Several factors affect the perceived quality of a video streamed over a network. Technical aspects related to the codec and the video play an important role besides the delivery of the video content in a timely and error-free manner. If the streamed video frames do not arrive in time, temporal artifacts may become observable during the video playback. This has potentially an adverse effect on the user experience. The study of temporal artifacts during the video playback is the area of concern for Quality of Delivery.

The licentiate work investigates the Quality of Delivery for Variable Bit Rate video sent over wireless technologies. The ability of the network to deliver data to the video’s jitter buffer before its playout deadline is studied. Jitter buffer exhaustions are of particular interest as they tell something about the Quality of Delivery. A number of experiments are conducted where a set of videos with different bit-rates are streamed to a mobile device over a wireless LAN and a W-CDMA network. The video streams are recorded and analyzed based on specific Quality of Service parameters and are related to the states of the jitter buffer. The statistical tools Support Vector Machines and the Mahalanobis distance are applied to the parameters to obtain a model that can classify the jitter buffer states. The performance evaluation indicates that the Mahalanobis distance can classify the jitter buffer state marginally better than the Support Vector Machines. However, the Support Vector Machines can produce more reliable predictions compared to the Mahalanobis distance. It is also observed that the perceived Quality of Delivery is not only affected by the behavior of the wireless networks but also by the behavior of the streaming server. Finally, the ability of the Quality of Service parameters to describe the Quality of Delivery is studied. The results indicate that metrics based on the packet arrival-rate and packet inter-arrival time are most suitable in this particular case.
On the Quality of Delivery for Variable Bit Rate Video

Karel De Vogeleer
On the Quality of Delivery for Variable Bit Rate Video

Karel De Vogeleer

Licentiate Dissertation in Telecommunications

School of Computing
Blekinge Institute of Technology
SWEDEN
Abstract

Several factors affect the perceived quality of a video streamed over a network. Technical aspects related to the codec and the video play an important role besides the delivery of the video content in a timely and error-free manner. If the streamed video frames do not arrive in time, temporal artifacts may become observable during the video playback. This has potentially an adverse effect on the user experience. The study of temporal artifacts during the video playback is the area of concern for Quality of Delivery.

The licentiate work investigates the Quality of Delivery for Variable Bit Rate video sent over wireless technologies. The ability of the network to deliver data to the video’s jitter buffer before its playout deadline is studied. Jitter buffer exhaustions are of particular interest as they tell something about the Quality of Delivery. A number of experiments are conducted where a set of videos with different bit-rates are streamed to a mobile device over a wireless LAN and a W-CDMA network. The video streams are recorded and analyzed based on specific Quality of Service parameters and are related to the states of the jitter buffer. The statistical tools Support Vector Machines and the Mahalanobis distance are applied to the parameters to obtain a model that can classify the jitter buffer states. The performance evaluation indicates that the Mahalanobis distance can classify the jitter buffer state marginally better than the Support Vector Machines. However, the Support Vector Machines can produce more reliable predictions compared to the Mahalanobis distance. It is also observed that the perceived Quality of Delivery is not only affected by the behavior of the wireless networks but also by the behavior of the streaming server. Finally, the ability of the Quality of Service parameters to describe the Quality of Delivery is studied. The results indicate that metrics based on the packet arrival-rate and packet inter-arrival time are most suitable in this particular case.
Acknowledgements

My deepest gratitudes and appreciation I would like to express to those who have encouraged me and contributed to the realization of this work. Foremost I thank my advisor Prof. Adrian Popescu for his guidance and belief in me. Also, I am imbedded to my supervisors Docent Markus Fiedler and Doctor David Erman. I appreciate the time they have invested in me to improve my researchers skills on this escapade. My fellow Ph.D. students and colleagues at Blekinge Institute of Technology (BTH) I also thank for the many discussion and our happy moments together.

Moreover I want to thank my whole family, although thousand kilometers away and located on different continents, their unconditional support and love has never fainted. Also much appreciation goes to my lovely wife whom I adore unlimited, thank you for baring with me during these sometimes exhausting years.

February 2012,
Karel De Vogeleer
Contents

1 Introduction 1
  1.1 Motivation ................................. 2
  1.2 Research questions ......................... 5
  1.3 Related Work ............................... 6
  1.4 Thesis outline .............................. 8

2 Background 11
  2.1 Video streaming ............................ 11
    2.1.1 Jitter buffer .......................... 13
    2.1.2 The jitter buffer fluid flow model .... 15
    2.1.3 Video transmission .................... 17
  2.2 Robust statistics .......................... 18
  2.3 Machine learning ........................... 20
  2.4 Summary .................................. 21

3 QoS Parameters 23
  3.1 Introduction ............................... 23
  3.2 QoS metrics ............................... 25
    3.2.1 Delay variation ......................... 26
    3.2.2 Zero throughput time ................... 27
    3.2.3 Packet loss ............................ 28
    3.2.4 Inter-loss distance ..................... 29
    3.2.5 Reordering ............................. 29
    3.2.6 Clumping ............................... 30
  3.3 Measurement under unreliable conditions .. 31
  3.4 Smoothing of measurement data .............. 33
  3.5 Summary .................................. 34

4 Multidimensional Statistical Methods 35
  4.1 Estimation of location and scatter .......... 35
    4.1.1 Estimation of location .................. 36
B List of Publications
  B.1 List of publications ......................................... 99

Bibliography ................................................. 101

Abbreviations ............................................. 109

Index ......................................................... 111
## List of Figures

1.1  Situation of the Quality of Delivery concept in the overall picture of video quality impairments ................................................. 2

2.1  Four different states of the jitter buffer ................................................. 14

2.2  97.5\% tolerance ellipses for the Packet Delay Variation and the Zero Throughout Time ......................................................... 19

3.1  Examples of the clumping effect: contraction and dispersion in packet streams .......................................................... 30

4.1  The One-Way Delay of operator A and its estimate of location and dispersion for different types of estimators with regards to the packet size ............................................................ 41

4.2  Exemplification of the difference between the Euclidean and the Mahalanobis distance ......................................................... 43

4.3  Example of a linear separable data set with a separating hyperplane and its support vectors .............................................. 45

5.1  Difference between the sampling time of video frames and the associated departure time ...................................................... 53

5.2  Efficiency of the delay variation estimation for the measurement algorithms compared to the results of ping .............................. 56

5.3  Efficiency of the packet loss estimation for the measurement algorithms compared to the results of ping ........................................... 57

5.4  Efficiency of the reordering rate estimation for the measurement algorithm compared to the estimation via tcpdump ............. 58

6.1  Distribution of the Delay Jitter for measurements over both W-CDMA and WLAN networks ................................................. 63

6.2  Distribution of Zero Throughput Time for the WLAN and W-CDMA measurements .......................................................... 64
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>CCDF of Zero Throughput Time for WLAN and W-CDMA measurements</td>
<td>65</td>
</tr>
<tr>
<td>6.4</td>
<td>Distribution of the Packet Arrival Rate for various video bit rates</td>
<td>68</td>
</tr>
<tr>
<td>6.5</td>
<td>Cross-validation of the Mahalanobis distance for WLAN and W-CDMA,</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>and Support Vector Machine for WLAN</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>Cross-validation of the precision and recall for Support Vector</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Machines – W-CDMA and WLAN</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>Effect of the smoothing coefficients on the precision and the recall</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>for EWMA and LMA</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>Illustration of the decision boundary of the Mahalanobis distance</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>and SVM classifier for the scenarios {D_i,T_Z} and {PAR,T_Z}</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>Illustration of the decision boundary of the Mahalanobis distance</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>and SVM classifier for the scenarios {PAR,D_i} and {PAR,T_Z}</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

5.1 Summary of the fixed and variable parameters during the video streaming experiments ................................... 51
5.2 Summary of the simulation settings for the validation and verification of the QoS metrics assessment algorithms ............. 54

6.1 Recorded number of jitter buffer starvations for the W-CDMA and WLAN measurements .................................... 62
6.2 Mean Delay Jitter for the W-CDMA and WLAN measurements ................................................................. 62
6.3 Distribution of Clumping for WLAN and W-CDMA measurements ............................................................... 65
6.4 Total number of measured packet losses for both the W-CDMA and WLAN networks ............................................. 66
6.5 Total number of recorded reordered packets for both W-CDMA and WLAN networks ............................................. 67
6.6 Precision and recall ranking of the Mahalanobis distance for the W-CDMA and WLAN data set smoothed with LMA and EWMA ................................................................. 71
6.7 Precision and recall ranking of SVM for the W-CDMA data set smoothed with LMA and EWMA ............................... 72
6.8 Precision and recall ranking of SVM for the WLAN data set smoothed with LMA and EWMA ............................... 73
6.9 Presence of the QoS metrics in the top ranked scenarios for the Mahalanobis distance and SVM ............................... 80
Chapter 1

Introduction

*A new type of thinking is essential if mankind is to survive and move toward higher levels.*

*Atomic Education Urged by Einstein - New York Times*

High demand for multimedia rich applications in contemporary Internet services encourages the development of good performing networks by using advanced mechanisms [83]. Wireless technologies are expected to provide better performance as the engineering and research process is pushed to its limits. The growing market share of smartphones and the never-ending demand for more voice and data bandwidth instigates a need for *seamless communication* in a way that has not been exploited before. Among others, this demands for Quality of Service (QoS) assurance, service portability, and application persistence across heterogeneous wireless networks. The network’s quality needs to be monitored to achieve seamless communication and changes should be anticipated accordingly such that the user’s experience can be maximized.

Measuring the performance of a link or an Internet service in terms of QoS is an integral part of quality assurance for user satisfaction during seamless communication. It is thus important that the QoS is quantified correctly and, even more importantly, interpreted correctly. Failing to do so may result in lesser seamless communication. Therefore, it is crucial for decision-making processes or other quality monitoring and prediction systems to use proper QoS analysis tools with high fidelity.

An approach is advanced for end-to-end QoS measurement inference with a focus on the transmission of variable bit rate video. The inference is based on a multidimensional view of the QoS metrics, i.e., the metrics are treated simultaneously not separately. Performance reports show that today video contributes a large amount to the Internet’s traffic and it is likely to even grow in the future [17]. Optimizing the user experience of Internet streamed videos is thus
an interesting area of research. The presented analysis also aims at contributing to the understanding of QoS on the Internet in the context of streaming video. The foundations of this work were laid while conducting research as part of the STREP FP7 PERIMETER project [90, 106]. PERIMETER’s main objective was to establish a new paradigm for user-centricity in advanced networking architectures to enable seamless communication driven by user preferences. PERIMETER tackled the seamless communication concept from a user-centric point of view where the user’s preferences, network measurements and experiences are taken into consideration for roaming decisions. This is in contrast with today’s subscriptions where a user is bound to its Service Provider (SP) and it is governed by business considerations rather than by the user’s actual needs.

1.1 Motivation

The user experience of Internet streamed videos can be affected by many factors. Figure 1.1 shows an overview of the causes and effects of video quality impairments as originally presented by Zepernick et al. [127]. The user experience is depicted on top and referred to as Quality of Experience (QoE)\(^1\). Before a video is displayed to a user, the video passes many processes between capturing the image and the decoding of the video signal that may affect the

\(^1\)We refer to QoE as per the International Telecommunication Union ITU P.10/G.100 definition: *the overall acceptability of an application or service, as perceived subjectively by the end-user* [60]. Alternatively, the definition of Nokia can be used: *how a user perceives the usability of a service when in use - how satisfied he or she is with a service* [85].
video stream. These processes include, but not limited to, the capturing, encoding, transmitting, decoding, and displaying of the video images. During any of these steps quality loss may occur that can potentially affect the QoE of a user using a video service. Zepernick et al. distinguishes two general types of causes affecting video quality that can be categorized by Quality of Delivery (QoD) and Quality of Presentation (QoP). QoP issues result in spatial degradation of the video quality whereas QoD issues manifest as temporal artifacts. For example, video codec, bit rate, loss recovery techniques are examples of application layer factors that may affect the QoP, whereas delay jitter and packet loss are examples of network factors that could potentially affect the QoD. This work focuses mainly on pauses and breaks that may occur during the playback of a video. Other factors and their effects are also depicted in Figure 1.1.

An important goal of the PERIMETER project was to minimize deterioration of the QoD in scenarios involving handovers between heterogeneous networks [24, 26, 92]. Another important goal was to derive QoE models for video streaming over wireless links [52, 53]. In these models, QoE parameters were directly mapped to network level QoS metrics, circumventing so the separation of QoP and QoD. The QoE models used in these experiments were of low complexity, namely exponential and power laws over single QoS metrics, e.g., delay variation and packet loss. For audio streaming and web browsing, these simple QoE models seem to perform well [49, 108]. For video streaming however, it was observed that these models do not yield accurate results.

