Master Thesis
Electrical Engineering
January 2013

A Study of Factors Which Influence QoD of HTTP Video Streaming Based on Adobe Flash Technology

Bin Sun and Wipawat Uppatumwichian

School of Computing
Blekinge Institute of Technology
371 79 Karlskrona
Sweden

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis on Telecommunication Systems. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Authors: Bin Sun, Wipawat Uppatumwichian
E-mail: BinSun@mail.com, u_wipawat@yahoo.com

University Advisor:
Patrik Arlos

School of Computing
Blekinge Institute of Technology
371 79 KARLSKRONA SWEDEN

Internet: www.bth.se/com
Phone: +46 455 385000
SWEDEN

Abstract

Recently, there has been a significant rise in the Hyper-Text Transfer Protocol (HTTP) video streaming usage worldwide. However, the knowledge of performance of HTTP video streaming is still limited, especially in the aspect of factors which affect video quality. The reason is that HTTP video streaming has different characteristics from other video streaming systems.

In this thesis, we show how the delivered quality of a Flash video playback is affected by different factors from diverse layers of the video delivery system, including congestion control algorithm, delay variation, playout buffer length, video bitrate and so on. We introduce Quality of Delivery Degradation (QoDD) then we use it to measure how much the Quality of Delivery (QoD) is degraded in terms of QoDD. The study is processed in a dedicated controlled environment, where we could alter the influential factors and then measure what is happening. After that, we use statistic method to analyze the data and find the relationships between influential factors and quality of video delivery which are expressed by mathematic models.

The results show that the status and choices of factors have a significant impact on the QoD. By proper control of the factors, the quality of delivery could be improved. The improvements are approximately 24% by TCP memory size, 63% by congestion control algorithm, 30% by delay variation, 97% by delay when considering delay variation, 5% by loss and 92% by video bitrate.

Keywords: Quality of Delivery, QoD, Quality of Delivery Degradation, QoDD, influential factors, congestion control algorithm, TCP memory size, delay, delay variation, playout buffer length, video bitrate.

Acknowledgements

When finishing this work, we wish to thank various people for their contribution to this project.

First of all, the research included in this thesis could not have been performed without the enthusiastic encouragement, the useful critiques of research and the patient guidance from our supervisor, Dr. Patrik Arlos.

We would like to express thanks to David Sveningsson, Vamsi Krishna Konakalla, Ramu Chakravadhanula and so on for their help of handling DPMI and a lot of advices from the Network Performance Lab and school of computing.

We would also like to extend the appreciation to our friends for the improvement of English grammar and expressions.

Finally, we want to thank our parents for their support and encouragement throughout our study and life.

Regards

Bin and Wipawat
2013, Sweden

Contents

List of Figures

List of Tables

List of Acronyms

ACK	Acknowledgment
AMD	Advanced Micro Devices, Inc.
ANN	Artificial Neural Network
AS3	ActionScript 3
BDP	Bandwidth Delay Product
CCA	Congestion Control Algorithm
CDN	Content Delivery Network
CPU	Central Processing Unit
CV	Coefficient of Variation
DPMI	Distributed Passive Measurement Infrastructure
DV	Delay Variation
DW	Durbin-Watson statistic
GPS	Global Positioning System
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
IBT	Initial Buffering Time
IPT	Inter-Packet Time
MDRB	Mean Duration of Re-Buffering events
MOS	Mean Opinion Score
MP	Measurement Point
NTP	Network Time Protocol
OS	Operating System
PCI	Peripheral Component Interconnect
PSNR	Peak Signal-to-Noise Ratio
QoD	Quality of Delivery
QoDD	Quality of Delivery Degradation
QoE	Quality of Experience
QoP	Quality of Presentation
QoS	Quality of Service
RBF	Re-Buffering Frequency
RQ	Research Question
RTCP	Real-Time Control Protocol
RTO	Retransmission Time-Out
RTP	Real-Time Protocol
RTT	Round-Trip Time
SD	Standard Deviation
SDK	Software Development Kit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

1 Introduction

Recently, Hyper-Text Transfer Protocol (HTTP) [1] video streaming is one of well-known systems to provide videos to users as seen in the large number of audiences. However, the knowledge of performance of HTTP video streaming is still limited, especially in the aspect of factors which affect video quality. The reason is that HTTP video streaming has different characteristics from other video streaming systems.

In this work, we study the factors that cause video artifacts of HTTP video streaming in temporal domain. Such video artifacts could influence Quality of Delivery (QoD). Despite the fact that many video artifacts affect QoD, only video artifacts which lead to extra required time to finish video playback are concerned in our study. Therefore, we always consider the extra required time to finish HTTP video streaming when talking about QoD in this work.

To gain the knowledge which was previously mentioned, we develop a new metric namely Quality of Delivery Degradation (QoDD) to quantify QoD in numerical method, and investigate factors which potentially influence QoD. After that, relationships between influential factors and QoD are revealed through empirical experiments.

Experiment results show that Transmission Control Protocol (TCP) [2] memory size, TCP Congestion Control Algorithm (CCA), Delay Variation (DV), playout buffer length and video bitrate are the influential factors, and there are relationships between the influential factors and QoD which can be expressed by mathematic models.

The knowledge derived from this research could expand the limited knowledge of performance of HTTP video streaming, and could provide a guideline for video quality optimization.

1.1 Motivation

HTTP video streaming is widely used today [3, 4]. Examples of some popular video content publishers are YouTube [5], Dailymotion [6] and Metacafe [7]. These video content publishers are improving their service quality to retain user stickiness in order to increase revenues [8, 9]. As QoD has a significant impact on user satisfaction [10], for example, only 5% of enhancement in customer retention could raise profit 25% to 85% [11], hence, it is worthwhile to identify and explore which factors can affect QoD.

1.2 Research Questions

Question 1 How to quantify QoD?

Question 2 What are factors which influence QoD?

Question 3 How is QoD affected from changing TCP memory sizes?

Question 4 How is QoD affected from changing TCP congestion control algorithm?

Question 5 How is QoD affected from changing delay variations?

Question 6 How is QoD affected from changing playout buffer lengths?

Question 7 How is QoD affected from changing video bitrates?

Question 8 Is there a relationship between TCP memory sizes and QoD? And can we model it, if any?

Question 9 Is there a relationship between delay variation and QoD? And can we model it, if any?

Question 10 Is there a relationship between playout buffer length and QoD? And can we model it, if any?

Question 11 Is there a relationship between video bitrate and QoD? And can we model it, if any?

1.3 Research Methodology

To answer the previously proposed RQs, we conduct the following methods to acquire knowledge and answer the RQs. First, we perform literature review to find QoD related video artifacts, and inherit Initial Buffering Time (IBT), Mean Duration of Re-Buffering events (MDRB) and Re-Buffering Frequency (RBF). Then we develop a metric, QoDD, to quantify QoD from IBT, MDRB and RBF. Later, we study previous works to explore factors which affect QoD by investigating mathematic models of IBT, MDRB and RBF. So we could answer RQs 1–2. This is detailed in Chapter 3.

RQs 3–7 are preliminarily answered by experiments to reveal relationships between influential factors and QoD. Briefly, we change numerical values of influential factors and observe IBT, MDRB and RBF. After that, we perform post processing to get QoDD and models. This is how we analyze experiment results and answer RQs. Details about the experiment design and post processing can be found in Chapter 4 and 5.

RQs 3–11 could be completely answered by further post processing. Shortly, data modeling on IBT, MDRF, RBF and QoDD can derive mathematic models which represent the relationships between influential factors and QoD. Besides, we also supplement result analysis by plotting the relationship models so that visual result analysis is possible.

1.4 Related Works

Previous works usually correlate impact of Quality of Service (QoS) on user perceived video quality with subjective evaluation, Mean Opinion Score (MOS) [12]. Research in [13] finds the relationship between network QoS and MOS. But the subjective evaluation is resource extensive and requires careful control over factors. Failing to do so will lead to inaccurate result. One alternative evaluation method is objective evaluation such as Peak Signal-to-Noise Ratio (PSNR). Nevertheless, PSNR is an inappropriate method to evaluate video quality of HTTP video streaming. The reason is that the video quality artifacts in HTTP video streaming are usually not picture artifact, but the additional time required to complete video playback. One work suggests that video qualities can be classified into two domains: spatial and temporal qualities [14]. These two can be observed at application layer and are named Quality of Presentation (QoP) and QoD respectively.

A research [15] develops new full-reference metrics for video quality evaluation. Despite the high accuracy offered by full-reference paradigm, the Internet always comes with limited bandwidth and cannot transmit full-reference information along with video stream. Otherwise video stream and full-reference influence each other and lead to inaccurate video quality evaluation. Another study [16] proposes non-reference metric to quantify video quality. However, the metrics is designed to evaluate over all video quality, not only temporal quality. Works in [13, 17] propose non-reference metrics for evaluation of temporal quality which represent startup delay and pause of video playback. However, [14] suggests other possible artifacts which are hack and break in addition to other artifacts.

Identification of influential factors is usually limited in network layer. Packet loss is often seen as an influential factor within this research domain. Some works [16, 18, 19] investigate artifacts of packet loss on video quality. Interestingly, study in [13] introduces DV into consideration. In spite of these network factors, one research [20] reveals various factors located on different layers also influencing video quality. These factors are, for examples, protocol, memory, and configuration. However the research does not deeply investigate all proposed factors and leaves them uninspected.

Although many works have well-done investigated factors and video quality, User Datagram Protocol (UDP) [21] was widely used as underlying protocol in their experiments as it is an ideal way to send video stream. For example, research in [18, 13] relates factors to video quality using UDP as transmission protocol. However the result of these works cannot characterize properties of HTTP video streaming as HTTP has packet loss recovery mechanism provided by TCP. In [19], research investigates pause-related video quality using TCP protocol.

One study [22] suggests various video player models and investigates performance of them with HTTP video streaming system. The work also performs experiments and explores artifact of packet loss and delay on re-buffering frequency and average re-buffering time. However, the work uses a simple model based simulator which is not suitable to represent some real video content publisher such as YouTube. Research in [23] introduces non-reference metrics to evaluate temporal quality of HTTP video streaming which are IBT, RBF and MDRB. Startup delay is evaluated by IBT while pause is evaluated by RBF and MDRB.

1.5 Aim and Objective

Aims

1. Develop a technique to evaluate QoD
2. Explore factors which influence QoD
3. Investigate relationships between the factors and QoD

Objectives

1. Develop a metric to quantify QoD
2. Develop a practical technique to determine and quantify QoD
3. Develop a video player for QoD quantification
4. Explore factors which influence QoD
5. Conducting experiments to reveal relationships between the influential factors and QoD
6. Develop an automatic system to assist empirical experiments
7. Analyze results of the experiments

1.6 Thesis Outline

Chapter 1 gives an overview of this research. Topics are developed to describe boundary of research, show importance of study and hint direction of the work. Chapter 2 describes background information of the research. The knowledge shall assist readers to understand the concept of QoD and HTTP video streaming. Chapter 3 provides a methodology to answer RQs 1–2 as well as the foundation of experiments. The chapter explains in detail how QoD quantification is developed and what influential factors are. Chapter 4 presents experiment design to preliminarily answer RQs 3–7. The chapter explains how experiments are conducted and how required components and tools are developed. Chapter 5 explains post processing to completely answer RQs 3–11. The chapter introduces how models of relationships between influential factors and QoD are developed. Chapter 6 presents experiment results, which are plots of the models, together with result analysis. Chapter 7 draws conclusions of research and re-answers all the RQs again. Besides, discussion and future work are proposed at the end.

2 Background

2.1 Video Streaming

There are two methods to transmit a stored video over the Internet, downloading and streaming [24]. The first method needs a long time to “download before play” while streaming indicates “play while downloading” (also known as “video on demand”) [25]. Considering streaming, there are two techniques usually used on the transport layer, UDP based video streaming and TCP based video streaming (particularly HTTP/TCP video streaming). Below is an overview of those two streaming methods.

2.1.1 Classical UDP Streaming

UDP video streaming relies on just-in-time data delivery and rendering. It could use low bandwidth to transmit video. However, these unreliable streaming protocols, such as RTSP over UDP, usually lead to degradation of picture quality and rendering distortions which are noticeable to the user. It cannot guarantee very high quality delivery of their videos which is desired by video content providers [26]. Additionally, the requirement of four UDP channels [27, 28, 29] will consume more server resources, and will make it complicated to design and implement the network. This scenario is shown in Figure 1 (simplified from [26]). Although CDNs could support RTSP and other UDP based streaming [30], in addition to dedicated resources, most firewalls are configured to block the dynamic UDP ports required by protocols like RTSP [31].

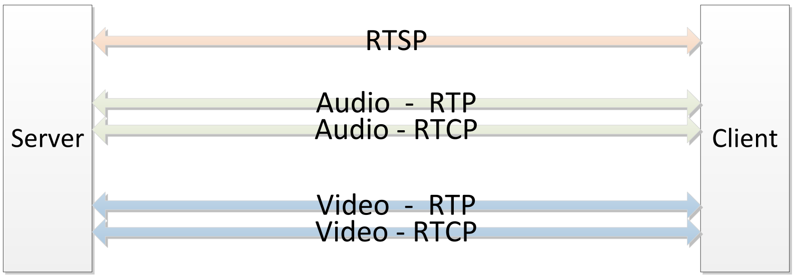


Figure 1 RTSP/RTP Connection Setup

2.1.2 HTTP Video Streaming

Unlike previous streaming protocols, HTTP has some primary advantages – data integrity, omnipresence, and wide firewall friendliness. Initially, it provided only straight download-and-play (Figure 2a [26]) for videos where the entire file had to be downloaded before playback would begin. HTTP progressive download based video streaming could be used to make full use of the benefits and avoid the unnecessary download time. This HTTP over

TCP [3, 32] streaming method is able to provide those advantages as well as getting the media started play before it has been completely received [33, 34]. Figure 3 shows a YouTube video under playing while downloading is in progress.

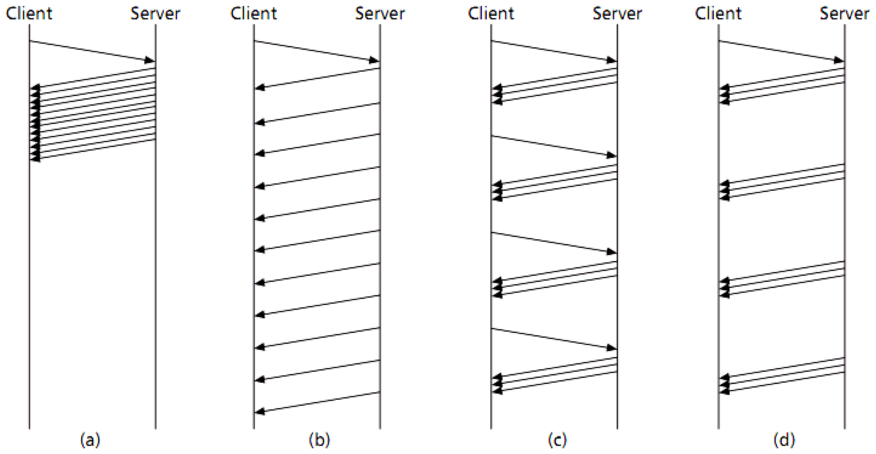


Figure 2 Video Delivery Methods

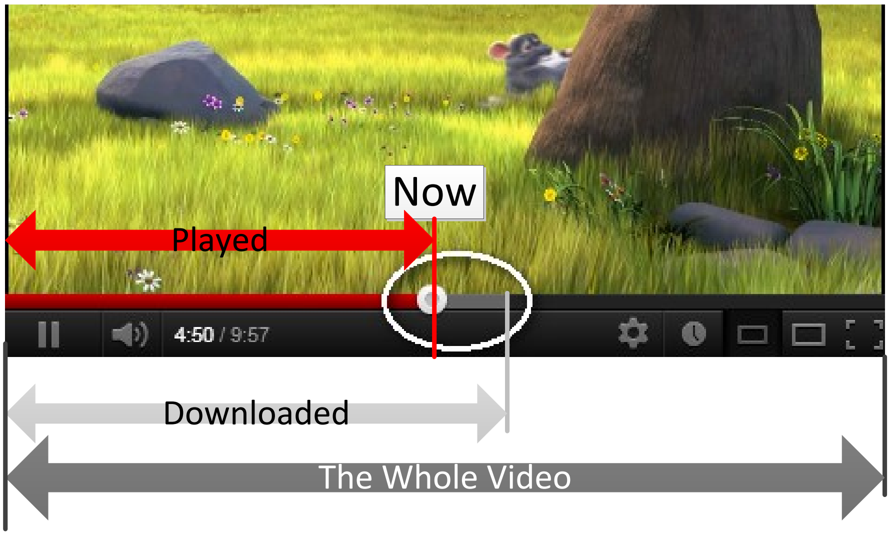


Figure 3 Progressive Download

Some new techniques, such as dynamic transcoding and client-side / server-side paced download (Figure 2c,d), are trying to combine the low bandwidth usage of RTSP (Figure 2b) with the data integrity of HTTP. Dynamic transcoding is very flexible, which gives the possibility to change the video bitrate during the transmission to make full use of the bandwidth and try to avoid congestion problem at the same time [35]. However, some of the hybrid solutions are currently not worth the cost of design and implementation, and additional pre-processing is also required [26].

2.2 Quality of Delivery

2.2.1 What is Quality of Delivery

In [36], Quality of Delivery (QoD) is described as the quality of delivered data. However, such definition is too general since the quality of delivered data has many aspects such as correctness, ordering, latency. Considering HTTP video streaming system where TCP is an underlying transport protocol, latency should be the main aspect in quality determination since incorrectness and out-of-ordering of delivered data are not existent at TCP-session layer. Another definition of QoD is derived from [14] by defining it as the capability of delivering data on time. Such definition is narrower than the previous one which makes it more suitable to study. Moreover, it also relates QoD with the latency of delivered data which is the major effect of using TCP as transport protocol. Therefore, we apply this definition to our research. In addition, we also improve the definition to make it more suitable for video quality study by defining QoD as the capability of delivering video frame on time.

From the definition which is previously mentioned, limited QoD results in the delay of delivered video frames that consequently causes non-smooth video motion since video frames are delayed from original source. Examples of video artifacts as results of limited QoD are jitter, jerkiness and freezing which are detailed in Chapter 3. Figure 4a illustrates occurrence of freezing.

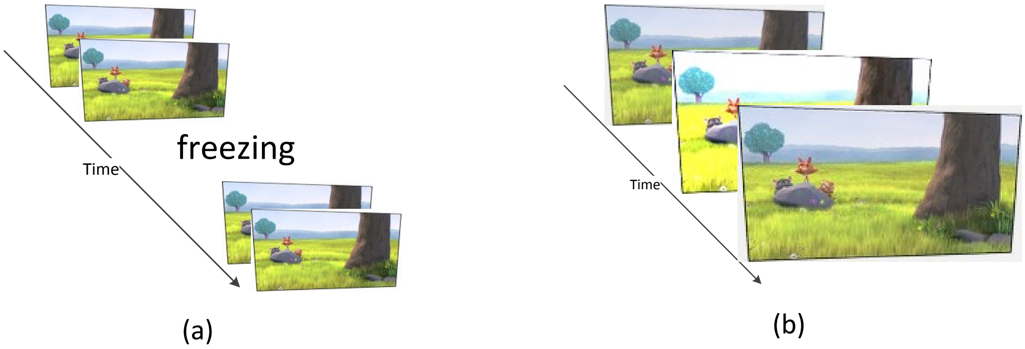


Figure 4 Examples of Video Artifacts

Note that not all temporal artifacts are results of limited QoD. One example of temporal artifacts which is not a result of limited QoD is flickering. To go into detail, flickering is a phenomenon that the light amplitude between video frames varies. Figure 4b illustrates occurrences of the flickering.

It is worthwhile to mention that the concept of QoD can apply to other streaming technologies which have similar characteristics when being compared with video streaming. One of these streaming technologies which can apply the concept of QoD is audio streaming. This is because delayed IP packets causes non-smoothness of audio playback in quite the same way as it happens in video streaming.

The video artifacts' influence on QoD contains two parts, QoD_A (QoD by application) and QoD_N (QoD by network) [36] as shown in Figure 5. To go into detail, QoD_A is described as the result of handling data by the application while QoD_N is described as the result of handling data by the network. Figure 6 illustrates the previously mentioned explanation.

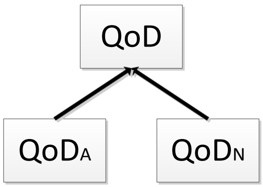


Figure 5 QoD_N and QoD_A

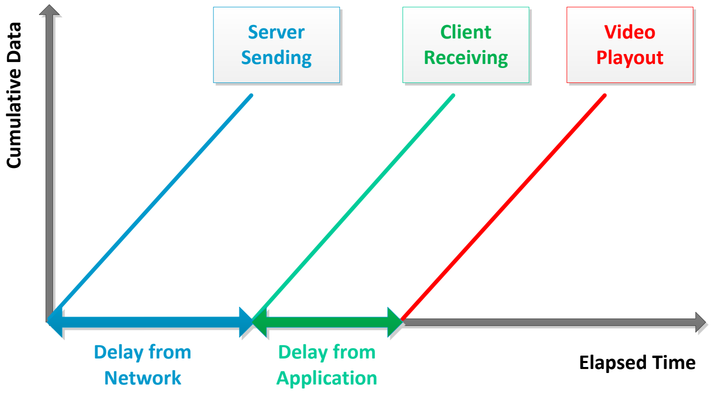


Figure 6 Delays of Data Processing

However, with the high performance of CPU and high reliability of software today, it is likely that the effect of application's performance is far less than the effect of QoD_N. Therefore, QoD is mainly influenced by QoD_N. One research proposes that network performance in terms of loss, jitter of delay, reordering and capacity assignment handover, influences QoD as shown in Figure 7 [14].

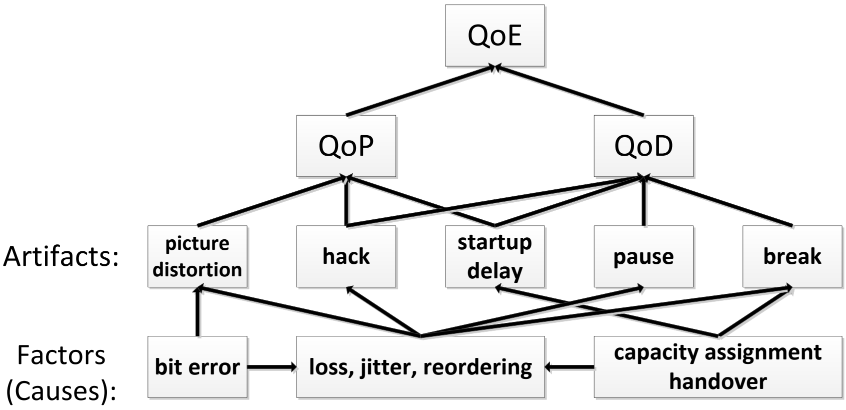


Figure 7 Overview of Video Quality Relationship

2.2.2 How Quality of Delivery is Affected

Figure 8 shows the QoE hourglass model (simplified from [36, 37]), which could be used to match specific networking layers with corresponding video quality layers.

