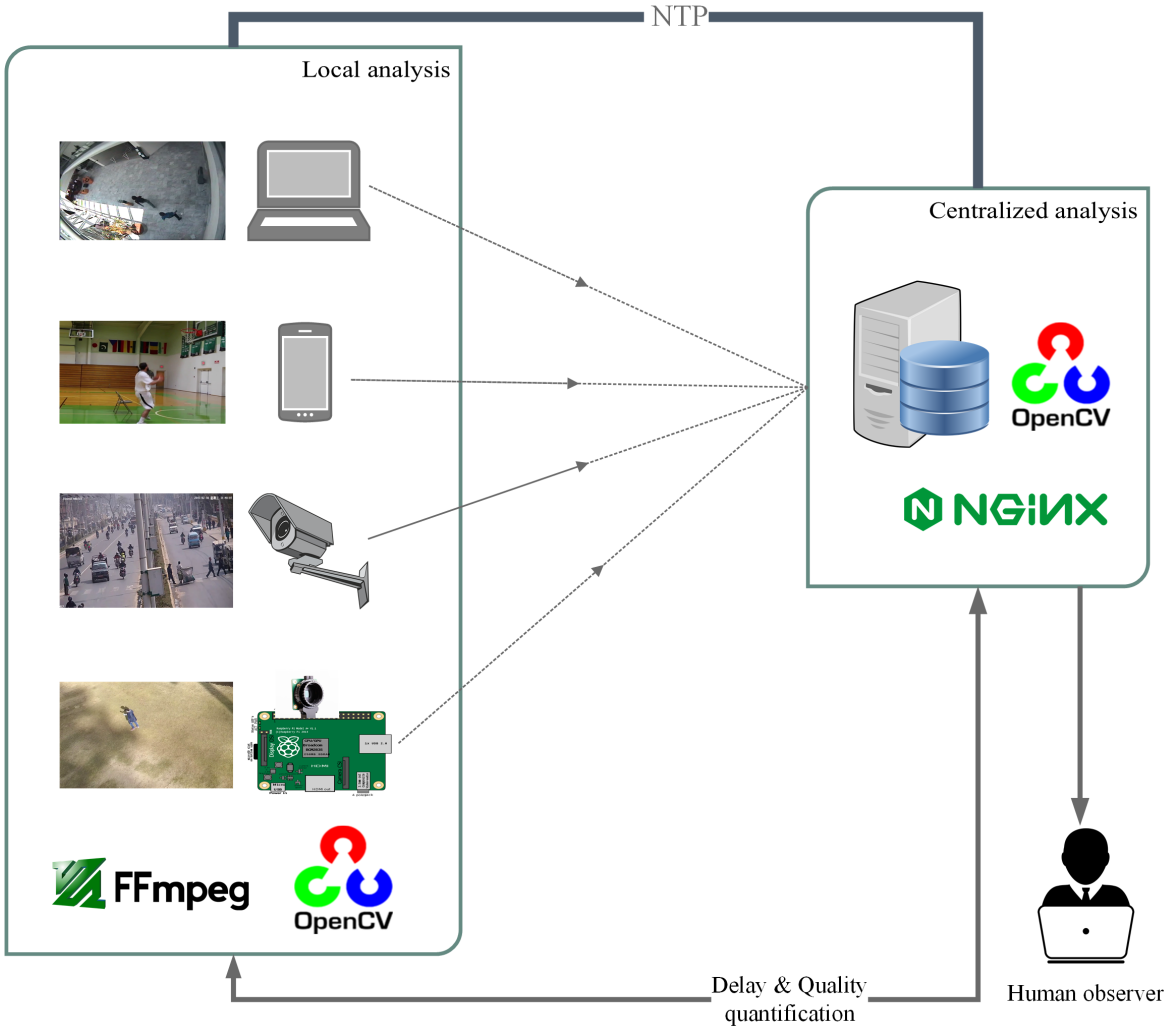


Figure 4.2: Edge computing-based video analytics architecture

The edge node emulates the camera behavior with specific video-adjustment abilities,

and the computing node performs a more extensive analysis. For the sake of practicality, the edge node is composed of a laptop that uses FFmpeg to stream locally stored videos via Real-Time Messaging Protocol (RTMP). To better establish an adequate streaming, the receiver side is configured with NGINX, which is an open-source software that supports web serving, reverse proxying, caching, load balancing, and media streaming. NGINX creates an RTMP session between the transmitting and receiving nodes to ensure optimal delivery of the video.

Additionally, the testbed relies on Network Time Protocol (NTP) synchronization to guarantee an accurate and unique time source. In this case, Chrony [119] is implemented in a server-client mode to meet such requirements. The edge node acts as a client entity that receives synchronization from the server side. More details about the platform can be found in Section 5.3.

4.3.2 Object Detection Model and Dataset

Depending on the intended application, any deep learning-based video analytics model can be used with our platform. We used Hara et al.'s [120] ResNet-based human activity recognition convolutional neural network (CNN), which was trained on the Kinetics 400 dataset [121]. The Kinetics 400 dataset includes 160000 video clips containing 400 different activities, which have been previously labeled and recognized. The activity recognition model was implemented with the help of OpenCV.

The platform supports real-time video input from cameras (e.g., webcam or dedicated camera sensor) as well as stored videos. The receiver node (edge device or server) can handle both types of video signals. However, if the testbed is used for video analytics research, as we do in this study, a labeled video dataset (with ground truths) would help investigate the detection accuracy.

4.4 Study of Quality and Latency for Edge Computing

There are two primary aspects of the overall quality of edge computing-based video analytics: total latency and analysis quality or accuracy. Different video analytics applications have different requirements/thresholds for latency and accuracy. The total latency in edge computing is the sum of computational latency and communication latency. The main contributing factors to computational latency are raw computational power of the edge and cloud nodes, task scheduling efficiency, the computational complexity of the analytics algorithm, and neural network (NN) training challenge due to a fragmented knowledge base. On the other hand, the primary factors that impact communication latency are bandwidth limitation, unpredictable latency, and abrupt service outage. The quality or accuracy of video analytics predominantly depends on the quality of input video frames. The other factors that influence the analysis quality are video content, video compression method, computing capacity, detector algorithms, and learning rate of the NN. This work focused on the latency and video quality factors that have the largest influence on overall analysis quality.

Table 4.1: Range of input settings for videos used in our experiments

Parameter	Values
Video Resolution (px)	192×144, 180×320, 240×320, 288×384, 288×612, 324×492, 352×640, 360×480, 422×622, 480×270, 540×960, 720×1280, 916×1904, 1080×1920
Frame Rate (fps)	4, 8, 16, 24, 30
QP	15, 20, 25, 30, 35, 40, 45, 50

To determine how the different aspects of the source video quality influence real-time activity detection in our video analysis platform, we streamed videos with a wide range of input settings listed in Table 4.1. In addition, the videos were streamed under different channel bandwidths ranging from 500 kbps to 10 Mbps. Testing the platform under different bandwidths allowed us to control the communication latency.

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.3: Snapshots of video sequences used for activity recognition

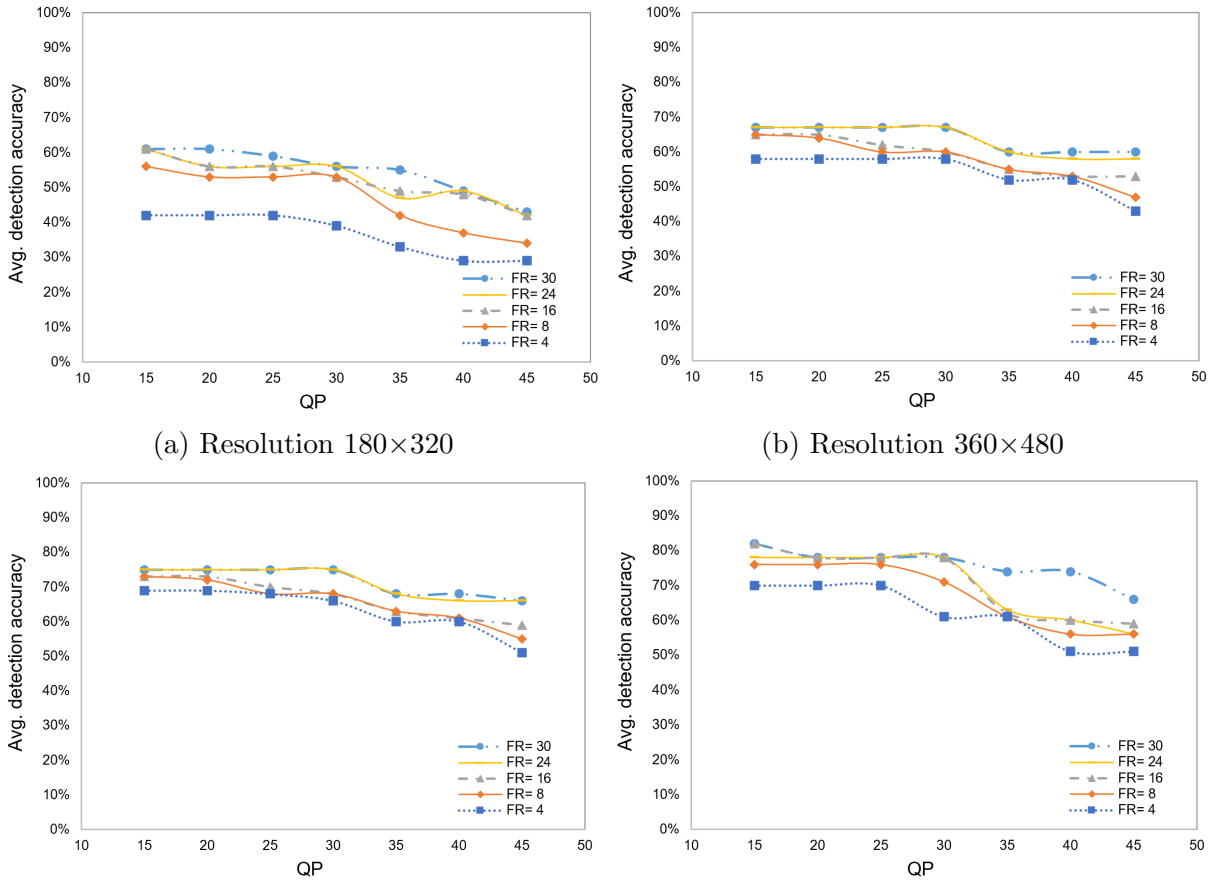
4.4.1 Dataset and CNN Model

To test the video analysis accuracy, we used 341 videos containing 17 activities such as running, tai chi, dribbling basketball, sitting, arson, car accident, driving, hitting, kicking, and vandalism. The video sequences had different scene characteristics, illumination levels, and object scales. The videos were taken from two datasets: Surveillance Perspective Human Action Recognition Dataset (SPHAR) [122], and Kinetics 400 [121]. SPHAR is a video dataset for human action recognition that contains 7759 videos divided into 14 activity classes. Fig. 4.3 shows some snapshots of the videos used in our video analysis experiments. The videos feature variations in video quality, aspect ratio, color, camera angle, orientation, and distance from the object.