The initial efforts in QoE/QoS modeling were directed towards modeling a truncated reality, i.e., single QoS metrics. In anticipation of more descriptive QoS indicators, the research was extended with QoS analysis within a multidimensional setting to reveal a better understanding of the underlaying network. To address the approach of multidimensionality, research on dimension reduction techniques was done with a focus on the Mahalanobis distance [27, 28]. The goal was to extract network information from video stream measurements that only exhibits within a multidimensional QoS analysis. The hypothesis was that unfavorable network behavior is one of the causes of user experience degradation that manifests through outlying QoS measurements. For example, in a packet switched Internet a lack of QoS guarantees such as fluctuating bandwidth can degrade Internet streaming significantly [42]. In this case, resource demanding media types, such as high-definition television (HDTV), are particularly vulnerable. This concept was further mapped to the human perception of the video streams on mobile phones over wireless links. Promising results were obtained by using the Mahalanobis distance. Among others, it was possible to identify a threshold for problematic network conditions in 3rd Generation (3G) mobile networks. It is however important to point out that the approach of mapping QoE to network level QoS is opportunistic as the model is potentially intractable when spacial factors also affect the video quality. To gain a better understand-
ing of the effects on the video quality by network level impairments, focus must be laid on the consequences for the QoD more closely.

The prospect is that a more elaborate study of the QoD in terms of a multidimensional analysis of QoS can, in general, disclose more information about a network compared to a one-dimensional analysis. In particular, multidimensional or multivariate data can reveal outlying data much better compared to what a single parameter analysis can achieve. Under the assumption that a network provides on the average good QoS, these outliers are indicators of temporal changes in network conditions. The data representation at the application layer could potentially be adversely affected by outliers. Previous work on the Mahalanobis distance [27, 28] showed that QoS outliers affect the application layer and that this influences the user experience adversely. The multidimensional analysis approach towards QoD advanced here is directed at the identification of outlying QoS data as an indicator of performance degradation. For this purpose the Mahalanobis distance is used again. Besides that, the goal is also to identify the Mahalanobis metrics that are most descriptive in the context of QoD.

For effective outlier detection robust statistical methods are used. Robust statistics is a branch of statistics that focuses on adapting statistical methods to cope with outliers or, more generally, contamination [75]. Besides that, the robust statistical methods provide tools to identify and classify the outlying data. The robust statistical methods are used to detect influential QoS conditions on the QoD of variable bit rate video. A wide array of robust statistical methods exists and frequently used especially in the field of chemometrics and econometrics [29]. Both research fields often produce measurements of high dimensionality with few samples, so robust methods are of vital importance [22]. Robust statistics often appears to increase the reliability of measurements in other research domains as well. Robust statistical methods should however be applied in the right context to be effective.

Throughout this work a robust multidimensional analysis technique to QoS measurements is proposed based on the Mahalanobis distance. Additionally the performance to another method, which does not focus on outlying objects, is collated. The performance of these two techniques are compared as an indicator for network performance deterioration. The following techniques are studied and applied:

- **Mahalanobis Distance** – is a distance metric that measures the dissimilarity between two vectors from the same (multivariate) distribution [73]. As observed from previous research, it is expected that Mahalanobis distances larger than a particular $\chi^2_p$ quantile correlate to problematic network conditions. With focus on QoD, a continuation of the work in the previous CTRQ paper [27] is done.

- **Support Vector Machines** – seek a hyperplane to separate different class-
es [117]. The Support Vector Machine (SVM) would have the ability to classify measurements into a group that is related to QoD impairments and another group that is not.

The QoD is measured by the state of the *jitter buffer* of the streaming protocol. Before streamed media is processed by the decoder at the receiver’s side, the data is buffered. For different types of real-time applications the queue length may vary. The QoS measurements are mapped to the states of the jitter buffer. The target is that, via the Mahalanobis distance and SVMs consistency is found in the mapping of the QoS metrics to the buffer states. If such consistency can be found, then this knowledge is useful for devices to estimate the QoD based merely on end-to-end measurements. A case study is presented on the QoD of a jitter buffer while streaming a variable bit rate video (without audio support). The QoS indicators are matched to events in the jitter buffers and a relationship is deduced. The event of concern is an *empty buffer*, or *starvation of the buffer*. Empty buffers may result in the freeze of the video during playback, but the actual effect depends on the behavior of the used video decoder. Empty buffers may thus induce temporal effects that affect the user experience. The effectiveness of the classifiers are measured with the so-called *precision* and *recall*. The methods and QoS indicators are defined to be effective if the precision and recall is above a certain threshold. This implies that the used methods and QoS metrics reveal a strong relation to critical events in the jitter buffer. As a result, it is expected to be able to infer problematic network conditions with multidimensional QoS indicators in a more efficient way than with single QoS metrics.

The multidimensional QoS analysis is expected to reveal a better understanding on seamless communication of Internet services, in particular streaming video. The outcomes of the research is assumed to be useful for the interpretation of QoS measurements in contexts like cognitive radio, handover decision making, performance analysis, or Always Best Connected (ABC). The analysis approach is valuable in contexts where the QoS of a link or service needs to be monitored in real-time from an end-to-end perspective, to identify quality deterioration, preferably at an early stage.

### 1.2 Research questions

The following research questions are addressed with the aim of finding a set of descriptive indicators that govern the QoS in a network with a focus on QoD and the media buffers of streaming video protocols:

- What QoS metrics are essential in the study of QoD for streaming video?
• Can QoS indicators be found that are more descriptive than the conventional QoS variables via outlier analysis such as the Mahalanobis distance?

• Do network types like Wireless Local Area Network (WLAN) and Wideband Code Division Multiple Access (W-CDMA) show different behavior with regards to outlying QoS data?

• Is it possible to infer the state of jitter buffers, such as buffer exhaustion, from network level measurements?

• Are the suggested techniques deployable in a real-time environment with limited resources?

The answers to these research questions are obtained by measuring QoS metrics and the jitter buffer’s state for streaming videos with varying settings like, e.g., bit-rate and resolution. Robust statistical methods and the SVM algorithm is then applied to the QoS measurements to predict the state of the jitter buffer.

1.3 Related Work

In the field of networking, multidimensional QoS analysis has not yet received much attention with regards to quality purposes. Although, multidimensional reduction techniques are frequently applied to network related problems, linear Principle Component Analysis (PCA) seems to be a popular choice. The Mahalanobis distance is especially used in contexts where reliability plays an important role and cross-correlation of variables is of importance.

Chen et al. [15] studied the relationship between user satisfaction and packet latency aspects via PCA in a General Packet Radio Service (GPRS) network. The authors defined a user experience metric based on the first principle component of the PCA analysis. Correlation figures between the experience and the packet latency aspects are reported. However, no thorough analysis is presented regarding the PCA results.

Ramirez-Velarde et al. [95] suggested that PCA can help in reducing dimensionality in computer related modeling of, e.g., user preferences and behavior, as well as resource popularity.

A series of papers elaborate on anomaly detection in networks to identify security breaches with the help of network level QoS. Lakhina et al. [67] and Hakami et al. [44] expound on diagnosing anomalies in traffic flows in fixed networks and wireless networks, respectively. The authors employ PCA to identify two disjoint subsets in high dimensional QoS data for normal and anomalous network conditions. QoS metrics and traffic flows are measured and analyzed.

\[\text{The first principle component is the component with the largest eigen value in the PCA analysis.}\]
to detect anomalous network behavior like, e.g. Denial-of-Service (DoS) attacks, port scans, and signal jamming. The authors claim that they accurately detect anomalous behavior and false alarms are minimized.

Yoshihiro et al. [57] suggests a mapping between application-level QoS and user-level QoS. The authors measured nine metrics for media synchronization quality for audio-video transmission and reduced its dimensions first with PCA. A Multiple Regression Analysis (MRA) was applied to the reduced dimensions to deduce a relationship with Thurstone’s law of comparative judgment. The authors report that they could accurately estimate the “user-level QoS” via their MRA.

Roy et al. [101] used the Mahalanobis distance to model multidimensional QoS assurance to provide eventually QoS guarantees to ubiquitous users by Grid Service Providers (GSP). The aim of the work was to create game-theoretic models to maximize revenues of GSPs and provide differentiated QoS to ubiquitous applications. The model is theoretically expounded and a simulation study is provided.

With the aim of creating robust computing systems, Eslamnour et al. [34] suggested to quantify the robustness of resource management systems, with the eventual objective to generate a robustness metric for a given computing system. Based on the Mahalanobis distance, a robustness evaluation of system configurations was performed. Mahalanobis was shown to perform better than other conventional metrics in distributed computing systems.

Schneps-Schneppe et al. [103] studied the design issues of Service Level Agreement (SLA) agreements in 3G networks. The authors presented a theoretical framework for a SLA classification method based on the Mahalanobis distance. The classification method is intended to detect SLA violations. The theory is applied to a real company’s SLA achievements and it was shown how SLA violation may result into penalties.

The presented work on PCA and the Mahalanobis distance uses these techniques to measure the performance as QoS and other metrics in networking environments. In general, the referenced material reports that the use of these techniques increased the reliability and robustness of their objectives. Although these papers take some QoS parameters into consideration, the aim of their QoS analysis differs from ours. The concern of this work is the quality of an Internet multimedia service while the referred papers have focus on security, SLA violation detection, or revenues and reliability maximization. However, we aim to contribute in a similar fashion to the analysis of QoS via the multidimensional approach.

Furthermore, experimental studies of QoD with regards to QoS parameters are not abundantly available. Though, theoretical studies are conducted that are devoted to the analysis of jitter buffers and streaming server buffers. Fiems et al. [36] developed an analytical model of the output buffer of an on-demand
video server using a queueing model with dependent transmission times. Using probability generating functions, the performance was measured with regards to packet loss ratio and average frame transmission time. Idle and busy periods of the buffer were the main focus of their work. Fiems et al. showed that a reduction of transmission times can lead to lower values of the frame loss ratio.

ParandehGheibi et al. [89] studied the trade-off between QoE metrics, the initial start-up latency and the jitter buffer starvation probability. The analysis was placed in a Peer-to-Peer (P2P) scenario where a random linear combination of packets is received. ParandehGheibi et al. showed that this simplifies the selection strategies of P2P systems and reduces duplicate packet reception.

Similar to the work of ParandehGheibi et al., Liang and Liang [69] developed an analytical model of jitter buffer exhaustion based on a Markov Variable Bit Rate (VBR) channel model. The model was studied under different buffer sizes at the receiver, initial start-up delays and video freeze recovery schemes. Simulations were conducted over wireless links with the extended Gilbert loss model. The authors claim that their analytical model quantifies the tradeoffs between the start-up delay, buffer size, and video freeze frequency.

The referred articles approach the QoD from a theoretical point of view. In their frameworks simplified models of data transmission over the Internet are used. On the other hand, we are studying the notion of QoD from a more practical point of view. The goal is to identify problematic network conditions in the context of QoD in contrast to theoretical bottlenecks. Real-world experiments are used not only to capture the behavior of wireless technologies but also to identify QoS metrics that are useful in the definition of QoD. The focus is however not on the optimal start-up delay and buffer size.

1.4 Thesis outline

The remainder if the thesis is organized as follows. Chapter 2 provides a background on video streaming and the notion of the jitter buffer. The concept of robust statistical methods is introduced as well. An elaboration on how robust statistical methods can potentially increase the reliability of measurements is given. Chapter 3 presents an overview and defines the QoS metrics that are used during the analysis. The calculation of the metrics under various network conditions is described. Chapter 4 expounds on the methods used to analyze the QoS measurements. A theoretical background is provided on the Mahalanobis distance and SVM, their formal definitions and how to produce models. The usability of the techniques in real-time resource-constrained environments is developed as well. Chapter 5 presents the experimental environment used to measure the QoS and captures the buffer states for our multidimensional anal-
ysis. The scenarios are described under which the data is acquired. In chapter 6 the analysis of the measurements are reported. Both the Mahalanobis distance and SVMs are applied to our QoS measurements. Furthermore, a comparison is done between the QoS of the different wireless internet access technologies used. Chapter 7 concludes the work where the conclusions are listed and future directions of research are put forward.
Chapter 2

Background

It requires a very unusual mind to undertake the analysis of the obvious.

Alfred North Whitehead

In this chapter the basic concepts used in the multidimensional QoS inference for video streaming are introduced. An overview of video streaming and the Real-time Transport Protocol (RTP) is provided. Next, the main aspects related to the jitter buffer in a video context is presented. The notion of machine learning is introduced as well. The concept of robust statistics is finally demystified with a brief introduction and some examples.