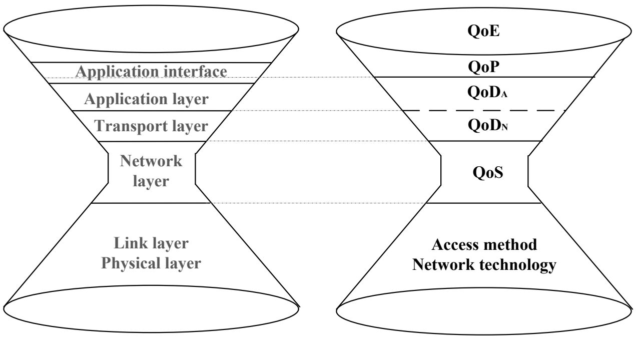


Figure 8 QoE Hourglass Model

However, there are mismatches when making use of this model together with the relationships in Figure 5 and Figure 7. Hence, we modify the hourglass model and get a new basic QoD hourglass model in Figure 9a and our own QoD hourglass model in Figure 9b. Figure 9a is for a system where the network condition could influence QoP directly, for example, lossy streaming. Figure 9b is for HTTP video streaming scenario since the data integrity could be guaranteed by HTTP/TCP protocol suits.

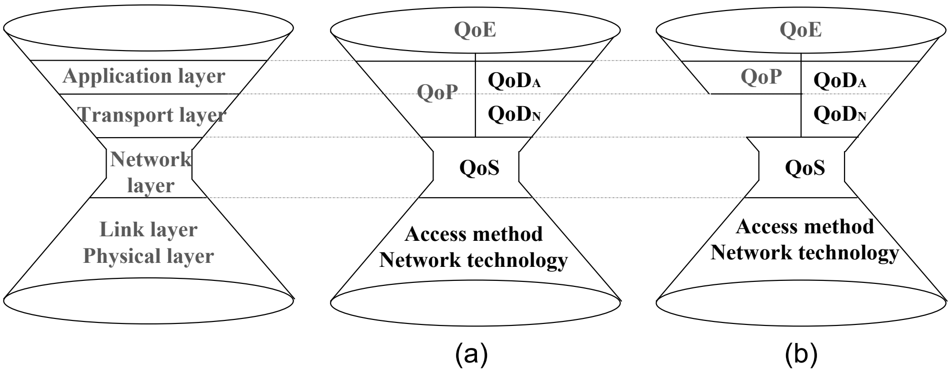


Figure 9 QoD Hourglass Model

2.2.3 Why Study Quality of Delivery

QoD depends on QoS and it is at higher level than QoS. The concept of QoD includes more factors and gets closer to users. For example, QoS is usually measured in the aspects of delay, loss rate, and bandwidth etc. [38] By contrast, QoD could be quantified by checking the startup delay, pauses, and breaks [14] and so on. QoS parameters are easy to measure, but they are far away from the user level. Hence, there is a need for the mapping from QoS level to near-user level.

Besides QoS, QoE is another term increasingly used [39] to evaluate how much the whole service system's quality is satisfying users. Although QoE provides the highest level value in video quality analysis, using QoE to quantify video quality brings up two concerns. The first is that QoE relies on human opinion (Figure 10) which has low reliability. For example, the score may vary from person to person when using the Mean Opinion Score (MOS) method [12]. Reducing the variation should be carried out through taking more samples. However, this is not always possible in most research because of limited budget and resources. The second concern is that the environment and other factors are also influencing QoE besides the factors this project is interested in. For example, MOS may be lower due to time of day when subjective candidates are getting tired from work. Some procedures are developed to reduce these artifacts. However, some hidden factors may exist, and not explored yet. These concerns suggest that QoE is not suitable in this project where resources are limited.

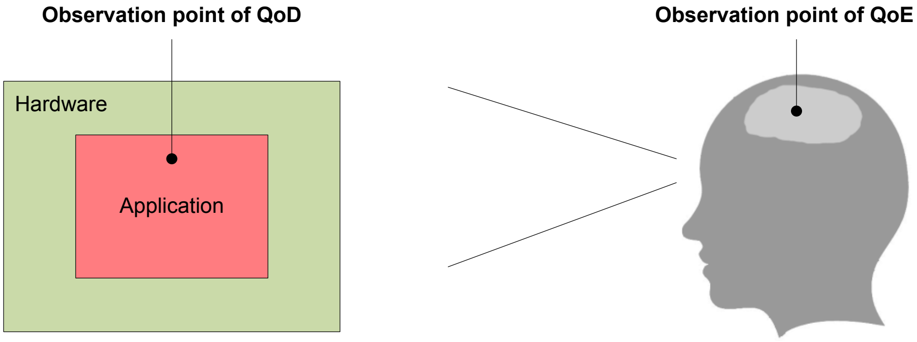


Figure 10 Different Observation Points of QoD and QoE

In summary, QoD is more reliable than QoE and closer to users than QoS. Besides, it may cost only a small quantity of resources to measure QoD. These reasons make QoD the best trade-off choice to study video quality in our work though it could not reflect users' final subjective satisfaction. Nevertheless, we have not found one definition that provides an objective manner which could quantify QoD yet.

3 QoD Quantification and Influential Factors

In this chapter, we establish knowledge to answer the first two RQs: (1) How to quantify QoD? (2) What are the factors which cause QoD? To response these questions, we investigate video artifacts which influence QoD and scope video artifacts to study. After that we establish metrics to quantify the scoped video artifacts by applying IBT, RBF and MDRB. Then, we develop a metric, namely QoD Degradation (QoDD), to quantify QoD in an objective manner. At last, we reveal that DV, TCP memory size, playout buffer length, and video bitrate are influential factors which cause QoD. The first RQ is answered by the development of QoDD. The second RQ is answered by the introduction of influential factors. The knowledge of this chapter provides bases for revealing the relationships between influential factors and QoD through empirical experiments which can be found in the next chapter.

3.1 QoD and Related Video Artifacts

Despite the fact that many video artifacts are results of limited QoD, only some are concerned in this research. We only focus on video artifacts which cause waiting time during video playback for the three following reasons. The first reason is that the waiting time is a result of limited QoD. This is because limited QoD increases time interval when delivering video frames. The second reason is that the waiting time significantly impacts user satisfaction [40]. The third reason is that observation of the waiting is possible at application layer since various functions are available for monitoring the waiting time.

The result of the waiting time is that there is extra required time added to playback duration. To go into detail, the extra required time is a period of time which video player waits for a certain video frame to arrive. For example, freezing unexpectedly increases time interval between video frames. The phenomenon then requires video player to wait until the freezing frame is completely delivered which consequently adds extra required time to complete video playback to playback duration.

Flickering, jerkiness and jitter are also major video artifacts found in video streaming [41]. Flickering is a fluctuation of light magnitude at different temporal positions. Jerkiness is a temporal visual freezing due to dropping video frame by video encoder. Jitter is a temporal visual freezing due to skipping video frame by decoder or frame loss in transmission process. Besides these artifacts, research in [14] suggests other four temporal video artifacts, namely hack, startup delay, pause and break. To explain further about each artifact, hack is a temporal visual freezing without human intervention. Startup delay is the extra required time before video playback can begin. Pause is quite similar to the hack but it occurs in longer time period than the hack, and human likely reacts to such artifact. Break is a permanent visual termination. From the previously mentioned video artifacts, only hack, startup delay, pause and break artifacts are concerned in this research since occurrences of flickering, jerkiness and jitter do not add extra required time to video playback duration. In addition, hack artifact is also neglected because the observation of a very short period of time is not possible without the support from extra tool.

In conclusion, startup delay, pause and break are related video artifacts which are concerned in this research. In the next section, quantification of these artifacts will be discussed.

3.2 Quantification of Artifacts

In this section, we are presenting metrics to quantify startup delay, pause and break artifacts so that extra required time can be evaluated. The process shall begin with exploring previous works to inherit their metrics. Later, the inherited metrics are improved so that startup delay, pause and break artifacts can be completely evaluated.

Previous works have developed various frameworks for video artifacts quantification. However, most of them focus on spatial issues and picture quality, which does not correspond to the concept of QoD. Several works concentrate on temporal issue, delay of video playback. One of them [42] develops a new metric, pause intensity, which is a product of pause event number per unit time and pause duration, to quantify pause artifact of TCP video streaming. Besides, research in [23] also introduces new metrics to evaluate temporal quality of HTTP video streaming. The metrics are IBT, RBF, and MDRB. Looking in detail, the IBT quantifies startup delay artifact by indicating the required time for filling playout buffer before playback is able to start; MDRB quantifies pause artifact by indicating the average time required for refilling playout buffer when the re-buffering event occurs; and the RBF quantifies the pause artifact by indicating the frequency of re-buffering events. Therefore, the total period of pause artifact can be evaluated by product of MDRB, RBF and video length.

We decide to inherit the metrics from [23] to quantify startup delay and pause artifacts. The aforementioned metrics are chosen because they can quantify three from four artifacts while the metric from [42] can quantify only one from four artifacts. Another advantage of inheriting the metrics from [23] in comparison with the metric from [42] is that the metrics independently provide magnitude and time period of re-buffering event rather than their product. This information will provide more understanding about QoD [17].

Since the inherited metrics have not been able to evaluate break artifact yet, we improve their metrics so that break artifact can be evaluated. To do so, we propose that the break artifact can be determined by defining the thresholds of IBT and MDRB as two hours. The reason behinds this is that TCP requires at least two hours to get break artifact. Because in a strong network outage, TCP will cut down a connection due to the timeout of keep-alive timer. The keep-alive timer is usually set to two hours in most implementation [43].

In conclusion, we use IBT, RBF and MDRB metrics to quantify startup delay, pause and break artifacts. The waiting time in terms of extra required time can be computed according to Eq. 1.

$$X = \begin{cases} \text{undefined,} & \text{if break artifacts occur} \\ IBT + (RBF \times MDRB \times L), & \text{otherwise} \end{cases} \quad \text{Eq. 1}$$

Where:
X = total extra required time [second]
IBT = initial buffering time [second]
RBF = rebuffering frequency [Hz]
MDRB = mean duration of a rebuffering event [second]
L = total video length [second]

3.3 QoD Quantification

As previously mentioned, only extra required time in video playback is concerned in this research, magnitude of extra required time is a good indicator of level of QoD. We call such level of QoD as degradation of QoD. To go into detail, the longer extra required time, the higher level of degradation of video quality in temporal domain.

To compute degradation of QoD, we develop a new metric, namely QoD Degradation (QoDD) to present degradation of QoD due to occurrence of extra required time. QoDD is defined as the amount of extra required time (X) to complete video playback in proportion of video length (L). The definition formula is shown in Eq. 2.

$$QoDD = \frac{X}{L} \quad \text{Eq. 2}$$

Where:
X= total extra required time [second]
L = video length [second]

In an ideal case, the extra required time, *X*, is equal to zero. Hence, the highest level of QoDD is zero. In such case, an audience will not experience any buffering while watching video. In practice, however, the extra required time exists. In such case, the audience will experience some buffering while watching video and perceives lower video quality than in the ideal case. Eq. 3 presents QoDD in terms of IBT, RBF, MDRB and video length.

$$QoDD = \begin{cases} \text{undefined,} & \text{if break artifacts occur} \\ \dfrac{IBT + (RBF \times L \times MDRB)}{L}, & \text{otherwise} \end{cases} \quad \text{Eq. 3}$$

Where:
IBT = initial buffering time [second]
RBF = rebuffering frequency [Hz]
MDRB = mean duration of a rebuffering event [second]
L = total video length [second]

It is important to mention that, for valid QoDD comparison between different cases, the same video length and video content should be applied. Otherwise, dissimilar workloads are given, which causes invalid comparison between different cases. In general, the exact same video file should be used on the different cases for valid comparison to ensure the same workload.

By developing QoDD, we can determine QoD in an objective manner. In the next section, we are going to discuss artifacts which influence QoD since occurrences of these artifacts increase the extra required time to complete video playback and degrade QoD.

3.4 Exploration of Influential Factors

Evidences of influential factors are found in mathematic models of IBT, RBF, and MDRB. These equations are proposed in [23]. The equations of IBT, MDRB and RBF are shown in Eq. 4 – Eq. 6. In these equations, playout buffer length, empty threshold of playout buffer, average TCP throughput, video bitrates, and video length are variables. Therefore, the fluctuating magnitude of these variables changes the values of IBT, RBF, and MDRB which exhibits influences QoD in terms of QoDD.

$$IBT = \frac{B_F \times V}{BW} \quad \text{Eq. 4}$$

$$MDRB = \frac{(B_F - B_E) \times V}{BW} \quad \text{Eq. 5}$$

$$RBF = \begin{cases} 0, & \text{if } BW \geq V, \\ \dfrac{\left\lceil \dfrac{L - \dfrac{B_F}{\left(1 - \frac{BW}{V}\right)}}{\dfrac{B_F - B_E}{1 - \frac{BW}{V}}} \right\rceil}{L}, & \text{otherwise.} \end{cases} \quad \text{Eq. 6}$$

Where:
B_F = a playout buffer length threshold which causes a FULL signal [second]
B_E = a playout buffer length threshold which causes an EMPTY signal [second]
V = video bitrate [bit per second]
BW = average TCP throughput [bit per second]
L = total video length [second]

Based on the conclusions above, we explore influential factors by investigating what cause changes of the variables in the functions. We discard some variables in the writing as they rarely altered or their fluctuations do not correspond to system variance. Empty threshold of playout buffer is the first variable to be neglected. The reason is that the empty threshold often cannot be modified. For example, Adobe does not provide any method to configure the empty threshold [44]. The second variable to be discarded is video length as it depends on video material, but it is not related with system performance.

In conclusion, we explore influential factors by concentrating on investigating factors which cause changes to threshold (B_F), average TCP throughput (BW), or video bitrates (V). The exploration shall begin with factors in the network-stack layer to a factor in the network layer. At last, factors in the application layer are revealed.

3.4.1 TCP Memory Size

At the network-stack layer, various implementations of TCP result in differences of average TCP throughput. The research in [45] shows that size of TCP socket buffer size greatly correlates with TCP throughput. This is because the size of TCP socket buffer regulates sending windows by limiting advertised receiver window and limiting sending buffer space. The work also reveals that if the size of TCP socket buffer is correctly configured, TCP throughput is on a high level which is close to maximum link capacity. However, if the size is poorly chosen, the throughput is limited by maximum sending window size, or from burst of TCP flow. In addition, the research in [46] illustrates that tuning size of socket buffer on both client and server are more effective than tuning only one side. However, socket buffer size is limited by TCP memory size since Linux kernel 2.4 [47].Therefore, TCP memory size influence average TCP throughput. The relationship between TCP memory size and average TCP throughput is shown in Eq. 7.

3.4.2 TCP Congestion Control Algorithm

Apart from TCP memory size, various implementations of TCP Congestion Control Algorithms (CCAs) bring about heterogeneous TCP throughput. Examples of such case are found in [48, 49]. The reason which causes diverse throughput is that CCA manipulates the size of congestion window, and different algorithms provide different ways to control. As a result, sending window size is diverse from one algorithm to another, which consequently results in heterogeneous throughput. The relationship between TCP memory size, congestion window size and average TCP throughput is shown in Eq. 7.

$$\begin{aligned} BW &= \frac{S_{wnd}}{RTT} \times 8 \\ &= \frac{\min\{W_{con}, W_{ack}\}}{RTT} \times 8 \\ &(\text{since: } W_{con} < S_{buf}; W_{ack} < R_{buf}) \\ &\le \frac{\min\{S_{buf}, R_{buf}\}}{RTT} \times 8 \\ &(\text{since: } S_{buf} < W_{mem}; R_{buf} < R_{mem}) \\ &\le \frac{\min\{W_{mem}, R_{mem}\}}{RTT} \times 8 \end{aligned} \qquad \text{Eq. 7}$$

Where:
BW = average TCP throughput [byte per second]
RTT = connection round-trip time [second]
Swnd = sending window size [byte]
Wcon = sender congestion window size [byte]
Wack = receiver acknowledged window size [byte]
Sbuf = sender socket buffer size [byte]
Rbuf = receiver socket buffer size [byte]
Rmem = TCP read memory [byte]
Wmem = TCP write memory [byte]

3.4.3 Delay Variation

By investigating at network layer, Delay Variation (DV) significantly causes changes in TCP throughput. The research in [50] reveals that high degree of DV causes spurious TCP timeout which consequently induces unnecessary data retransmission. The phenomenon of unnecessary data retransmission occurs because the packet with extra high delay exceeds Retransmission Time-Out (RTO) timer. The packet is then treated as packet loss which triggers retransmission and reduces sending windows size. In order to mitigate the effect of such phenomena, increasing tracked packets' Round-Trip Time (RTT) to increase DV has been proved to regain some throughput since RTO timer is raised higher [51]. Figure 11 illustrates unnecessary retransmission which is caused by a high-delay packet and spurious timeout.

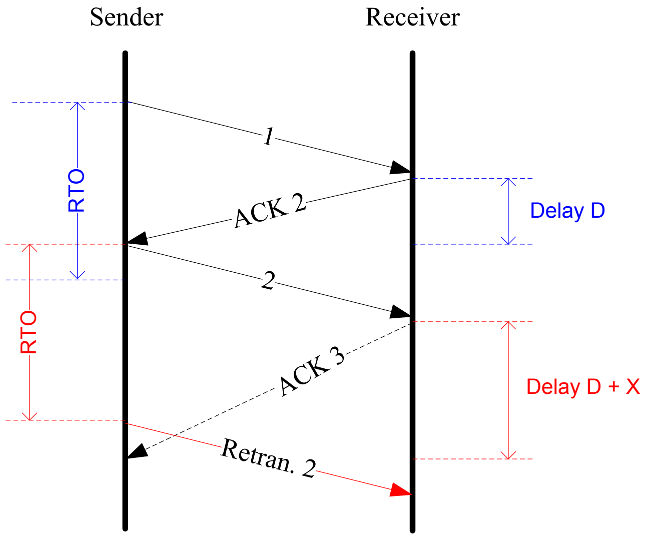


Figure 11 Spurious Retransmission

3.4.4 Playout Buffer Length

At application layer, various designs of video player lead to different video quality. One research proposes several models of video player from simple stalling model, which has very short length of playout buffer, to YouTube model, which has a longer playout buffer [22]. Their experiments reveal that the number and duration of re-buffering events change when playout buffer length is changed. Similar results are found in [52] where long length of playout buffer reduces probability of re-buffering event while increasing re-buffering duration.

3.4.5 Video Bitrate

Another factor in application layer which greatly influences video quality is video bitrate. The higher bitrate, the more data needed to fill the playout buffer (given a fixed buffer length, for example, 3 seconds), so the more time to start play or resume from rebuffering. Today, many video content publishers offer various video bitrates to users. Examples can be found on YouTube [5] and Metacafe [7] where choices of video bitrate (in terms of resolution) are provided to users.

In conclusion, TCP memory size, TCP CCA, DV, playout buffer length and video bitrate are influential factors of QoD. Other factors are neglect in the writing since they are rarely altered or their fluctuations do not correspond with system variance. To provide a clear picture of the influential factors, [36] categorizes these factors into different layers. Figure 12 illustrates these new proposed factors according to our QoD hour glass model (Figure 9, page 9).

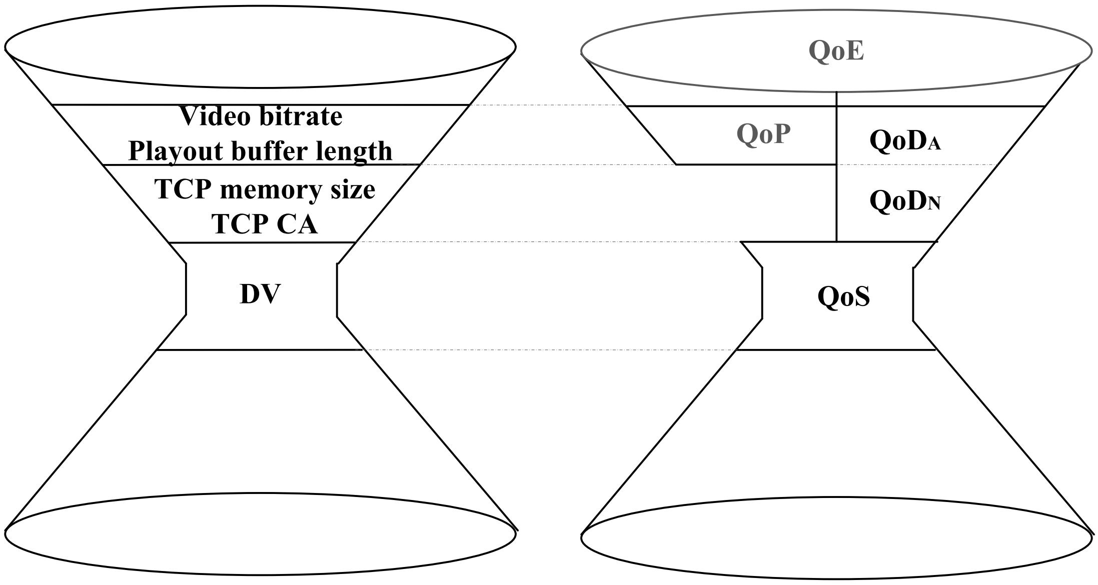


Figure 12 Factors and the QoD Hour Glass Model

Up to now, we developed a technique to quantify QoD and got influential factors which degrade QoD. Furthermore, the RQs 1–2 are answered. In the next chapter, empirical experiments are designed to investigate the relationships between the influential factors and QoD.

4 Experiment Design

In this chapter, we study relationships between influential factors and QoD in terms of QoDD so that RQs 3–7 can be preliminarily answered. To do so, we design experiments to observe the magnitude of IBT, MDRB, RBF and QoDD when influential factors are varied. In brief of experiment design, three main computers are used to delivery video. These are a streaming server where video files are stored, a network emulator for emulating network factors and a client where video files are played. Figure 13 illustrates the aforementioned three main computers and their connections.