The CNN model goes through each video blob by blob and identifies the activity based on the frames processed within that blob. The default number of frames in a blob is set to 16. If the detected activity is the same as the manually set label, we consider that to be an accurate detection. So, the detection accuracy is calculated by comparing the number of correctly detected activities (N_c) and the total number of videos processed (N_T):

$$Accuracy = \frac{N_c}{N_T} \tag{4.1}$$

4.4.2 Impact on Accuracy



(a) Resolution 180×320

(b) Resolution 360×480

(c) Resolution 720×1280

(d) Resolution 1080×1920

Figure 4.4: Average detection accuracy for different video resolutions

Fig. 4.4 shows the average detection accuracy for different video resolutions. We tested the detection accuracy for each resolution for different combinations of QP and frame rate (FR). The overall trend for different video resolutions was found to be similar, with better detection accuracy observed for higher resolutions. The maximum accuracy observed for the lowest resolution (180×320) was 61% (Fig. 4.4(a)) while maximum accuracy for 1080×1920 was 82.2% (Fig. 4.4(d)). Note that these numbers are dependent on the dataset used to test

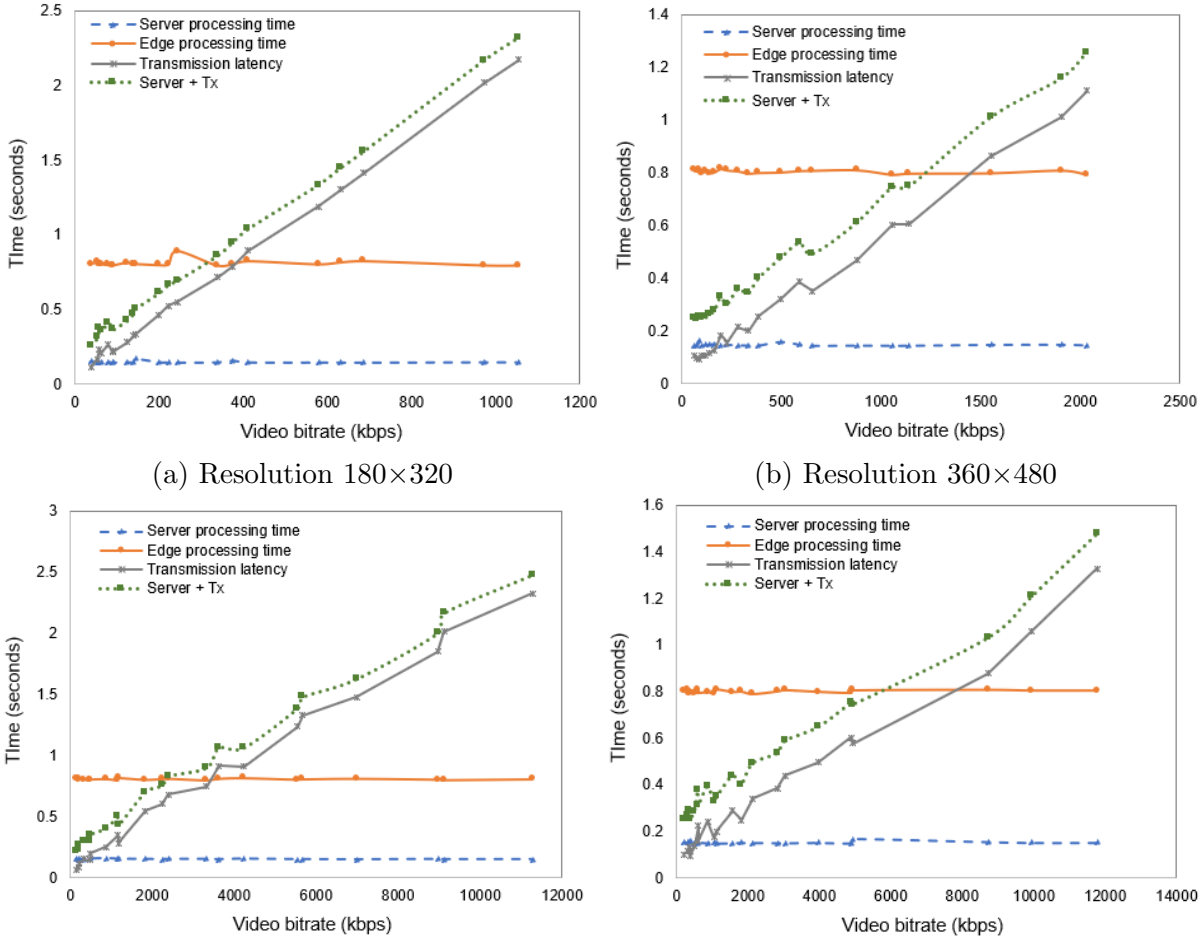
the CNN model.

The impact of QP on detection accuracy was noticed across all video resolutions. For lower QP values, i.e., 25 and below, the detection accuracy was generally very good, and the accuracy usually began to be affected after QP=30. For instance, in Fig. 4.4(c), the accuracy remained constant for QP = 15 to 30 for both 24fps and 30fps frame rates.

We also studied the impact of video frame rate on average detection accuracy. Interestingly, we observed a comparatively smaller impact of frame rate on detection accuracy for most videos. We found the detection accuracy to be largely unaffected for frame rates ranging from 16 fps to 30 fps in the majority of videos. Only when the frame rate was reduced to or below 8 fps, the average detection accuracy went down notably. For example, in Fig. 4.4(b) and Fig. 4.4(c), we can see the average detection accuracies are very similar for 30fps, 24fps, and 16fps. This phenomenon could be understood by realizing how activity detection and video analytics work. While the frame rate is a key aspect for the human viewing experience, and lower frame rate results in poor QoE, the significance of frame rate for activity detection is different. If an activity happens very quickly, e.g., hitting, a lower frame rate would affect the detection rate. However, if the activity in question is not time-sensitive, e.g., walking, a low frame rate would still produce good detection results. Therefore, the minimum frame rate requirement is highly dependent on the type of video analytics application.

4.4.3 Impact on Latency

To find the impact of video quality on overall latency, we studied the variation of processing time and transmission latency. We also quantified the trade-off between computation at the edge vs. server in terms of latency to illustrate which option generates minimal total delay. Fig. 4.5 shows how the edge processing time, server processing time, and transmission latency vary with video bitrate. Fig. 4.5 also shows the combined time it takes for the video to be transmitted to the remote server (communication delay) and then analyze the video (processing delay).



(a) Resolution 180×320

(b) Resolution 360×480

(c) Resolution 720×1280

(d) Resolution 1080×1920

Figure 4.5: Processing and transmission latency for different video resolutions

We found that the processing times at the edge and the server remained mostly unchanged when the video bitrate was varied. The same trend was noticed across all resolutions. Due to the greater computing capacity of the server, the server processing time was on average 5.356 times shorter than the edge processing time. Table 4.2 lists the average processing delays and the corresponding standard deviations. We can see the variation in processing time is minimal, and there is no significant impact of bitrate on the processing time, at least not in the range of bitrates we tested. Therefore, communication latency plays a more critical role in deciding where to process the video.

Table 4.2: Processing delays for edge vs. server

Resolution	Processing location	Avg. processing latency (sec)	Standard deviation
180×320	Edge	0.810275793	0.020575198
	Server	0.148663691	0.005894285
360×480	Edge	0.804905051	0.005989622
	Server	0.147724946	0.00475499
720×1280	Edge	0.806213322	0.006051314
	Server	0.152425845	0.001932372
1080×1920	Edge	0.800968919	0.005982224
	Server	0.152935982	0.003905544

Fig. 4.5(a) through 4.5(d) also illustrate the trade-off between computation at the edge vs. server in terms of total latency. It takes less time to transmit the video from the edge to the server for processing at lower bitrates. At that stage, the summation of the transmission latency and server processing time (denoted by *Server+Tx* in Fig. 4.5) is lower than the edge processing time. As the video bitrate goes up, the transmission latency starts to increase. Therefore, after a certain bitrate, i.e., turning point, processing at the edge incurs lower delay. In Fig. 4.5, the turning points are the intersections of the *Edge processing time* and *Server+Tx*.

The value of the turning point depends on two factors: video bitrate and channel bandwidth. In our experiments, we used different bandwidths for different video resolutions: 500 kbps for 180×320, 2 Mbps for 360×480, 5 Mbps for 720×1280, and 10 Mbps for 1080×1920. With those bandwidth settings, the bitrates where processing at the edge became the better option were found to be the following: 338.56 kbps for 180×320, 1550.31 kbps for 360×480,

2397.41 kbps for 720×1280, and 8722.19 kbps for 1080×1920. Of course, if either the channel bandwidth or the bitrate is changed, the turning points will change accordingly.

4.5 Quality-Aware Video Analytics based on Quality-Latency Trade-off

Computation at the network edge offers low latency at the expense of lower processing power and consequently lower accuracy or quality. On the other hand, transmitting the data to the more computationally powerful server from the data source or device leads to higher communication delay, but better quality of analytics can be achieved. This trade-off between quality and latency is a critical factor for the overall video analytics quality.

We propose a quality-aware video analytics framework (QVAF) for edge-computing environments that aims to maintain satisfactory analysis quality or accuracy while minimizing total latency (Δ_T). The proposed framework optimizes real-time video analytics in edge networks through two steps.

First, QVAF intelligently adjusts the video size to minimize Δ_T while maintaining the detection accuracy. The framework harnesses the relationship between source video quality parameters and detection accuracy described in Section 4.4 to find out how the parameters could be adjusted to reduce file size while offering the best analysis quality. Based on the previous streaming data and required detection accuracy, QVAF sets the thresholds for QP (Θ_{QP}) and frame rate (Θ_{FR}) for each video resolution. Then, QVAF computes the estimated bitrates of the output video for different combinations of QP and frame rate using the rate-control relationships of the video encoder. We used the H.264/AVC encoder in our experiments to adjust the QP and frame rates.

In H.264/AVC, the relationship between QP and bitrate or video file size is dictated by the quantizer step size (Qstep). The ratio between successive Qstep values is chosen to be 1.2246 so that Qstep doubles in size when QP increases by 6 [123]. As a result, the video

bitrate or file size is doubled if QP is reduced by 6 and halved if QP is increased by 6. Similarly, the encoder scaling matrix (V_i) could be used to predict bitrates for other values of QP [124] using the following equation:

$$Qstep(QP) = Qstep(QP\%6) * 2floor(QP/6) \tag{4.2}$$

Please refer to Section 7.2.3.5 in [123] for more details on the scaling matrix and H.264 standard.

On the other hand, in general, we can assume the video bitrate or file size to change at the same rate as the frame rate (FR) does (24fps to 30fps would mean 125% increase in file size) [125]. Utilizing these relationships between QP, frame rate, and output bitrate, QVAF can calculate the predicted bitrate and thus associated latency.