2.1 Video streaming

Video-on-Demand (VoD) services are very popular among Internet users. Companies such as Netflix, Youtube and others, have been successful in disseminating video content to any customer, irrespective of the customer’s geographical location. Besides VoD, video streaming over the Internet is increasingly used in applications such as remote surveillance, security monitoring, smart home, environmental tracking, battlefield intelligence, distance learning, collaboration and interactive virtual environments [47, 128]. Cisco Systems claims in its annual networking index of 2010 that back then 40% of Internet’s traffic is related to video [17]. By 2014 Cisco Systems forecast that more than 90% of Internet’s traffic will be originating from video services, of which 13% will have its origin in real-time video applications. Cisco Systems also predicts that up to 66% of the world’s mobile data traffic by 2014 will be video. Besides the growing demand for VoD, traditional video providers adopt emerging Internet Protocol (IP) technologies for the delivery of video content, which contributes to the growing IP
traffic [86].

Today, a wide array of video streaming technologies exist. Both open-source and proprietary solutions are available. RTP is one of the most popular open standard streaming techniques [42]. The 3rd Generation Partnership Project (3GPP) Packet-switched Streaming Service (PSS) [1] puts RTP forward as the main component for the transport of media streams. Adobe’s HyperText Transport Protocol (HTTP) Dynamic Streaming System [2] is an example of a proprietary streaming solution and it is used in, e.g., Akamai’s High Definition (HD) network [3]. Real Networks [96] are pioneers in streaming multimedia and offer proprietary solutions such as RealVideo. YouTube [126] encodes the videos in a Flash Video (FV) container for playback in Flash enabled browsers and it uses RTP for its mobile variant. In the following, the focus will be on RTP as it is an open standard and knows wide adoption in all kinds of environments.

The Real-time Transport Protocol (RTP) is defined in RFC 3550 and it is a common choice to transport multimedia streams over the Internet. RTP provides loss detection, synchronization and sequencing services as well as security [46]. Designed by the Internet Engineering Task Force (IETF), RTP provides end-to-end network transport functions suitable for applications transmitting real-time data (such as audio, video or simulation data), over multicast or unicast network services [104]. RTP does not provide any QoS guarantees however. Each RTP stream maintains a queue, the jitter buffer, in which it buffers incoming data. Every RTP packet is labelled with a sequence number and timestamp. The incoming data in the queue is arranged such that the data assumes the same sequence as it was originally coded on the streaming server. This is essential for the playback of the media. The jitter buffer is discussed in more detail in section 2.1.1.

The Real-time Transport Control Protocol (RTCP) is used in connection with RTP to create a closed-loop feedback system between the media-player and the streaming server. RTCP disseminates periodic statistical performance measures between all participants in the RTP session, including all active media players and the streaming server. The primary function of RTCP is to provide feedback on the quality of the data distribution [104]. Based on this information, the RTP streaming server may optimize the performance through adaptive encoding if supported. Other parties can use this feedback to detect and anticipate network problems. The RTCP feedback is also used for congestion and traffic control purposes, although RFC 3550 doesn’t state explicitly how to achieve this. Congestion control in RTP is not always straightforward as RTP streams are often inelastic, i.e., generated at a fixed or controlled rate [104].

While RTP is used to transport time-sensitive data the stateful Real-time Transport Streaming Protocol (RTSP) is used to control the on-demand delivery of real-time data [105]. RFC2326, which describes RTSP, defines request messages such as play, stop, pause, and describe. Most RTSP messages are sent by
the client to the RTP streaming server. RTSP shows overlapping features with the Session Initiation Protocol (SIP) and the Session Description Protocol (SDP). SIP/SDP is however used to establish multimedia sessions whereas RTP addresses streaming media systems [70].

The Darwin Streaming Server (DSS) [4] is an open-source streaming server released under the Apple Public License Software that shares the same code base as Apple’s proprietary QuickTime Streaming Server (QTSS). DSS supports digital media standards such as 3GPP and Motion Picture Experts Group (MPEG) version 4 for streaming over RTP/RTSP. DSS provides Reliable User Datagram Protocol (UDP) support. Reliable UDP is an extension to the default UDP that enables Transport Control Protocol (TCP)-like features such as retransmission and congestion control but also properties such as overbuffering\(^1\) [5]. Reliable UDP is enabled by setting the proper fields in the RTSP SETUP request during client-server negotiation. Major content distribution companies such as Akamai [3] and Apple TV have DSS/QTSS based solutions for their video services. The DSS is available online from Mac OS Forge [4] and it can be installed and launched under Linux and Mac OS.

### 2.1.1 Jitter buffer

In streaming media applications, the jitter buffer, or play-out buffer, is a queue at the receiver’s side that temporarily buffers data frames before these are processed by the media decoder or media player. The motivation behind the deployment of a jitter buffer is to mask the effects of fluctuating network QoS on the delivery of media. The length of the jitter buffer varies depending upon the particular application. For example, for Voice over IP (VoIP) and video conferencing applications the queue length is minimized to reduce the communication latency. In VoIP settings the jitter buffer length varies around 15 ms with a maximum of 30 ms [112]. For other video streaming applications, e.g., video streaming in browsers such as YouTube, the queue length is the size of the media itself. In such cases, the necessary storage space should be available beforehand. For desktop computers and laptops this is usually not a problem, but hand-held devices are often constrained in memory space. The RTP implementation available in Android 1.6 often allocates a buffer size of around 170 kB. When using the DSS and Android 1.6 together, a buffer usage of about 60% is targeted\(^2\). The actual buffer usage shows a standard deviation around 25 kB, peaks larger than this are not unusual. This behavior is inherent for VBR video and fluctuating network resources.

\(^{1}\)Overbuffering is referred to as a situation where the streaming rate is faster than the playback rate. Browser embedded players often use overbuffering. Windows Media server for example names this as fast streaming [77].

\(^{2}\)At least when 3GPP-adaptation is enabled.
Chapter 2. Background

The media player can maintain several jitter buffers in parallel, depending on the amount of channels defined in the media file to be streamed. For a basic video stream, there is usually one channel for audio and a second one for video. Each jitter buffer can be considered as a G/D/1 queue. Here the data arrival process is assumed to be a general process, and the frames\footnote{Frames can be fragmented in different packets for transmission.} are consumed at deterministic time-intervals. This often resembles the frame-rate of the video or the sample-rate of the audio. Other research assumes M/D/1 queues which include Poison arrival rates [89].

The goal is to infer the state of the jitter buffer via network (multidimensional) QoS indicators. This is because the state of the jitter buffer is an indicator of the network layer’s performance and it has an effect on the media player’s ability to display streamed media in a timely fashion. Four types of states of the jitter buffer are distinguished: full buffer, empty buffer, critical holes and non-critical holes. An overview is shown in Figure 2.1.

A full buffer (Figure 2.1(a)) is defined to be a state when there is at least one media packet in the queue and all sequence numbers starting from the first packet after the last consumed, till the last received packet are present in the buffer. An empty buffer (Figure 2.1(b)) on the other hand doesn’t have any packet in the queue. Figure 2.1(c) shows non-critical holes. These are missing sequence numbers, which are not first in the queue. Last, critical holes are shown in Figure 2.1(d). Critical holes manifest when packets that are supposed to be first in the queue are missing.

It is clear that an empty buffer will temporarily halt the media playback. For video streaming the image on the screen could freeze or, in case of audio, the sound could stop. The behavior of the play-backed media in such cases is depended on the used decoder. Similarly, a missing packet, i.e., critical hole, may effect the media quality. Depending on the codec of the streamed media, the media quality may be impaired in different ways, e.g., for video: color changes or blurred pictures, short break. An empty buffer is considered to be...
more critical for the timely playback than a critical hole. A non-critical hole can become a critical hole when the non-critical hole progresses to the first place in the queue. The non-critical hole could also turn into a full buffer when the missing sequence numbers arrive at a later point in time. This can happen as a result of reordering effects on the network or because of packet retransmission due to losses or corruptive communication. When the jitter buffer is empty for some time the streaming application may decide to temporarily halt the playback of the media. During this time, the jitter buffer has the opportunity to populate again while no frames are retrieved, this is also called rebuffering.

The focus is on the identification of empty buffers as this particular case has the most severe impact on the application layer. Empty buffers affect the temporal playback of the media. Critical holes may mostly affect the spacial quality of the media. In this context, the notion of empty buffers and critical holes is in line with the Continuity Index (CI) that was initially defined in the work of Erman [32]. The CI is interpreted as the percentile of packets, or pieces in BitTorrent terms, that meet their play-out deadline. The CI was previously used in research on the evaluation of swarm video streaming over P2P networks [32, 33].

### 2.1.2 The jitter buffer fluid flow model

The jitter buffer is modeled as a fluid flow model. For now, the media stream arriving at, and departing from the jitter buffer is considered to be a bit stream. Let \( \Psi(t) \) be the amount of data in the jitter buffer at time \( t \). \( \Psi(t) \) is the amount of downloaded data, minus the amount of data retrieved by the video decoder up to time \( t \). The rate at which \( \Psi(t) \) changes over time is given by

\[
\frac{d\Psi(t)}{dt} = \varphi(t) - \xi(t),
\]

(2.1)

where \( \xi(t) \) is the data rate at which the decoder retrieves data from the jitter buffer and \( \varphi(t) \) is the arrival rate of the data to the buffer. \( \varphi(t) \) is a random process equal to the transmission rate and depending on the transmission policy of the streaming server. \( \xi(t) \) is a batch process proportional to the frame rate of the video. \( \varphi(t) \) may be affected by the network if it is unable to sustain the transmission. Thus, during transmission \( \varphi(t) \) is modulated by events that can be characterized by QoS parameters.

If \( \varphi(t) = \xi(t) \) then \( \Psi(t) \) will be constant, hence the amount of data in the jitter buffer remains stable. When \( \varphi(t) > \xi(t) \), the buffer size grows and for \( \varphi(t) < \xi(t) \) the buffer size shrinks. Depending on the application, \( \varphi(t) \) is aimed to be larger than \( \xi(t) \) or equal by controlling the transmission rate. When no constraints exist on memory, in fast streaming scenarios such as watching a video in a browser, \( \varphi(t) \) is often much larger than \( \xi(t) \). Other applications target \( \varphi(t) = \xi(t) \), e.g., for media with unknown length or for resource-constraint
devices. In both cases \( \frac{d\Psi(t)}{dt} \geq 0 \) to sustain a seamless media playback from the jitter buffer. \( \frac{d\Psi(t)}{dt} < 0 \) for a considerable time might indicate unreliable network conditions. This will eventually lead to buffer underflow or exhaustion if the network performance doesn’t improve. Thus \( \frac{d\Psi(t)}{dt} \) indicates the ability of the network to deliver a media stream over a network in a timely fashion. \( \frac{d\Psi(t)}{dt} \) is defined as a parameter that describes the QoD of a video stream and henceforth referred to as \( \omega(t) \):

\[
\Psi(t) = \int_0^t \omega(t) \, dt + c \quad \text{and} \quad \omega(t) = \varphi(t) - \xi(t) \quad (2.2)
\]

where \( c \) is a constant that reflects the initial conditions of the buffer at \( t = 0 \). \( \Psi(0) \) is usually zero. This implies that no data is available for the decoder at the start of the media streaming. Based on equation 2.2, a jitter buffer starvation is formally defined as follows.

**Definition 1** When streaming a video with length \( \gamma \), an empty buffer, or jitter buffer starvation, manifests when the buffer on the receiver’s side contains no information:

\[
\Psi(t) = 0, \quad 0 < t < \gamma. \quad (2.3)
\]

Note that \( \omega(t) \) has a lower bound defined by \( \xi(t) \) given that \( \xi(t) \geq 0 \) as long as \( \Psi(t) \geq 0 \) and \( \varphi(t) \) is always larger or equal to zero. The RTP streaming server usually streams at the same rate as media is consumed by the media player for real-time streaming. Under ideal conditions \( \xi(t) \approx \varphi(t) \), hence the \( \omega(t) \) is close to zero. This holds for constant bit rate videos. For variable bit rate videos \( \omega(t) \) will vary more than for constant bit rate videos. But the average \( \omega(t) \) will be close to zero. Under certain circumstances, based on RTCP feedback, the transmission rate \( \varphi(t) \) may be increased or decreased temporarily, e.g., to avoid congestion on the network. Also traffic control is sometimes used to amend the transmission rate in real-time with the target of a constant jitter buffer population. This can be useful when streaming variable bit rate media.