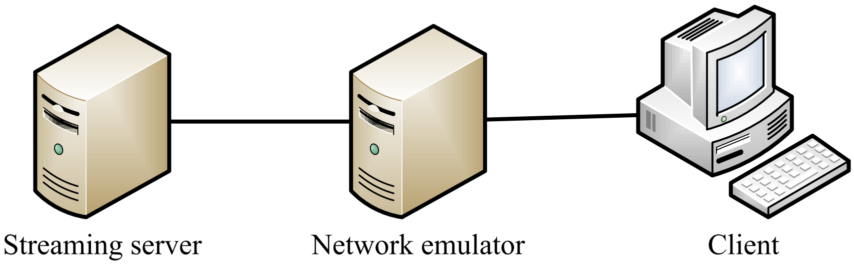


Figure 13 Basic Experiment System

The design process begins with "Parameter Space" where influential factors are factorized, and experiments are defined. After that, test bed and tools which are required to fulfill the experiments are chosen or developed. Finally, a new developed video player for QoD quantification is validated through performance evaluation.

4.1 Parameter Space

Here we design the parameter space for a series of experiments as shown below.

4.1.1 Experiment 1: TCP Memory Size

This experiment tries to answer RQ 3.

To observe how TCP memory size affects QoD, we increase TCP-write-memory size on server and TCP-read-memory on client from 16 KB to 64 MB while fixing other TCP memories at 250 KB. There are two main reasons for choosing such TCP memory sizes. The first reason for fixing other TCP memories at 250 KB is that other TCP memories, which are TCP-read-memory on server and TCP-write-memory on client, do not involve in downlink throughput as described in [45]. The second reason for setting 16 KB as a minimum size is that it is the default initial TCP memory size in Linux kernel 2.6 [47]. Table 1 on page 21 reviews previously mentioned configurations. The rest of configurations are set according to baseline 1 which is shown in Table 2 on page 22.

4.1.2 Experiment 2: TCP Congestion Control Algorithm

This experiment tries to answer RQ 4.

To observe how different TCP CCAs affect QoD, we change TCP CCA from the default CUBIC to Reno, Highspeed, Westwood, Hybla, Illinois, Scalable, Vegas, Veno, BIC, CTCP, HTCP and YEAH. The reason behind choosing such TCP CCAs is that these algorithms are dominantly available on Internet today [53]. However, we have to exclude CTCP from this experiment since the website which provides a required patch for enabling CTCP on Linux is not available during the project time [54]. Besides, LP is available by default on Ubuntu 10.04 in addition to previously mentioned algorithms.

In addition, the research in [55] found that some CCA performed better than others under a specific delay, DV and loss. Therefore, we also changes one-way DV for each direction between 0–50ms and loss between 0–1% in order to observe relationships between these three factors. It is worth to note that we always mean the same “one-way” configuration for both directions unless with other statements. Table 1 reviews previously mentioned configurations. The rest of configurations are set according to baseline 2 which is shown in the Table 2.

4.1.3 Experiment 3: Delay Variation

This experiment tries to answer RQ 5.

To observe how different levels of DVs affect QoD, we increase DV from 0ms to 50ms. In addition, we also change delay between 1 and 245ms in order to observe how various levels of delay together with DV affect QoD. It is important to note that TCP-write-memory sizes of server and TCP-read-memory size of client are set to 1 MB since we observed from previous experiment that QoD is low when the sizes are set to 1MB. Table 1 reviews previously mentioned configurations. The rest of configurations are set according to baseline 2 which is shown in the Table 2.

4.1.4 Experiment 4: Playout Buffer Length

This experiment tries to answer RQ 6.

To observe how TCP playout buffer length affects QoD, we change playout buffer length between 1 second and 6 seconds. The reason for choosing such playout buffer lengths is that the YouTube uses 3 seconds in its video player [23]. Moreover, we observes that a playout buffer length which is shorter than 1 second is not suitable for being used since it generates unexpected *NetStream* signals which do not correspond with a state diagram shown in Figure 13. Therefore, we investigate playout buffer length between 1–6 seconds.

In addition, we also change random loss rate and DV together with changes of playout buffer length so that we can observe relationships between these three factors. To do that, we separate this experiment into two sub-experiments. In the first sub-experiment, we change loss between 0–1% together with changes of playout buffer length between 1–6s. For the second sub-experiment, we change DV between 0–10ms together with changes of playout buffer length between 1–6 seconds. Table 1 reviews previously mentioned configurations. The rest of configurations are set according to baseline 3 which is shown in the Table 2.

4.1.5 Experiment 5: Video Bitrate

This experiment tries to answer RQ 7.

To observe how video bitrate affects QoD, we change video resolution between 360P, 480P, 720P and 1080P in order to produce video bitrates of 378, 624, 1426 and 3364 Kbps. The reason for choosing such video resolution is that we observe that YouTube offers those resolutions to user.

Furthermore, we also change random loss rate and DV together with changes of video bitrate so that we can observe relationships between these three factors. To do that, we separate this experiment into two sub-experiments. In the first sub-experiment, we change loss between 0–1% together with the changes of video resolutions which were previously described. For the second sub-experiment, we change DV between 0–14ms together with the changes of video resolutions. Table 1 reviews previously mentioned configurations. The rest of configurations are set according to baseline 3 which is shown in the Table 2.

Table 1 Parameter Space

Factors	Experiment Number	Independent Variables	Control Variables
TCP memory size	Experiment 1: TCP memory sizes	▪ TCP-write-memory 16 KB – 64 MB [by 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 4MB, 8MB, 16MB, 32MB, 64 MB] ▪ TCP-read-memory 16 KB – 64 MB [by 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 4MB, 8MB, 16MB, 32MB, 64 MB]	Baseline 1
TCP CCA	Experiment 2.1: TCP CCAs with different levels of random loss rate	▪ Reno, Highspeed, Westwood, Hybla, Illinois, Scalable, Vegas, Veno, BIC, CUBIC, HTCP, YEAH and LP. ▪ Random loss 0 – 1% [0%, 0.2%, 0.4%, 0.6%, 0.8%, 1.0% for both directions]	Baseline 2
	Experiment 2.2: TCP CCAs with different levels of DV	▪ Reno, Highspeed, Westwood, Hybla, Illinois, Scalable, Vegas, Veno, BIC, CUBIC, HTCP, YEAH and LP ▪ DV 0 – 50 ms [10, 20, 30, 40, 50 ms for both directions]	
Delay Variation	Experiment 3: DVs with different levels of delay	▪ DV 0 – 50 ms [1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49 for both directions] ▪ Delay 1 – 245 ms [1, 11, 21, 31, 41,50, 65, 80, 95,110, 125, 140, 155,170, 185, 200, 215,230, 245 ms for both directions]	
Playout buffer length	Experiment 4.1: Playout buffer length with different levels of loss rate	▪ Playout buffer length 1 – 6 seconds ▪ Random loss 0 – 1% [0, 0.2%, 0.4%, 0.6%, 0.8%, 1.0% for both directions]	Baseline 3
	Experiment 4.2: Playout buffer length with different levels of loss rate	▪ Playout buffer length 1 – 6 seconds ▪ DV 0 – 10 ms [0, 2, 4, 6, 8, 10 ms for both directions]	
Video bitrate	Experiment 5.1: Video bitrate with different levels of loss rate	▪ Video resolution 360P (378Kbps), 480P (624Kbps), 720P (1426 Kbps) and 1080P (3364Kbps) ▪ Random loss 0 – 1% [0, 0.2%, 0.4%, 0.6%, 0.8%, 1% for both directions]	
	Experiment 5.2: Video bitrate with different levels of loss rate	▪ Video resolution 360P (378Kbps), 480P (624Kbps), 720P (1426 Kbps) and 1080P (3364Kbps) ▪ DV 0 – 14 ms [0, 2, 4, 6, 8, 10 ms for both directions]	

Table 2 Experiment Baselines

Control variables	Configurations
Baseline 1	Server ▪ CCA: CUBIC ▪ TCP-write-memory size: 250 KB ▪ TCP-read-memory size: 250 KB ▪ Ubuntu 10.04 Client ▪ CCA: CUBIC ▪ TCP-write-memory size: 250 KB ▪ TCP-read-memory size: 250 KB Video ▪ Playout buffer length: 3 seconds ▪ Video resolution: 1080P ▪ Video file: Sintel.flv Network condition ▪ One-way delay: 200ms for each direction ▪ Link bandwidth: 10 Mbps Others ▪ Repetition of each test: 32 times, otherwise specified
Baseline 2	▪ Based on Baseline 1 ▪ Replace TCP-write-memory size on server with 1MB ▪ Replace TCP-read-memory size on client with 1MB
Baseline 3	▪ Based on Baseline 2 ▪ Replace CCA on server side with Hybla

4.2 Development of Test Bed and Tools

After the decision of tuning parameters, this subchapter discusses about components and tools to conduct and measure the previously designed experiments.

4.2.1 Test Bed

Though there are three main computers involved in video delivery process which was mentioned before, additional two computers are designed to support experiments. Therefore, total five computers are designed to use in our study. These computers are streaming server, network emulator, client, measurement point (MP), and experiment controller.

To go into detail of each computer, the streaming server is set up with Nginx version 1.2.0. The reason for choosing Nginx is that it consumes lower memory and is more efficient than Apache [56]. The low memory consumption of Nginx is critical in our experiment since our streaming server has quite small amount of RAM.

The client is setup with a video player for QoD quantification. More details about the video player can be found in section 4.2.2.

The network emulator is set up with NetEm. There are two reasons for choosing NetEm. The first reason is that it is built into Linux kernel. Therefore, no additional software is required. The second reason is we experience that NetEm could work in an accurate way [57, 58] and it provides higher accuracy for random loss rate when being compared with KauNet.

Distributed Passive Measurement Infrastructure (DPMI) version 0.7.6 is installed on the MP computer [59]. There are two reasons for choosing DPMI. Firstly, DPMI provides higher time accuracy than general tcpdump since DPMI is built with high-precision hardware timestamp clock [59, 60]. Secondly, we could get support from DPMI developers so that the modification of DPMI to match project requirement is more convenient than using other alternative tools.

In addition to the previously mentioned four computers, we also setup another computer to be an experiment controller. The controller is setup with our "automatic experiment controller" which is detailed in section 4.2.3.

Clocks on all the computers are properly synchronized with a GPS-NTP server. All the computers are set up with Ubuntu 10.04 32 bit, except MP which is setup with Crux 2.6. The hardware specification of streaming server and client can be found in Table 3 below.

Table 3 Server and Client Specification

Server	Client
CPU: Intel Celeron 700 MHz RAM: 384 MB Network Card: D-Link DFE-530TX REV-A3-1 10/100 PCI Ethernet Adapter OS: Ubuntu 10.04 32 bit (server version)	CPU: AMD Athlon 64 X2 5000+ 1GHz RAM: 2 GB. Network Card: D-Link DFE-530TX REV-A3-1 10/100 PCI Ethernet Adapter OS: Ubuntu 10.04 32 bit (desktop version)

For networking information, the network is setup with a 10 BASE-T full-duplex connection which offers a maximum bandwidth of 10Mbps. Two wiretaps (VSS monitoring 10/100 1x1) are installed on the network so that more accurate one-way measurement is possible. Figure 14 illustrates all aforementioned components together with the network setup. It is notable that the experiment controller is not shown in the figure since it is not directly involved in the experiments and result.

At this point, computers and network are setup. In the next paragraph, we are going to choose a video streaming platform for the experiments.

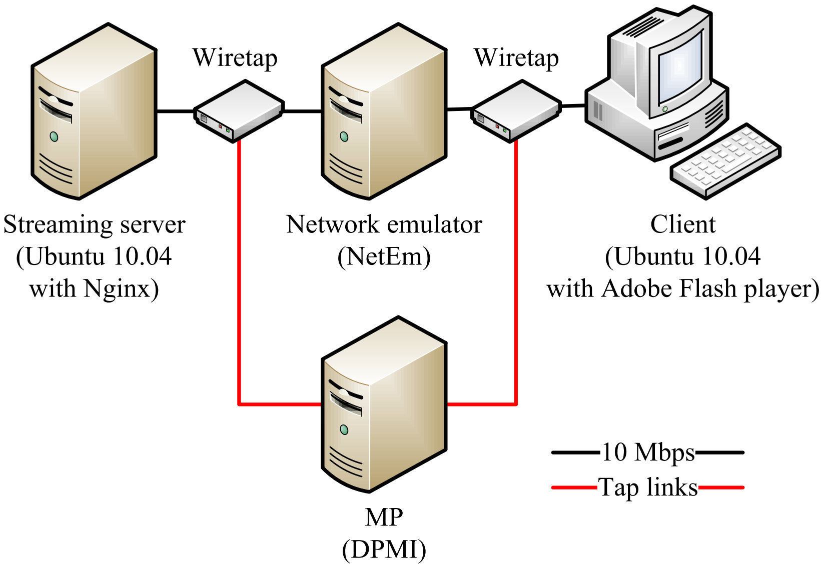


Figure 14 Complete Experiment System

Despite many video platforms have been developed to offer video streaming over HTTP (for example, QuickTime, Adobe Flash, Silverlight, and HTML5), not all the platforms are widely available today. One video platform which has been available and widely used for a long time is Adobe Flash [61]. The availability of Adobe Flash can be seen from supports by popular video content publishers such as YouTube, Vimeo and Metacafe. Though some of these video content publishers are evolving their systems to support HTML5 together with Adobe Flash, the development of HTML5 has not completed yet [62] and may require many years to be done. For the aforementioned reasons, Adobe Flash is chosen to be a multimedia framework in this research.

With regard to video delivery method, we apply progressive download because it provides simpler implementation than the one produced by HTTP dynamic streaming since progressive download does not require extra software installed on server side [63] as required by HTTP dynamic streaming [64].

In the following sections we are presenting development of two tools which are used in the experiments. The first tool is a video player for QoD quantification. The second tool is an automatic test system for assisting empirical experiments.

4.2.2 Development of Video Player for QoD Quantification

In order to quantify QoD, a video player is developed for quantification of startup delay and pause artifacts in terms of IBT, MDRB and RBF. To develop the video player, Adobe-Apache Flex [65] is chosen as a development tool for developing the video player, based on Adobe Flash technology. The benefit of using such development tool is that it is publicly available throughout the Internet while the other tool, namely Adobe Flash CS5, is only available for commercial purpose.

Development of video player begins with writing a code in ActionScript3 (AS3), followed by compiling the code to an “swf” file (the player) with Apache Flex SDK. Usually, the completed video player is embedded to an HTML webpage which is later executed and run in a web browser.

However, in this research, we decide to run the video player on Adobe Flash debugger instead of web browser. Since the debugger offers trace generation which is useful to track video player operation, and making use of debugger could get rid of the influence of the browsers. For more details about software versions, Flex version 4.6 and Flash debugger version 11.2 are used in this research.

With regard to coding, the main classes which are used to implement a video player are *NetConnection*, *NetStream* and *StageVideo* classes. To go into detail of each class, the *NetConnection* class is used to establish a two-way connection between client and server. The *NetStream* class is used for accessing video content over an established connection. The *StageVideo* class provides decoding and visualization of video picture. To code a simple video player, a *NetConnection* object is constructed first. Secondly, the *NetStream* object is constructed over the *NetConnection*, and then the object is attached to *StageVideo*. Finally, *NetStream.play()* is executed to begin video playback. Code 1 is an example of basic video player code which is previously described.

Code 1

```
nc = new NetConnection();
nc.connect (null);
ns = new NetStream(nc);
vid.attachNetStream ( ns );
ns.play("http://.../vdo.flv");
```

To track startup delay and pause artifacts, *NetStatusEvent* class that contains signals and informs playback status is used. To get the signals, an event listener must be attached to *NetStream* object and passes inherited signals to another function for further processing. Table 4 shows the signals and corresponding playback status. Code 2 shows an example of getting playback signals.

Measuring IBT and MDRB can be achieved by the following processes. Measuring IBT by reading a timestamp before executing *NetStream.Play()*, and reading another timestamp after receiving *NetStream.Buffer.Full*. Then IBT is the time difference between these two timestamps. Measuring MDRB could be done by reading a timestamp after receiving *NetStream.Buffer.Empty*, and reading a timestamp after receiving *NetStream.Buffer.Full*. The time difference is added to a total re-buffering time counter. Besides, the counter of re-buffering events is also increased for every received *NetStream.Buffer.Empty* event. The MDRB is an average of time differences between the empty-full timestamp pair of each re-buffering event. Such procedures are briefly described in Table 5. The flowing diagram of overall process is illustrated in Figure 15. It is important to note that in Figure 15 we observe unexpected behaviors that the *NetStream.Buffer.Full* and *NetStream.Buffer.Empty* may repeat their state when playout buffer length is not set properly. However, with a playout buffer which is not less than 1s and careful detection in the coding, we don't experience such case during the experiment.

Table 4 Adobe Flash Signals and Status

Signals	Playback Status
NetStream.Play.Start	Video streaming has started
NetStream.Play.Stop	Video streaming has stopped
NetStream.Buffer.Empty	Data in playout buffer reach the low threshold and video playback is pause
NetStream.Buffer.Full	Data in playout buffer reach the high threshold and video playback is resume
NetStream.Buffer.Flush	Steaming is completed. All video data is resided in playout buffer

Code 2

```
ns.addEventListener(NetStatusEvent.NET_STATUS, statusHandler);
function statusHandler(event:NetStatusEvent):void{
    trace(event.info.code);
}
```

Table 5 Artifacts, Metrics and Procedures

Artifacts	Metrics	Procedures
Hack	Not available due to limitation of study (described in Chapter 3)	
Startup Delay	IBT	1. Record a timestamps when executes *NetSream.play()* 2. Record another timestamp when receive *NetStream.Buffer.Full.* 3. Calculate the time difference for IBT.
Pause	MDRB	1. Record a timestamps when receives *NetStream.Buffer.Empty* while playing video 2. Record another timestamp when receiving *NetStream.Buffer.Full*. 3. Calculate the time difference for increasing the total re-buffering time. 4. Count number of *NetStream.Buffer.Empty* signals 5. Once video playback is complete, calculate MDRB from the total re-buffering time by the number of *NetStream.Buffer.Empty* signals
	RBF	Once video playback is complete, calculate RBF from the number of *NetStream.Buffer.Empty* signals by video length
Break	IBT & MDRB	Observe thresholds of IBT and MDRB not to exceed 2 hours

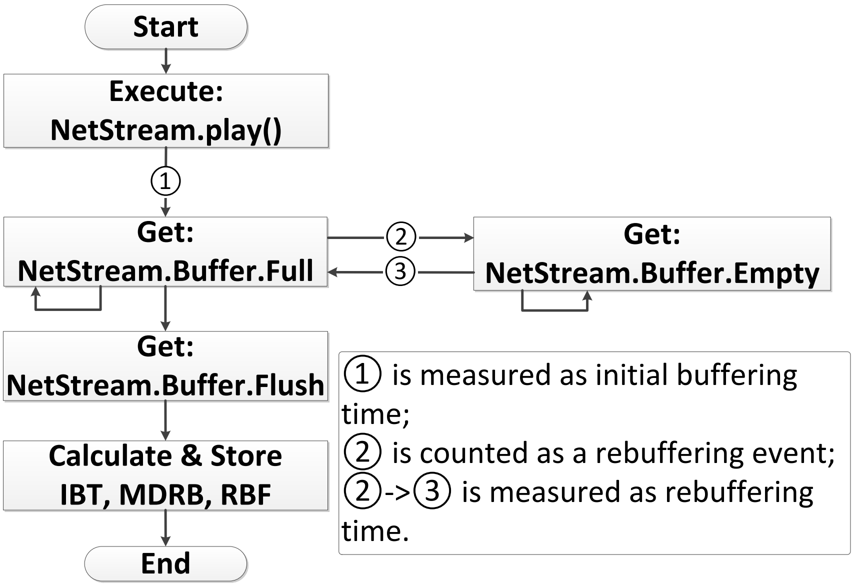

Start
Execute: NetStream.play()
①
Get: NetStream.Buffer.Full
②
③
Get: NetStream.Buffer.Empty
Get: NetStream.Buffer.Flush
Calculate & Store IBT, MDRB, RBF
End
① is measured as initial buffering time;
② is counted as a rebuffering event;
②->③ is measured as rebuffering time.

Figure 15 NetStream Diagram

4.2.3 Development of Automatic Test System

Since this research is based on empirical experiments, it is found that a lot of manual operations are required to be done in order to perform the experiments. Examples of such operations are setting configurations, monitoring system and collecting experiment results. Nevertheless, there is limited man power to handles such operations in this research. Therefore, it is necessary to develop an automatic system so that all experiments can be done within available man power.

From the aforementioned need, in the research, an application named experiment controller is developed to assist the experiments. The experiment controller is designed to provide automatic tasks handling so that experiments require lower human intervention than before. Besides, the software also improves reliability of performing experiment since computers has higher reliability than human being, especially when repeating actions [66].

Tasks which are handled by the experiment controller are preparing database, distributing experiment configurations and saving results of the experiment to database. To go into detail about preparing database, the experiment controller reads a configuration file then creates a table in database. In the next process, each row in the table is filled with a unique test number, and each column is filled with experiment configurations. Experiment configurations are loss rate, delay, DV, playout buffer length, video bitrate, TCP memory size and TCP CCA. For details about distributing experiment configurations, experiment controller reads experiment configurations from database, and then sends to other computers over a control network. On each computer, there is a manager application, an additional application designed for task handling, which receives the experiment configurations and applied such configuration to its machine.

To go into detail about manager applications, we develop the following managers to assist experiment controller: TCP configuration manager, MP manager and emulator manager. The TCP manager takes responsibilities for adjusting TCP memory size and changing TCP CCA. The applications are installed on webserver and client. MP manager is responsible for enabling data capturing. Emulator managers take responsibilities for setting loss rate, setting delay and setting DV. The player manager will restart the player for every video playback. Apart from these managers, the video player also communicates with the experiment controller to receive playout buffer length and video bitrate configurations, and to send results of the experiment back to experiment controller once each test is done.

To implement the experiment controller and their related managers, Perl 5 is used. The reason for choosing Perl is that it is a scripting language so that modification of these applications is more convenient than choosing other languages. In addition, Perl is ready to use without extra installation on Ubuntu 10.04.

For more information about implementation, the TCP memory size is adjusted via commands "*sysctl -w net.ipv4.tcp_rmem*" and "*sysctl -w net.ipv4.tcp_wmem*". The minimum, default and maximum values of TCP memory are assigned to be the same value in order to force OS to only use the given size regardless of server memory and connection performance [47, 67]. Emulating delay and DV are done by setting one-way delay and one-way DV on down and up links. Such settings are done via commands "*tc qdisc add dev ethX parent 1: handle 10: netem delay ${delay}ms ${DV}ms distribution normal loss ${loss}%*". Playout buffer length is set via *NetSteam.bufferTime*. Video files are encoded to different resolution from frame-pictures using software FFmpeg.