Table 4.3: List of variables used in QVAF

Variable	Definition
L	Latency requirement of the application
A	Accuracy requirement of the application
Θ_{QP}	Threshold for QP
Θ_{FR}	Threshold for frame rate (FR)
Δ_{TE}	Total estimated latency for edge processing
Δ_{TS}	Total estimated latency for server processing
Δ_{Tmin}	Minimum of Δ_{TE} and Δ_{TR}

Second, it determines the best location, i.e., edge vs. remote server, for processing or analyzing the video by comparing the total estimated latency (Δ_T). The total latency, Δ_T, is the summation of the communication latency (Δ_c) and processing latency (Δ_p):

$$\Delta_T = \Delta_c + \Delta_p \tag{4.3}$$

Communication latency (Δ_c) depends on the link quality or channel bandwidth between the video source and the processing location. If the video is processed at the edge, Δ_c would be zero. The value of processing latency (Δ_p) depends on the raw computing power of the machine used for video analysis. Algorithm 3 outlines the working mechanism of the

proposed framework, and Table 4.3 lists the variables used in QVAF.

Algorithm 3 Quality-aware Video Analytics Framework (QVAF)

```
Set L and A
Copy the detection data from previous streaming sessions
Set Θ_QP, and Θ_FR that satisfies A for each resolution
Compute both Δ_TE and Δ_TS and find the minimum of the two (Δ_Tmin)
if Δ_Tmin < L then
    Conduct video analysis at corresponding location
else
    Change QP and FR to adjust video quality and bitrate using the rate-control relationships
    Select values such that QP ≥ Θ_QP and FR ≥ Θ_FR
    Compute new Δ_TE, Δ_TS, and Δ_Tmin
    Conduct video analysis at corresponding location
end if
Update the detection data
```

To illustrate how the QVAF framework works, let us consider a video analytics application that demands a minimum detection accuracy of 70% and requires total latency to be no more than 0.8 seconds. QVAF will start by setting the values of A and L according to the application requirements. Then, the thresholds for QP (Θ_{QP}) and frame rate (Θ_{FR}) will be decided based on the existing data. Next, QVAF will compute the predicted total latency and adjust QP or FR if required using the rate-control relationships of the encoder. Lastly, QVAF will decide the best location, i.e., edge or server, and send the video for processing.

4.6 Performance Analysis

4.6.1 General Performance

Our edge computing-based video analytics platform can provide real-time activity detection results for video resolutions up to 1080×1920 with frame rates up to 30 fps and QP up to 15. The bitrates we tested ranged from 38.873 kbps to 11782.34 kbps. Observation of the experimental results indicates that the platform could support higher bitrates if the channel bandwidth between the edge and the server is higher (≥10 Mbps) or the processing power of

the edge device is increased. The general performance of the platform in terms of processing delay, communication delay, and detection accuracy is outlined in Table 4.4.

Table 4.4: General performance of the video analytics platform

	Edge processing delay (sec)	Server processing delay (sec)	Communication delay (sec)	Detection accuracy
Minimum	0.790779829	0.144020319	0.068426244	29.3%
Maximum	0.890771627	0.167339563	2.328599791	82.2%

The edge and server processing delays show small differences between their minimum and maximum values; this finding has also been discussed in Section 4.4.3. Communication delay varies significantly and has a major impact on deciding the processing location. The maximum accuracy achieved in our testbed, 82.2%, is comparable to rank-1 accuracies reported on state-of-the-art models trained on ImageNet [120]. The minimum accuracy of 29.3% illustrates the adverse effects of higher QP and lower frame rates.

4.6.2 Performance of QVAF

To evaluate the performance of our proposed QVAF framework, we conducted real-time video analysis in our experimental testbed. We used a total of 140 videos with four resolutions: 180×320, 360×480, 720×1280, and 1080×1920. The videos were taken from the testing category of the Kinetics 400 [121] dataset. The following activities were shown in the videos: lighting fire, javelin throw, watering plants, running, and rope climbing. The edge node and the high-performance server were connected via Ethernet, and the videos were transmitted under the following channel bandwidths: 500 kbps, 2000 kbps, 5000 kbps, and 10000 kbps.

We studied the performance of QVAF against two benchmarks: Baseline-Edge and Baseline-Server. The benchmarks work in the following ways.

- Baseline-Edge: The videos are processed without any input parameter adjustments. All the processing or video analysis is performed at the edge regardless of the available bandwidth or computing power.

- Baseline-Server: The videos are streamed/transmitted without any input parameter adjustments. All the processing or video analysis is performed at the high-performance server regardless of the available bandwidth or computing power.

We also tested how QVAF performs under different accuracy requirements. To that end, we chose two sets of minimum detection accuracy for the four video resolutions: high accuracy and moderate accuracy, and observed the comparative performance gain in both cases.

Table 4.5: Video properties and processing location: high accuracy requirement

Resolution	Scheme	Processing location	Video settings QP, Frame rate, Encoding rate
180×320	Baseline-Edge	Edge	25, 30 fps, 374.373 kbps
	Baseline-Server	Server	25, 30 fps, 374.373 kbps
	QVAF	Server	25, 16 fps, 243.895 kbps
360×480	Baseline-Edge	Edge	20, 16 fps, 877.672 kbps
	Baseline-Server	Server	20, 16 fps, 877.672 kbps
	QVAF	Server	25, 16 fps, 496.571 kbps
720×1280	Baseline-Edge	Edge	15, 16 fps, 8973.58 kbps
	Baseline-Server	Server	15, 16 fps, 8973.58 kbps
	QVAF	Edge	20, 16 fps, 5541.95 kbps
1080×1920	Baseline-Edge	Edge	20, 24 fps, 4936.72 kbps
	Baseline-Server	Server	20, 24 fps, 4936.72 kbps
	QVAF	Server	30, 16 fps, 1129.50 kbps

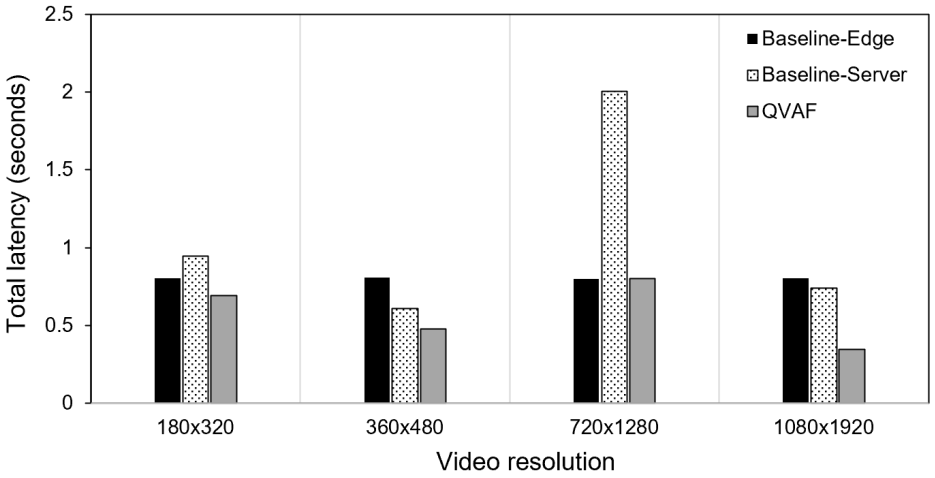


Figure 4.6: Total latency for QVAF vs. baseline configurations: high accuracy requirement

For high detection accuracy requirement, Fig. 4.6 and Table 4.5 show the comparison of total latency for Baseline-Edge, Baseline-Server, and QVAF. We set the accuracy requirement

for each video resolution (56% for 180×320, 66% for 360×480, 73% for 720×1280, and 78% for 1080×1920) and then examined which model produces the smallest total latency. The accuracy thresholds were different for different resolutions because lower resolution videos can not attain the same accuracy as high-resolution ones, which has been demonstrated in Section 4.4.2.

Table 4.6: Video properties and processing location: moderate accuracy requirement

Resolution	Scheme	Processing location	Video settings QP, Frame rate, Encoding rate
180×320	Baseline-Edge	Edge	25, 16 fps, 243.895 kbps
	Baseline-Server	Server	25, 16 fps, 243.895 kbps
	QVAF	Server	30, 8 fps, 73.07 kbps
360×480	Baseline-Edge	Edge	30, 16 fps, 281.884 kbps
	Baseline-Server	Server	30, 16 fps, 281.884 kbps
	QVAF	Server	30, 8 fps, 190.626 kbps
720×1280	Baseline-Edge	Edge	35, 30 fps, 1176.965 kbps
	Baseline-Server	Server	35, 30 fps, 1176.965 kbps
	QVAF	Server	35, 30 fps, 1176.965 kbps
1080×1920	Baseline-Edge	Edge	25, 30 fps, 3043.696 kbps
	Baseline-Server	Server	25, 30 fps, 3043.696 kbps
	QVAF	Server	30, 8 fps, 562.741 kbps

In most cases, QVAF performed notably better than the other two models across all resolutions. The latencies generated by QVAF were on average 27.86% lower than Baseline-Edge and 40.44% lower than Baseline-Server. The performance difference between edge and server processing was dictated by the communication delay and computing power. In particular, channel bandwidth significantly affected the communication delay for high bitrate videos. An example of this can be seen in Fig. 4.6 for resolution 720×1280. The source video had a comparatively higher bitrate (8973.58 kbps), and with the available 5000 kbps bandwidth, processing at the server resulted in a large delay.

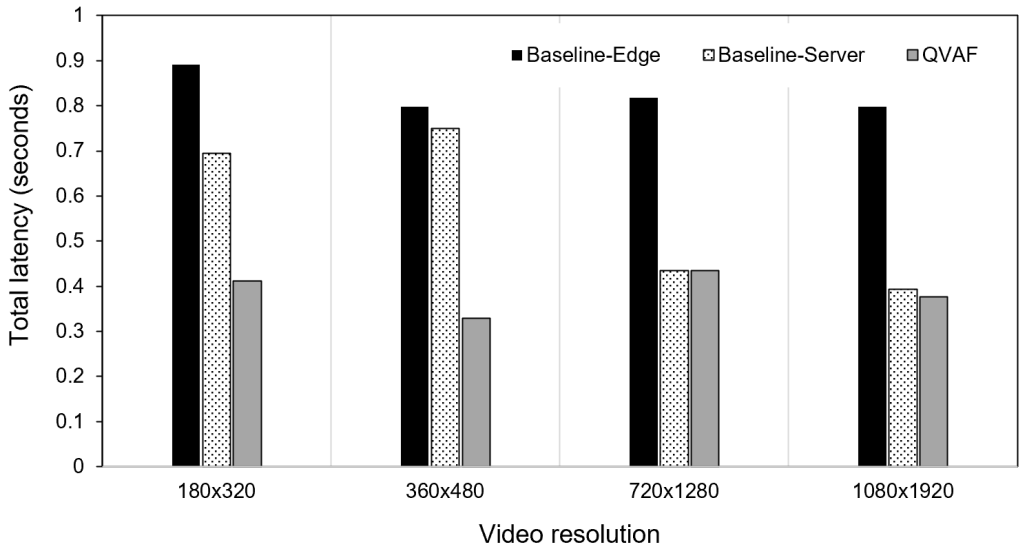


Figure 4.7: Total latency for QVAF vs. baseline configurations: moderate accuracy requirement

Then, we set the accuracy requirement to moderate level: 53% for 180×320, 60% for 360×480, 69% for 720×1280, and 72% for 1080×1920. The overall performance trend was similar to the previous case (high accuracy requirement). However, we took note of one difference in performance when the accuracy requirement was lowered. As QVAF had more room for adjustment due to a lower accuracy threshold, it was able to adjust video settings more drastically and produce video streams with lower bitrates. Consequently, QVAF dictated all videos be processed at the server as the communication delay was brought down. This ultimately resulted in a bigger performance gap between Baseline-Edge and QVAF: 51.81%, whereas it was 27.86% for the high accuracy requirement. This behavior further exhibits the gravity of source video quality and processing location on overall latency.