The jitter buffer can starve because data isn’t transmitted at a minimal rate to maintain a certain population of the jitter buffer. On the other hand, starvation of the jitter buffer could be the consequence of packet losses on the path to the jitter buffer. These causes should be distinguished. They could potentially take place at the same time.

Also, there is a small lag between the actual start of the streaming and the start of the media consumption from the jitter buffer. This is to allow the jitter buffer to populate before the actual playback of the media starts. It has been shown that for a longer start-up latency, the chance of jitter buffer starvation decreases [69, 89].
2.1.3 Video transmission

A video encoder produces a video at a constant or variable bit rate. The bit rate of a Constant Bit Rate (CBR) video is fixed over time whereas a Variable Bit Rate (VBR) video exhibits a fluctuating bit rate. The produced variable bit rate is dependent on the codec and content of the video, and shows long-range dependent properties [11]. Scenes with lots of motion are often encoded with more bits to maintain the quality level of the images. The minimum bit rate for videophones is 16 kbit/s, HDTV requires 8 to 15 Mbit/s, the Blue-Ray optical disc format has a maximum bit rate of 40 Mbit/s.

Both TCP and UDP can be used to transport media. Some protocols allow to switch in between protocols during runtime on client’s request. TCP provides more features, compared to UDP, such as reliable end-to-end communication and congestion control. Sometimes an application layer process is programmed to provide congestion control for UDP as well as other features, e.g., retransmission. Reliable UDP used by the QuickTime Streaming Server (QTSS) is such example [5].

TCP’s congestion control is realized by altering the transmission rate of packets. While streaming media, the application layer has also other alternatives to control the transmission rate. For example, frames can be dropped before the actual transmission, this is referred to as stream thinning. Stream thinning results in a lower transmission rate without affecting the timeline of the media. This is possible because frames in a video stream maintain a hierarchical structure called the Group of Pictures (GOP). The structure of a video frame is a combination of I, P, and B frames. I frames are independent reference pictures, fully specified. P, and B frames can be seen as enhancement layers dependent on other frames. Scalable Video Coding (SVC), part of the H.264/MPEG-4 AVC standard, defines a video bitstream to constitute of one or more subsets of bitstreams (inclusion of P and B frames). When stream thinning is enabled one or more enhancement layers or bitstreams are pruned and, as a result the timeline of the media will not be affected. TCP’s adaptation of the packet transmission rate results in a stretch or contraction of the timeline as after all, the whole media file will be send. This may result in a jitter buffer exhaustion, or possibly the freeze of the video, if the frames did not arrive before their playout deadline. Due to the use of stream thinning, the streamed media is less prone to jitter buffer exhaustion compared to congestion control on the transport layer. However, as part of the media content is dropped, the quality of the video may degrade. But the advantage is that the media is less likely to freeze during play-back.

Multiple Bit Rate (MBR) [113] or Adaptive Multi Rate (AMR) encoding is another technique for real-time congestion control. MBR allows for dynamic switching between precoded videos with different bit rates showing the same
content. The 3GPP PSS proposes the use of MBR for adaptive streaming. As with stream thinning, switching between bit rates in MBR encoded videos is useful to adapt to the changing bandwidth constraints on a network. When bandwidth becomes more scarce, the MBR stream server switches to the video with a lower bit rate, resulting in a lower network load. Similarly, MBR can switch to a higher bandwidth when more network resources are available. Fast streaming is frequently used by web browser players. Fast streaming is a technique where the media file is delivered as fast as possible to the user in order to reduce the startup latency and to protect against negative effects of bandwidth fluctuations. Even if MBR, fast streaming and rate adaption are novel techniques, Guo et al. reported that they tend to over-utilize CPU usage and bandwidth resources to provide the user with a better experience [42]. Effective and efficient quality control of streamed media is not a straightforward task.

2.2 Robust statistics

In the conventional linear regression model, the sum of squared residuals is minimized. The standard deviation of the residuals or error terms are assumed to be constant and normally distributed. The treatment of random variables can be represented with the location model [75]:

\[ x_i = \mu + u_i, \quad i = 1, \ldots, n, \]

where \( \mu \) is the true value of \( X \) and the errors \( u_i \) are random variates from \( U \) that act additively. \( U \) can be considered as noise emanating from, e.g., natural deviations, measurement errors, environmental influences. The aim of statistical methods is to estimate \( \hat{\mu} = \hat{\mu}(x_1, x_2, x_3, \ldots, x_n) = \hat{\mu}(x) \) such that it is as close to the real value \( \mu \) as possible.

When a vector of random variates \( x_i \) show equal finite variance, the sequence of \( x_i \) is called homoscedastic [123]. Homoscedasticity and normality is the basic assumption for many classical regression models and other statistical methods. The assumption of homoscedasticity and normal distributed errors does, however, not always hold in reality. When this happens, classical statistical methods may yield biased results. Robust statistical methods can be used in this case to produce more accurate estimations.

Figure 2.2 shows an example of the difference in performance between a robust and classical statistical method. This is an excerpt from previous work on the application of the Mahalanobis distance to QoS measurements [27]. The theoretical \( \chi^2 \) 97.5% tolerance ellipse is plotted for the Delay Jitter (\( D_J \)) and Zero Throughput Time (\( T_Z \)) with a classical method based on the arithmetic mean, and a robust statistical method based on the Minimum Covariance Determinant (MCD). The left crosshair is the estimation of location with
the robust estimator, and the right crosshair is computed via the classical way. The ellipses try to capture the variance of the bivariate data. It is observed that the ellipse and the estimation of location for the classical method is inflated towards the outliers. The robust ellipse and estimation of location describes the bulk of the point cloud well, regardless of the outliers. The glyphs in the plot are related to the user’s perception during the experiment of concern. It is observed that the robust ellipse is able to separate the triangular points from the round points in the center of the point cloud with about 95% accuracy. The performance is different from the classical ellipse’s classification accuracy. In this context, the robust analysis revealed more information than the classical statistical procedure did.

Robust statistics can be considered as the stability theory of statistical procedures [45]. Robust statistics aims to minimize the influence of outlying data objects and construct models and estimates that well describe the bulk of the point cloud [22]. This is often achieved by down weighting outlying objects to minimize their influence on an estimator. The trimmed mean is a straightforward application of down weighting the $n$ most outlying objects to zero. In general, robust procedures not necessarily down weights outlying objects to zero. Robust statistics provide objective methods to select the objects to label as outlying and how to down weight them. Objective methods outperform...
subjective methods for identifying outliers, especially in small sample sets [45]. Besides poor effectiveness, there are other reasons to avoid subjective outlier detection methods. The classification of good measurement points as atypical must be minimized, outliers inherent to the observed phenomenon should not be removed, and, to assess the statistical behavior of subjective procedures is a challenge [75].

Robust statistical methods are of particular interest for the purpose of multidimensional QoS inference. Understanding the processes that govern the QoS on the network level is partly based on the interpretation and numerical study of QoS data exploration. The goal is to find out QoS indicators that identify the influential measurement points with regard to application layer performance of multimedia streaming. In Previous work on the Mahalanobis distance [27, 28] it was observed that the QoS values most outlying with regards to the measurement’s average state were most influential on the quality perception of streamed video. The plot of Figure 2.2 is part of the outcome of this study. To identify these leverage points (influential QoS measurement points), there is a need to apply proper statistical methods. For this purpose robust statistical methods are used. Chapter 4 presents a theoretical expound on the robust statistical methods used in this research.

A general introduction to robust statistics can be found in the works of Hampel [45], Hubert [50] and Shurygin [109]. Additionally, Maronna et al. [75] provide an elaborate background and a detailed overview of the latest evolutions in the theory and methods of robust statistics.

### 2.3 Machine learning

The objective of the multidimensional end-to-end QoS analysis is to deduce a set of rules that allows a device to estimate the QoD of a video stream based on network conditions. Accordingly, user experience deterioration can be anticipated. With the help of the Mahalanobis distance and SVM classifiers are established that can distinguish problematic network conditions from the acceptable conditions. These methods are dimensioned to perform optimally based on a training set consisting of QoS measurements. This process is also referred to as supervised learning in machine learning terminology. The scheme operates under supervision in the sense that the process learns based on the actual outcomes [124].

More formally, supervised learning is defined as the maximization of a score function to approximate the target function. The target function $f$ classifies each $x \in X$ into one of $Y$ classes. A supervised learning method searches a mapping function $h$, or hypothesis, from the hypothesis space $H$ that maps an input space $X$ to an output space $Y$ given a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. A score
function $k : X \times Y \to \mathbb{R}$ is selected from the score function space $K$ such that $h$ yields the highest scores for $y$:

$$h(x) = \arg\max_y k(x, y). \quad (2.5)$$

In this context, the multivariate QoS measurements are used as the input space (representing the QoD), which is mapped to the output space, defined by the state of the jitter buffer. A set of QoS metrics describing the network constitute the feature vector in machine learning terms, or the attributes of the learning process. A feature vector is a vector of values describing the object. The states of the jitter buffer are referred to as classes.

Proper dimensioning of the Mahalanobis distance and SVM is of vital importance for the success of the supervised machine learning and hence, to increase the reliability of the model’s ability to predict. In the case of the Mahalanobis distance this is achieved by the use of robust statistical methods. This produces estimators that are robust to outlying objects, as frequently occurs in QoS measurements. Chapter 4 applies the machine learning concept in the context of multidimensional QoS analysis.

### 2.4 Summary

In this chapter basic concepts of multidimensional analysis have been discussed. The notion of machine learning was briefly introduced and showed how the statistical analyses fits into the concept of supervised machine learning. Robust statistics have been introduced as a tool to improve the reliability of the measurement data. Robust statistical methods are expected to help in identifying problematic network conditions. The jitter buffer was described along with an overview of streaming video techniques. Four states of the jitter buffer have been identified which will be matched against QoS measurements in the experiments.
Chapter 3

QoS Parameters

We see only what we know.
Johann Wolfgang von Goethe

The notion of QoS is introduced in this chapter and an overview of some common QoS metrics are provided. Additionally, some new QoS metrics are discussed that were introduced in previous research on video streaming. The metric’s measurement requirements are described and their applicability in the streaming media context is discussed.

3.1 Introduction

The term Quality of Service (QoS) has been popular for more than 20 years. Though, there is little consensus of what QoS exactly incorporates [61]. The term QoS can be construed intuitively as a measure for how well a particular service performs [55]. In the context of video transmission QoS is used as the quality quantification of a communication channel in terms of measurable parameters meaningful to a computer network. QoS in the context of computer networks, however, has a wider variety of interpretations. For example, in traffic engineering, QoS also refers to the capability of service guarantee and the provisioning of network resources [37, 120].

Conventional QoS metrics can be measured and assessed from an end-to-end point of view. In particular, a set of QoS metrics is needed that are applicable and essential in the streaming media context. An overview of common QoS metrics include but is not limited to:

- **Latency** – also referred to as One-Way Delay (OWD), is the time taken for a data packet to travel from its source to destination [7]. Delay is induced by queues and transmission delays on the path of the packet.
• **Delay variation** – the OWD of packets commonly varies per packet [82]. The differences in latencies of consecutive packets is referred to as the Inter-packet Delay Variation (IPDV).

• **Reordering** – a stream of packets sometimes arrives at its destination in a different order than it was initially transmitted [93].

• **Errors** – or corruption, occur during transmission when the content of packets on the path change due to unforeseen circumstances.

• **Bandwidth** – knows several definitions but in this context bandwidth is defined to be the data rate or capacity of a communication channel [8]. The bandwidth availability of a channel may fluctuate over time.

• **Packet loss** – some packets may be lost during transmission. This can for example be the result of packet collisions, queue overflows, or physical damage to the transmission medium.

Any of the above phenomena can affect the quality of a multimedia stream when they manifest or fluctuate. Though, not all of the stated QoS metrics are easily measurable. For instance, to measure the OWD accurately, one needs synchronized clocks between sender and receiver [6]. The synchronization is not a simple task, special equipment or protocols need to be deployed to achieve high accuracy. For every day use, a accurate synchronization system is not feasible for current retail devices.

Measuring the number of erroneous packets is also not straightforward. Error control is done at various layers of the network stack. For example, every Ethernet datagram carries a Cyclic Redundancy Check (CRC) checksum, the Internet Protocol (IP) provides error control for the header, and TCP calculates a checksum for both data and header. UDP provides, similar to TCP, checksum capabilities, but this is not a mandatory feature. Depending on the policy of the protocols, erroneous packets are dropped or retransmitted. This happens transparently to higher layers in the network stack and it is therefore hard to track from the application layer. Consequently, the QoS metrics error and OWD are not viable metrics for our purposes.