In conclusion, the experiment controller and their related managers are developed in this project in order to reduce manual operation required to perform experiment. The design is illustrated in Figure 16.

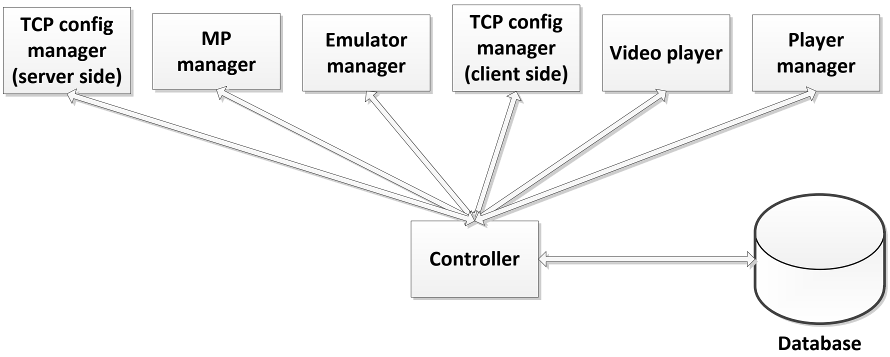


Figure 16 Automatic Test System

4.3 Validation of Video Player Performance

In this section, we evaluate a developed tool to ensure suitability of experiment design which will bring about quality of experiment result. To go into detail, there are two concerns which are found by using our video player. The first one is that performance of Adobe Flash debugger is currently unknown. The second concern is that timestamp accuracy of Adobe Flash is never revealed before. These concerns bring about evaluation of correctness of IBT, MDRB and RBF since these metrics depends on the performance of video player and timestamp accuracy. To clear the aforementioned concerns, two experiments are conducted. The performance of Adobe debugger when being compared with web browser is examined in one experiment while the timestamp accuracy of Adobe Flash is inspected in the other one.

4.3.1 Performance of Adobe Debugger

The reason why Adobe Flash debugger may show different performance from realistic system, a web browser, is that the debugger normally handles more activities than a normal application. Such extra activities are code debugging, log generation and resource monitoring. Debugger application therefore runs slower than normal application, and may provide unreliable performance on time sensitive tasks such as measuring IBT, MDRB and RBF.

To investigate the performance of Adobe Flash debugger when being compared with web browser, two different systems are set up. The first system runs video player application in a web browser, Firefox version 10.9.8 with Adobe Flash plugin version 11.10.98. For the second system, it runs the video player application on Adobe Flash debugger version 11.2. Besides, other configurations are set up according to baseline 1 which can be found in Table 2 (page 22). For further details, video playback is repeated 750 times. Average and Standard Deviation (SD) of IBT, MDRB and RBF of each system are computed and shows in Table 6 below. The result shows that the performance of Flash debugger is different from the performance of web browser plugin.

Table 6 Web Browser and Flash Debugger

Players	IBT[ms]		MDRB[ms]		RBF[H$_Z$]	
	Mean	SD	Mean	SD	Mean	SD
Web Browser	4216.48	49.09	2432.00	60.51	0.01	0
Flash Debugger	4207.03	61.19	2931.78	72.46	0.01	0

4.3.2 Accuracy of Timestamp Function

Since measurement of IBT and MDRB are implemented using Adobe's timestamp function, it is necessary to ensure accuracy of the timestamp function since inaccurate timestamp brings about untrustworthiness measurement.

Research in [68] develops a method to evaluate timestamp accuracy of application by referencing with high accuracy timestamp system, named DPMI (detailed in Section 4.2.1). We apply the aforementioned method since access to DPMI is available. To implement the method, three applications are developed. These applications are packet sender, packet receiver and network observer. Sequences of verification process are described as follows. The sender sends out one packet every 10ms for 1 500 000 packets (the packets which cannot be distinguished by the player are discarded). To go into detail, the number of generated packet is 100 times of the number of packets for each video which is used in this research. The receiver records all the generated packets, generates timestamp of each packet. Then we calculate the Inter-Packet Time (IPT) between every two continuous packets (IPT$_{\mathbf{r},\,\mathbf{a}}$). The network observer reads high-accuracy timestamp from MP and then calculates the time interval at link layer (IPT$_{\mathbf{r},\,\mathbf{l}}$) using high accuracy timestamp. At last, the timestamp accuracy (T_Δ) is calculated by equations Eq. 8 - Eq. 10 [68]. The result shows that each timestamp's accuracy error is 22ms.

Figure 17 illustrates the aforementioned process. Figure 18 shows ε of each IPT pair as well as the statistics of ε. Figure 19 provides the distribution of ε.

$$IPT_{x,y}(K,K+1) = T_{x,y}(K+1) - T_{x,y}(K) \qquad \text{Eq. 8}$$

$$\varepsilon_{K,K+1} = IPT_{r,a}(K,K+1) - IPT_{r,l}(K,K+1) \qquad \text{Eq. 9}$$

$$T_\Delta = \left|\min(\varepsilon_{K,K+1})\right| + \left|\max(\varepsilon_{K,K+1})\right| \qquad \text{Eq. 10}$$

Where:
x = side (sender / receiver)
y = layer (application / link)
K = packet index number
T = a timestamp of one packet [ms]
ε = difference of one IPT pair (application layer IPT and corresponding link layer IPT) [ms]
T_Δ = timestamp accuracy [ms]

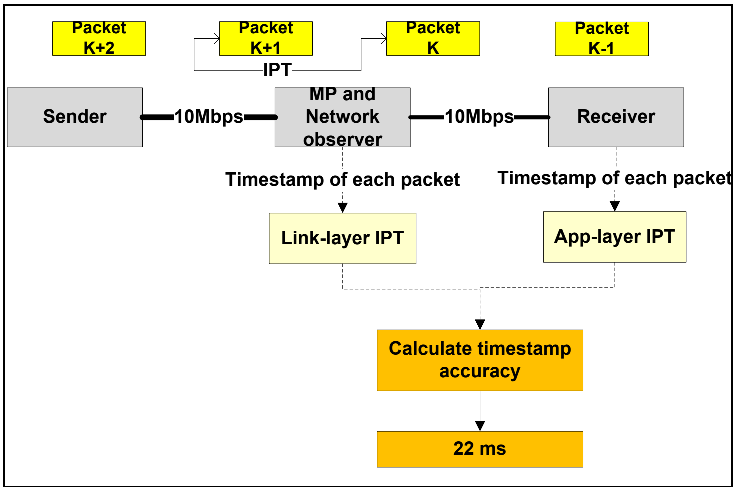


Figure 17 Accuracy of Timestamp Function

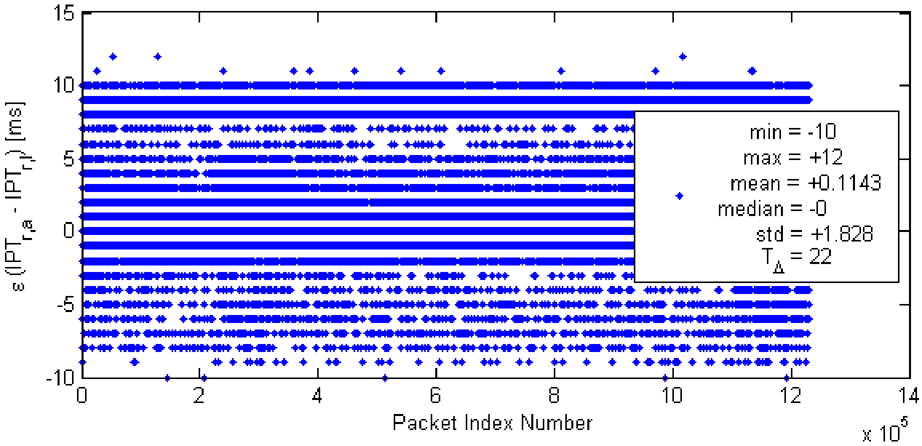


Figure 18 ε of Each IPT Pair

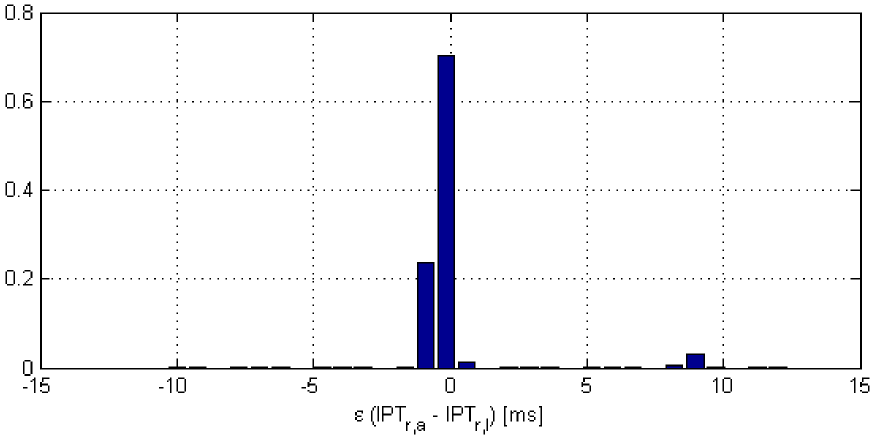


Figure 19 Distribution of ε

At this phase, experiments are completely designed. A video player for QoD quantification is evaluated. Then the designed experiments will be conducted. The experiment results are IBT, MDRB and RBF. However, these data could not completely answer RQs 3–11 since QoDD and models of relationships do not exist. In the next chapter, we are going to discuss post processing which generates QoDD and models using IBT, MDRB and RBF.

5 Post Processing

In this chapter, we discuss about post processing which is used to develop QoDD and models of relationships. The post processing is a crucial operation in this research since the data which are derived from the experiments cannot yet answer the RQs 3–11. Therefore, the post processing is required for further analysis on the derived data so that the RQs can be properly answered.

Briefly, only timestamps of video playback status are gain during the experiments. The timestamps are then computed for metrics, which are IBT, MDRB and RBF, inside the video player. After that, the metrics are stored into a database for post processing. Later, the post processing calculates QoDD from the stored metrics, and all the aforementioned metrics are models to establish the relationships between influential factors and QoD. By doing so, the RQs 3–11 are completely answered. Furthermore, plots of the relationship models are also plotted at the end of post processing to provide easier visual analysis on experiment results. The Figure 20 illustrates the previously mentioned procedures.

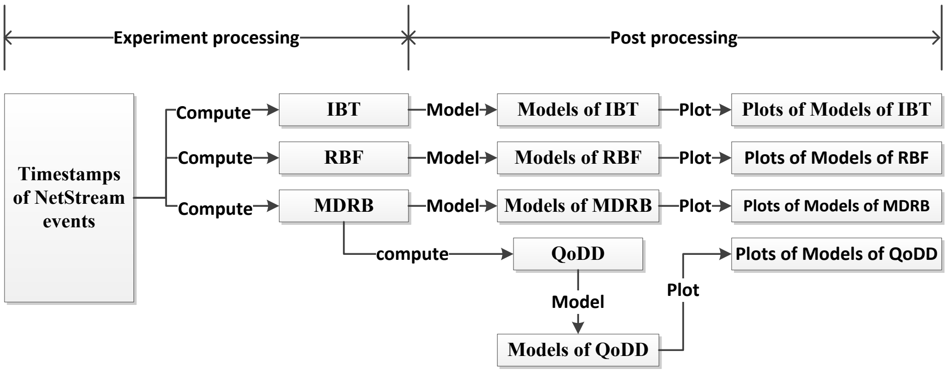


Figure 20 Post Processing

The information in this chapter begins with data modeling to explain how the relationships between influential factors and QoD are revealed using numerical methods. Together with data modeling, plotting of models are explained and verification of models are discussed at the end.

5.1 Data Modeling

When there is one predictor, 2D plotting is used. Sometimes, there are two predictors. Therefore, we generate 3D scatter plots. However, it is hard to analyze the patterns or trends in the scatter plots. Fortunately, the visual information can be greatly enhanced by computing and plotting smoothed response surface [69]. Therefore, modeling the relationships in the data is done first which is then followed by analyzing the model and the surface instead of analyzing the original scatter plots directly.

There are many modeling methods separated into two categories, parametric regression analysis [70] and non-parametric regression analysis [71]. The data-driven non-parametric regression analysis, for example, Artificial Neural Network (ANN) modeling [72] could provide more flexibility [73]. But the relationships given by them could not be known and are unspecified, and are not convenient to be calculated [71, 73]. On the other hand, the parametric regression continues to be one of the most useful tools in the quantitative analysis phase [72]. In this paper, the parametric regression is used so we and any other researchers could get the models and analyze them more easily in a quantitative way.

5.2 Parametric Regression Analysis

There are two kinds of parametric regression analysis. One is linear regression, the other one is non-linear regression. When the true relationships are unknown and the finality is to adequately represent the response in the experimental domain, making use of general linear regression is more suitable for the project [74].

In statistics, linear regression models often take the form of Eq. 11 [70, 75, 76, 74, 77, 78]. The response y is modeled as a linear combination of (not necessarily linear) functions of the predictors, plus a random error ε. The $\beta_{\mathrm{j}}(j = 1, \dots, p)$ are the coefficients. Error ε is assumed to be uncorrelated and distributed with mean 0 and constant (but unknown) variance.

$$y = \beta_{00} + (\beta_{10}x_1^{\,1} + \beta_{11}x_2^{\,1}) + (\beta_{20}x_1^{\,2} + \beta_{21}x_1^{\,1}x_2^{\,1} + \beta_{22}x_2^{\,2}) + \cdots \sum_{j=0}^{n} \beta_{n\,j}x_1^{\,n-j}x_2^{\,j}) + \varepsilon \qquad \text{Eq. 11}$$

$$(n = 0,1,2,\ldots)$$

This polynomial model is selected under linear regression considering our project's situation and the previous discussion. There are two reasons for selecting this model. Firstly, it could generally represent complicated relationship [79]. Polynomials are widely used in situations where the response is curvilinear, as even complex nonlinear relationships can be adequately modeled by polynomials over reasonably small ranges of the x's. Secondly, it is mathematically simple. The polynomial model is in fact Taylor Series Expansion. It is easy to alter the order (degree) of one model [80].

When considering the order, on one hand, the lack of fit is not serious enough to alter a model, especially when the model is used for explanation instead of prediction [81]. On the other hand, the plot of a good regression model should have a random style [82]. Otherwise, there is high possibility that the assumptions of regression analysis are violated [83]. Ideally, the model residuals are randomly independently distributed. That is, the residuals should have no particular trend / pattern / autocorrelation [84, 85]. But detecting such an autocorrelation is difficult; thus the Durbin-Watson statistic (DW) is generally used [86]. There are two statistic values derived from DW: pDW and dw. The higher the pDW, the less autocorrelation obtained [78, 87]. So the best regression analysis result has "the smallest pDW value" without "0" correlation coefficient in the regression analysis result, as the 0 coefficient is meaningless [88].

5.3 Implementation and Demonstration

As the calculation is time-consuming, Matlab is used to get the regression analysis results instead of manual computation. The used function is named "$regress$" which implements the least-square method to perform the regression. Then the response surface is plotted by using the function "*surfc*".

In practice, the "$dwtest$"function in Matlab is being used to process DW statistic: "$[pDW, dw] = \text{dwtest}(res, X)$". Where "$res$" is the array of the residuals, "X" is the design matrix of the empirical data [78, 87].

For example, Figure 21 is the experimental data, it seems that there are two groups of data, but it is not easy to see the patterns clearly.

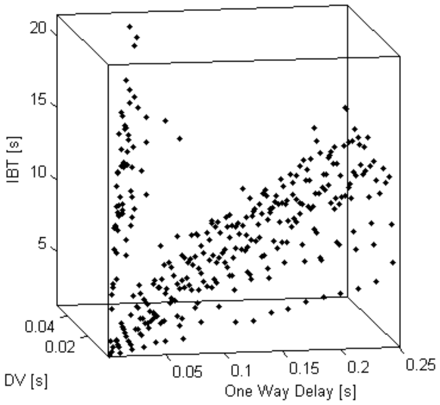
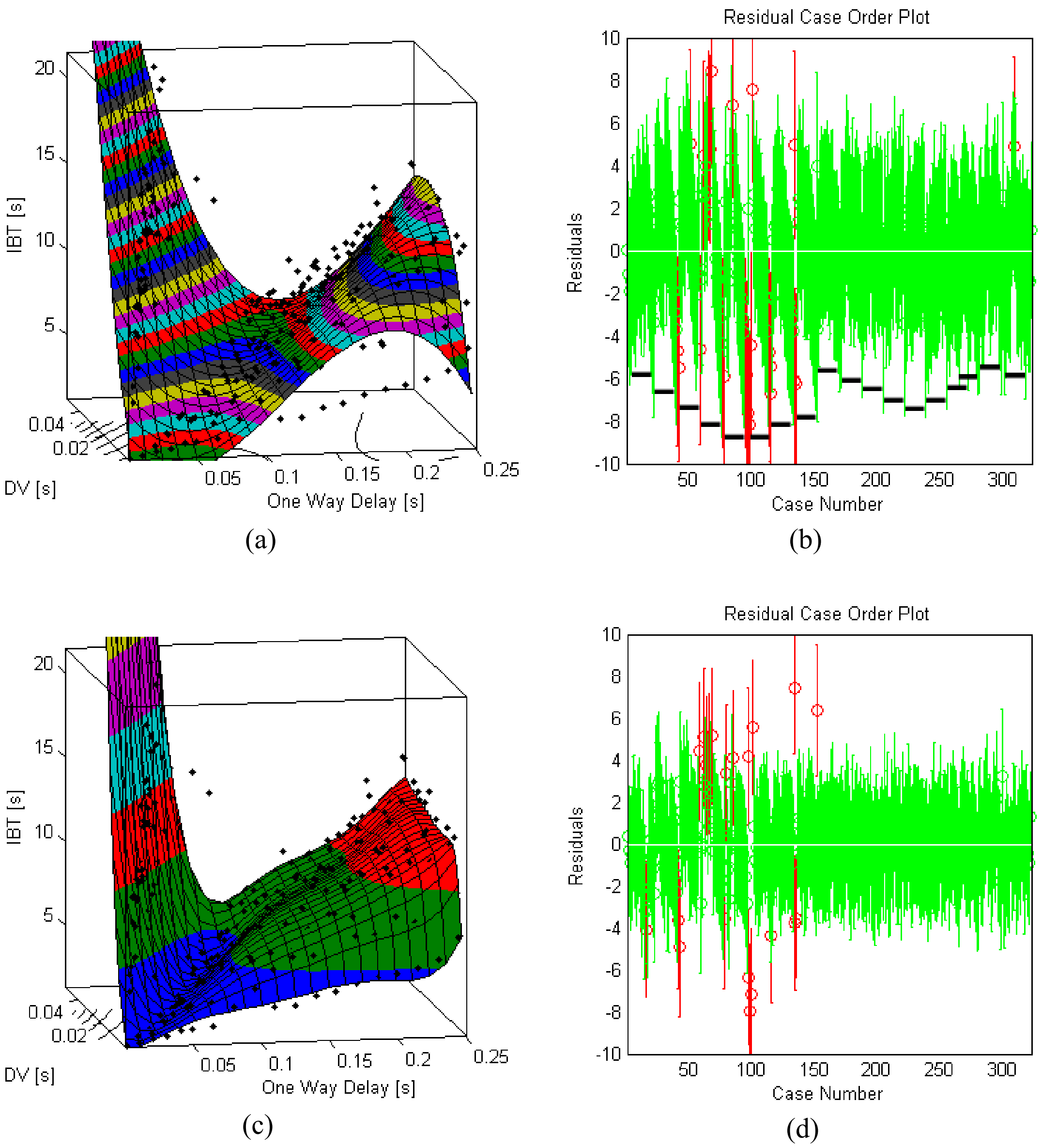


Figure 21 Original Experimental Data

Figure 22a shows the response surface calculated by regression analysis having the order of 3, where the $pDW = 2.31 \times 10^{-25}$. The Corresponding residual case plot is shown in Figure 22b. There are obvious repetition patterns / similarities (marked by black bars) in the plot. The values have a period around 20.

When the order increases to 6, the regressed response surface becomes Figure 22c, where $pDW = 0.000338$. The new residual case plot (Figure 22d) does not show clear patterns.



(a) (b)

(c) (d)

Figure 22 Demonstration of Regression

6 Results, Analysis and Verification

In this chapter, results of experiments are presented as models of relationships between influential factors and QoD. Besides, analyses of the results are also developed under the relationship models. The presented results are complete answers to RQs 3–11. The summary of experiment results can be found in Table 9 in Appendix B.