4.7 Conclusion

We designed and built an edge computing-based video analytics platform that allows real-time activity detection. Then, we employed this platform to investigate the relationships between video properties such as bitrate, QP, and frame rate and video analytics metrics

such as latency and accuracy. Additionally, we inspected how QP and frame rate impacts detection accuracy. Furthermore, we analyzed the pros and cons of processing the video at the network edge vs. the server. Finally, we proposed a quality-aware video analytics algorithm (QVAF) to provide minimum latency while maintaining required detection accuracy via optimizing source video properties. Experimental results exhibited that QVAF could significantly boost the response time of video analytics applications.

Chapter 5

Developed Experimental Platforms

We designed and built three video streaming platforms based on three different networking architectures. These platforms can be very beneficial to future researchers willing to conduct experiments in similar fields. In this section, we will describe the major components, design challenges, and applications of our developed platforms.

5.1 QoE-Aware Multi-Source Video Streaming Platform over CCN

We built a QoE-aware multi-source CCN video streaming platform based on CCNx 1.0 [8]. This platform supports scalable video streaming from multiple CCN nodes across both wired and wireless connections.

A standard CCN router (shown in Fig. 5.1) comprises of three primary modules: content store (CS), forwarding information table (FIB), and pending interest table (PIT). The FIB maps content names to the output interfaces that should be used to forward interest messages (requests) towards appropriate data sources. The PIT tracks the incoming interfaces from which pending requests have arrived. Finally, the CS serves as a local cache for content objects that have gone through the CCN router. We added a new component to the CS of

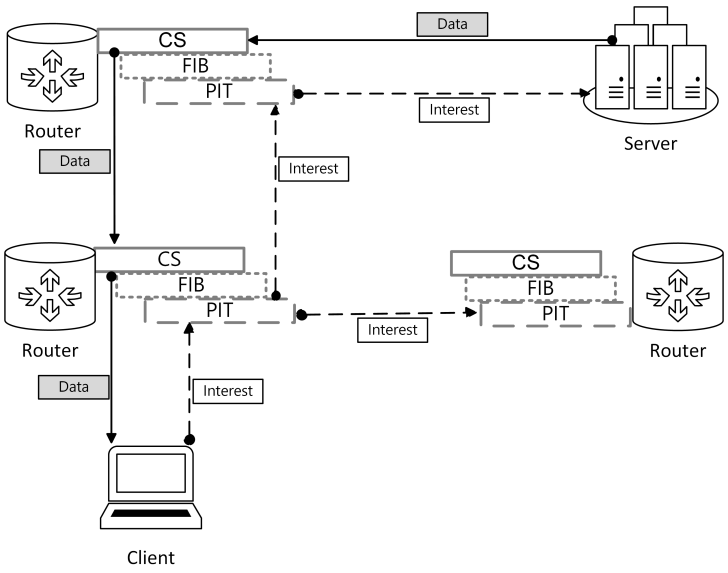


Figure 5.1: Content-Centric network architecture

the CCN routers: the content-listing table (CLT). At each router, the CLT will store the cached file names and the range of packets cached for each file. This additional component allows QoE-aware video streaming optimization by adjusting the video quality based on the content distribution (more details in Section 2.7).

The CCNx 1.0 (Distillery 2.0) software distribution brings together a set of modules that are needed to build a complete CCN-based system. These modules are independent of each other and fully customizable (written in C++). To enable multi-source streaming and implement the QoE-aware streaming mechanism, we implemented a customized VLC plugin (*CCN-VLC-AccessModule*) to play the videos on the client side. Plus, we maintained realistic content (i.e., CCN chunks) distributions (determined by ccnSim) in the CCN server nodes by pre-chunking the video files using *ccnxSimpleFileTransfer_Server*. Moreover, we also modified the built-in CCN forwarder named *Athena* to log CCN chunk transmission and latency data. The major system components are shown in Fig. 5.2 and discussed below.

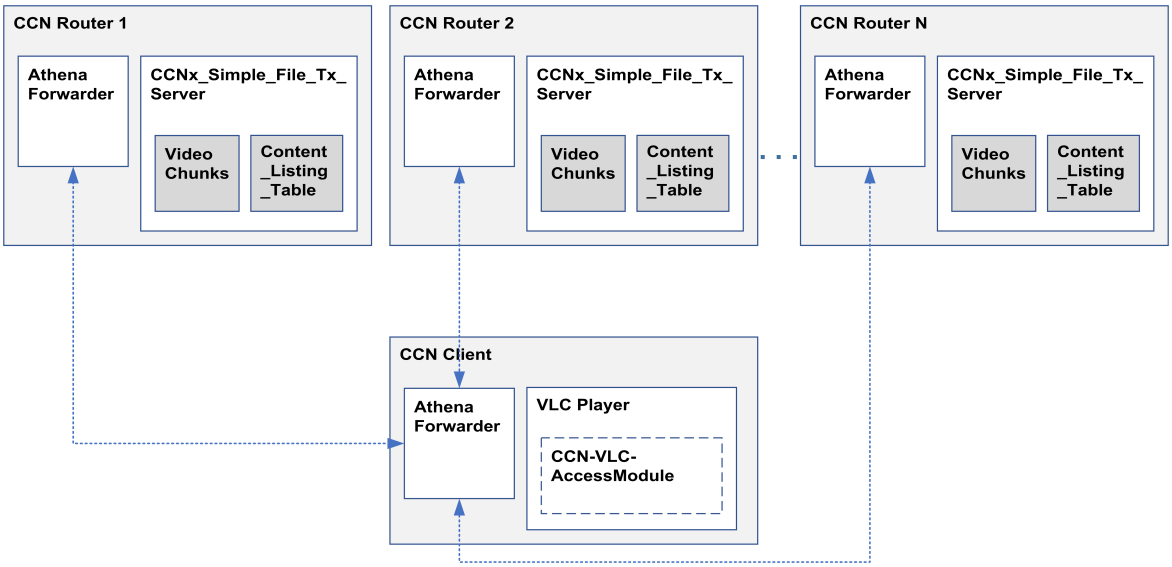


Figure 5.2: Client-server connection in our platform

5.1.1 System Components

- **CCNx 1.0:** The CCNx software was built from source and installed on computers running Ubuntu 14.04 LTS. CCNx comes with two CCN forwarders: *Athena* and *Metis* and two file transfer modules: *ccnxSimpleFileTransfer_Server* and *ccnxFileRepo_Server*. All of these modules can be customized to add or change functionalities.

- ***Athena* Forwarder:** We used the CCN *Athena* forwarder to establish dynamic routing paths between content servers. We employed public keys and passwords as the chunk-level encryption method. We modified the *Athena* module to keep track of chunk transmission and reception times as well as the source of each chunk.

- ***ccnxSimpleFileTransfer_Server:*** This module was used to segment each video into the desired number of CCN chunks (encrypted and named accordingly) and then create a uniform resource identifier (URI) for the video file. On the client-side, *ccnxSimpleFileTransfer_Client* can be used to get a list of available video files hosted by the servers in the network.

- ***CCN-VLC-AccessModule:*** The CCN 1.0 compatible version of the *CCN-VLC-*

AccessModule was used in our platform. This plugin works with VLC Player v2.1.6 (later versions might be incompatible due to VLC API changes). This plugin can be used to play videos streamed over CCN. We customized this plugin and the VLC player core to integrate QoE measurement features and allow updating CCN chunk request messages.

5.1.2 Challenges

One of the major challenges was the lack of documentation regarding video streaming using the CCNx platform. The existing CCNx manuals and wikis are focused on general data transfer over CCN, and none has outlined multi-source streaming. Plus, the installation procedures of both the CCNx 1.0 and the *CCN-VLC-AccessModule* software require very specific dependencies, which were challenging to complete without sufficient resources. Note that CCNx does not have any built-in module for multi-source video streaming or video quality adjustment. Therefore, we had to modify the existing modules and add those features.

5.1.3 Applications

Our multi-source CCN video streaming platform could be helpful for studying video transmission over CCN and investigating the impacts of key CCN features such as encryption, routing, and naming on video streaming. Although our platform provides QoE prediction and quality adjustment mechanisms, it is possible to integrate other QoE models and video quality adaptation schemes into this platform. Moreover, the modular structure and the highly customizable nature of the CCNx codebase means that researchers could use our platform as a base and add their own modules for supplemental features.

5.2 SDR-Based QoE-Aware Video Streaming Platform

We designed and implemented an SDR-based QoE-aware video streaming platform, which is depicted in Fig. 5.3. The main hardware and software components of our platform are described below.

Figure 5.3: A picture of our SDR platform

5.2.1 System Components

Hardware Modules

- **USRP:** The platform uses Ettus B210, a single-board USRP platform with continuous frequency coverage from 70 MHz – 6 GHz and up to 56 MHz of real-time bandwidth. Both USRPs were connected to the same computer via USB 3.0. One USRP was used as the transmitter while the other one acted as the receiver. The B210 has full support for UHD software and allows easy integration with GNU Radio.

- **Antennas:** Ettus VERT2450 antennas (dual Band $2.4 - 2.48$ GHz and $4.9 - 5.9$ GHz omnidirectional vertical antenna at 3dBi Gain) are used with the USRPs.

- **Computer:** The USRPs are connected to a laptop computer with Intel Core i5-5200U (2.20 GHz) processor, 16 GB of RAM, Intel HD Graphics 5500, and Ubuntu 18.04 LTS operating system.

- **WebCam:** Real-time video is captured using a webcam and then streamed to the encoder. We used the built-in webcam with a maximum resolution of 720p.