Streamed multimedia needs some minimum bandwidth, constant or variable, in order to have a guaranteed quality level. If this bandwidth is not available, the streamed media may show quality deterioration, e.g., in the form of visual impairments. Measuring the bandwidth availability is difficult without perturbing the network. Currently, generating packet trains is the best known methodology to estimate the available bandwidth in high-speed networks [63]. This means that some data sequences are generated from one point in the network to another, to assess the end-to-end network characteristics. In this case,
3.2 QoS metrics

the measurements are called active measurements. This is in contrast to passive measurements that infers information at a given point in the network [94].

Additionally, the media arrival rate can be measured. The Packet Arrival Rate (PAR) is the a straightforward measure to quantify the arrival rate. PAR can be defined in bytes or packets received per time interval. PAR is however content dependent on the media of concern. A deviation in PAR for CBR media is more likely to be an indicator of changing network conditions than it would be for the VBR case. Though, extreme values for PAR in the VBR case might give insight in degrading network conditions as well.

The delay variation, reordering, and dropped packets are conventional QoS metrics that are easily measurable. The focus is therefore on these parameters and some derivatives. Reordering and packet loss can however potentially be masked by stateful protocols such as TCP. Stateless protocols, such as UDP, don’t interfere in the assessment of these QoS metrics as effects on the network can propagate to the application layer. To assess the delay variation, reordering, and dropped packets, one needs information given by the timestamp and the sequence number of every packet. This means that only passive measurement suffices. Application layer protocols such as RTP have dedicated header fields for the timestamp and sequence number, and consequently can be useful in the assessment of QoS metrics. Extending existing protocol headers with extra fields is also an option. The feasibility of extending the UDP header was shown in previous work [24, 52]. However, one must be careful in the sense that end-to-end quality metrics not necessarily yield the same estimates when measured on different layers in the network stack. The implementation of a network stack in many modern Operating Systems (OSs) involves multiple queues and can thus affect the end-to-end QoS on its way to the user space\(^\text{1}\). In the Linux OS for example, almost all network stack layers have their own buffers [19].

The computation of QoS metrics is not always straightforward and various definitions are used for the implementation of the metrics. A more in-depth study on some selected metrics is presented in the following.

3.2 QoS metrics

In the sequel the parameters delay variation, packet loss and reordering are developed. Also some alternative QoS metrics are presented that showed their usefulness in previous research on video streaming. Formal definitions of the QoS metrics are given and practical implementation issues are discussed.

\(^\text{1}\)User space is the portion of the memory that is allocated for user processes. In contrast to kernel space, which is assigned for kernel operations. The network protocol operations and conventional Network Interface Card (NIC) drivers are located in kernel space in the standard Linux kernel.
3.2.1 Delay variation

Different definitions of Delay Variation (DV) are circulating and standardized within the research community [30, 58, 59, 104]. DV is sometimes referred to as jitter, inter-arrival jitter, or packet delay variation [30]. RFC 5481 addresses the fuzzy definition of DV and puts forward the DV definition based on the IETF [30] and the International Telecommunication Union (ITU) [58] to be most appropriate for packets when referring to data streams: the quantification of a path’s ability to transfer packets with consistent delay. Two definitions of DV are mainly used in industry and academia [82]: the Inter-packet Delay Variation (IPDV), which describes the difference in consecutive OWDS, and the Packet Delay Variation (PDV), which compares the difference in OWD to a fixed reference point.

The actual computations of PDV and IPDV avoid the necessity to measure the OWD with synchronized clocks. Instead, the unsynchronized OWD $D_n$ of packet $n$ is measured and calculated as

$$D_n = T_{R,n} - T_{S,n},$$

(3.1)

where $T_{S,n}$ is the local departure time of packet $n$ at the sender’s side, and $T_{R,n}$ is the local arrival time of corresponding packet at the receiver’s side. The IPDV of packet $n$ is obtained by comparing the unsynchronized OWD of packet $n$ to the preceding packet

$$IPDV_n = D_n - D_{(n-1)}$$

(3.2a)

$$= (T_{R,n} - T_{S,n}) - (T_{R,(n-1)} - T_{S,(n-1)})$$

(3.2b)

$$= (T_{R,n} - T_{R,(n-1)}) - (T_{S,n} - T_{S,(n-1)}).$$

(3.2c)

To estimate the delay variation, the RTP RFC recommends the IPDV definition. The computation of the IPDV with reference to the previous packet (as in equation (3.2c)) can yield biased results. For example, a long lasting increase in OWD is only reflected in two IPDV estimate, at the start and the end. To minimize the bias in PDV's calculation, RFC 5481 recommends not to use the previous packet as a reference but instead to use $D_{\text{min}}$, the lowest $D$ measured. Equation 3.2a thus becomes:

$$PDV = D_n - D_{\text{min}}.$$  

(3.3)

In real-time environments $D_{\text{min}}$ is however not always known in advance. Given that $D_n$ might be available, e.g., at the start of a measurement, $D_{\text{min}}$ may change over time. To address this problem the PDV can be computed over a given time interval as the standard deviation $D_J$ of $D_n$ with reference to the first unsynchronized OWD in that particular time interval. $D_J$ is then defined as

$$D_J = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} (D_d - \overline{D})^2},$$

(3.4)
where $\overline{D}$ is the average unsynchronized OWD of the time interval, $D_d$ is the PDV with reference to the first packet of the time interval: $D_d = D_n - D_1$, where $D_n$ is the unsynchronized OWD of packet $n$. $N$ is the total number of packets within the time interval. Equation (3.4) in its current form requires to store all values of $D_d$ in order to compute $(D_{S,n} - \overline{D})^2$ at the end of the time interval. This might not be favorable or feasible for memory constrained devices. By rearranging the symbols under the root of equation (3.4) we obtain:

$$D_J = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} D_{d_n}^2 - \frac{N}{N-1} \overline{D}^2}. \tag{3.5}$$

As a result, for the calculation of $D_J$, one needs to maintain only three variables: $N$, $\sum D_{d_n}^2$ and $\overline{D}$. Clearly, this is more advantageous than maintaining an array of a priori unknown length for the computation of $D_J$ (as in equation (3.4)).

The $D_J$ calculation from equation (3.5) describes a Ring Buffer (RB) algorithm [56]. Non-RB calculation algorithms store historical values in a database and compute the estimates afterward. The RB algorithm does not need to store any historical timestamps or sequence numbers. Thus a RB is more efficient and desirable in resource constraint environments.

In the study by Verdoy [119] six different delay jitter methods were studied. The study pointed out that the $D_J$ from equation (3.5) is the optimal method with regard to packet inconsistencies, memory requirements and ease of deployment. An overview of other DV definitions is available in Verdoy’s work and RFC 3393 [30].

### 3.2.2 Zero throughput time

Previous experiments [52, 53] has shown the disadvantage of PDV being undefined in time intervals where no packets arrivals are observed. Hence, PDV does not adequately reveal what happens on the network level when packet arrival rates drop, especially in more critical situations. For example, when congestion occurs somewhere in the network, zero packets may arrive within a particular time interval. Although a serious network impairment is manifesting, the resulting PDV of that particular time interval is not defined. In the quest of finding more resilient delay variation definitions and better delay variation indicators, the Zero Throughput Time ($T_Z$) was studied [52].

$T_Z$ is defined as the time between the arrival of the last packet $T^N$ of the previous time interval $(i-1)$, and the arrival of the first packet $T^1$ in the current interval $i$ [52]. If no packet has arrive in the time interval $i$ ($N_i = 0$) then the current time is used as a reference and subtracted from the last known packet
arrival. \( T_z \) for interval \( i \) is then defined as:

\[
T_z(i) = \begin{cases} 
T_i^1 - T_{i-1}^N, & \text{for } N > 0 \text{ in } i \text{ and } i - 1, \\
T_{\text{current}} - T_{i-k}^N, & \text{for } N = 0 \text{ in } i \text{ to } i - k + 1,
\end{cases}
\]  

(3.6)

where \((i - k)\) is the interval where the last known packet arrival occurred before the interval \( i \). This definition allows \( T_z \) to span across multiple time intervals when no packets arrive, e.g., in the case of network congestion or severe packet losses. \( T_z \) provides thus additional information compared to PDV. \( T_z \) shows its significance for a continuous stream of packets, such as a video stream, in the event of exceptionally large packet inter-arrival times. For bursty traffic, e.g., web-browsing, \( T_z \) has a different meaning than in the streaming media case.

### 3.2.3 Packet loss

Packet Loss (\( P_L \)), or the number of dropped packets, is calculated as:

\[
P_L = \text{number of expected packets} - \text{number of arrived packets.} \quad (3.7)
\]

The sequence numbers of packets can be used to identify the packets. For real-time assessment of the packet loss, the last received sequence number is usually used as a reference.

When a packet trace is recorded and the whole trace is analyzed afterwards, applying equation (3.7) is straightforward. During real-time measurements, on the other hand, only the sequence numbers from the past are known, and the future arrival of sequence numbers is unknown. Identifying a missing sequence number can then have two causes, either the packet is really lost or the packet arrives in the future. The latter can happen because of reordering effects or because of the lost packet’s retransmission. In both cases, a missing sequence number is a false indicator for packet loss and should be addressed properly.

When the packet loss is measured for real-time purposes, the packet loss is often measured within a specific time interval. The packet loss rate is defined as the number of packets lost per time interval. It is easily observed that the chance for a false packet loss measurement, as described above, increases when the time interval, over which the packet loss is computed, decreases.

The RTP RFC considers packets lost as per equation (3.7). The packet loss is computed over the time between each transmission of receiver reports. Also, packets that were lost but recovered by retransmission don’t fall under the RTP’s perception of packet loss. RTP considers packet loss as persistent congestion whereas the delay variation is regarded as transient congestion. An overview of more definitions of packet loss is presented in RFC 3357 [66].
3.2 QoS metrics

3.2.4 Inter-loss distance

Packet losses occur sometimes in bursts. Quantifying the bursty behavior of packet losses is difficult to capture with a single metric such as the packet loss. Gilbert modeled packet losses by a simple model similar to a Bernoulli process [38]. The Gilbert model was suggested to be extended to a Markov model to better capture the bursty behavior or the temporal loss dependency [125]. The Inter-loss Distance (ILD) describes the distance of packet losses in terms of sequence numbers [66] and quantifies the temporal dependency of packet losses. A sequence of small ILD values indicate packet loss bursts. ILD is also referred to as inter-loss-period-length [66].

Measuring the ILD is quite straightforward, the number of packets between packet losses needs to be counted. A larger ILD implies better performance as less packet losses occur. However, if no packet loss occurs in a given time interval the ILD is not defined. Therefore we track the ILD in the form of the Inverse Inter-loss Distance (IILD). In real-time settings, it is more meaningful to use the IILD per time interval as IILD is always defined. The IILD for interval $i$ is defined as

$$IILD_i = \begin{cases} 0, & \text{for } L = 0 \text{ in time interval } i, \\ \frac{1}{\min(ILD_i)}, & \text{for } L > 0 \text{ in time interval } i, \end{cases} \quad (3.8)$$

where $L$ is the number of packet losses during time interval $i$. If a packet loss occurs, IILD is the inverse of the lowest ILD measured. This way the IILD yields 0 when no packet losses are measured. Consequently, poor performance with regards to packet loss is translated in an increased value of IILD.

3.2.5 Reordering

Reordering occurs when packets do not arrive in the same sequence as they are transmitted [93]. Reordering effects are claimed to occur naturally due to local parallelism in Internet components and links [10]. Packets can traverse the Internet over different paths or packets can be placed in different queues of the same network devices. In both cases there is a chance that the sequence of packets of the same stream is changed.

An arriving packet is labeled as reordered if its associate sequence number is smaller than that of the previously arrived packet [119]. For example, if a packet sequence is initially transmitted as $\{1, 2, 3, 4, 5, 6\}$, and received as $\{1, 2, 3, 5, 4, 6\}$, then packet 4 can be labelled as reordered. Other definitions and calculations of reordering also exist and are discussed in RFC 4737, e.g., reordering ratio and reordering density [81]. The most straightforward reorder-
ing definition is the reordering ratio \( RR \) defined as:

\[
RR = \frac{N_R}{N_T},
\]

(3.9)

where \( N_R \) is the number of reordered packets and \( N_T \) the total number of packets. Alternatively, the reordering density takes the distance \( R_E \) (in packets) into account over which a packet is reordered:

\[
R_E = S_E - S_R,
\]

(3.10)

where \( S_E \) is the expected sequence number and \( S_R \) the sequence number of the reordered packet. It is clear that \( R_E \) will increase the later a packet arrives when reordered. The \( R_E \) values from each packet are combined in a density plot and presented as the reordering density. Thus the reordering density, in contrast to the reordering rate, also provides some extent to which the packets are reordered.