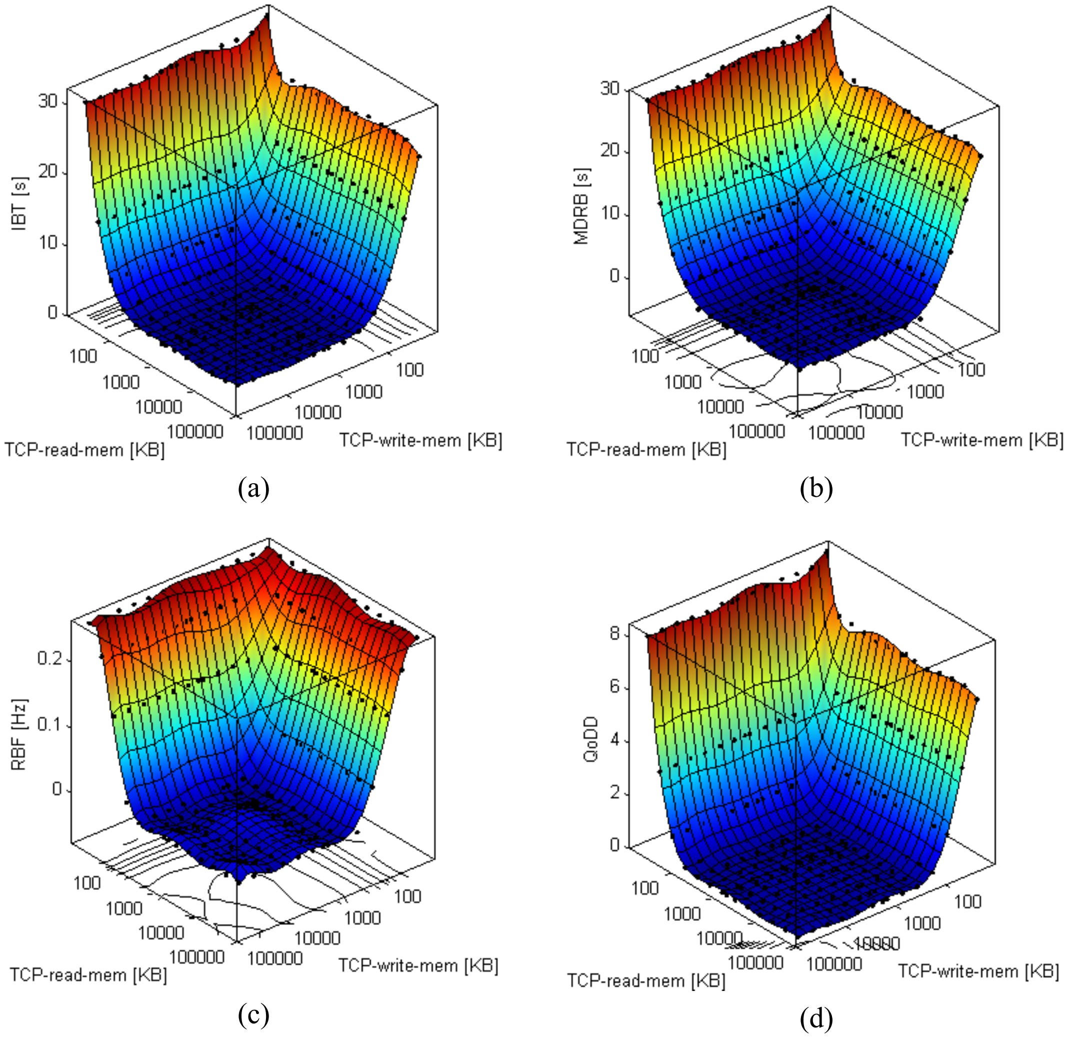
6.1 Experiment 1: TCP Memory Size

Figure 23a-c show the data of IBT, RBF and MDRB, when TCP memory size is changed between 16KB-64MB. The goodness of fit is measured by R^2, coefficient of determination, which is 0.9988 for the model of IBT, 0.9914 for the model of RBF, 0.9976 for the model of MDRB, and 0.9987 for the model of QoDD. The equations which express the relationships can be found in Appendix A. Note that one black point means a single test for each configuration, that is, no repetition in this experiment.

The models in Figure 23a-d reveal that TCP memory influences QoD since levels of IBT, MDRB, RBF and QoDD change when there are changes of TCP memory size as described in the following sentence. Reducing TCP memory size below 250 KB sharply increases value of all metrics while increasing TCP memory size over 250 KB slightly decreases value of all metrics. The rising value of all metrics corresponds to the suggestion found in [45], which states that a TCP socket buffer size which is smaller than Bandwidth Delay Product (BDP) limits maximum throughput of connection. The limited throughput consequently slows down buffering process which produces higher IBT, MDRB, RBF and QoDD. However, the research cannot explain why a TCP memory size, larger than BDP, offers lower IBT, MDRB, RBF and QoDD than the level which a TCP memory size at BDP can provide. One possible explanation is that actual TCP socket buffer size is smaller than TCP memory size as described in Linux manual [47]. This explanation however needs further investigations since the actual socket buffer size is not logged during the experiment.

Besides, the confirmation of influence, Figure 23a, b, and d exhibits that a size of TCP-write-memory affects IBT, MDRB and QoDD differently from the same size of TCP-read-memory can do so. This is because in the models of the Figure 23a, b, and d, there is a slope on the left edge which is steeper than the one on the right edge. Furthermore, Figure 23a-d reveal that the lowest level of IBT, MDRB, RBF and QoDD can be achieved only when size of TCP-write memory and TCP-read memory are augmented at the same time. Increasing only one type of memories is not sufficient to gain the lowest level of them.

The results of the experiment suggest that managing TCP memory size is a way to improve QoD. In order to achieve the lowest QoDD, TCP-write-memory size and TCP-read-memory size should be set to be larger than BDP. The results of this experiment reveals that setting TCP-write-memory size to 1MB together with setting TCP-read-memory size to 1MB yields the lowest QoDD at 0.05 while setting the memory size to BDP yields QoDD at 0.07. Increasing TCP memory size can be implemented in Linux kernel 2.6 by augmenting default and maximum values of *tcp_wmem* and *tcp_rmem*, to force operating system to allocate memory size which is larger than BDP [47, 67]. The values should be set as high as possible in order to handle high BDP connection. In addition, minimum value of *tcp_wmem* and *tcp_rmem* should be judged carefully in order to retain QoD at a high level in case operating system operates on memory pressure mode.



(a)

(b)

(c)

(d)

Figure 23 TCP Memory Size

Though the lowest QoDD can be gained only when the sizes of TCP-write-memory and TCP-read-memory are increased at the same time, however, access to client is usually limited; hence adjusting TCP-read-memory size is almost impossible. Figure 24 shows QoDD in another way, for example, the red one is fixing TCP-read-memory at BDP while changing TCP-write-memory from 64 KB to 64 MB. The result shows that increasing only TCP-write-memory size over BDP still reduces QoDD at almost the same level as increasing sizes of the both memories.

Apart from sizing TCP memory, the result of the experiment hints another clue that auto tuning TCP memory size on modern OSes is an important technique to retain high QoD as they can adjust TCP memory size according to a connection's BDP, and proactively prevent QoD from being trapped in a low level [46]. Examples of OSes with auto tuning functions are Windows Vista, Linux kernel 2.6 and OSX 10.5. Therefore, these modern OSes are likely to offer higher QoD than the level their preceding versions provide.

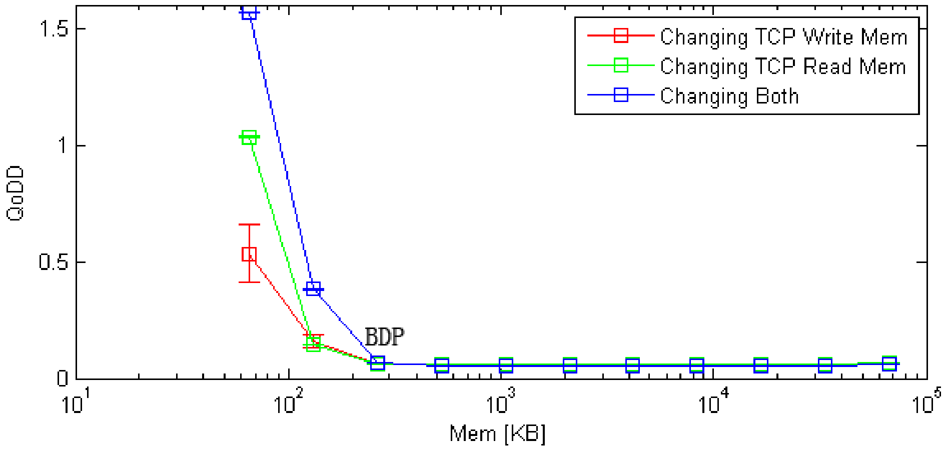


Figure 24 TCP Memory Size and BDP

6.2 Experiment 2: TCP Congestion Control Algorithms

6.2.1 Subject to Different Levels of Loss Rate

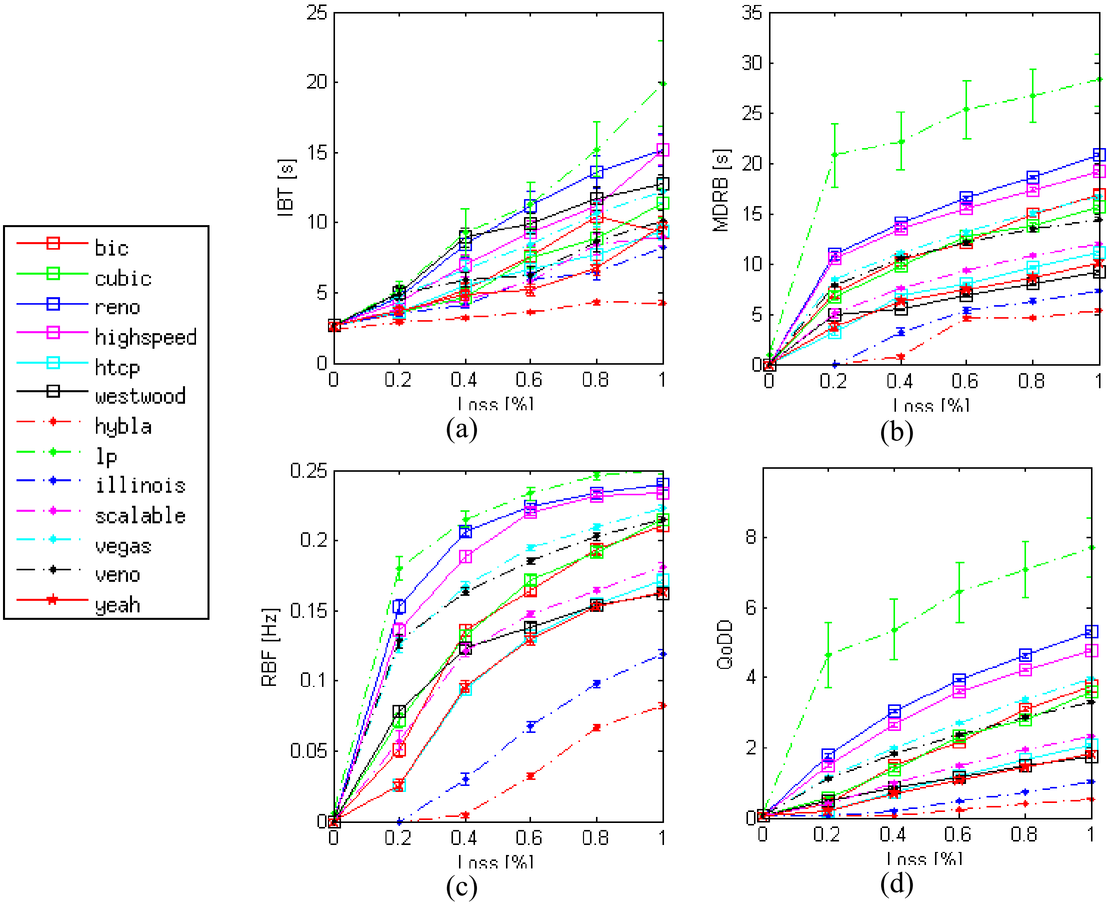
Figure 25a-d reveal that CCA is another influential factor which influences QoD. This is because a certain algorithm shows different IBT, MDRB, RBF and QoDD from the other CCAs, and the difference is more noticeable as the loss rate increases.

To go into details, Figure 25a shows that, at loss rate of 0%, most of TCP CCAs provide IBT at 2.6 seconds with the exception of Hybla which provides IBT at 2.3 seconds. The evidences also show that almost all TCP CCAs develop higher IBT, RBF, MDRB and QoDD when the loss rate augments, except BIC which swings IBT at the loss rate between 0.8% and 1%. This suggests that BICu and Yeah do not provide linear IBT degradation as the loss rate increases.

Figure 25a-d reveal that Hybla always provides the lowest IBT, RBF, MDRB and QoDD during the whole experiment. At the loss rate of 1%, Hybla provides 4.29 seconds of IBT, 0.08 Hz of RBF and 5.39 seconds of MDRB which are equal to 62%, 61% and 65% of relative improvement when being compared with CUBIC which provides 11.36 seconds of IBT, 0.21Hz of RBF and 15.62 seconds of MDRB. Hybla also provides the smallest variation as it has the smallest error bars when being compared to other TCP CCAs. These evidences suggest that Hybla provides the highest QoD and the highest reliability among other TCP CCAs. Note that in contrast with Hybla, LP always provides the lowest QoD during the whole experiment. This is probably because LP is designed for low priority data transfer [89].

The result of the experiment suggests that upgrading CCA greatly improves QoD. For example, the result shows that upgrading from Reno to Hybla reduces QoDD by 4 at loss rate of 1%. Furthermore, the result hints that various operating systems will offer different

QoD since they support only certain algorithms. For example, Linux supports Hybla while Windows does not support. Apart from that, the results of the experiment suggests that applying Hybla on server will offer better QoD than the level that other algorithms can provide on lossy link. Such lossy link is typically found in wireless, mobile, and satellite networks.



(a) (b)

(c) (d)

Figure 25 TCP CCA (Different Loss Rates)

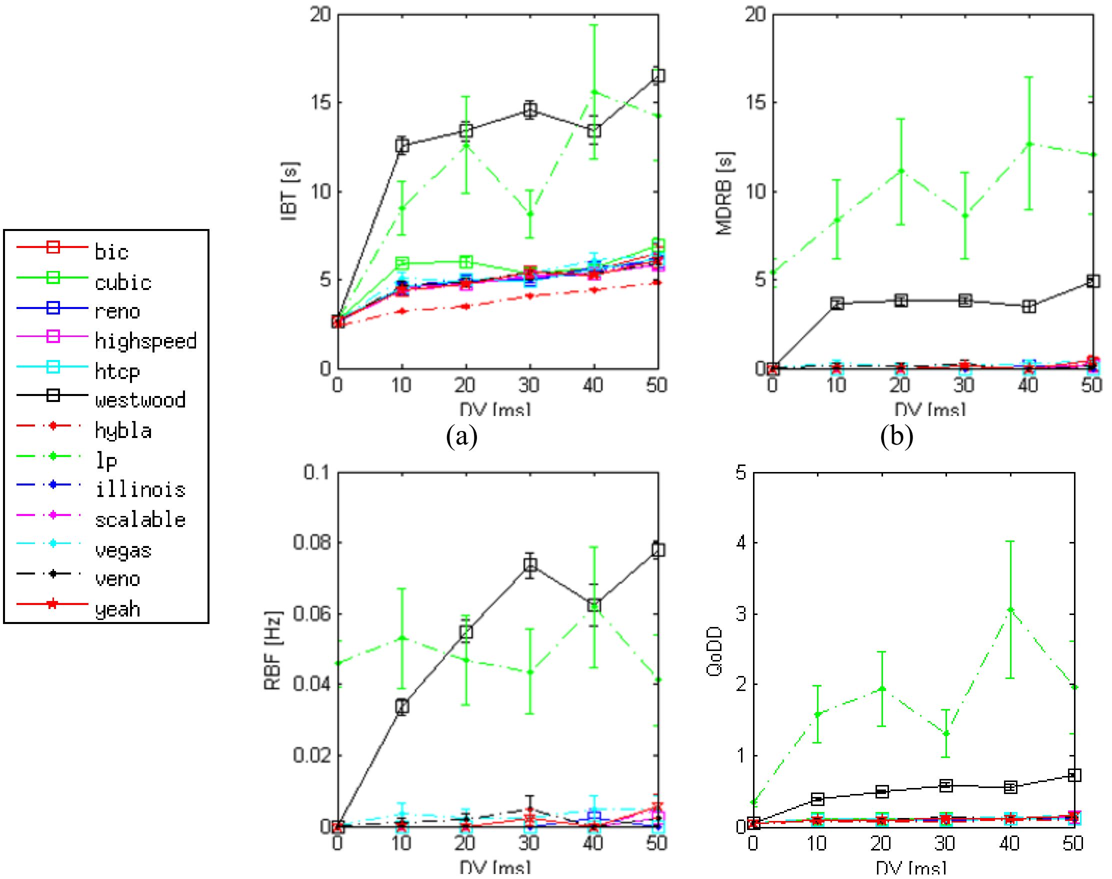
6.2.2 Subject to Different Levels of DV

Figure 26a-d shows that, under a network with DV, most of TCP CCAs produce quite the same level of IBT, MDRB, RBF and QoDD, except Westwood which produces distinctly high level of IBT, MDRB, RBF and QoDD when being compared with other algorithms. The reason behind the high QoDD of Westwood is probably because Westwood detects network congestion from frequency of ACK [55]. Such frequency is biased due to influence of DV. Besides that, some TCP CCAs do not gradually develop higher IBT, MDRB, RBF and QoDD when DV augments, except Reno, HTCP, Hybla, Illinois and Veno.

Apart from the previous analysis, Figure 26a also reveals that Hybla always gains the lowest IBT and the smallest variation of IBT when being compared with other algorithms. For example, at 50ms of DV, Hybla produces 4.8 seconds of IBT while CUBIC produces 7.3 seconds of IBT. This suggests that Hybla shows 34% of relative difference of IBT when being compared with CUBIC. In addition, Figure 26b-c expose that Westwood, HTCP, Scalable, LP, and YEAH cause unanticipated re-buffering events as they develop higher

RBF and MDRB while other TCP CCAs do not cause any re-buffering event, and maintain RBF and MDRB at level of zero.

The result of experiment suggests that using Westwood should be avoided in network with high DV for the three reasons. The first reason is that Westwood produces non-linear IBT degradation as DV augments. The second reason is that Westwood generates unanticipated re-buffering event. The third reason is that Westwood produces the highest QoD when coping with DV.



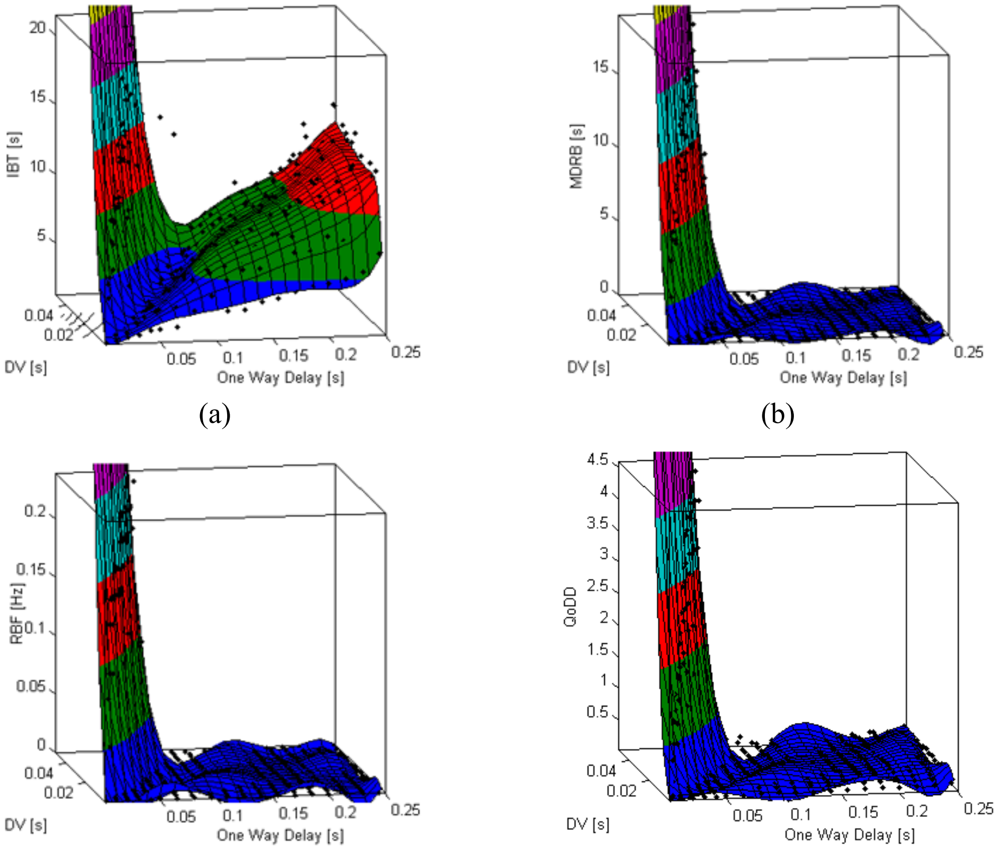
(a) (b)

(c) (d)

Figure 26 TCP CCA (Different DVs)

6.3 Experiment 3: Delay Variation

Figure 27a-c show the data of IBT, RBF and MDRB, of which DV is changed from 1-49ms, and of which (one-way) delay is changed from 1-245ms. The goodness of fit is measured by R^2, coefficient of determination, which is 0.826 for the model of IBT, 0.878 for the model of RBF, 0.87 for the model of MDRB, and 0.863 for the model of QoDD. The equations which express the relationships can be found in Appendix A. Note that one black point means a single test for each configuration, that is, no repetition in this experiment.



(a) (b)

(c) (d)

Figure 27 DV (Different One-Way Delays)

Figure 27a-d reveal that high IBT, RBF, MDRB and QoDD usually occur in a certain area where proportional delay and DV maintain in a specific ratio. For instance, at 40ms of DV and 245ms of delay, IBT is at 13 seconds. But when the delay is increased to 110ms, IBT is at 2.8 second. To enhance the analysis, Figure 28a-d are developed to represent plots with Coefficient of Variation (CV), defined as a ratio of DV to delay.

Figure 28a-d show that CV which is higher than 0.6 produces extremely high MDRB, RBF and QoDD while lower CV maintains MDRB, RBF and QoDD around the level of zero. The research in [51] gives one possible explanation that high DV causes spurious TCP timeout which degrades TCP throughput. Our research then suggests that increasing delay will extend the TCP timeout period and thus amends such TCP throughput degradation.

In addition to aforementioned explanation, the research in [90] also suggests another explanation describing that high DV leads to spurious fast retransmission since high DV causes out-of-order arriving packets. These out-of-order arriving packets may cause three duplicate ACKs which triggers TCP fast recovery process. However, this explanation cannot advise why increasing network delay lowers QoDD. Therefore, further investigations are required in order to explain such phenomena.

The result of the experiment suggests that CV which is larger than 0.6 causes an adverse effect on QoDD, and thus should be avoided. In order to solve this problem, injection of extra delay to network can be a solution to mitigate such degradation as being exhibited in [51]. Furthermore, upgrading CCA to make use of TCP Eifel [91] or F-RTO [92] can be options to reduce QoDD, too.