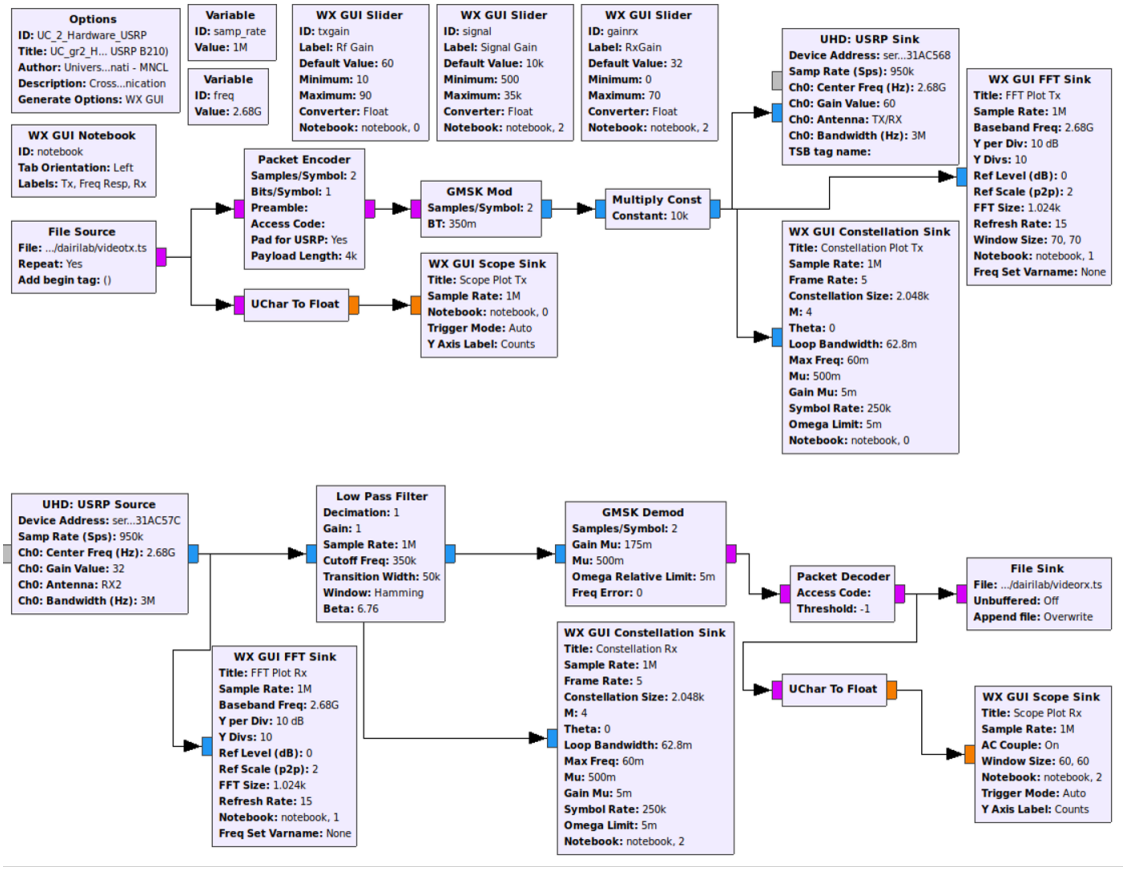


Figure 5.4: Flowgraph in GNU Radio

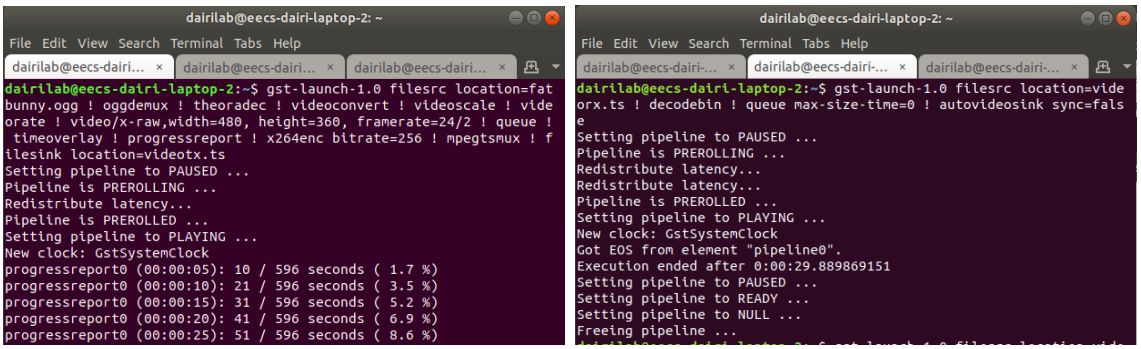
Software Modules

- **GNU Radio:** We designed the transceiver using GNU Radio software. It is a free software development toolkit that can be used with external RF hardware to implement SDRs. Our integrated transmission/reception flowgraph is shown in Fig. 5.4. One of the key advantages of GNU Radio is the availability of a large number of built-in signal processing blocks and the capability of designing custom blocks using Python.

- **GStreamer:** We used Gstreamer [4] to process (encoding and multiplexing) the video before transmission and after reception. Fig. 5.5 shows the operation of the commands

at the transmitter/receiver side. An input video is compressed using MPEG-4 Part 10 (a.k.a H.264 or Advanced Video Coding). The x264 open source software library is used for this implementation.

- **Video player:** GStreamer *playbin*, VLC Player, and MPlayer are three video players that we used to play the received videos.



(a) Transmitter / Encoding (b) Receiver / Player

Figure 5.5: GStreamer commands

5.2.2 Video Transmission and Reception

The block diagrams for video transmission are shown in the upper half part in Fig. 5.4. The first block in the flow graph is a file source that receives an H.264 encoded video input stream. An example of the input stream is shown in Fig. 5.6. The stream is passed to a packet encoder to be encoded in packets. After forming packets, the packetized data is modulated using GMSK modulation followed by multiplying the signal in order to increase the amplitude of the modulated signal. Then the samples are sent out to the USRP connected to GNU Radio via a USB port.

The bottom part of Fig. 5.4 shows the receiver implementation. The first block in video reception is the USRP source. The USRP captures the transmitted data and converts it to a form suitable for processing by the computer (GNU Radio). An example of video reception is shown in Fig. 5.7. The data is demodulated using the GMSK demodulator and forwarded to

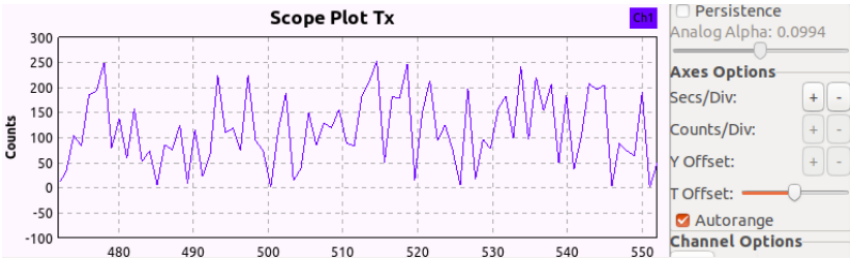
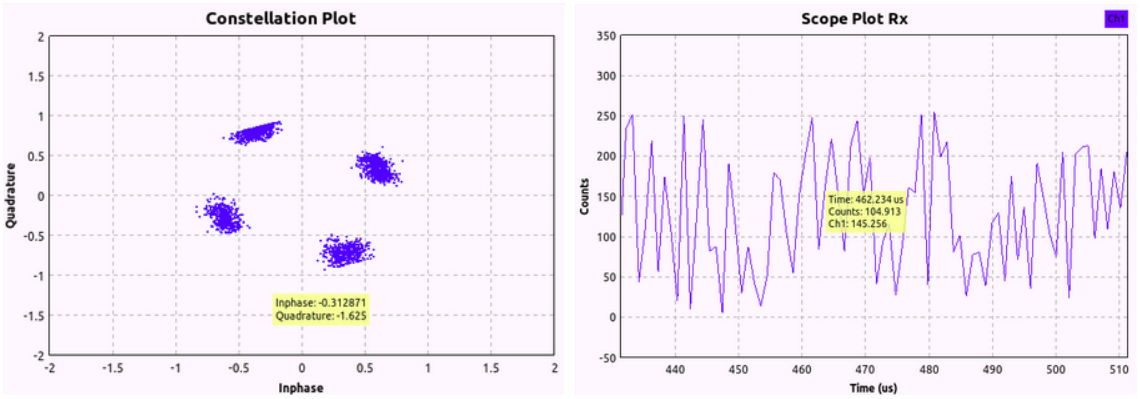


Figure 5.6: Video input stream



(a) Constellation (Rx side) (b) Output/Decoded stream

Figure 5.7: An example of video reception

the packet decoder, which performs the inverse operation of the packet encoder to generate a bitstream. This bitstream is then sent to either a file sink or played in real-time.

5.2.3 Challenges

The initial setup of both UHD and GRC requires a detailed process that is usually accomplished by installing these tools from the source code. The source code has a lot of prerequisite libraries, and they vary depending on the Linux distribution being used. Moreover, we had to sort through issues arising from version incompatibility between GRC and UHD. The default branches of the Git repositories of these two software may not always work together. Ensuring this compatibility helps to avoid complications at more advanced stages.

Additionally, the flowgraph design represents one of the most challenging situations as it defines the communication architecture of the solution. It is important to note that,

when it comes to the transmission of video, there are limited references in regards to SDR implementations. Therefore, the selection of an adequate modulation/demodulation scheme became one of the most elaborate duties that, combined with the lack of well-documented debugging strategies, clearly impacted the overall progress. We also faced some difficulties due to the GNU radio's inability to handle certain types of video streams (e.g., different compression and multiplexing methods).

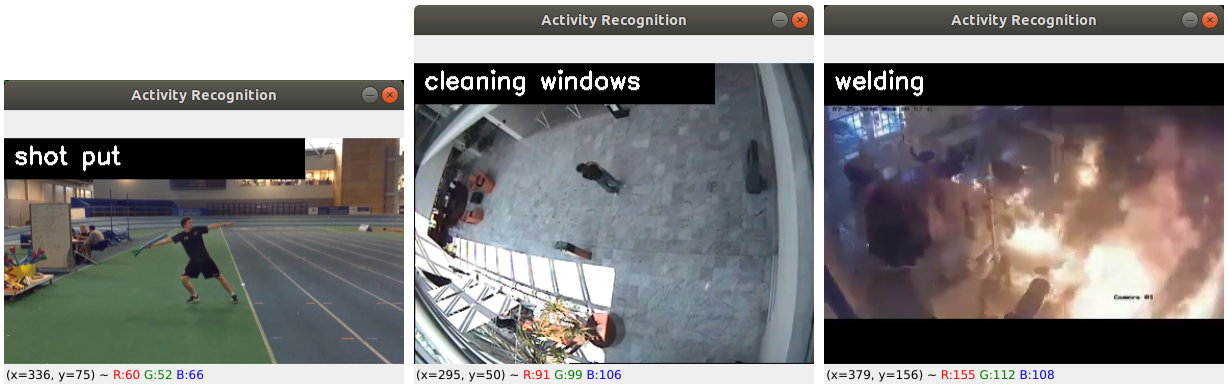
5.2.4 Applications

Our developed platform could be used to establish SDR-based real-time video streaming between different devices connected wirelessly. This platform can also act as a base to implement and test other SDR-based video streaming solutions. The integrated QoE support could be especially helpful to investigate streaming performance in terms of users' QoE.

Although the current implementation considers single-hop connectivity, the system could be scaled up to build more complicated network topologies. Another possible application of this platform is using it in conjunction with Unmanned Aerial Vehicles (UAVs), which will help to study the impact of mobility on video streaming quality.

5.3 Quality-Aware Edge Computing-based Video Analytics Platform

We built an edge computing-based video analytics platform featuring real-time quality-aware video analysis functionalities. This particular testbed allows a comprehensive examination of the critical attributes of edge computing scenarios. The proposed platform is aimed to resemble a high-performance architecture capable of transporting video with low latency and processing video data at both the edge and remote server depending on the quality and latency requirements. Fig. 5.8 shows an example of how the platform detects different activities. The main components of the system are described below.