When the reordering is measured per time interval this is referred to as the Reordering Rate (\( R_R \)). In this work and experiments, the reordering rate calculation from equation (3.9) is used. Içkin [51] claims that this definition is the most practical to implement and interpret. Although the reordering density is more descriptive, the histogram’s interpretation is more complex to automatize. Içkin also provides an elaborate study of other reordering techniques [51].

### 3.2.6 Clumping

Clumping (CL) is an effect that results in the alteration of the inter-arrival time of packets traversing a network. CL effects are mostly visible in packet streams. Clumping occurs in two forms and Figure 3.1 depicts both cases. Figure 3.1(a) shows packets that arrive at the receiver with larger inter-arrival times than the original associated inter-departure times at the sender’s side. In this case we
3.3 Measurement under unreliable conditions

refer to dispersion of packets. Figure 3.1(b) shows the opposite effect: contraction of packets. This means that the packets arrive with smaller inter-arrival times than their associated inter-departure times. For example, if a sequence of packets is queued and released simultaneously then the packets will most likely be contracted. This could potentially affect the playback of the media therein.

Identification of dispersion and contraction can be done via packet timestamp tracking. Consecutive inter-arrival times are compared to their inter-departure times, from this comparison an estimate of the clumping can be established [119]. Alternatively, the contraction can be measured as initially proposed by ˙Içkin [25], with a less complex algorithm. The time interval $t$ is subdivided into $i$ smaller sub-intervals $\Delta t_i$ and the number of arrivals per sub-interval is counted. Assuming that under regular conditions the average number of arrivals per $\Delta t$ will be constant, peaks in number of packet arrivals in $\Delta t_i$ are indicators of clumping effects. Clumping can then be defined as

$$CL = \max\{\Delta N_1, \Delta N_2, \ldots, \Delta N_n\},$$  

(3.11)

where $\Delta N_i$ is the number of arrivals in sub-interval $\Delta t_i$. When the size of $\Delta t$ approaches 0, CL becomes the number of packets that arrive at the exact same time instant, i.e., carrying the same arrival timestamp. The “same time instant” is dependent on the OS clock’s resolution. ˙Içkin showed that the $\Delta t \rightarrow$ clock resolution approach is feasible on Linux based devices with a millisecond resolution and correlates well with application layer events [25].

Varga [118] studied a similar metric, the kurtosis$^2$ of the Packet Inter-arrival Time (PIT). Varga observed that the PIT Probability Density Function (PDF) reveals distinct peaks, especially for low PIT values, under bottleneck conditions in a network. The kurtosis, as proposed by Varga, is meant to quantify the departure from an assumed uniformly dispersed PIT PDF and hence reveals the occurrence of peaks. The kurtosis is shown to correlate with the user experience on the application layer. The CL is in line with the suggestion to use the kurtosis for measuring the congestion in a network. Though, CL focuses only on the back-to-back packets, indicating that packets might have been queued in the network and released simultaneously.

3.3 Measurement under unreliable conditions

When network conditions deteriorate, tricky situations may arise, when multiple of the previously described QoS metrics manifest at the same time, such as packet loss and reordering. In this case, the effects of the different metrics can become difficult to distinguish and might impair the calculation thereof.

$^2$Kurtosis, is a statistical metric that describes the peakedness of a probability distribution of a real valued random variable.
Measuring the reordering rate and packet loss can be a delicate undertaken. Reordering can easily be mistaken for packet loss and vice versa. Two scenarios can occur. First, the reordered packet arrives in the same time interval where it originated. In the second case, the packet that is reordered arrives in a next time interval after which it was expected to arrive. In the first case, the reordering can be recorded adequately. In the second case, the reordering may be recorded too late. Or, if the packet arrives first in the next time interval, the reordering may not be detected at all. A false packet loss may be recorded instead. Similar scenarios can be sketched for the packet loss.

To address these issues, the following rule is used to distinguish between reordered packets and lost packets: if a missing sequence number arrives within four consecutive arrivals of other packets, the packet is considered reordered, otherwise the packet is considered lost [51]. In comparison, TCP uses a timer instead of a packet counter to decide from which point on data is considered lost. Waiting for four packets before the determination of reordering or loss induces a delay depending on the packet rate. RTP video streams however commonly transmit more than ten packets per second. As a result, waiting four packets in the case of 10 packets/s video stream yields an average measurement delay of 400 ms.

Furthermore, packet loss and reordering can bias the estimation of delay jitter. If $T_{S,(n-1)}$ and $T_{R,(n-1)}$ are not available in equation (3.2c) then $D_n$ and IPDV$_n$ cannot be calculated. The PDV and IPDV computation algorithms must be adjusted in order for packet losses and reordering not to take effect on the delay variation estimate. The calculation of the $D_J$ in equation (3.5) is not susceptible to missing or reordered packets as they can safely be ignored in the calculation. Because the $D_J$ estimation computes the average and squared sum of $D_d$, the $D_J$ is still defined when packets are missing. Ignoring lost packets in the computation of PDV is also put forward by RFC 3393 [30] and it is referred to as reducing the event space by conditioning$^3$.

The RTP implementation in Android 1.6, implemented by PacketVideo, ignores packet loss and reordering effects in the computation of the DV. PacketVideo computes the IPDV from equation (3.2c) as prescribed by the RTP RFC. Moreover, no reordering effects are recorded by the implementation. The RTP RFC doesn’t state anything about reordering. It is thus not surprising that PacketVideo doesn’t either.

Appendix A.2 shows code snippets that compute the discussed QoS metric algorithms. These snippets were used in the parsing tool that processed the measurement of the experiments. The implementation of the algorithms are partially based on Içkin’s QoS metrics assessment algorithms [25, 51].

$^3$RFC 3393: . . . conditional statistics are considered; namely the mean IPDV is estimated (or other derivative statistic) conditioned on the event that selected packet pairs arrive at the destination . . . [30]
3.4 Smoothing of measurement data

Before processing the measurements of QoS metrics it is not unusual to smooth or average the data, depending on the purpose of the measured parameters. Smoothing is commonly used to cancel out noise in time-series to reveal its true trend. For example, smoothing can minimize the noise factor \( U \) in the location model of equation (2.4). Smoothing can also be used to simulate more complex processes. For example, Exponentially Weighted Moving Average (EWMA) is frequently used to simulate human cognitive behavior such as the recency effect [78]. Averaging can also be used to simulate a delay in measurements. This can be useful to relate a response value to an input parameter when detecting level shifts that take some time to propagate through a system.

The simplest smoothing algorithm over a time-series \( x(i) \) is the Linear Moving Average (LMA), given by

\[
y(i) = \frac{x(i) + x(i-1) + \cdots + x(i-(n-1))}{n}.
\]  

(3.12)

In this linear smoothing algorithm, \( n \in \mathbb{Z}_+^+ \) is the smoothing coefficient. \( n \) can be seen as the number of historical observations used for averaging.

Exponential based moving averages are non-linear smoothing algorithms. The Exponentially Weighted Moving Average (EWMA) is a common exponential smoothing algorithm used in many fields of research, e.g., econometrics and signal processing. The EWMA \( y(i) \) of a time-series \( x(i) \) is computed as [84]:

\[
y(i) = (1 - \beta) y(i-1) + \beta x(i),
\]  

(3.13)

where \( \beta \in [0,1] \) is the smoothing coefficient. \( \beta \) is sometimes expressed as \( 2/(n+1) \), where \( n \) is expressed in units of time. The definition of EWMA in equation (3.13) can be shown to be the convolution of \( x(i) \) with a geometrical distribution, the discrete counterpart of the exponential distribution, and lends its name from this property. The EWMA is comparable to the analogue RC-filters, which are used in many signal processing applications.

The RTP RFC prescribes the use of EWMA in equation (3.13) with \( \beta \) set to \( \frac{1}{16} = 0.0625 \) to smooth its measured IPDV values. The RTP RFC claims that \( \beta = \frac{1}{16} \) yields a good noise reduction ratio while maintaining a reasonable rate of convergence. This was inferred from an analysis by Cadzow [14]. \( \beta = \frac{1}{16} \) seems to be inherent to a series of RFCs [12, 30, 41, 82, 110, 121].

One should not forget that there is a trade-off between response time and smoothing period. The larger the smoothing period, the longer it will take for a smoothed step function to reach a certain threshold. As a rule of thumb one can say that it takes EWMA \( n = 2/\beta - 1 \) time intervals to reach 85% of a step function. The response time may be too slow if \( \beta = \frac{1}{16} \) to anticipate poor QoS conditions for a one second interval in a real-time setting. Of course, the time
scale of the data to which the smoothing is applied plays an important role in the outcome of the smoothing. The RTP RFC suggests to use EWMA over each IPDV value while EWMA can also be applied to QoS measurements over given time intervals. This means that the timescale of the IPDV values in RTP is a few magnitudes smaller than for QoS metrics over 1 s time intervals. In previous research EWMA with $\beta$ set to 0.25 was used, which yielded favorable results in the context of audio and video streaming [16, 43, 114]. $\beta = 0.25$ for $D_j$ is similar to a $\beta = \frac{1}{16}$ for the IPDV in RTP when streaming videos at low bit rates: $\approx 5$ packets/s.

In chapter 6, both averaging in equation (3.12) and (3.13) will be used, and checked whether $\beta = 0.25$ or any other value is the optimal smoothing coefficient for EWMA in a streaming video context.

### 3.5 Summary

The end-to-end QoS parameters have been analyzed. Practical operation issues are discussed. It is shown that some QoS metrics are difficult to measure in an every-day environment. This is due to the need of special equipment or masking effects by the OSs. Other metrics were discussed that are of practical use. These include the conventional definitions of Delay Jitter ($D_j$), Packet Loss ($P_L$), Packet Arrival Rate (PAR), and Reordering Rate ($R_R$). Besides these, some alternative metrics are advanced, namely the Zero Throughput Time ($T_Z$), Clumping (CL), and Inverse Inter-loss Distance (IILD). These presented QoS metrics are used in the sequel.
Chapter 4

Multidimensional Statistical Methods

Quapropter bono christiano, sive mathematici, sive quilibet impie divinantium . . . cavendi sunt, ne consortio daemoniorum irretiant.

De Genesi ad Litteram – Augustine of Hippo

The chapter is about the statistical and decision learning methods to be used for the analysis of QoS measurements. An intuitive explanation of multidimensional QoS analysis is presented as well as the idea behind outlier identification. This is helpful in the understanding of the Mahalanobis distance statistics. A theoretical foundation of SVM is then presented.

4.1 Estimation of location and scatter

Consider a set of independent and identically distributed (i.i.d.) observations \( x_1, x_2, \ldots, x_n \in X \subset \mathbb{R}^1 \) with a density function given by \( f(x; \theta) \), which depends on a parameter of interest \( \theta \in \Theta \subset \mathbb{R}^1 \). The Maximum Likelihood Estimation (MLE) of \( \theta_0 \) is given by [109]:

\[
\hat{\theta} = \arg \max_{\theta \in \Theta} \prod_{i=1}^{n} f(x_i; \theta). \tag{4.1}
\]

The MLE in this equation boils down to solving:

\[
\sum_{i=1}^{n} \psi(x_i; \theta) = 0, \tag{4.2}
\]

where \( \psi(x; \theta) = \partial \rho(x; \theta) / \partial \theta \), \( \rho(x; \theta) = -\log f(x; \theta) \) and \( \theta \) is the parameter to be estimated [116]. \( \psi(x) \) is known as the score function whereas \( \rho(x) \) is also denoted...
as the *contrast function*. Estimators obtained via the MLE in equation (4.2) are sometimes referred to as M-estimators.

The estimation of location $\mu$ and the variance $\sigma$ are the corner stones in many multivariate data analysis techniques, including the Mahalanobis distance. The MLE function can be used to obtain an estimate of $\mu$ and $\sigma$. Estimators that produce estimates close to the true value turn out to be of vital importance for the QoS analysis. In the sequel it is shown how the $\mu$ and $\sigma$ can be made robust against departure of the normal assumption.