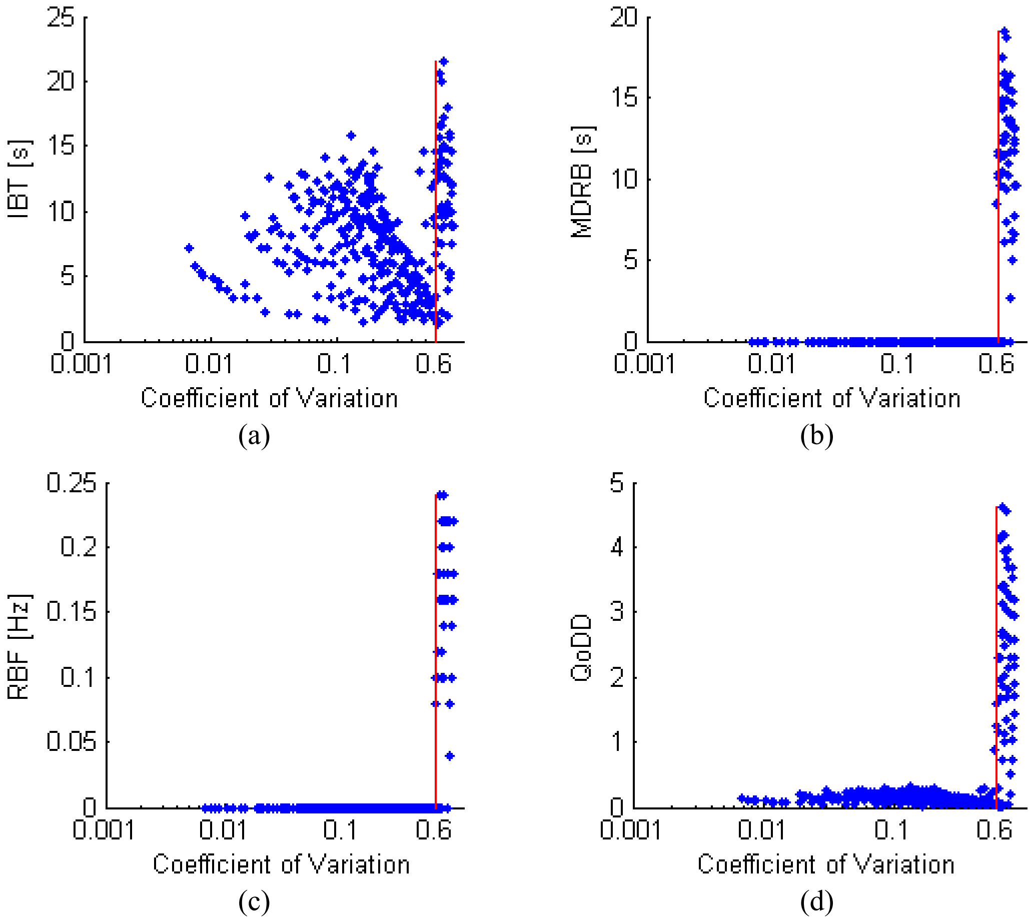


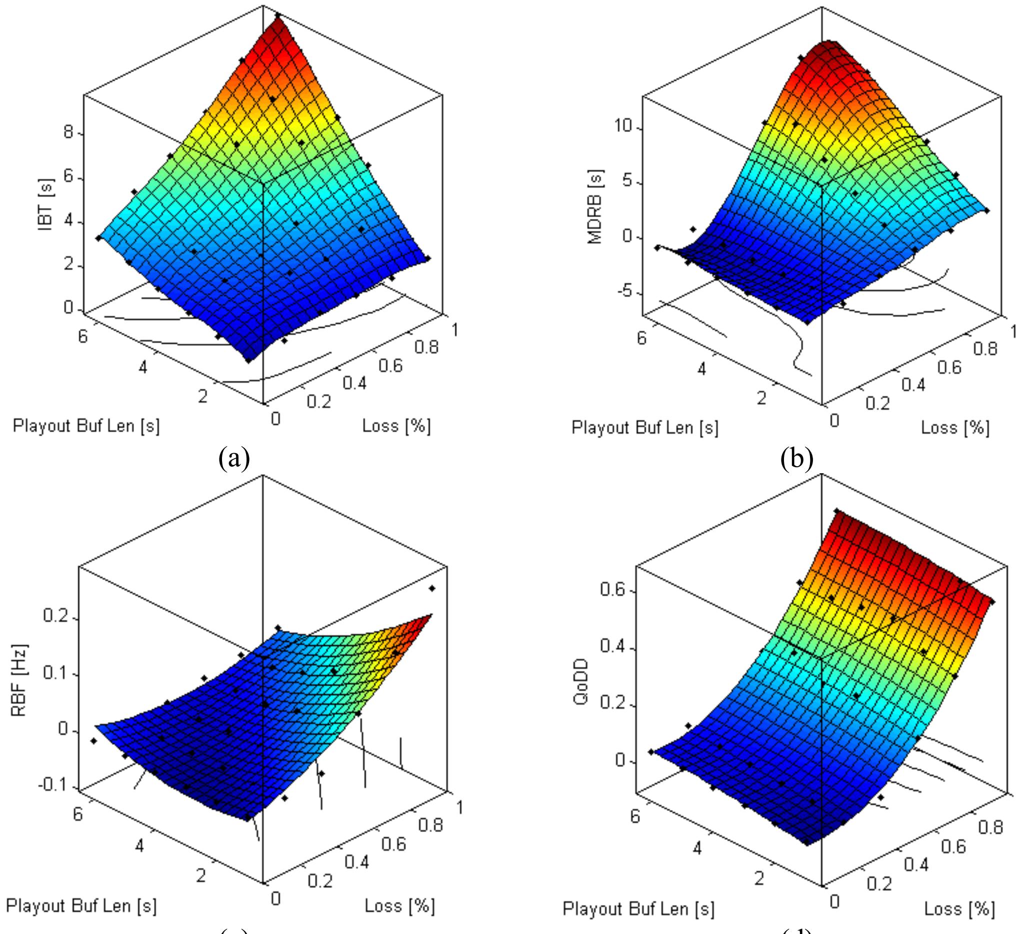
Figure 28 CV

6.4 Experiment 4: Playout Buffer Length

6.4.1 Subject to Different Levels of Loss Rate

Figure 29a-c show the data of IBT, RBF and MDRB, of which playout buffer length size is changed between 1 and 6 seconds, and of which loss rate is changed between 0 and 1%. The goodness of fit is measured by R^2, coefficient of determination, which is 0.992 for the model of IBT, 0.979 for the model of RBF, 0.969 for the model of MDRB, and 0.994 for the model of QoDD. The equations which express the relationships can be found in Appendix A.

Figure 29a-d demonstrate that at the network loss between 0 and 1%, increasing playout buffer length produces lower RBF but higher IBT, MDRB and QoDD. On the other hand, decreasing playout buffer length produces higher RBF while lower IBT, MDRB and QoDD. For example, at the loss of 1%, setting playout buffer length at 1 second produces 0.24361 Hz of RBF, 2.062 seconds of MDRB and 0.51 of QoDD. Once increase playout buffer length from 1 second to 6 second, RBF is reduced to 0.0379 Hz but MDRB is increased to 10.412 seconds and QoDD is increased to 0.591. In addition, increasing or decreasing the playout buffer length offers a trade between RBF with IBT, MDRB and QoDD.



(a) (b)

(c) (d)

Figure 29 Playout Buffer Length (Different Loss Rates)

The result of the experiment suggests that increasing playout buffer length indeed slightly produces higher QoDD at any loss rate. Therefore, the playout buffer length should be short as much as possible so that QoDD maintains in a low level. Despite the fact that short playout buffer length is desired, the playout buffer length should long enough to keep IBT, MDRB and RBF in a certain ratio since the research in [23] finds that proportional IBT, MDRB and RBF also has an impact on QoE in addition to QoD. In case a long playout buffer length may be desired, dual-threshold buffering technique [93] could be an interesting solution for maintaining IBT at a low level since the technique offers dynamic playout buffer length.

6.4.2 Subject to Different Levels of DV

Figure 30a-c show the data of IBT, RBF and MDRB, of which playout buffer length size is changed between 1 and 6 seconds, and of which DV is changed between 0-10ms. The goodness of fit is measured by R^2, coefficient of determination, which is 0.996 for the model of IBT, and 0.996 for the model of QoDD. The equations which express the relationships can be found in Appendix A.

Figure 30a-d show that, at the DV between 0–10ms, QoDD solely depends on IBT since there is no re-buffering event occurring, and increasing playout buffer length slightly produces higher QoDD. The result of the experiment suggests that, in the network with the DV between 0–10ms, increasing playout buffer length only produces higher QoDD.

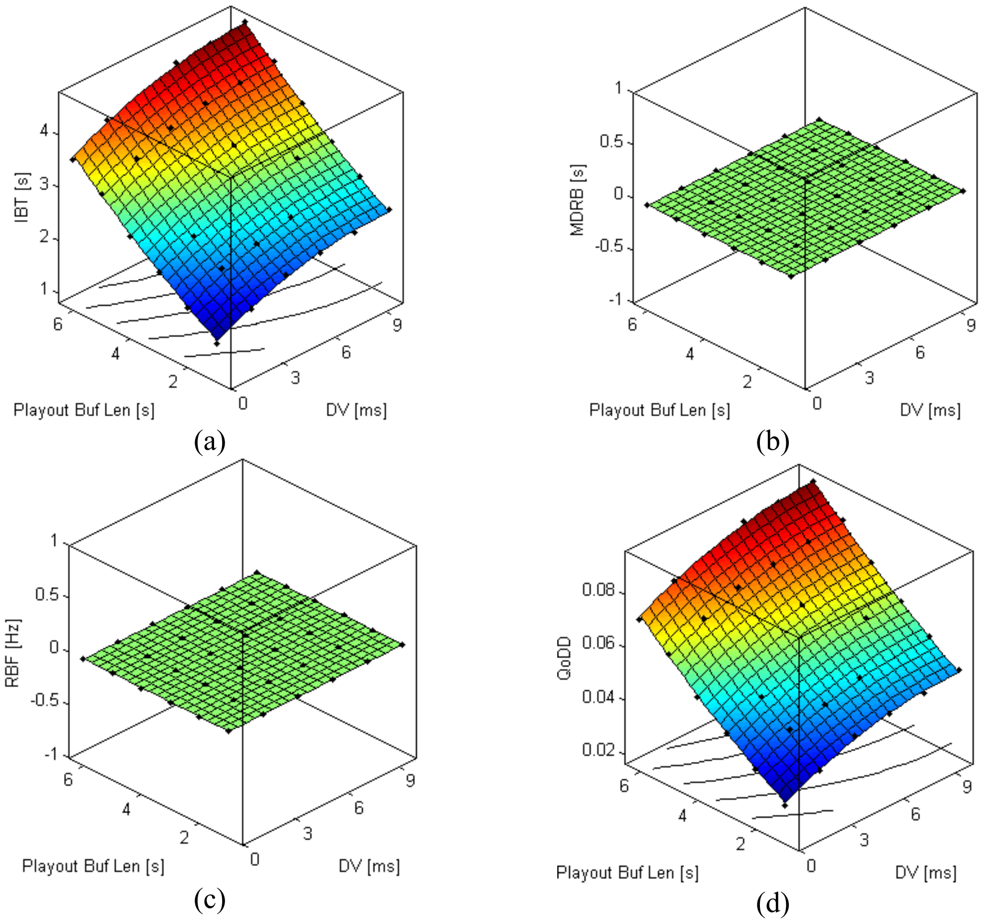


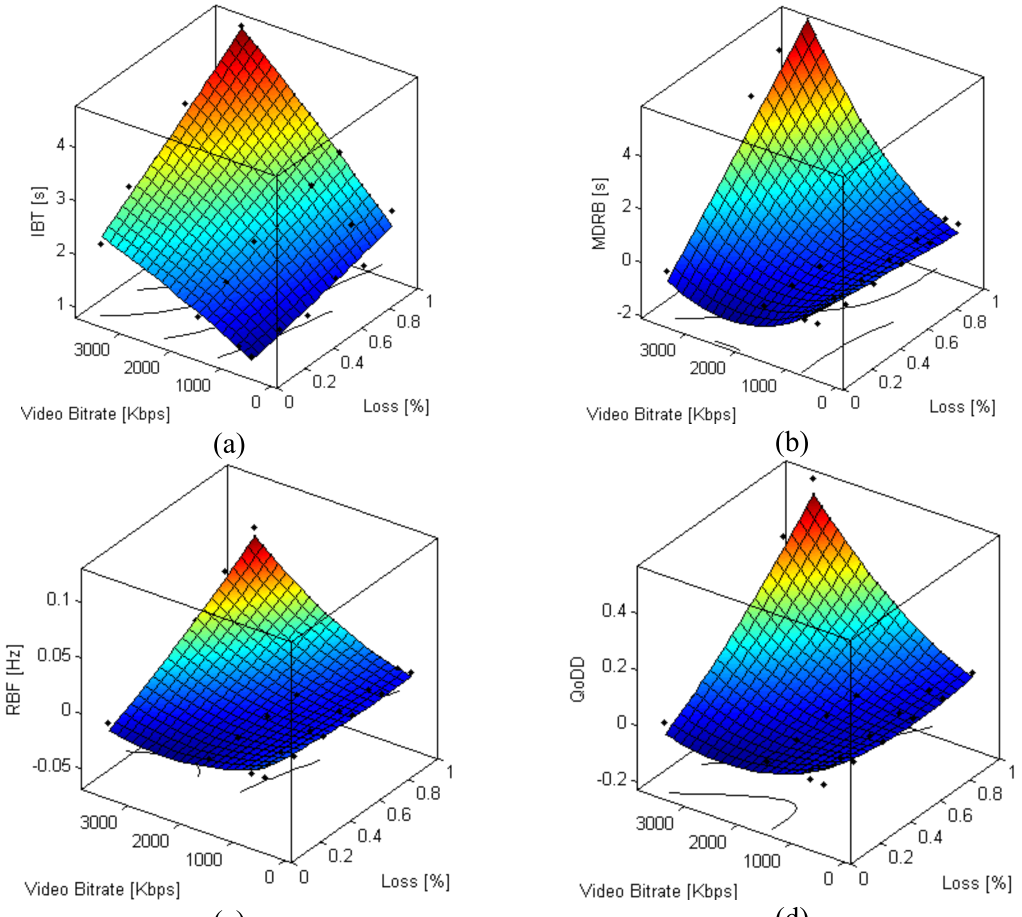
Figure 30 Playout Buffer Length (Different DVs)

6.5 Experiment 5: Video Bitrate

6.5.1 Subject to Different Levels of Loss Rate

Figure 31a-c show the data of IBT, RBF and MDRB, of which video bitrate is changed between 378–3364Kbps, and of which loss rate is changed between 0 and 1%. The goodness of fit is measured by R^2, coefficient of determination, which is 0.978 for the model of IBT, 0.924 for the model of RBF, 0.911 for the model of MDRB, and 0.919 for the model of QoDD. The equations which express the relationships can be found in Appendix A.

Figure 31a-d demonstrate that reducing video bitrate drastically decreases IBT, MDRB, RBF and QoDD especially when loss rate is at a high level. The result of the experiment suggests that a high video bitrate produces higher QoDD than a low video bitrate. In order to maintain QoDD at a low level, the video bitrate should be low as much as possible. Reducing video bitrate can be implemented by various techniques. One common technique is lowering video resolution. However, lowering video resolution often degrades picture quality which causes an impact on QoE beside QoD .Another possible solution is upgrading video codec since modern video codec can provides a lower bitrate with the same or better picture quality. Despite the fact that H.264 is currently one of the most widely accepted codec today, the research in [94] finds that H.264 requires a higher bitrate than the level need by VP6 for almost the same video quality. This may hint that upgrading from VP6 to H.264 may not help reduce QoDD.



(a) (b)

(c) (d)

Figure 31 Video Bitrate (Different Loss Rates)

6.5.2 Subject to Different Levels of DV

Figure 32a-d show the data of IBT, RBF and MDRB, of which video bitrate is changed between 378-3364Kbps, and of which loss rate is changed between 0 and 1%. The goodness of fit is measured by R^2, coefficient of determination, which is 0.983 for the model of IBT, and 0.983 for the model of QoDD. The equations which express the relationships can be found in Appendix A.

Figure 32a-d illustrate that, at the DV between 0–10ms, QoDD solely depends on IBT since there is no re-buffering event occurring, and reducing video bitrate slightly decreases QoDD. The result of the experiment suggests that with the DV between 0–10ms, reducing video bitrate would decrease QoDD.

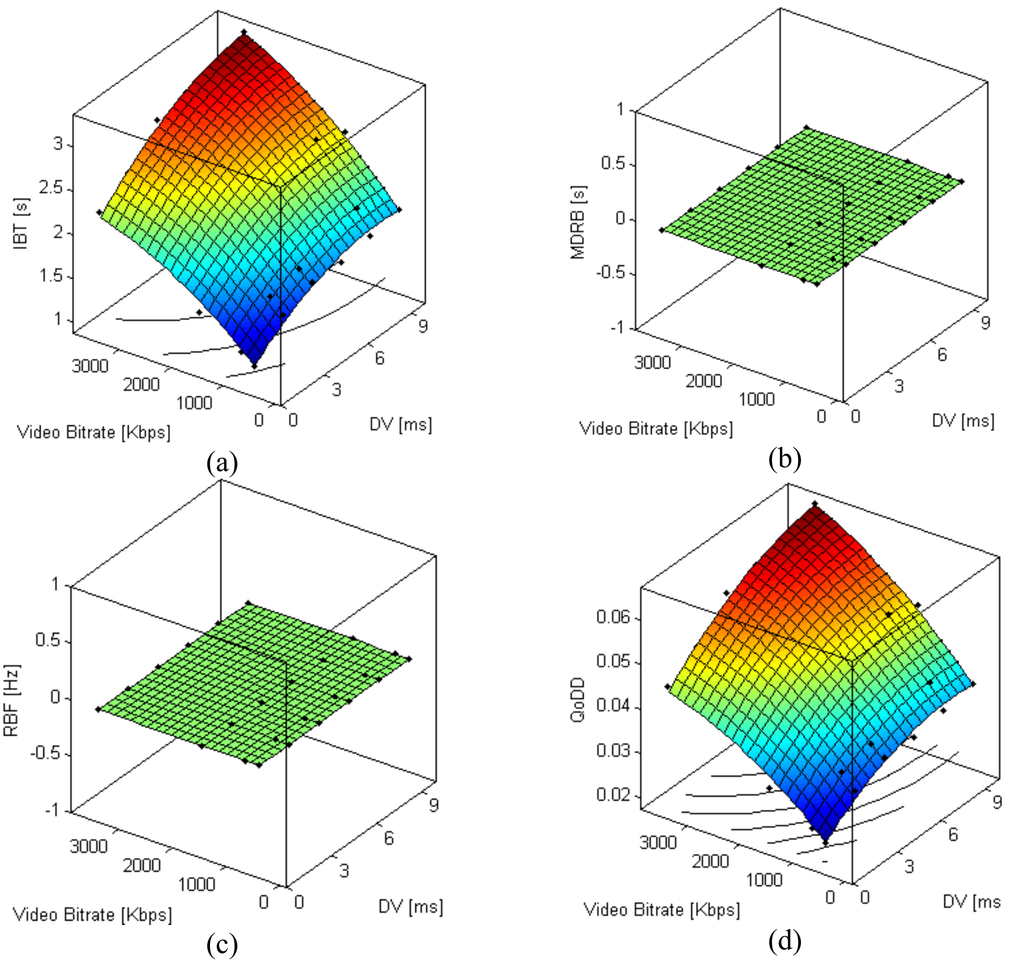


Figure 32 Video Bitrate (Different DVs)

6.6 Verification of Result Accuracy

Based on the timestamp accuracy found in the previous chapter, errors of IBT, MDRB and QoDD in all the experiments are computed to verify reliability of the experiment results. From the fact that measuring IBT and MDRB involve with reading timestamp for two times, and each timestamp comprise 22ms of maximum possible error. Each measurement therefore contains 44ms of maximum possible error in its value. The minimum IBT in this project is 1212ms which gives the maximum possible relative error of IBT as shown in Eq. 12:

$$\frac{44}{1212} \times 100\% = 3.63\% \qquad \text{Eq. 12}$$

For RBF, we keep tracking the signals in the player and ensure no occurrence of wrong count.

For MDRB, no matter how many rebuffering events in one single experiment, the accumulative maximum possible error will be divided by the count of rebuffering events. So the accuracy of MDRB is also 44ms which gives the maximum possible relative error of MDRB as shown in Eq. 13:

$$\frac{44}{575} \times 100\% = 7.65\% \qquad \text{Eq. 13}$$

The calculation of the maximum possible relative error of QoDD is shown below:

$$\begin{aligned}
\max\left(\frac{QoDD_1 - QoDD}{QoDD}\right)\times 100\% &= \max\left(\frac{\frac{X_1}{L}-\frac{X}{L}}{\frac{X}{L}}\right)\times 100\% \\
&= \max\left(\frac{X_1 - X}{X}\right)\times 100\% \\
&= 2\times 22\times \max\left(\frac{1+rebuf\_count}{IBT + rebuf\_count\times MDRB}\right)\times 100\%
\end{aligned} \qquad \text{Eq. 14}$$

Where*:*
QoDD: the measured QoDD value;
QoDD$_1$: QoDD with max possible error.

We apply this calculation for each single test and get the value as 4.40%.

Table 7 provides the result of previous analysis.

Table 7 Max Possible Relative Error

Term	Max Possible Relative Error [%]
IBT	3.63
RBF	0
MDRB	7.65
QoDD	4.40

7 Conclusions and Future Work

Research develops the following five conclusions from the results of the experiment that were previously shown in the last chapter. First of all, a technique to quantify break artifact of HTTP video streaming is developed, and no experiment shows occurrence of break artifact. For the second conclusion, QoDD is proposed as a metric to quantify QoD. The third conclusion is that DV, TCP memory size, TCP CCA, playout buffer length and video bitrate are factors which affect QoD. The fourth conclusion is there are relationships between QoD and influential factors. The last conclusion is the relationships can be modeled.

7.1 Answers to Research Questions

Below are answers to RQs. It is important to note that QoD is referred in terms of QoDD.

Question 1 How to quantify QoD?
Ans. QoD can be quantify in numerical value by using QoD degradation (QoDD)

Question 2 What are factors which influence QoD?
Ans. Influential factors which cause QoD are TCP memory size, DV, TCP CCA, playout buffer length and video bitrate.

Question 3 How is QoD affected from changing TCP memory sizes?
Ans. QoD changes when TCP memory sizes are changed as described in the following two ways. Firstly, QoD is better when TCP memory size is increased from BDP. Secondly, QoD is worse when TCP memory size is reduced from BDP.

Question 4 How is QoD affected from changing TCP congestion control algorithm?
Ans. QoD changes when TCP CCA is changed as described in the following two ways. Firstly, QoD is changed to a specific level when a certain CCA is applied. Secondly, at loss rate between 0–1% and DV between 0–50ms, The best QoD is gained by applying Hybla on server side.

Question 5 How is QoD affected from changing delay variations?
Ans. QoD is worse when delay variation is increased from 0 to 50ms.

Question 6 How is QoD affected from changing playout buffer lengths?

Ans. QoD is worse when playout buffer length is increased from 1 to 6 seconds.

Question 7 How is QoD affected from changing video bitrates?

Ans. QoD is worse when video bitrate is increased from 378Kbps to 3364 Kbps.