(a) (b) (c)

Figure 5.8: Activity recognition on the receiver side

5.3.1 System Components

Receiver Side: The receiver side can be the edge-computing device (e.g., laptop) or the remote server depending on the particular analysis task. Regardless of the physical location of the video receiver, the following components are used for video reception and analysis.

- **Open CV:** Open CV offers a broad set of libraries that handles advanced functions for computer vision and deep learning. We installed Open CV using Python virtual

environment to easily develop the project in our Ubuntu 18.04 system with isolated packages.

- **NGINX:** NGINX [126] with Real Time Messaging Protocol (RTMP) module provides the system with a reliable yet flexible client-server connection. RTMP was an essential part of this platform as it provides the experiments with sufficient capabilities and has a low demand for system resources. NGINX was installed on our Ubuntu-based machine, and by default, it uses port 1935 to accept the streams.

- **Chronyc:** Chronyc operates as an easy-to-use implementation of the Network Time Protocol (NTP). It allows the platform to create a common timing reference that is highly necessary for delay-quantification purposes. Figure 5.9 shows the execution of the *chronyc sources* command that also indicates the node that is currently acting as a timing reference (*). In this case, the server node (192.168.0.106) has been declared as the machine with the highest priority (stratum 10) and thus is able to proportionate a common clock to the system.

```
dairilab@eecs-dairi-laptop-1:~/activity_recognition/ALL_4$ chronyc sources -v
210 Number of sources = 9

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* 192.168.0.106                10   7   377    49  -3342ns[  -19us] +/-  381us
^? 91.189.91.157                 2  10     0   14d  +201.0s[ +336us] +/-   96ms
^? 91.189.94.4                   2  10     0   14d  +201.0s[-1531us] +/-   86ms
^? 91.189.89.198                 2  10     0   14d  +201.0s[+1712us] +/-   95ms
^? 91.189.89.199                 2  10     0   14d  +201.0s[+1638us] +/-   82ms
^? 208.75.88.4                   2  10     0   14d  +201.0s[  +14ms] +/-   98ms
^? 47.190.36.235                 2  10     0   14d  +201.0s[-1015us] +/-   29ms
^? 50.205.244.22                 2  10     0   14d  +201.0s[+2580us] +/-   69ms
^? 44.190.6.254                  1  10     0   14d  +201.0s[ +131us] +/-   40ms
```

Figure 5.9: Time synchronization between transmitter and receiver

- **Wireshark:** Wireshark [127] is an open-source packet analyzer that supports net-

work troubleshooting, analysis, and communications protocol development. We used Wireshark to identify and analyze the RTMP packets, which helped us verify the communication latency between the nodes.

Transmitter Side: The transmitter side denotes the source of the video, which can be a surveillance video camera, webcam, or a camera sensor attached to a Raspberry Pi. For our experiments, we used labeled videos from datasets containing actual surveillance camera footage. As the transmitter node, we used a laptop with Ubuntu. The platform supports wireless (Wi-Fi) and wired (Ethernet) connections between the transmitter and the receiver.

- **FFmpeg:** We used FFmpeg for streaming the videos from the transmitting node to the receiver node, where they were received by the NGINX server. FFmpeg also allowed us to adjust the video properties such as QP and frame rate.
- **Chronyc:** Chronyc was installed and run on both the transmitter and the receiver sides for time synchronization.
- **Wireshark:** Packets from the transmitter side were captured and analyzed using Wireshark. Note that we ran Wireshark on both sides so that the captured frames could be compared via saved network traces.
- **Wonder Shaper:** Wonder Shaper [128] is a Linux command-line tool that allows the user to adjust the bandwidth of one or more network adapters. We used this tool to control the bandwidth between the nodes and experiment with communication latency.

5.3.2 Challenges

One of the major implementation challenges was ensuring that the correct number of frames are received by the CNN-based activity detection model. We found that the receiver was losing some frames at the beginning of the streaming session and thus processing an incorrect number of frames during blob construction. This frame loss at the very beginning of the

streaming caused a domino effect and resulted in erroneous detection results for the entire session. We identified the number of frames that were lost and then adjusted the frames sent from the transmitter side to solve the issue.

Another challenge was the quantification of delay between the transmitter and the receiver. As the communication latency was measured in microseconds, it was essential to maintain the highest possible precision in the time measurements. Therefore, we used the NTP protocol to synchronize the time references at the transmitter and the receiver. We also ran Wireshark at both ends to further quantify and verify the packet transmission delays.

5.3.3 Applications

Our real-time video analytics platform could be used to implement edge computing-based video applications. Moreover, our quality-aware video analytics framework (QVAF) offers optimized video streaming and processing. Furthermore, the ability to monitor processing and communication latencies separately is beneficial for edge-computing research. Although the current implementation comprises laptops as edge devices to support real-time processing by the CNN-based detection model, other devices such as cellphones and Raspberry Pi's could be easily integrated based on the needs of the intended application.

Chapter 6

Conclusion and Future Work

6.1 Research Contributions

We studied quality-aware video communication for both human viewers and video analytics tools in three different networking paradigms: Content-Centric Networking (CCN), Software-Defined Radio (SDR), and Edge Computing. Based on the findings of our extensive experimental data, we proposed new video communication frameworks for each of the networking architectures.

For CCN, we investigated the impact of ubiquitous in-network caching on content distribution across multiple CCN nodes and how that influences video streaming in CCN. We implemented actual CCN caching mechanisms to determine realistic chunk distributions among CCN routers. We found that caching a single video file across multiple CCN nodes could be a common scenario, which would result in source switching and video stalling. Then, we conducted human subjective tests to find out which factors impact the overall quality of experience (QoE) for human viewers. Our subjective test data showed that video stalling and video clarity had the highest impact on viewers' QoE. Next, we proposed a QoE model to accurately predict video quality from the perspective of human viewers. To achieve that, we used mean opinion scores (MOS) from our subjective tests. After that, we proposed a new

QoE-aware multi-source video streaming framework that maximizes perceptual video quality by minimizing stalling in CCN. Experimental results showed that our proposed algorithm, ASDC, can outperform state-of-the-art streaming standards such as DASH and SVC.

We designed and built an SDR platform that supports real-time video streaming and QoE monitoring. Our USRP B210 based SDR platform can stream HD videos and maintain satisfactory QoE. We also investigated how different application and physical layer parameters shape the QoE of received videos. Then, based on our measurement results, we proposed a cross-layer video streaming protocol for SDR (CL-SDR) to maximize QoE. Performance evaluation illustrated that CL-SDR could improve the quality of received videos in terms of frame loss rate, video clarity, and playback stalling. We also found that CL-SDR's cross-layer approach helped it achieve higher video quality compared to just application layer-based streaming solutions.

We developed an edge computing-based video analytics platform that supports real-time video streaming as well as stored video transmission. This platform could stream Full HD surveillance footage from the camera sensor to the processing location and provide real-time analytics results. Then, we studied which factors impact latency and detection accuracy and to what degrees. We looked at both communication and computation latencies and identified the trade-off between edge vs. centralized processing. Moreover, we explored how video properties such as QP and frame rate influences detection accuracy and how they control the video bitrate and thus transmission delay. Finally, we proposed a novel quality-aware video analytics scheme that minimizes overall latency while meeting the quality requirements. Our experimental results demonstrated that QVAF could significantly improve response times in edge-based video analytics applications.

6.2 List of Publications

1. Mohammad Nazmus Sadat, Erwin Vargas-Alfonso, Rui Dai, Ziqian Huang, Yiling Fu, and Sunmeng Lin. QoE-driven cross-layer design for video communication over software-defined radio. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–9. IEEE, 2021

2. Mohammad Nazmus Sadat, Erwin Vargas-Alfonso, Rui Dai, Ziqian Huang, Yiling Fu, and Sunmeng Lin. QoE-VS: A cross-layer QoE-aware video streaming platform using software-defined radio. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–6. IEEE, 2020

3. Mohammad Nazmus Sadat, Rui Dai, Lingchao Kong, and Jingyi Zhu. QoE-aware multi-source video streaming in content centric networks. *IEEE Transactions on Multimedia*, 22(9):2321–2330, 2020

4. J. Joseph, M. Radmanesh, M. N. Sadat, R. Dai, and M. Kumar. UAV path planning for data ferrying with communication constraints. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–9, 2020

5. Lingchao Kong, Jingyi Zhu, Rui Dai, and Mohammad Nazmus Sadat. Impact of distributed caching on video streaming quality in information centric networks. In *2017 IEEE International Symposium on Multimedia (ISM)*, pages 399–402. IEEE, 2017

6.3 Future Work

The expanding usage of video-focused applications over wireless networks has necessitated efficient and scalable video communication strategies. As new networking concepts are being realized, developing and fine-tuning quality-driven video transmission schemes is crucial. Additionally, the popularity of M2M video communication and intelligent video analysis

means that the quality from the perspective of machines or software will need closer inspection. Furthermore, video content plays a significant role in human viewing experience, video encoding, and video analytics. Therefore, optimizing video streaming by taking the content into consideration would be a big step towards more efficient video communication in general.

IoT-based smart environments are becoming ubiquitous, and consequently, guaranteeing satisfactory QoE to the consumers in these scenarios is vital. Investigating the factors that impact the perceptual video quality in the IoT applications, including those that require real-time response, can lead to the development of robust communication protocols. While received video properties such as resolution, bitrate, frame rate, and QP have major impacts on QoE, other facets of the human viewing experience such as viewing devices, interactive interfaces, and shared wireless resources should be studied more closely to provide a better user experience. This dissertation could serve as a foundation for the following prospective future research directions.

6.3.1 ICN Video Streaming: Other Facets

In Chapter 2 of this dissertation, we have studied the impact of CCN in-network caching on video streaming and how the new challenges could be tackled for better QoE. This study has given us a closer look into some of the crucial differences between traditional IP-based networks and content-centric networks. Core networking aspects such as content naming, security, routing and forwarding, and flow control are significantly different in CCN, even more broadly, ICN. Future works could explore how these factors bring in new challenges for ICN video streaming and what could be done to mitigate the issues.

6.3.2 More Comprehensive QoE Model

We have developed an accurate QoE prediction model (Section 2.5) that considers two of the most important factors for viewer satisfaction: video stalling and clarity. While these

two factors significantly influence the overall viewing experience, there are other factors that could be considered to build a more comprehensive QoE prediction model. As predicting human perceptual quality is a complex task, integrating more parameters into the QoE model would help in developing a more fine-tuned prediction framework. Our work on human subjective tests could be expanded to collect mean opinion scores (MOS) from viewers on additional video playback characteristics such as video content, viewer engagement, fatigue, and user interface. Additionally, new use cases such as virtual reality (VR)-based video streaming adds elements such as immersion and usability. Incorporation of these factors could result in a more accurate approximation of viewer satisfaction.