### 4.1.1 Estimation of location

The *arithmetic mean* $\mu$ is the most popular and intuitive estimate of location for a random variable $X$:

$$
\mu = \frac{1}{m} \sum_{i=1}^{m} x_i. \quad (4.3)
$$

Via the MLE it can be proven that the arithmetic mean is the theoretical estimate for $X$ if the random variable is normal distributed [75]. When the distribution of $X$ differs from the normal assumption, the arithmetic mean may not be the most efficient estimator of location. If, for example, 100 samples are drawn from a $\mathcal{N}(0, 1)$ distribution, the sample arithmetic mean will be close to 0. When an arbitrary large value is assigned to one sample, then this sample can dominate the arithmetic mean easily [123]. It is said that the arithmetic mean has a breakdown point of 0% [31]. The breakdown point of an estimator is the maximal fraction of phony objects in the data, that the estimator can handle without perturbation [22].

The *median* $\kappa$ is the solution of equation (4.2) for the double exponential or Laplace distribution $f(x) = e^{-|x|}/2$. The median $\kappa$ is defined as:

$$
\#(x_i > \kappa) - \#(x_i < \kappa) = 0. \quad (4.4)
$$

It is observed that the breakdown point of the median is 50%. Up to 50% of outlying objects may be introduced until the median becomes abstract. The median is thus a more robust estimate of location than the arithmetic mean. Kolmogorov recommended the use of the median for the center estimation of normal distributions in applications\(^1\) [109].

The trimmed mean is an example of an estimator with a variable breakdown point. The trimmed mean drops the $m$ most outlying points on both sides of a distribution. $m = [(n - 1) \alpha]$, and $\alpha \in [0, 1/2)$ is proportional to the amount of trimmed data. $\alpha$ of the trimmed mean can be shown to be the breakdown point.

\(^1\)Kolmogorov recommended this in 1950, long before the field of robust statistics was developed. Nowadays there are even better performing estimators available.
point [87]. For $\alpha = 0$, the trimmed mean becomes the arithmetic mean and $\alpha = 0.5$ yields the median.

The median and arithmetic mean are estimators of center for univariate distributions. The estimators can also be extended to produce estimates for multivariate distributions. In non-robust settings the estimation of center for multivariate data is often defined as the average object, the arithmetic mean of each dimension in the data set:

$$
\mu(x) = \frac{1}{n} \sum_{i=1}^{n} x_i
$$

(4.5)

This approach is however prone to outliers for the same reasons as the univariate arithmetic mean is. Finding the center of location can also be formulated as an optimization problem. A more robust estimate is the minimization of the Euclidean distances of all points in the data set to the center $\mu$, also know as the $L_1$-median:

$$
\min_{\mu_{L1}} \sum_{i=1}^{n} \|x_i - \mu_{L1}(x)\|,
$$

(4.6)

where $\| \cdot \|$ is the Euclidean norm. The $L_1$-median is any point that minimizes the sum of Euclidean distances from $\mu_{L1}(x)$ to all points in the data set $x$. The $L_1$-median exhibits a 50% breakdown point as opposed to the 0% breakdown of the average object method in equation (4.5).

Other common robust estimators of location exist as well, e.g., the Stahel-Donoho estimator and the $S$-estimates. Theoretical background of these estimators and others are found in the work of Small [111], Maronna et al. [75, 76], and Davies [23]. The effect of outliers on M-estimator and their robustness are extensively studied in the work of Huber [50] and Shurygin [109].

### 4.1.2 Covariance Matrix

In the computation of the Mahalanobis distance the estimation of scatter or dispersion represented by the covariance matrix plays an important role. The classical estimation of the covariance matrix $C$ for a multivariate data set $X = (x_1, x_2, \ldots, x_n)$ in $\mathbb{R}^{n \times p}$ is defined as:

$$
C = \frac{1}{m-1} \cdot (X - 1^T \mu(X))^T \cdot (X - 1^T \mu(X))
$$

(4.7)

The non-diagonal elements in $C$ are the pair-wise covariances of the variables in $X$ whereas the diagonal elements in the covariance matrix are the variance of the individual variables. The arithmetic mean and the variance, defined by

$$
\sigma^2 = \text{var}(X) = E[(X - \mu)^2],
$$

(4.8)
are used in equation (4.7) and are both prone to outliers. Consequently, the classical computation of the covariance matrix will be biased by outliers too. The covariance matrix can be made more robust by using the Mean Absolute Deviation (MAD) instead of the variance from equation (4.8). The MAD, also known as: median absolute deviation about the median, is defined by:

\[ s^*_k = \text{median} |x_i - \text{median}|. \]

To make the MAD consistent to \( \sigma \) for the normal distribution, \( s^*_k \) is often multiplied by the scaling factor \( c_k = 1.4826 \). This is more formally expressed as:

\[ s_k = 1.4826 s^*_k \rightarrow p \sigma, \]

where \( \rightarrow_p \) is interpreted as “tends in probability to”.

Alternatively, robust estimators can be used, which produce both estimations of center and scatter. The MCD is an example of such an estimator. In the QoS analysis the MCD will be used as a basis for the computation of the Mahalanobis distance. The MCD’s objective is to minimize the determinant of the classical covariance matrix over \( h \) out of \( n \) observations \[97\]. \( \alpha = h/n \) is considered to be the breakdown point of the MCD algorithm. The robust estimation of scatter \( S_{\text{MCD}} \) and location \( \mu_{\text{MCD}} \) over the \( h \)-sample subset \( \{x_{i1}, \ldots, x_{ih}\} \) is then given by:

\[ \mu_{\text{MCD}} = \frac{1}{h} \sum_{j=1}^{h} x_{ij}, \]

\[ S_{\text{MCD}} = c_{\text{ccf}} c_{\text{sscf}} \frac{1}{h-1} \sum_{j=1}^{h} (x_{ij} - \mu_{\text{MCD}})(x_{ij} - \mu_{\text{MCD}})^T, \]

where \( c_{\text{ccf}} \) is a consistency correction factor and \( c_{\text{sscf}} \) is a small sample correction factor introduced to make \( S \) consistent to the normal distribution and unbiased for small samples respectively \[20\].

The original version of the MCD by Rousseeuw \[97\] suffered from large computational requirements, which was not straightforward for the 90’s technology. The solvable problems were limited to a few hundreds of objects in a few dimensions. To address these limitation, a modified version of MCD was introduced namely: the Fast Minimum Covariance Determinant (FAST-MCD) \[99\]. FAST-MCD aims at reducing the computation time of the original MCD. The authors claim that the FAST-MCD is faster in order of magnitudes and more accurate than existing algorithms of the MCD by selective iteration and nested extensions.

Other robust covariance matrix estimators exist such as Multivariate Trimming (MVT), the Stahel-Donoho estimator and the Minimum Volume Estimator (MVE). The MVE searches the ellipsoid covering \( h \) data points, where
4.1 Estimation of location and scatter

\[ n/2 \leq h < n, \] with the smallest volume. The FAST-MCD is, however, among the fastest robust covariance matrix estimators and it is often used in practice [115]. More information on other estimators of scatter can be found in the work of Maronna et al. [75, 76], Rousseeuw [98], Gnanadesikan et al. [39], and the references therein.

Implementations of the seemingly complex MCD algorithm, and many other robust estimators, are available in a multitude of mathematical software packages including MATLAB, S-PLUS, and R. The R implantation of the MCD is used in this work, which is part of the \texttt{rrcov} library [115]. In this implementation, \( h \) is available as an argument to the MCD function. Fauconnier and Haesbroeck [35] showed that optimal performance can be achieved for \( h/n = 0.75 \) and default remaining parameters. This corresponds to a breakdown point of 25\%, which ensures reasonable efficiency and high robustness against outliers [35].

4.1.3 Example

A practical example shows the different estimators and the usefulness of the robust estimators. Arlos and Fiedler [7] measured the OWD in 3G networks for three different network operators with respect to packet sizes. The OWD measurements of operator A as per their paper is shown in Figure 4.1. The estimation of location and scatter is shown in Figure 4.1(a) and 4.1(b), respectively. A box-and-whisker plot of the data is depicted in 4.1(c). The boxes encapsulates all data points within the 25\% and 75\% quantiles (the Interquartile range (IQR)). The whiskers equals the maximum and minimum data values no larger than 1.5 times the IQR. Data points larger than 1.5 times the IQR are plotted. The median is marked inside the boxes. A linearly increasing trend is observed until packet size 496 bytes, after that the median OWD drops to 45 ms. The packet sizes 496 bytes to 608 bytes show a significantly larger variance than the other measurement points. It is believed that in this region the 3G connection switches from W-CDMA to High-speed Downlink Packet-data Access (HSDPA).

The estimation of location and scatter is computed and plotted in Figure 4.1(a) and 4.1(b), respectively. The solid lines were obtained via classical estimates, the arithmetic mean and standard deviation. The long dashed line was computed via the median and MAD. The short dashed lines were estimated with MCD. From the packet size 500 bytes the arithmetic mean is showing large variations. This indicates that from that point on the dataset contains measurements that would be considered outliers for a normal distribution. By analyzing the box-plots in Figure 4.1(c), it is indeed observed that the data contains several outliers especially in the tail of the distribution. The major mass of the distributions are however estimated well by the MCD and median. The estimation of dispersion with the standard deviation shows some large deviations, for
similar reasons as the arithmetic mean. The robust methods are fairly stable, though the MCD shows some increased variance around 600 bytes, which is also identified by the standard deviation. The MAD failed to identify the increased variance.

### 4.2 Distances

A distance metric is equivalent to a set of rules that assign positive numbers between pairs of objects [102]. Many statistical methods are based on the analysis of distances between objects, matrixes, or tables. The Euclidean and Mahalanobis distance are such examples.

The *Euclidian distance* is probably the most known distance metric and it is defined as:

\[
D_E(a, b) = \sqrt{(a - b)^T(a - b)} = \sqrt{\sum_{i=0}^{N} (a_i - b_i)^2},
\]

(4.12)

where \(a = (a_1, a_2, \ldots, a_n)\) and \(b = (b_1, b_2, \ldots, b_n)\) are two vectors in \(\mathbb{R}^n\) with \(n\) elements. The Euclidean distance is a special case of the *Minkowski distance* family. A Minkowski distance \(M_p\) with \(p\) degrees of freedom has the form [102]:

\[
M_p = \left(\sum_{i=1}^{n} |a_i - b_i|^p\right)^{1/p}, \quad p \in [1, \infty).
\]

(4.13)

\(p\) is not necessarily an integer. The Minkowski distance with \(p\) degrees of freedom is also referred to as the \(L_p\)-norm. For \(p = 2\), the Minkowski distance becomes the Euclidian distance. \(p = 1\) is also known as the *Manhattan distance* or the *taxi-cab distance*. For the *Chebyshev distance*, \(p\) is equal to \(\infty\). Equation (4.13) then reduces to \(\max(|a_i - b_i|)\).

#### 4.2.1 Mahalanobis distance

The *Mahalanobis distance* is a generalization of the Euclidean distance. Mahalanobis distance expresses the distance between two vectors taking into account constraints expressed by a covariance matrix [73]. This distance differs from the Euclidian distance in the use of the correlation between the components of the data set. Mahalanobis distance is formally defined as:

\[
D_M(a, b) = \sqrt{(a - b)^T C^{-1} (a - b)},
\]

(4.14)

where \(C\) is covariance matrix, defined by equation (4.7).

Equation (4.14) calculates the distance between two arbitrary points. When the distance of an arbitrary point to the center of a data set is required then \(a\)
Figure 4.1: The One-Way Delay (OWD) of operator A and its estimate of location and dispersion for different types of estimators with regards to the packet size [7]. Both classical and robust statistical methods were used to estimate the location and dispersion of the data set.
is replaced with $x = (x_1, x_2, \ldots, x_n)$: a vector containing a set of arbitrary coordinates, $b$ with $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$: the center of the multivariate distribution from which $x$ are samples, and $C$ by $S$: the covariance matrix of the multivariate data set. Whence equation (4.14) results in:

$$D_M(x, \mu) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)},$$  \hspace{1cm} (4.15)

Equation (4.15) is interpreted as a measure of the probability of $x$ in the multivariate data set. $\mu$ and $S$ are very sensitive to outliers as shown in section 2.2. Robust estimators are used to obtain more reliable values. $\mu$ and $S$ are replaced with the estimates of the MCD, so robust distances are obtained [100].

It can be shown that if a data set is multi-normal distributed then the Mahalanobis distances of the data set are distributed approximately as a chi-square distribution:

$$D_M^2 \sim \chi^2_p,$$  \hspace{1cm} (4.16)

with $p$ degrees of freedom [107]. Given that $D_M^2$ is approximately $\chi^2_p$ distributed, distances larger than some predefined cutoff can be considered as outliers. A popular theoretical cutoff value is the square root of $\chi^2_{p,0.975}$ [64]. For practical use, a cutoff value can be used as a threshold or decision boundary by a classification method.