Question 8 Is there a relationship between TCP memory sizes and QoD? And can we model it, if any?

Ans. There is a relationship between TCP memory sizes and QoD. The equation which expresses such relationship can be found in Appendix A. The model which represents such relationship can be found in Figure 23d (page 36).

Question 9 Is there a relationship between delay variation and QoD? And can we model it, if any?

Ans. There is a relationship between DV and QoD. The equation which expresses such relationship can be found in Appendix A. The model which represents such relationship can be found in Figure 28d (page 41).

Question 10 Is there a relationship between playout buffer length and QoD? And can we model it, if any?

Ans. There is a relationship between playout buffer length and QoD. The equation which expresses such relationship can be found in Appendix A. The model which represents such relationship can be found in Figure 29d (page 42) and Figure 30d (page 43).

Question 11 Is there a relationship between video bitrate and QoD? And can we model it, if any?

Ans. There is a relationship between playout buffer length and QoD. The equation which expresses such relationship can be found in Appendix A. The model which represents such relationship can be found in Figure 31d (page 44) and Figure 32d (page 45).

7.2 Discussion

Based on the analysis of the results, in this research, the following five suggestions to video content publishers are developed to improve QoD:

- Considering increases of TCP memory size to be larger than minimum BDP
- Considering upgrade of TCP CCA
- Maintaining CV to be lower than 0.6
- Avoiding setting excessive playout buffer length
- Minimizing video bitrate while maintaining picture quality

Table 8 shows examples of the results derived from the experiment, corresponding to the aforementioned suggestions.

However, the aforementioned suggestions are also derived with the following concerns:

- Increasing TCP memory size reduces maximum number of connections per server since OS kernel has limited direct access/mapping to physical memory. In Linux 32 bit, there is only 896 MB physical memory available for OS kernel [95, 96]
- Upgrading TCP CCA may bring up other problems such as RTT-fairness, resource consumption and OS compatibility
- Shortening playout buffer length causes more re-buffering events
- Lowering video bitrate usually causes lower picture quality

Table 8 Suggestions and Improvement

Case	Description	Relative Improvement [%]			
		IBT	MDRB	RBF	QoDD
1	Increasing TCP memory to be 4 times as large as BDP	23.5	0	0	23.5
2	At loss rate of 1%, upgrading CCA from CUBIC to Hybla	62.3	65.5	61.6	63.1
3	At DV of 50ms, upgrading CCA from CUBIC to Hybla	30.3	0	0	30.3
4	At DV of 40ms, increasing delay from 60 to 90ms (CV is changed from 0.83 to 0.56)	65.0	100	100	97.3
5	At loss rate of 1%, reducing playout buffer length from 3 seconds to 1 seconds	58.0	63.4	-190	4.91
6	At loss rate of 1%, reducing video bitrate from 3364 Kbps to 378 Kbps	54.4	100	100	92.2

7.3 Future Work

In addition to the proposed factors, other influential factors still remain and wait for future investigations. Examples of other influential factors which are found during research are bandwidth variation, loss pattern, TCP loss recovery technique, TCP slow-start threshold.

Aside from the extension of influential factors, some video content publishers do not employ progressive download since other new methods provide better bandwidth efficiency, for example, HTTP paced download. Therefore, these video delivery methods should be taken into consideration together with the progressive download in order to cover more realistic implementations.

Though in the research, it is found that Hybla offers the best QoD, other aspects of using Hybla for HTTP video streaming however have not been revealed yet . An example of other aspects is RTT fairness of Hybla on the Internet instead of just simulation. Therefore, further investigations of Hybla on HTTP video streaming may be required.

Appendix

A. QoD Relationships

$$\begin{aligned}
&\mathit{MDRB}[\mathrm{s}] = +1.1619e+005 - 1.2673e+005x_1^0x_2^1 - 13118x_1^1x_2^0 + 60495x_1^0x_2^2 + 6466.9x_1^1x_2^1 + 4795.4x_1^2x_2^0 - 16600x_1^0x_2^3 - 366.31x_1^1x_2^2 \\
&-2570.3x_1^2x_2^1 - 788.33x_1^3x_2^0 + 2651.3x_1^0x_2^4 + 328.43x_1^1x_2^3 - 333.04x_1^2x_2^2 + 803.62x_1^3x_2^1 - 20.504x_1^4x_2^0 - 248.23x_1^0x_2^5 - 62.444x_1^1x_2^4 \\
&+43.831x_1^2x_2^3 + 6.3016x_1^3x_2^2 - 101.17x_1^4x_2^1 + 17x_1^5x_2^0 + 12.851x_1^0x_2^6 + 3.9375x_1^1x_2^5 + 0.17155x_1^2x_2^4 - 4.7756x_1^3x_2^3 + 3.1849x_1^4x_2^2 + 5.2275x_1^5x_2^1 \\
&-1.4946x_1^6x_2^0 - 0.28757x_1^0x_2^7 - 0.076491x_1^1x_2^6 - 0.082991x_1^2x_2^5 + 0.12518x_1^3x_2^4 + 0.065886x_1^4x_2^3 - 0.14405x_1^5x_2^2 - 0.090726x_1^6x_2^1 + 0.040479x_1^7x_2^0 \\
&\text{where: } x_1 = \log_{10}(\mathit{WriteMem}[\mathrm{Byte}]), x_2 = \log_{10}(\mathit{ReadMem}[\mathrm{Byte}]); 4.2144 < x_1 < 7.8268, 4.2144 < x_2 < 7.8268; \mathit{pDW} = 0.2636, R^2 = 0.9976
\end{aligned}$$

$$\begin{aligned}
&\mathit{IBT}[\mathrm{s}] = -9195.7 - 54195x_1^0x_2^1 + 66296x_1^1x_2^0 + 29701x_1^0x_2^2 - 6251.6x_1^1x_2^1 - 29878x_1^2x_2^0 - 8593.8x_1^0x_2^3 + 1884.3x_1^1x_2^2 + 354.23x_1^2x_2^1 \\
&+8178.7x_1^3x_2^0 + 1346.3x_1^0x_2^4 + 108.4x_1^1x_2^3 - 726.71x_1^2x_2^2 + 434.23x_1^3x_2^1 - 1463.7x_1^4x_2^0 - 117.93x_1^0x_2^5 - 52.523x_1^1x_2^4 + 74.84x_1^2x_2^3 \\
&+39.919x_1^3x_2^2 - 74.036x_1^4x_2^1 + 158.97x_1^5x_2^0 + 5.531x_1^0x_2^6 + 4.1213x_1^1x_2^5 - 1.6735x_1^2x_2^4 - 5.6574x_1^3x_2^3 + 1.0967x_1^4x_2^2 + 4.3878x_1^5x_2^1 \\
&-9.3528x_1^6x_2^0 - 0.11046x_1^0x_2^7 - 0.09697x_1^1x_2^6 - 0.042375x_1^2x_2^5 + 0.15556x_1^3x_2^4 + 0.071512x_1^4x_2^3 - 0.079982x_1^5x_2^2 - 0.091538x_1^6x_2^1 + 0.2288x_1^7x_2^0 \\
&\text{where: } x_1 = \log_{10}(\mathit{WriteMem}[\mathrm{Byte}]), x_2 = \log_{10}(\mathit{ReadMem}[\mathrm{Byte}]); 4.2144 < x_1 < 7.8268, 4.2144 < x_2 < 7.8268; \mathit{pDW} = 0.000947, R^2 = 0.9988
\end{aligned}$$

$$\begin{aligned}
&\mathit{RBF}[\mathrm{Hz}] = +2176.2 - 620.91x_1^0x_2^1 - 2120.3x_1^1x_2^0 + 24.893x_1^0x_2^2 + 648.3x_1^1x_2^1 + 789.77x_1^2x_2^0 + 34.384x_1^0x_2^3 - 149.92x_1^1x_2^2 - 124.24x_1^2x_2^1 \\
&-188.99x_1^3x_2^0 - 11.249x_1^0x_2^4 + 27.598x_1^1x_2^3 + 8.9393x_1^2x_2^2 + 21.661x_1^3x_2^1 + 27.599x_1^4x_2^0 + 1.5879x_1^0x_2^5 - 3.0586x_1^1x_2^4 - 0.70472x_1^2x_2^3 - 0.83886x_1^3x_2^2 \\
&-2.2305x_1^4x_2^1 - 2.4472x_1^5x_2^0 - 0.10712x_1^0x_2^6 + 0.16855x_1^1x_2^5 + 0.076501x_1^2x_2^4 - 0.021655x_1^3x_2^3 + 0.0866x_1^4x_2^2 + 0.1091x_1^5x_2^1 + 0.12432x_1^6x_2^0 \\
&+0.0028189x_1^0x_2^7 - 0.0035641x_1^1x_2^6 - 0.0029165x_1^2x_2^5 + 0.00066728x_1^3x_2^4 + 0.00016159x_1^4x_2^3 - 0.0029206x_1^5x_2^2 - 0.0019299x_1^6x_2^1 - 0.0028175x_1^7x_2^0 \\
&\text{where: } x_1 = \log_{10}(\mathit{WriteMem}[\mathrm{Byte}]), x_2 = \log_{10}(\mathit{ReadMem}[\mathrm{Byte}]); 4.2144 < x_1 < 7.8268, 4.2144 < x_2 < 7.8268; \mathit{pDW} = 3.324\text{e-}012, R^2 = 0.9914
\end{aligned}$$

$$\begin{aligned}
&\mathit{QoDD}[1] = -13560 - 8876.3x_1^0x_2^1 + 25201x_1^1x_2^0 + 4925.2x_1^0x_2^2 - 1324.2x_1^1x_2^1 - 11857x_1^2x_2^0 - 1450.7x_1^0x_2^3 + 519.96x_1^1x_2^2 - 109.79x_1^2x_2^1 \\
&+3321.5x_1^3x_2^0 + 213.33x_1^0x_2^4 + 44.192x_1^1x_2^3 - 216.46x_1^2x_2^2 + 181.02x_1^3x_2^1 - 594.72x_1^4x_2^0 - 16.228x_1^0x_2^5 - 17.038x_1^1x_2^4 + 21.715x_1^2x_2^3 \\
&+12.225x_1^3x_2^2 - 28.956x_1^4x_2^1 + 64.343x_1^5x_2^0 + 0.59824x_1^0x_2^6 + 1.3201x_1^1x_2^5 - 0.47337x_1^2x_2^4 - 1.6504x_1^3x_2^3 + 0.28897x_1^4x_2^2 + 1.7887x_1^5x_2^1 - 3.7887x_1^6x_2^0 \\
&-0.0079957x_1^0x_2^7 - 0.032343x_1^1x_2^6 - 0.010555x_1^2x_2^5 + 0.041479x_1^3x_2^4 + 0.024532x_1^4x_2^3 - 0.024869x_1^5x_2^2 - 0.040227x_1^6x_2^1 + 0.093169x_1^7x_2^0 \\
&\text{where: } x_1 = \log_{10}(\mathit{WriteMem}[\mathrm{Byte}]), x_2 = \log_{10}(\mathit{ReadMem}[\mathrm{Byte}]); 4.2144 < x_1 < 7.8268, 4.2144 < x_2 < 7.8268; \mathit{pDW} = 7.58e-005, R^2 = 0.9987
\end{aligned}$$

Eq. 15 Relationship between TCP Memory Sizes and QoD

$$\begin{array}{l}
MDRB[s] = -0.279 - 144x_1^0x_2^1 + 27.8x_1^1x_2^0 + 1.02e + 005x_1^0x_2^2 - 2.93e + 004x_1^1x_2^1 + 1.19e + 003x_1^2x_2^0 - 8.55e + 005x_1^0x_2^3 \\
-2.79e + 006x_1^1x_2^2 + 8.95e + 005x_1^2x_2^1 - 4.35e + 004x_1^3x_2^0 + 7.79e + 007x_1^0x_2^4 - 2.93e + 007x_1^1x_2^3 + 3.47e + 007x_1^2x_2^2 \\
-9.46e + 006x_1^3x_2^1 + 4.52e + 005x_1^4x_2^0 - 6.88e + 008x_1^0x_2^5 - 2.36e + 008x_1^1x_2^4 + 1.96e + 008x_1^2x_2^3 - 1.67e + 008x_1^3x_2^2 \\
+4.11e + 007x_1^4x_2^1 - 1.9e + 006x_1^5x_2^0 - 4.79e + 009x_1^0x_2^6 + 7.33e + 009x_1^1x_2^5 - 1.9e + 009x_1^2x_2^4 - 2.13e + 006x_1^3x_2^3 \\
+2.43e + 008x_1^4x_2^2 - 6.19e + 007x_1^5x_2^1 + 2.82e + 006x_1^6x_2^0
\end{array}$$

where: $x_1 = OneWayDelay[s], x_2 = DV[s]; 0.0034 < x_1 < 0.2483, 0.0017 < x_2 < 0.0494; pDW = 8.93e - 033, R^2 = 0.87$

$$\begin{array}{l}
IBT[s] = 0.485 + 106x_1^0x_2^1 + 57.2x_1^1x_2^0 + 3.11e + 004x_1^0x_2^2 - 1.69e + 004x_1^1x_2^1 - 56.4x_1^2x_2^0 + 3.61e + 006x_1^0x_2^3 \\
-2.7e + 006x_1^1x_2^2 + 7.04e + 005x_1^2x_2^1 - 1.62e + 004x_1^3x_2^0 - 1.34e + 008x_1^0x_2^4 + 5.49e + 006x_1^1x_2^3 + 2.6e + 007x_1^2x_2^2 \\
-7.36e + 006x_1^3x_2^1 + 2.12e + 005x_1^4x_2^0 + 3.76e + 009x_1^0x_2^5 - 9.96e + 008x_1^1x_2^4 + 2.32e + 008x_1^2x_2^3 - 1.4e + 008x_1^3x_2^2 \\
+3.29e + 007x_1^4x_2^1 - 1.01e + 006x_1^5x_2^0 - 3.51e + 010x_1^0x_2^6 + 9.6e + 009x_1^1x_2^5 - 7.33e + 008x_1^2x_2^4 - 2.89e + 008x_1^3x_2^3 \\
+2.37e + 008x_1^4x_2^2 - 5.24e + 007x_1^5x_2^1 + 1.67e + 006x_1^6x_2^0
\end{array}$$

where: $x_1 = OneWayDelay[s], x_2 = DV[s]; 0.0034 < x_1 < 0.2483, 0.0017 < x_2 < 0.0494; pDW = 0.000338, R^2 = 0.826$

$$\begin{array}{l}
RBF[\mathrm{Hz}] = -0.00744 - 2.04x_1^0x_2^1 + 0.03x_1^1x_2^0 + 2.44e + 003x_1^0x_2^2 - 754x_1^1x_2^1 + 59.6x_1^2x_2^0 - 6.58e + 004x_1^0x_2^3 \\
-4.62e + 004x_1^1x_2^2 + 1.95e + 004x_1^2x_2^1 - 1.48e + 003x_1^3x_2^0 + 2.67e + 006x_1^0x_2^4 - 1.89e + 005x_1^1x_2^3 + 5.03e + 005x_1^2x_2^2 \\
-1.87e + 005x_1^3x_2^1 + 1.34e + 004x_1^4x_2^0 - 3.58e + 007x_1^0x_2^5 - 4.78e + 006x_1^1x_2^4 + 1.87e + 006x_1^2x_2^3 - 2.27e + 006x_1^3x_2^2 \\
+7.62e + 005x_1^4x_2^1 - 5.22e + 004x_1^5x_2^0 + 1.14e + 008x_1^0x_2^6 + 9.71e + 007x_1^1x_2^5 - 2.02e + 007x_1^2x_2^4 + 4.43e + 005x_1^3x_2^3 \\
+3.24e + 006x_1^4x_2^2 - 1.11e + 006x_1^5x_2^1 + 7.37e + 004x_1^6x_2^0
\end{array}$$

where: $x_1 = OneWayDelay[s], x_2 = DV[s]; 0.0034 < x_1 < 0.2483, 0.0017 < x_2 < 0.0494; pDW = 1.15e - 036, R^2 = 0.878$

$$\begin{array}{l}
QoDD[1] = -0.0465 - 35.3x_1^0x_2^1 + 12.7x_1^1x_2^0 + 1.24e + 004x_1^0x_2^2 - 3.14e + 003x_1^1x_2^1 - 132x_1^2x_2^0 + 2.67e + 005x_1^0x_2^3 \\
-6.34e + 005x_1^1x_2^2 + 1.63e + 005x_1^2x_2^1 - 4e + 003x_1^3x_2^0 + 1.83e + 007x_1^0x_2^4 - 1.42e + 007x_1^1x_2^3 + 9.85e + 006x_1^2x_2^2 \\
-2.12e + 006x_1^3x_2^1 + 7.04e + 004x_1^4x_2^0 - 3.45e + 008x_1^0x_2^5 + 5.78e + 007x_1^1x_2^4 + 5.87e + 007x_1^2x_2^3 - 4.93e + 007x_1^3x_2^2 \\
+1.02e + 007x_1^4x_2^1 - 3.64e + 005x_1^5x_2^0 + 1.04e + 009x_1^0x_2^6 + 9.39e + 008x_1^1x_2^5 - 5.04e + 008x_1^2x_2^4 - 1.08e + 007x_1^3x_2^3 \\
+7.47e + 007x_1^4x_2^2 - 1.64e + 007x_1^5x_2^1 + 6.07e + 005x_1^6x_2^0
\end{array}$$

where: $x_1 = OneWayDelay[s], x_2 = DV[s]; 0.0034 < x_1 < 0.2483, 0.0017 < x_2 < 0.0494; pDW = 4.12e - 045, R^2 = 0.863$

Eq. 16 Relationship between DV and QoD (Different One-Way Delay)

$$\begin{array}{l}
MDRB[s] = -0.63153 + 1.0254x_1^0x_2^1 + 845.8x_1^1x_2^0 - 0.60229x_1^0x_2^2 - 491.76x_1^1x_2^1 - 2.7119e + 005x_1^2x_2^0 \\
+0.13929x_1^0x_2^3 + 28.506x_1^1x_2^2 + 1.5976e + 005x_1^2x_2^1 + 3.3439e + 007x_1^3x_2^0 - 0.010545x_1^0x_2^4 \\
-4.6196x_1^1x_2^3 + 1940.1x_1^2x_2^2 - 1.0832e + 007x_1^3x_2^1 - 1.3181e + 009x_1^4x_2^0
\end{array}$$

where: $x_1 = Loss[1], x_2 = PlayoutBufLen[s]; 0.0000 < x_1 < 0.0100, 1.0000 < x_2 < 6.0000; pDW = 3.4e - 005, R^2 = 0.969$

$$\begin{array}{l}
IBT[s] = -0.50909 + 2.8375x_1^0x_2^1 + 541.27x_1^1x_2^0 - 1.1441x_1^0x_2^2 - 147.33x_1^1x_2^1 - 1.4722e + 005x_1^2x_2^0 \\
+0.21337x_1^0x_2^3 + 50.765x_1^1x_2^2 + 14830x_1^2x_2^1 + 1.8984e + 007x_1^3x_2^0 - 0.013598x_1^0x_2^4 \\
-4.1711x_1^1x_2^3 - 362.04x_1^2x_2^2 - 4.3685e + 005x_1^3x_2^1 - 9.0955e + 008x_1^4x_2^0
\end{array}$$

where: $x_1 = Loss[1], x_2 = PlayoutBufLen[s]; 0.0000 < x_1 < 0.0100, 1.0000 < x_2 < 6.0000; pDW = 0.866, R^2 = 0.992$

$$\begin{array}{l}
RBF[\mathrm{Hz}] = +0.0031812 - 0.014418x_1^0x_2^1 + 0.77496x_1^1x_2^0 + 0.0076622x_1^0x_2^2 - 7.5682x_1^1x_2^1 \\
+4839.4x_1^2x_2^0 - 0.00092137x_1^0x_2^3 + 1.1263x_1^1x_2^2 - 426.36x_1^2x_2^1 - 1.4945e + 005x_1^3x_2^0
\end{array}$$

where: $x_1 = Loss[1], x_2 = PlayoutBufLen[s]; 0.0000 < x_1 < 0.0100, 1.0000 < x_2 < 6.0000; pDW = 0.405, R^2 = 0.979$

$$\begin{array}{l}
QoDD[1] = +0.067912 - 0.048828x_1^0x_2^1 - 4.3391x_1^1x_2^0 + 0.022514x_1^0x_2^2 + 17.334x_1^1x_2^1 - 13297x_1^2x_2^0 \\
-0.0036273x_1^0x_2^3 - 3.3556x_1^1x_2^2 - 981.49x_1^2x_2^1 + 4.0785e + 006x_1^3x_2^0 + 0.00021367x_1^0x_2^4 \\
+0.19756x_1^1x_2^3 + 90.79x_1^2x_2^2 + 11146x_1^3x_2^1 - 2.2709e + 008x_1^4x_2^0
\end{array}$$

where: $x_1 = Loss[1], x_2 = PlayoutBufLen[s]; 0.0000 < x_1 < 0.0100, 1.0000 < x_2 < 6.0000; pDW = 0.154, R^2 = 0.994$

Eq. 17 Relationship between Playout Buffer Length and QoD (Different Loss Rates)

$MDRB[\mathrm{s}] = 0$

where: $x_1 = DV[\mathrm{ms}], x_2 = PlayoutBufLen[\mathrm{s}]; 0.0000 < x_1 < 10.0000, 1.0000 < x_2 < 6.0000$

$IBT[\mathrm{s}] = +1.2773 + 0.30711 x_1^0 x_2^1 + 0.16777 x_1^1 x_2^0 + 0.017469 x_1^0 x_2^2 + 0.0016375 x_1^1 x_2^1 - 0.0088624 x_1^2 x_2^0$

where: $x_1 = DV[\mathrm{ms}], x_2 = PlayoutBufLen[\mathrm{s}]; 0.0000 < x_1 < 10.0000, 1.0000 < x_2 < 6.0000; pDW = 0.842, R^2 = 0.996$

$RBF[\mathrm{Hz}] = 0$

where: $x_1 = DV[\mathrm{ms}], x_2 = PlayoutBufLen[\mathrm{s}]; 0.0000 < x_1 < 10.0000, 1.0000 < x_2 < 6.0000$

$QoDD[1] = +0.025504 + 0.0061325 x_1^0 x_2^1 + 0.0033499 x_1^1 x_2^0 + 0.00034882 x_1^0 x_2^2 + 3.2698e - 005 x_1^1 x_2^1 - 0.00017697 x_1^2 x_2^0$

where: $x_1 = DV[\mathrm{ms}], x_2 = PlayoutBufLen[\mathrm{s}]; 0.0000 < x_1 < 10.0000, 1.0000 < x_2 < 6.0000; pDW = 0.842, R^2 = 0.996$