6.3.3 Quality-Driven Task Delegation

Different video analytics applications have different user requirements and available resources, and thus the demands for detection accuracy and response time vary widely. Our edge-computing platform (Chapter 4) integrated a deep learning-based activity detection model to achieve higher accuracy with minimal data preprocessing. It was an appropriate selection because we considered use cases with very high thresholds for accuracy and response time. However, for delay-tolerant video analytics applications and networks with both high-power and resource-constrained devices, a combination of traditional object detection methods and deep learning algorithms could prove more useful. Such implementation could benefit devices with limited computing power and sensor networks where energy consumption is a crucial factor. This cooperative processing from the video quality perspective could be an interesting area to investigate. Additionally, processing task distribution based on the requirements of the users may lead to the development of highly specialized video streaming frameworks.

Chapter 7

Bibliography

[1] Cisco. Cisco Annual Internet Report (2018–2023). white paper, 2018.

[2] Cisco visual networking index: Forecast and trends (2017-2022). white paper.

[3] GNU Radio - The Free & Open Source Radio Ecosystem. `https://www.gnuradio.org/`. Accessed: 2020-03-13.

[4] GStreamer. Open source multimedia framework. `https://gstreamer.freedesktop.org/`. Accessed: 2020-03-16.

[5] George Xylomenos, Christopher N Ververidis, Vasilios A Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V Katsaros, and George C Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys & Tutorials*, 16(2):1024–1049, 2014.

[6] Jacques Samain, Giovanna Carofiglio, Luca Muscariello, Michele Papalini, Mauro Sardara, Michele Tortelli, and Dario Rossi. Dynamic Adaptive Video Streaming: Towards a systematic comparison of ICN and TCP/IP. *IEEE Transactions on Multimedia*, 19(10):2166–2181, 2017.

[7] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[8] CCNx project web site. `https://wiki.fd.io/view/Cicn`. Accessed: 2018-04-05.

[9] Alessandro Finamore, Marco Mellia, Maurizio M Munafò, Ruben Torres, and Sanjay G Rao. Youtube everywhere: Impact of device and infrastructure synergies on user experience. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 345–360. ACM, 2011.

[10] Giuseppe Rossini and Dario Rossi. Evaluating CCN multi-path interest forwarding strategies. *Computer Communications*, 36(7):771–778, 2013.

[11] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492, 2015.

[12] Anush Krishna Moorthy, Lark Kwon Choi, Alan Conrad Bovik, and Gustavo De Veciana. Video quality assessment on mobile devices: Subjective, behavioral and objective studies. *IEEE Journal of Selected Topics in Signal Processing*, 6(6):652–671, 2012.

[13] Lingchao Kong, Jingyi Zhu, Rui Dai, and Mohammad Nazmus Sadat. Impact of distributed caching on video streaming quality in information centric networks. In *2017 IEEE International Symposium on Multimedia (ISM)*, pages 399–402. IEEE, 2017.

[14] Kamal Deep Singh, Yassine Hadjadj-Aoul, and Gerardo Rubino. Quality of experience estimation for adaptive HTTP/TCP video streaming using H. 264/AVC. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 127–131. IEEE, 2012.

[15] 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.

[16] Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. Quantification of youtube QoE via crowdsourcing. In *Multimedia (ISM), 2011 IEEE International Symposium on*, pages 494–499. IEEE, 2011.

[17] Quan Huynh-Thu and Mohammed Ghanbari. Temporal aspect of perceived quality in mobile video broadcasting. *IEEE Transactions on Broadcasting*, 54(3):641–651, 2008.

[18] Farhan Pervez and Muhammad Salman Raheel. QoE-based network-centric resource allocation for on-demand uplink adaptive HTTP streaming over LTE network. In *Application of Information and Communication Technologies (AICT), 2014 IEEE 8th International Conference on*, pages 1–5. IEEE, 2014.

[19] Konstantin Miller, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. Adaptation algorithm for adaptive streaming over HTTP. In *Packet Video Workshop (PV), 2012 19th International*, pages 173–178. IEEE, 2012.

[20] Waqas Ur Rahman, Dooyeol Yun, and Kwangsue Chung. A client side buffer management algorithm to improve QoE. *IEEE Transactions on Consumer Electronics*, 62(4):371–379, 2016.

[21] Suphakit Awiphan, Takeshi Muto, Yu Wang, Zhou Su, and Jiro Katto. Video streaming over content centric networking: experimental studies on PlanetLab. In *Computing, Communications and IT Applications Conference (ComComAp), 2013*, pages 19–24. IEEE, 2013.

[22] Zhengyang Liu and Yiran Wei. Hop-by-hop adaptive video streaming in content centric network. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–7. IEEE, 2016.

[23] Kenji Kanai, Takeshi Muto, Jiro Katto, Shinya Yamamura, Tomoyuki Furutono, Takafumi Saito, Hirohide Mikami, Kaoru Kusachi, Toshitaka Tsuda, Wataru Kameyama, et al. Proactive content caching for mobile video utilizing transportation systems and evaluation through field experiments. *IEEE Journal on Selected Areas in Communications*, 34(8):2102–2114, 2016.

[24] Yago Sánchez de la Fuente, Thomas Schierl, Cornelius Hellge, Thomas Wiegand, Dohy Hong, Danny De Vleeschauwer, Werner Van Leekwijck, and Yannick Le Louédec. iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 257–264. ACM, 2011.

[25] Raf Huysegems, Bart De Vleeschauwer, Tingyao Wu, and Werner Van Leekwijck. SVC-based HTTP adaptive streaming. *Bell Labs Technical Journal*, 16(4):25–41, 2012.

[26] Yago Sanchez, Thomas Schierl, Cornelius Hellge, Thomas Wiegand, Dohy Hong, Danny De Vleeschauwer, Werner Van Leekwijck, and Yannick Le Louédec. Efficient HTTP-based streaming using scalable video coding. *Signal Processing: Image Communication*, 27(4):329–342, 2012.

[27] Stefano Petrangeli, Niels Bouten, Maxim Claeys, and Filip De Turck. Towards SVC-based adaptive streaming in information centric networks. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[28] Abdelhak Bentaleb, Ali C Begen, Roger Zimmermann, and Saad Harous. SDNHAS: An SDN-enabled architecture to optimize QoE in HTTP adaptive streaming. *IEEE Transactions on Multimedia*, 19(10):2136–2151, 2017.

[29] Jonnahtan Saltarin, Eirina Bourtsoulatze, Nikolaos Thomos, and Torsten Braun. Adaptive video streaming with network coding enabled named data networking. *IEEE transactions on multimedia*, 19(10):2182–2196, 2017.

[30] CCN-VLC-AccessModule. `https://github.com/PARC/CCN-VLC-AccessModule`. Accessed: 2017-06-10.

[31] Xiph.Org. Video Test Media Collection, 2014. `https://media.xiph.org/video/derf/`, Accessed: March 10, 2018.

[32] Tears of Steel — Mango Open Movie Project, 2012. `https://mango.blender.org/`, Accessed: March 10, 2018.

[33] Valkaama, 2008. `http://www.valkaama.com/`, Accessed: March 10, 2018.

[34] ITUT Recommendation. P. 910, "Subjective video quality assessment methods for multimedia applications". *International Telecommunication Union, Tech. Rep*, 2008.

[35] ITUT Recommendation. BT. 500-13, "Methodology for the subjective assessment of the quality of television pictures". *International Telecommunication Union, Tech. Rep*, 2012.

[36] Zhengfang Duanmu, Kai Zeng, Kede Ma, Abdul Rehman, and Zhou Wang. A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing*, 11(1):154–166, 2017.

[37] Lucjan Janowski and Zdzislaw Papir. Modeling subjective tests of quality of experience with a generalized linear model. In *2009 International Workshop on Quality of Multimedia Experience*, pages 35–40. IEEE, 2009.

[38] Wang Kaiyu, Wang Yumei, and Zhang Lin. A new three-layer QoE modeling method for HTTP video streaming over wireless networks. In *2014 4th IEEE International Conference on Network Infrastructure and Digital Content*, pages 56–60. IEEE, 2014.

[39] Ching-Fang Yang. The regressive QoE model for VoLTE. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 409–414. IEEE, 2017.

[40] Friedemann Köster, Gabriel Mittag, and Sebastian Möller. Modeling the overall quality of experience on the basis of underlying quality dimensions. In *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2017.

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[42] Raffaele Chiocchetti, Dario Rossi, and Giuseppe Rossini. ccnSim: An highly scalable CCN simulator. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2309–2314. IEEE, 2013.

[43] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache "less for more" in information-centric networks. In *International Conference on Research in Networking*, pages 27–40. Springer, 2012.

[44] Michele Garetto, Emilio Leonardi, and Valentina Martina. A unified approach to the performance analysis of caching systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 1(3):12, 2016.

[45] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM, 2012.

[46] Andrea Araldo, Dario Rossi, and Fabio Martignon. Design and evaluation of cost-aware information centric routers. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, pages 147–156. ACM, 2014.

[47] Dong Doan Van and Dung Ong Mau. MS-CCN: Multi-source content centric networking. In *Information Technology, Networking, Electronic and Automation Control Conference, IEEE*, pages 843–847. IEEE, 2016.

[48] Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review*, 43(5):5–12, 2013.

[49] Of Forests and Men. `http://www.offorestsandmen.org/en/film-en`, 2011. Accessed: 2018-03-10.

[50] The Swiss Account. `http://www.lt11.com/2011/07/17/the-swiss-account/`, 2011. Accessed: 2018-03-10.

[51] Stefan Lederer, Christopher Müller, and Christian Timmerer. Dynamic adaptive streaming over HTTP dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 89–94. ACM, 2012.

[52] MainConcept H.264/SVC Encoder (proprietary). `https://www.mainconcept.com/us/products.html`, 2012. Accessed: 2018-01-16.

[53] Wei Quan, Changqiao Xu, Jianfeng Guan, Hongke Zhang, and Luigi Alfredo Grieco. Social cooperation for information-centric multimedia streaming in highway VANETs. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE, 2014.

[54] Suphakit Awiphan, Takeshi Muto, Zhou Su, and Jiro Katto. Outbound face selection considering response time and buffer usage for CCN adaptive video streaming. In *2015 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 181–186. IEEE, 2015.

[55] Dung Ong Mau, Tarik Taleb, and Min Chen. Mm3c: Multi-source mobile streaming in cache-enabled content-centric networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[56] 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):1717–1731, 2007.

[57] Kuldeep S Gill and Alexander M Wyglinski. Heterogeneous cooperative spectrum sensing test-bed using software-defined radios. In *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2017.

[58] Manolis Surligas, Antonis Makrogiannakis, and Stefanos Papadakis. Empowering the IoT heterogeneous wireless networking with software defined radio. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2015.

[59] Xingguang Wei, Zhiming Geng, Haitao Liu, Kan Zheng, and Rongtao Xu. A portable SDR non-orthogonal multiple access testbed for 5G networks. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2017.

[60] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492, 2014.

[61] Mohammad Nazmus Sadat, Rui Dai, Lingchao Kong, and Jingyi Zhu. QoE-aware multi-source video streaming in content centric networks. *IEEE Transactions on Multimedia*, 22(9):2321–2330, 2020.

[62] Fuzheng Yang and Shuai Wan. Bitstream-based quality assessment for networked video: a review. *IEEE Communications Magazine*, 50(11):203–209, 2012.

[63] Abdul Hameed, Rui Dai, and Benjamin Balas. A decision-tree-based perceptual video quality prediction model and its application in FEC for wireless multimedia communications. *IEEE Transactions on Multimedia*, 18(4):764–774, 2016.

[64] Demin Wang, Filippo Speranza, Andre Vincent, Taali Martin, and Phil Blanchfield. Toward optimal rate control: a study of the impact of spatial resolution, frame rate, and quantization on subjective video quality and bit rate. In *Visual Communications and Image Processing 2003*, volume 5150, pages 198–209. International Society for Optics and Photonics, 2003.

[65] Ting-Lan Lin, Sandeep Kanumuri, Yuan Zhi, David Poole, Pamela C Cosman, and Amy R Reibman. A versatile model for packet loss visibility and its application to packet prioritization. *IEEE Transactions on Image Processing*, 19(3):722–735, 2009.

[66] Péter Orosz, Tamás Skopkó, and Pál Varga. Towards estimating video QoE based on frame loss statistics of the video streams. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1282–1285. IEEE, 2015.

[67] Apurv Shaha, Duy HN Nguyen, Nathaniel Rowe, and Sunil Kumar. Real time video transceiver using SDR testbed with directional antennas. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 499–504. IEEE, 2017.

[68] S Nimmi, V Saranya, R Gandhiraj, et al. Real-time video streaming using GStreamer in GNU Radio platform. In *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, pages 1–6. IEEE, 2014.

[69] Octarina Nur Samijayani, Pramuditoruni Gitomojati, Dwi Astharini, Suci Rahmatia, and Nurul Ihsan Hariz Pratama. Implementation of SDR for video transmission using GNU radio and USRP B200. In *2017 5th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–4. IEEE, 2017.

[70] Amir Torabi, Michael W Shafer, Gabriel S Vega, and Kellan M Rothfus. UAV-RT: an SDR based aerial platform for wildlife tracking. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–6. IEEE, 2018.

[71] T. Radišić, M. Muštra, and P. Andraši. Design of an UAV equipped with SDR acting as a GSM base station. In *2019 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 31–34. IEEE, 2019.

[72] Limesdr: A low cost, open source (SDR) platform. https://limemicro.com/products/boards/limesdr/. Accessed: 2020-07-15.

[73] K Guevara, M Rodriguez, N Gallo, G Velasco, K Vasudeva, and I Guvenc. UAV-based GSM network for public safety communications. In *SoutheastCon 2015*, pages 1–2. IEEE, 2015.

[74] Gyeong Cheol Lee and Hwangjun Song. An effective cross layer-based video streaming algorithm over mobile ad hoc network. In *2009 6th IEEE Consumer Communications and Networking Conference*, pages 1–5. IEEE, 2009.

[75] Shoaib Khan, Yang Peng, Eckehard Steinbach, Marco Sgroi, and Wolfgang Kellerer. Application-driven cross-layer optimization for video streaming over wireless networks. *IEEE communications Magazine*, 44(1):122–130, 2006.

[76] Zhangyu Guan, Tommaso Melodia, and Dongfeng Yuan. Jointly optimal rate control and relay selection for cooperative wireless video streaming. *IEEE/ACM Transactions on Networking*, 21(4):1173–1186, 2013.

[77] Hsuan-Li Lin, Tung-Yu Wu, and Ching-Yao Huang. Cross layer adaptation with QoS guarantees for wireless scalable video streaming. *IEEE communications letters*, 16(9):1349–1352, 2012.

[78] Honghai Zhang, Yanyan Zheng, Mohammad A Khojastepour, and Sampath Rangarajan. Cross-layer optimization for streaming scalable video over fading wireless networks. *IEEE Journal on Selected Areas in Communications*, 28(3):344–353, 2010.

[79] Pinghua Zhao, Yanwei Liu, Ruixiao Yao, Song Ci, and Hui Tang. Perceptual quality driven cross-layer optimization for wireless video streaming. In *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pages 210–215. IEEE, 2013.

[80] Shu Fan and Honglin Zhao. Delay-based cross-layer QoS scheme for video streaming in wireless ad hoc networks. *China Communications*, 15(9):215–234, 2018.

[81] Dalei Wu, Song Ci, Haiyan Luo, Wendy Zhang, and Jinfang Zhang. Cross-layer rate adaptation for video communications over LTE networks. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 4834–4839. IEEE, 2012.

[82] Mohammad Tahir, Hafizal Mohamad, Nordin Ramli, and Sigit PW Jarot. Experimental implementation of dynamic spectrum access for video transmission using USRP. In *2012 International Conference on Computer and Communication Engineering (ICCCE)*, pages 228–233. IEEE, 2012.

[83] Jason Robert Carey Patterson. Video Encoding Settings for H.264 Excellence. http://www.lighterra.com/papers/videoencodingh264/, April 2012. Accessed: 2020-03-21.

[84] X. Chen, J. Hwang, C. Lee, and S. Chen. A near optimal QoE-driven power allocation scheme for scalable video transmissions over MIMO systems. *IEEE Journal of Selected Topics in Signal Processing*, 9(1):76–88, 2015.

[85] M. Chino, H. Miyashiro, and J. L. Arizaca. Implementation of an adaptive control system of the power transmitted through the estimated SNR, over SDR platform. In *2019 IEEE XXVI International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–4, 2019.

[86] A. Prince, A. E. Abdalla, H. Dahshan, and A. E. Rohiem. Multimedia SDR-based cooperative communication. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pages 381–386, 2018.

[87] Anish Mittal, Anush K Moorthy, and Alan C Bovik. Blind/referenceless image spatial quality evaluator. In *2011 conference record of the forty fifth asilomar conference on signals, systems and computers (ASILOMAR)*, pages 723–727. IEEE, 2011.

[88] Michele A Saad, Alan C Bovik, and Christophe Charrier. Blind image quality assessment: A natural scene statistics approach in the DCT domain. *IEEE transactions on Image Processing*, 21(8):3339–3352, 2012.

[89] FFmpeg Developers, FFmpeg tool (ver. 4.1.5) [software]. https://ffmpeg.org/. Accessed: 2020-02-16.

[90] Peach open movie project, Big Buck Bunny. https://peach.blender.org. Accessed: 2020-04-09.

[91] Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, 2004.

[92] Lingchao Kong. *Modeling of Video Quality for Automatic Video Analysis and Its Applications in Wireless Camera Networks*. PhD thesis, University of Cincinnati, June 2019.

[93] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[94] Michael Isard and Andrew Blake. Condensation—conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.

[95] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 212–219. IEEE, 2005.

[96] Aaron F Bobick and Andrew D Wilson. A state-based approach to the representation and recognition of gesture. *IEEE Transactions on pattern analysis and machine intelligence*, 19(12):1325–1337, 1997.

[97] Andrew D Wilson, AE Bobick, and Justine Cassell. Temporal classification of natural gesture and application to video coding. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 948–954. IEEE, 1997.

[98] Matthew Brand and Vera Kettnaker. Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):844–851, 2000.

[99] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[100] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[101] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[102] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[103] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[104] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[105] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[106] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[107] Woonhyun Nam, Piotr Dollár, and Joon Hee Han. Local decorrelation for improved pedestrian detection. *Advances in neural information processing systems*, 27:424–432, 2014.

[108] Yinghao Xie, Yihong Hu, Yuejun Chen, Yaqiong Liu, and Guochu Shou. A video analytics-based intelligent indoor positioning system using edge computing for IoT. In *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 118–1187. IEEE, 2018.

[109] Yuejun Chen, Yinghao Xie, Yihong Hu, Yaqiong Liu, and Guochu Shou. Design and implementation of video analytics system based on edge computing. In *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 130–1307. IEEE, 2018.

[110] Peng Yang, Feng Lyu, Wen Wu, Ning Zhang, Li Yu, and Xuemin Sherman Shen. Edge coordinated query configuration for low-latency and accurate video analytics. *IEEE Transactions on Industrial Informatics*, 16(7):4855–4864, 2019.

[111] Qiong Wu, Haitao Zhang, Peilun Du, Ye Li, Jianli Guo, and Chenze He. Enabling adaptive deep neural networks for video surveillance in distributed edge clouds. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 525–528. IEEE, 2019.

[112] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. LAVEA: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–13, 2017.

[113] Hui Sun, Ying Yu, Kewei Sha, and Bendong Lou. mvideo: Edge computing based mobile video processing systems. *IEEE Access*, 8:11615–11623, 2019.

[114] Yang Deng, Tao Han, and Nirwan Ansari. Fedvision: Federated video analytics with edge computing. *IEEE Open Journal of the Computer Society*, 1:62–72, 2020.

[115] Shun-Ren Yang, Yu-Ju Tseng, Chen-Chia Huang, and Wo-Chien Lin. Multi-access edge computing enhanced video streaming: Proof-of-concept implementation and prediction/QoE models. *IEEE Transactions on Vehicular Technology*, 68(2):1888–1902, 2018.

[116] Ayman Younis, Tuyen X Tran, and Dario Pompili. On-demand video-streaming quality of experience maximization in mobile edge computing. In *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–9. IEEE, 2019.

[117] Chang Ge and Ning Wang. Real-time QoE estimation of DASH-based mobile video applications through edge computing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 766–771. IEEE, 2018.

[118] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski. Edge computing assisted adaptive mobile video streaming. *IEEE Transactions on Mobile Computing*, 18(4):787–800, 2019.

[119] Chrony. `https://chrony.tuxfamily.org/`. Accessed: 2021-01-11.

[120] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and Imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[121] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[122] SPHAR Dataset. `https://github.com/AlexanderMelde/SPHAR-Dataset`. Accessed: 2021-01-05.

[123] Iain E Richardson. *The H. 264 advanced video compression standard*. John Wiley & Sons, 2011.

[124] H.264/AVC 4x4 Transform and Quantization. `https://www.vcodex.com/h264avc-4x4-transform-and-quantization/`. Accessed: 2021-07-01.

[125] How Changing Video Frame Rate Affects File Size. `https://larryjordan.com/articles/how-changing-frame-rate-affects-file-size/`. Accessed: 2021-07-01.

[126] NGINX. `https://www.nginx.com/`. Accessed: 2021-07-04.

[127] Wireshark. `https://www.wireshark.org/`. Accessed: 2021-07-04.

[128] Wonder Shaper. `https://github.com/magnific0/wondershaper`. Accessed: 2020-11-01.

[129] Mohammad Nazmus Sadat, Erwin Vargas-Alfonso, Rui Dai, Ziqian Huang, Yiling Fu, and Sunmeng Lin. QoE-driven cross-layer design for video communication over software-defined radio. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–9. IEEE, 2021.

[130] Mohammad Nazmus Sadat, Erwin Vargas-Alfonso, Rui Dai, Ziqian Huang, Yiling Fu, and Sunmeng Lin. QoE-VS: A cross-layer QoE-aware video streaming platform using software-defined radio. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–6. IEEE, 2020.

[131] J. Joseph, M. Radmanesh, M. N. Sadat, R. Dai, and M. Kumar. UAV path planning for data ferrying with communication constraints. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–9, 2020.