Eq. 18 Relationship between Playout Buffer Length and QoD (Different DVs)

$MDRB[\mathrm{s}] = +1.1564 - 0.0020273 x_1^0 x_2^1 - 174.7 x_1^1 x_2^0 + 4.6967e - 007 x_1^0 x_2^2 + 0.23184 x_1^1 x_2^1 + 508.93 x_1^2 x_2^0$

where: $x_1 = Loss[1], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 0.0100, 378.2645 < x_2 < 3364.1742; pDW = 0.631, R^2 = 0.911$

$IBT[\mathrm{s}] = +0.95943 + 0.00055915 x_1^0 x_2^1 + 32.817 x_1^1 x_2^0 - 3.0409e - 008 x_1^0 x_2^2 + 0.047285 x_1^1 x_2^1 + 1301.6 x_1^2 x_2^0$

where: $x_1 = Loss[1], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 0.0100, 378.2645 < x_2 < 3364.1742; pDW = 0.00549, R^2 = 0.978$

$RBF[\mathrm{Hz}] = +0.017284 - 2.5637e - 005 x_1^0 x_2^1 - 4.0904 x_1^1 x_2^0 + 5.5026e - 009 x_1^0 x_2^2 + 0.0030933 x_1^1 x_2^1 + 182.74 x_1^2 x_2^0$

where: $x_1 = Loss[1], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 0.0100, 378.2645 < x_2 < 3364.1742; pDW = 0.847, R^2 = 0.924$

$QoDD[1] = +0.11675 - 0.00012339 x_1^0 x_2^1 - 25.571 x_1^1 x_2^0 + 2.675e - 008 x_1^0 x_2^2 + 0.017742 x_1^1 x_2^1 + 1419.7 x_1^2 x_2^0$

where: $x_1 = Loss[1], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 0.0100, 378.2645 < x_2 < 3364.1742; pDW = 0.887, R^2 = 0.919$

Eq. 19 Relationship between Video Bitrate and QoD (Different Loss Rates)

$MDRB[\mathrm{s}] = 0$

where: $x_1 = DV[\mathrm{ms}], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 10.0000, 378.2645 < x_2 < 3364.1742$

$IBT[\mathrm{s}] = +1.0187 + 0.00059768 x_1^0 x_2^1 + 0.13693 x_1^1 x_2^0 - 6.405e - 008 x_1^0 x_2^2 + 1.1205e - 005 x_1^1 x_2^1 - 0.0082025 x_1^2 x_2^0$

where: $x_1 = DV[\mathrm{ms}], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 10.0000, 378.2645 < x_2 < 3364.1742; pDW = 0.578, R^2 = 0.983$

$RBF[\mathrm{Hz}] = 0$

where: $x_1 = DV[\mathrm{ms}], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 10.0000, 378.2645 < x_2 < 3364.1742$

$QoDD[1] = +0.020341 + 1.1935e - 005 x_1^0 x_2^1 + 0.0027342 x_1^1 x_2^0 - 1.279e - 009 x_1^0 x_2^2 + 2.2375e - 007 x_1^1 x_2^1 - 0.00016379 x_1^2 x_2^0$

where: $x_1 = DV[\mathrm{ms}], x_2 = VideoBitrate[\mathrm{Kbps}]; 0.0000 < x_1 < 10.0000, 378.2645 < x_2 < 3364.1742; pDW = 0.578, R^2 = 0.983$

Eq. 20 Relationship between Video Bitrate and QoD (Different DVs)

B. Summary of Experiment Results

Table 9 shows the summary of experiment results.

Table 9 Summary of Experiment Results

Experiment Number	Detail	IBT [ms]	MDRB [ms]	RBF [H_z]	QoDD	Condition
Experiment 1: TCP memory sizes	Increasing TCP-write-memory size from 16 Kb to 64Mb	9.30 to 3.00	2.72 to 0	0.050 to 0	0.533 to 0.0597	Baseline 1
	Increasing TCP-read-memory size from 16 Kb to 64Mb	7.25 to 3.46	6.37 to 0	0.140 to 0	1.04 to 0.0586	
	Increasing TCP-with-memory size along with increasing TCP-read-memory size from 16 Kb to 64Mb	9.29 to 2.67	8.65 to 0	0.160 to 0	1.56 to 0.0533	
Experiment 2.1: TCP CCAs with different levels of random loss rate	Changing TCP CCA along with increasing one-way loss rate from 0 to 1%.	Min 2.37 Max 19.9	Min 0 Max 28.3	Min 0 Max 0.250	Min 0.0472 Max 7.69	Baseline 2
	At loss rate of 1%, changing TCP CCA	Min 4.29 Max 19.9	Min 5.40 Max 28.3	Min 0.0824 Max 0.250	Min 0.53 Max 7.69	
Experiment 2.2: TCP CCAs with different levels of DV	Changing TCP CCA along with increasing one-way DV from 0 to 50 ms.	Min 2.37 Max 16.5	Min 0 Max 12.7	Min 0 Max 0.0618	Min 0.0970 Max 1.96	
	At DV of 50 ms, changing TCP CCA	Min 4.83 Max 16.5	Min 0 Max 12.7	Min 0 Max 0.0618	Min 0.0474 Max 3.05	
Experiment 3: Delay variation	Increasing DV from 1 to 49 ms along with increasing one-way delay from 1 to 245 ms.	Min 1.28 Max 21.5	Min 0, Max 19.1	Min 0 Max 0.240	Min 0.0260 Max 4.60	
	At DV of 49 ms, increasing one-way delay from 1 to 245 ms.	Min 5.47 Max 21.5	Min 0 Max 19.1	Min 0 Max 0.240	Min 0.109 Max 4.60	
Experiment 4.1: Playout buffer length subject to different levels of loss rate	Increasing playout buffer length from 1 to 6 seconds along with increasing loss rate from 0 to 1%	Min 1.54 Max 9.80	Min 0 Max 10.4	Min 0 Max 0.0379	Min 0.0308 Max 0.591	Baseline 3
	At loss rate of 1%, increasing playout buffer length from 1 to 6 seconds	2.08 to 9.80	2.06 to 10.4	0.244 to 0.0379	Min 0.544 Max 0.591	
Experiment 4.2: Playout buffer length subject to different levels of DV	Increasing playout buffer length from 1 to 6 seconds along with increasing DV from 0 to 10 ms	Min 1.54 Max 4.64	Retain at 0	Retain at 0	Min 0.0308 Max 0.0927	
	At DV of 10ms, increasing playout buffer length from 1 to 6 seconds along with increasing DV from 0 to 10 ms	2.46 to 4.64	Retain at 0	Retain at 0	0.0491 to 0.0927	
Experiment 5.1: Video bitrate subject to different levels of loss rate	Increasing from 378 to 3364 Kbps along with increasing loss rate from 0 to 1%	Min 1.22 Max 4.57	Min 0 Max 5.48	Min 0 Max 0.0819	Min 0.0244 Max 0.540	
	At loss rate of 1%, increasing video bitrate from 378 to 3364 Kbps	2.08 to 4.57	0 to 5.48	0 to 0.0819	0.0416 to 0.540	
Experiment 5.2: Video bitrate subject to different levels of DV	Increasing video bitrate from 378 to 3364 Kbps along with increasing DV from 0 to 10 ms	Min 1.22 Max 3.24	Retain at 0	Retain at 0	Min 0.0244 Max 0.0647	
	At DV of 10ms, increasing from 378 to 3364 Kbps along with increasing DV from 0 to 10 ms	1.82 to 3.24	Retain at 0	Retain at 0	0.0365 to 0.0647	

Bibliography

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “RFC 2616 - Hypertext transfer protocol – HTTP/1.1,” IETF, 2006.

[2] J. Postel, “RFC 793 - Transmission control protocol,” IETF, 2003.

[3] A. Begen, T. Akgul, and M. Baugher, “Watching video over the web: Part 1: Streaming protocols,” *Internet Computing, IEEE*, vol. 15, no. 2, 2011.

[4] M. Saxena, U. Sharan, and S. Fahmy, “Analyzing video services in web 2.0: a global perspective,” in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2008.

[5] (2012) YouTube. [Online]. Available: http://www.youtube.com/

[6] (2012) Dailymotion. [Online]. Available: http://www.dailymotion.com/

[7] (2012) Metacafe. [Online]. Available: http://www.metacafe.com/

[8] (2012) YouTube video speed history. [Online]. Available: http://youtube.com/my_speed

[9] R. Walker, L. Johnson, and S. Leonard, “Re-thinking the conceptualization of customer value and service quality within the service-profit chain,” *Managing Service Quality*, vol. 16, no. 1, 2006.

[10] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, “Understanding the impact of video quality on user engagement,” in *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*. ACM, 2011.

[11] M. Söderlund and M. Vilgon, “Customer satisfaction and links to customer profitability: an empirical examination of the association between attitudes and behavior,” *SSE/EFI Working Paper Series in Business Administration*, 1999.

[12] I. Rec, “P. 800.1, mean opinion score (MOS) terminology,” *International Telecommunication Union, Geneva*, 2006.

[13] T. Minhas, O. Gonzalez Lagunas, P. Arlos, and M. Fiedler, “Mobile video sensitivity to packet loss and packet delay variation in terms of QoE,” in *Packet Video Workshop (PV), 2012 19th International*. IEEE, 2012.

[14] M. Fiedler, H. Zepernick, L. Lundberg, P. Arlos, and M. Pettersson, “QoE-based cross-layer design of mobile video systems: Challenges and concepts,” in *Computing and Communication Technologies, 2009. RIVF’09. International Conference on*. IEEE, 2009.

[15] T. Liu, Y. Wang, J. Boyce, H. Yang, and Z. Wu, “A novel video quality metric for low bit-rate video considering both coding and packet-loss artifacts,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 3, no. 2, 2009.

[16] Q. Dai and R. Lehnert, “Prediction of video perceptual quality in the presence of packet loss,” in *Telecommunications (ConTEL), Proceedings of the 2011 11th International Conference on*. IEEE, 2011.

[17] C. Bailey, M. Seyedebrahimi, and X. Peng, “Pause intensity: A no-reference quality assessment metric for video streaming in TCP networks,” in *Multimedia and Expo (ICME), 2012 IEEE International Conference on*. IEEE, 2012.

[18] N. Chen, X. Jiang, C. Wang, and J. Su, “Study on relationship between network video packet loss and video quality,” in *Image and Signal Processing (CISP), 2011 4th International Congress on*, vol. 1. IEEE, 2011.

[19] D. Rodriguez, J. Abrahão, D. Begazo, R. Rosa, and G. Bressan, “Quality metric to assess video streaming service over TCP considering temporal location of pauses,” *Consumer Electronics, IEEE Transactions on*, vol. 58, no. 3, 2012.

[20] W. Leister, S. Boudko, and T. Halbach, “Estimation of subjective video quality as feedback to content providers,” in *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*. IEEE, 2010.

[21] J. Postel, “RFC 768 - User datagram protocol,” IETF, 1980.

[22] F. Metzger, A. Rafetseder, D. Stezenbach, and K. Tutschku, “Analysis of web-based video delivery,” in *FITCE Congress (FITCE), 2011 50th*. IEEE, 2011.

[23] R. Mok, E. Chan, and R. Chang, “Measuring the quality of experience of HTTP video streaming,” in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011.

[24] D. Wu, Y. Hou, W. Zhu, Y. Zhang, and J. Peha, “Streaming video over the internet: approaches and directions,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 3, 2001.

[25] C. Palau, M. Esteve, J. Martnez, B. Molina, and I. Pérez, “Urban traffic control: a streaming multimedia approach,” in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 2005.

[26] K. Ma, R. Bartos, S. Bhatia, and R. Nair, “Mobile video delivery with HTTP,” *Communications Magazine, IEEE*, vol. 49, no. 4, 2011.

[27] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RFC 3550 - RTP: A transport protocol for real-time applications,” IETF, 2003.

[28] H. Schulzrinne, “RFC 2326 - Real time streaming protocol (RTSP),” IETF, 1998.

[29] D. Wing, “RFC 4961 - Symmetric RTP/RTP control protocol (RTCP),” IETF, 2007.

[30] T. Yoshimura, Y. Yonemoto, T. Ohya, M. Etoh, and S. Wee, “Mobile streaming media CDN enabled by dynamic SMIL,” in *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002.

[31] J. Kurose and K. Ross, *Computer Networks: A Top Down Approach*, 5th ed. Pearson Addison Wesley, 2010, ch. 7, pp. 612–613,631–639.

[32] S. Latré, N. Staelens, P. Simoens, B. De Vleeschauwer, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, S. Van den Berghe, R. Huysegems *et al.*, “On-line estimation of the QoE of progressive download services in multimedia access networks.” 2008.

[33] B. De Vleeschauwer, W. Van de Meerssche, P. Simoens, F. De Turck, B. Dhoedt, P. Demeester, E. Gilon, K. Struyve, and T. Van Caenegem, “On the enhancement of QoE for IPTV services through knowledge plane deployment.” 2006.

[34] Z. Yetgin and G. Seckin, “Progressive download for 3G wireless multicasting,” in *Future Generation Communication and Networking (FGCN 2007)*, vol. 1. IEEE, 2007.

[35] T. Stockhammer, “Dynamic adaptive streaming over HTTP - standards and design principles,” in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011.

[36] T. Minhas, “Network impact on quality of experience of mobile video,” Blekinge Institute of Technology, 2012.

[37] T. N. Minhas and M. Fiedler, “Quality of experience hourglass model,” in *Computing, Management and Telecommunications (ComManTel), 2013 International Conference on*. IEEE, 2013.

[38] J. Shaikh, M. Fiedler, and D. Collange, “Quality of experience from user and network perspectives,” *Annals of Telecommunications*, vol. 65, no. 1, 2010.

[39] M. Fiedler, H. Hlavacs, K. Hackbarth, and P. Arlos, “Quality of experience,” *Annals of Telecommunications*, vol. 65, no. 1, 2010.

[40] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler, “Waiting times in quality of experience for web based services,” in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. IEEE, 2012.

[41] K. Yang, G. Dane, and K. El-Maleh, “Temporal quality evaluation for enhancing compressed video,” in *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*. IEEE, 2007.

[42] T. Porter and X. Peng, “An objective approach to measuring video playback quality in lossy networks using TCP,” *Communications Letters, IEEE*, vol. 15, no. 1, pp. 76–78, 2011.

[43] B. Forouzan, *TCP/IP Protocol Suite*, 4th ed., ser. McGraw-Hill Forouzan Networking Series. McGraw-Hill, 2002, p. 482.

[44] (2012) ActionScript 3.0 reference for the Adobe Flash platform. Adobe Systems Inc. [Online]. Available: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/

[45] M. Jain, R. Prasad, and C. Dovrolis, “The TCP bandwidth-delay product revisited: network buffering, cross traffic, and socket buffer auto-sizing,” 2003.

[46] S. Tao, L. Jacob, and A. Ananda, “A TCP socket buffer auto-tuning daemon,” in *Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on*. IEEE, 2003.

[47] (2012) TCP(7) - TCP protocol - Linux Manual Page. Linux. [Online]. Available: http://kernel.org/doc/man-pages/online/pages/man7/tcp.7.html

[48] T. Kelly, “Scalable TCP: Improving performance in high-speed wide area networks,” *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, 2003.

[49] C. Caini and R. Firrincieli, “TCP Hybla: a TCP enhancement for heterogeneous networks,” *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, 2004.

[50] F. Xin and A. Jamalipour, “TCP throughput performance and fairness in wireless networks under spurious timeouts,” in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 3. IEEE, 2005.

[51] T. Klein, K. Leung, R. Parkinson, and L. Samuel, “Avoiding spurious TCP timeouts in wireless networks by delay injection,” in *Global Telecommunications Conference, 2004. GLOBECOM’04. IEEE*, vol. 5. IEEE, 2004.

[52] T. Kim, N. Avadhanam, and S. Subramanian, “Dimensioning receiver buffer requirement for unidirectional VBR video streaming over TCP,” in *Image Processing, 2006 IEEE International Conference on*. IEEE, 2006.

[53] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, “TCP congestion avoidance algorithm identification,” in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011.

[54] L. Andrew. (2008) Compound TCP Linux module. [Online]. Available: http://netlab.caltech.edu/lachlan/ctcp

[55] C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi, and R. Wang, “TCP Westwood: end-to-end congestion control for wired/wireless networks,” *Wireless Networks*, vol. 8, no. 5, 2002.

[56] W. Reese, “Nginx: the high-performance web server and reverse proxy,” *Linux Journal*, vol. 2008, no. 173, 2008.

[57] A. Jurgelionis, J.-P. Laulajainen, M. Hirvonen, and A. I. Wang, “An empirical study of NetEm network emulation functionalities,” in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*. IEEE, 2011.

[58] J. Shaikh, T. N. Minhas, P. Arlos, and M. Fiedler, “Evaluation of delay performance of traffic shapers,” in *Security and Communication Networks (IWSCN), 2010 2nd International Workshop on*. IEEE, 2010.

[59] P. Arlos, M. Fiedler, and A. Nilsson, “A distributed passive measurement infrastructure,” *Passive and Active Network Measurement*, 2005.

[60] *DAG 3.6EP/T card user guide, EDM01-05*. Endace Measurement Systems Ltd, 2005.

[61] C. Bernard, H. Debar, and S. Benayoune, “Cross-domain vulnerabilities over social networks,” in *Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on*. IEEE, 2012.

[62] I. Hickson and D. Hyatt, “HTML5: A vocabulary and associated APIs for HTML and XHTML,” *W3C Working Draft edition*, 2011.

[63] S. Lee and C. Burrell, “Introduction to streaming video for novices,” *Library Hi Tech News*, vol. 21, no. 2, 2004.

[64] A. Fecheyr-Lippens, “A review of HTTP live streaming,” Jan 2010.

[65] (2012) Apache Flex (incubating). The Apache Software Foundation. [Online]. Available: http://incubator.apache.org/flex/

[66] G. Seed, *An introduction to Object-Oriented programming in C++: With applications in computer graphics*, 2nd ed. Springer, 2001, p. 91.

[67] W. Wu, M. Crawford, and M. Bowden, “The performance analysis of Linux networking–packet receiving,” *Computer Communications*, vol. 30, no. 5, 2007.

[68] P. Arlos, “Application level measurement,” *Network Performance Engineering*, 2011.

[69] W. Cleveland, “Robust locally weighted regression and smoothing scatterplots,” *Journal of the American statistical association*, vol. 74, no. 368, 1979.

[70] G. Seber and A. Lee, *Linear regression analysis*, 2nd ed. Wiley, 2012, vol. 936, p. 4.

[71] P. Doruska, “Methods for quantitatively describing tree crown profiles of loblolly pine (pinus taeda l.),” Ph.D. dissertation, Dissertação de doutoramento. Virginia Polytechnic Institute and State University. USA, 1998.

[72] L. Marquez, T. Hill, R. Worthley, and W. Remus, “Neural network models as an alternative to regression,” in *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*, vol. 4. IEEE, 1991.

[73] W. Hardle, *Applied nonparametric regression*. Cambridge University Press, 1990, vol. 27, p. xi.

[74] S. Brown, R. Tauler, and B. Walczak, *Comprehensive chemometrics: chemical and biochemical data analysis*. Elsevier Science Limited, 2009, p. 348.

[75] J. Szymanski, *Basic Mathematics for Electronic Engineers: Models and Applications*, ser. Tutorial guides in electronic engineering 16. John Wiley & Sons, Incorporated, 1989, pp. 174–175.

[76] A. Hamadeh, “Calculus 3 for chemical engineering,” 2011, Math 217 Lecture, University of Waterloo.

[77] S. Ghorpade and B. Limaye, *A course in multivariable calculus and analysis*. Springer, 2009, p. 411.

[78] *Matlab Statistics Toolbox 7 User’s Guide*. The MathWorks Inc., 2010, pp. 9–3, 19–315.

[79] S. Billings and S. Chen, “Extended model set, global data and threshold model identification of severely non-linear systems,” *International Journal of Control*, vol. 50, no. 5, 1989.

[80] D. Montgomery, E. Peck, and G. Vining, *Introduction to linear regression analysis*. Wiley, 2012, vol. 821, ch. 7.

[81] P. Royston and W. Sauerbrei, *Multivariable model-building: a pragmatic approach to regression analysis based on fractional polynomials for modelling continuous variables*. Wiley, 2008, vol. 790, pp. 18–19, 229.

[82] A. Badiru, *Project management in manufacturing and high technology operations*. Wiley-Interscience, 1996, p. 263.

[83] D. Kleinbaum, L. Kupper, and K. Muller, *Applied regression analysis and other multivariable methods*. Duxbury Press, 2007, p. 300.

[84] G. Wang and C. Jain, *Regression analysis: modeling & forecasting*. Graceway Publishing Company, 2003, p. 72.

[85] B. Vasigh, T. Tacker, and K. Fleming, *Introduction to air transport economics: from theory to applications*. Ashgate Publishing Company, 2008, p. 269.

[86] C. Lee, J. Lee, and A. Lee, *Statistics for Business and Financial Economics*. World Scientific, 2000, vol. 1, p. 712.

[87] (2012) Durbin-Watson test. MathWorks Nordic. [Online]. Available: http://mathworks.se/help/stats/dwtest.html

[88] “Multiple regression,” COM531 Lecture, Cleveland State University.

[89] A. Kuzmanovic and E. Knightly, “TCP-LP: A distributed algorithm for low priority data transfer,” in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3. IEEE, 2003.

[90] U. Bodin and A. Simonsson, “Effects on TCP from radio-block scheduling in WCDMA high speed downlink shared channels,” *Quality for All*, 2003.

[91] R. Ludwig and R. Katz, “The Eifel algorithm: making TCP robust against spurious retransmissions,” *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 1, 2000.

[92] P. Sarolahti, M. Kojo, and K. Raatikainen, “F-RTO: an enhanced recovery algorithm for TCP retransmission timeouts,” *SIGCOMM Computer Communication Review*, vol. 33, no. 2, Apr 2003.

[93] N. Garapati, “Quality estimation of YouTube video service,” Blekinge Institute of Technology, 2010.

[94] J. Padia, “H.264 to VP6 transcoder,” Jul 2008, Interim Project Report, The University of Texas at Arlington.

[95] M. Gorman, *Understanding the Linux Virtual Memory Manager*, ser. Bruce Perens Open Source. Prentice Hall, 2004, ch. 2.

[96] D. Bovet and M. Cesati, *Understanding the Linux Kernel*, 3rd ed., ser. Oreilly Series. O’Reilly Media, Incorporated, 2005, ch. 2.