It is observed that the Mahalanobis distance in equation (4.14) can be reduced to the Euclidean distance from equation (4.12) when $C$ is replaced by an appropriate identity matrix. An identity matrix has ones on its main diagonal axis and zeros elsewhere. The equality of $C$ to the identity matrix indicates an absence of correlation between the dimensions of a multivariate data set.

Figure 4.2 exemplifies the difference between the Euclidean and the Mahalanobis distance. In Euclidean geometry, a circle is the collection of points in a plane with equal distance to a given point. Similarly, the Mahalanobis distance connects all points that have an equal probability in a multivariate distribution. 1500 random points are plotted, which are drawn from a bivariate normal distribution $\mathcal{N}_2(0, 1)$ with a correlation $\rho = 0.8$. A circle is drawn connecting all points with an Euclidean distance of $D_E = 2.63$ from the center $(0, 0)$. An ellipse is drawn, which connects all points with the same Mahalanobis distance of $D_M = 5.99$. In this case the ellipse represents also the 95% quantile of the bivariate normal distribution $\mathcal{N}_2(0, 1)$. The two (triangular) points $A$ and $B$ have the same Mahalanobis distance and are equally probable to occur. The Mahalanobis distance of a point tells us something about the location of the point in a distribution whereas the Euclidean distance describes the location of a point in an Euclidean space.

An ellipse defined by a given Mahalanobis distance can be used to classify instances. A dataset $\mathcal{D}$ is given by

$$\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^p, y_i \in \{1, -1\}\}_{i=0}^n,$$  \hspace{1cm} (4.17)
The theoretical cutoff for outliers is argued to be \( (\chi^2_{p,0.975})^{\frac{1}{2}} \). In practical applications this theoretical value might not yield the best classification accuracy. The following optimization problem is formulated to locate the optimized cutoff (OC) value:

\[
\min_{D_M} \max\{C_{1,F}(D_M), C_{2,F}(D_M)\}, \tag{4.19}
\]

where \( C_{i,F}(D_M) \) is the number of wrongly classified objects in class \( i \), for a given Mahalanobis distance \( D_M \). \( C_{1,F}(D_M) \) and \( C_{2,F}(D_M) \) can also be referred to as
false positives and false negatives. The minimization problem finds the value for \( D_M \) where the number of misclassifications in both classes are equal. For a data set whose classes are not elliptically separable, the optimized cutoff seeks the middle of the two classes overlap. For a data set whose classes are elliptically separable, the optimized cutoff is any ellipse that separates the two classes, i.e., \( C_{1,F}(D_M) = C_{2,F}(D_M) = 0 \). The \( C_{1,F}, C_{2,F} \) equality is the also referred to as the breakdown point in the precision-recall analysis. It is observed that the minimization problem is a convex problem and thus can be solved by an appropriate optimization solver.

### 4.3 Support Vector Machines

A Support Vector Machine (SVM) is a learning method that produces a dichotomous classifier [18]. The SVM training algorithm generates a hyperplane that aims to separate instances into two classes. SVM can be used for both classification and regression analysis. The hyperplane \( f_k \), defined by

\[
\{ x \mid f_k(x) = w \cdot x + b = 0 \},
\]

where \( w \) is the normal to the hyperplane, defines the decision boundary that divides the sample space into two classes. Given that instances belonging to different classes in a data set \( D \) (defined by equation (4.17)) are linearly separable, then SVM seeks the two parallel equidistant hyperplanes to \( f_k \), or support vectors, that separate the dataset with the largest margin. The margin is defined as the distance from the support vectors to the separating hyperplane \( f_k \). Figure 4.3 shows a separating hyperplane between a linear separable data set and its margin. The two support vectors \( f_+ \) and \( f_- \) can be written as

\[
\begin{align*}
\{ x \mid f_+(x) = w \cdot x + b = a \}, \\
\{ x \mid f_-(x) = w \cdot x + b = -a \},
\end{align*}
\]

for some constant \( a > 0 \). When the data points are multiplied by an arbitrary factor, the margin and \( a \) will scale with the same factor. Further on, \( a \) is set to 1 for simplicity. The points \( x \in D \) can then be classified in a class \( y \) as

\[
h(x_i) = \begin{cases} 
    y_i = 1, & \text{for } w \cdot x + b \geq 1, \\
    y_i = -1, & \text{for } w \cdot x + b \leq -1. 
\end{cases}
\]

Let's assume that the points \((x_+, x_-) \in D\) are chosen such that the margin is maximized. The points \((x_+, x_-)\) lie on the hyperplanes \( f_+ \) and \( f_- \), respectively, and are equidistant from the decision boundary \( f_k \). Subtracting equation (4.21b) from (4.21a), and dividing by the norm of the hyperplane \( f_k \) normal, yields

\[
\frac{1}{2} \frac{w}{\|w\|} \cdot (x_+ - x_-) = \frac{1}{\|w\|}. 
\]

\(4.23\)
4.3 Support Vector Machines

The left hand side of this equation is equal to the margin if the separating hyperplane $f_k$ [9]. The margin maximization problem is thus equivalent to the minimization of $\|w\|$. For mathematical convenience $\|w\|$ is divided by two and squared in the minimization problem. Hence, SVM becomes a minimization problem defined by

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2, \\
\text{subject to} & \quad y_i \cdot (w \cdot x_i + b) \geq 1, \quad (i = 1, \ldots, n).
\end{align*}$$

The constraint to the optimization problem in this equation is a compact formulation of the equation (4.22).

The SVM minimization problem from equation (4.24) does not work if the assumption of linearly separable classes is violated. In practical problems the classes are often overlapping. Moreover, larger margins could be achieved if some date points are misclassified. To allow this slack variables $\xi_i$ are introduced in the inequalities and the minimization problem results in

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i, \\
\text{subject to} & \quad y_i \cdot (w \cdot x_i + b) \geq 1 - \xi_i, \quad (i = 1, \ldots, n), \\
& \quad \xi_i \geq 0, \quad (i = 1, \ldots, n).
\end{align*}$$

The introduction of the slack variable allows points to be misclassified ($\xi_i > 1$) or in the margin ($0 \leq \xi_i \leq 1$). Hence, $\sum_i \xi_i$ is a bound on the number of
misclassified points. The term \( C \sum_{i=1}^{n} \xi_i \) in equation (4.25) is appended to the minimization problem to penalize misclassification and margin errors [9]. The value of \( C \) is then a trade-off between maximizing the margin and minimizing the number of misclassifications (amount of slack). Finding a hyperplane with SVM thus reduces to the minimization problem of equation (4.25). Usually the problem is expanded into a (dual) quadratic programming (QP) problem. The resulting problem can be solved by any QP solver. Special optimization algorithms have been developed, however, to perform optimally for SVM problems. For example, Sequential Minimal Optimization (SMO) [91] breaks the long QP problem into small pieces which can be solved more efficiently. The amount of memory used is linear to the size of the dataset. Via an iterative process, chunking incremental learning [62] adds samples to the working set in batches. This reduces computational complexity and increases the learning process. The optimized algorithms for SVM perform better in terms of speed, scalability, and memory usage [65].

Non-linear classification problems can also be tackled by the described SVM algorithm. Before applying SVM, the input space \( X \) is mapped to a higher-dimensional feature space \( H \), where the data can be separated by a hyperplane: \( \Phi : X \rightarrow H \). Each time the dot product is applied, the input space is mapped accordingly. For example, the first constraint from equation (4.25) becomes

\[
y_i \cdot (w \cdot \Phi(x_i) + b) \geq 1 - \xi_i, \quad (i = 1, \ldots, n).
\]

This process is also referred to as the kernel trick [9]. Popular kernels include the polynomial function, the hyperbolic tangent function, the Gaussian radial basis function and others. SVM can also be used for regression purposes. This is achieved by modifying the set of constraints and adding extra slack variables to the minimization problem. Novelty detection is also possible by creating a spherical decision boundary, defined by support vectors, around the data set [65].

### 4.4 Computational complexity

Real-time measurement systems preferably need algorithms that have low complexity such that the computation time is within certain limits, and the resource requirements are acceptable. The latter is especially important for devices with limited or scarce resources such as hand-held devices. The advantage of the Mahalanobis distance and SVM is that some variables of the algorithm can be considered fixed over time. Thus these variables do not need to be computed each time an QoD estimation needs to be produced.

For the Mahalanobis distance the \( S \) matrix (equation (4.15)) is not likely to change and can thus be precomputed for specific environments. This is similar for the center of the data set \( \mu \). If these values are available beforehand
then the computation of the Mahalanobis distance is reduced to the multiplication of a matrix and two vectors. These are operations that do not require excessive computational power. In such a measurement system the accuracy of the Mahalanobis estimates is dependent on the precomputed $S$ matrix and $\mu$. To yield reliable distance estimations, these variables should be computed in environments with an overall good QoS.

To classify measurements with SVM during run-time is less complex than for the Mahalanobis distance. It suffices to solve the optimization problem beforehand on training data. The resulting separating hyperplane can then be used in real-time as decision boundary for future classifications. Classifying a measurement is then the computation of a linear function whose complexity is proportional to the input space. Similar to the Mahalanobis distance, the training set over which the decision boundary is computed should be representative of the real-world.

4.5 Summary

The theoretical background of the methods, SVM and the the Mahalanobis distance, used for the analysis of the QoS metrics have been presented. The computational complexity of the presented methods has been briefly discussed for use in a real-time environment.

The Mahalanobis distance is based on statistical properties of a data set. The concept of robust statistics was introduced and also shown how to apply these to improve the accuracy of the Mahalanobis distance. In contrast with the Mahalanobis distance, the SVM algorithm doesn’t depend on statistical properties of the data.
Chapter 5

Measurement Environment

In science one tries to tell people, in such a way as to be understood by everyone, something that no one ever knew before. But in poetry, it's the exact opposite.

Paul Dirac

This chapter is about the measurements acquisition and testbed used to collect the data for the QoS analysis. The technical components of the testbed are described and also the used software and tools are discussed. The scenarios and their settings under which the measurements are collected are summarized. Finally, the measurement software is validated via some experiments.

5.1 Measurement scenario outline

For the purposes of the QoS metrics analysis with a focus on jitter buffers, it is needed to measure two different processes: the network’s QoS parameters and the RTP jitter buffer state. The aim is to relate the jitter buffer states of a video player, to conditions on the network described by QoS metrics. A series of experiments are conducted where a video is streamed over wireless networks to a mobile handset while recording both jitter buffer states and QoS metrics. Depending on the behavior of the wireless network towards different video quality levels, the jitter buffer is affected accordingly. These phenomena are captured and behavioral consistencies are identified.

The specifications of the video may have an affect on the continuity of the video playback. It is thus of great importance to cover as many possible network conditions as possible to secure a meaningful view of the effects of QoS fluctuations on the jitter buffer state. This is theoretically desired, however, practically this is not easily attained. The experiments are organized to simulate scenarios as they could present by ordinary usage of wireless technologies, e.g.,
in an office space or around the house. Obviously, if the wireless connection is optimal, the performance of the media applications is plausibly also optimal. It is therefore needed to measure on the transition phase between suboptimal and poor network conditions to exhibit the network effects on the video QoD. To achieve this, the load on the wireless network is increased by increasing the quality of the video, i.e., a larger video bit rate. An increased video bit rate may also imply an increased packet rate while streaming.

5.1.1 Technical settings

The experiments were conducted over a Wireless Local Area Network (WLAN) and a Wide-band Code Division Multiple Access (W-CDMA) network. A video is streamed with different parameter settings from a Darwin Streaming Server (DSS) to an HTC Dream running Android 1.6 OASP with kernel version 2.6.29. The screen size of the mobile device is $320 \times 480$ pixels. Though the videos were resized to Quarter Common Intermediate Format (QCIF) format, i.e., $172 \times 144$ pixels, for smooth play-back purposes. It was observed that for higher resolutions the HTC’s device seems to have difficulties with processing the video, resulting in short video freezes. To minimize this effect, a smaller video resolution was used.

The video resolution and bit rates were chosen so that the threshold between smooth streaming and degrading QoS performance is covered. Two different videos: Elephants Dream (ED) and Big Bug Bunny (BBB) [13] were streamed, both released under the Creative Commons Attribution license. The ED video contains much motion whereas the BBB has more still frames. The videos were downscaled and compressed with the H.264/MPEG-4 AVC codec based on the original video file without audio. After the compression, the videos were hinted with MP4BOX, an open-source multimedia packager as part of GPAC [40]. The hinted video file suggests the streaming server how to package the video for transmission over the Internet. The videos were hinted with a Maximum Transmission Unit (MTU) of 1440 bytes.\(^1\)

During the experiments, the bit rate of the video stream was increased gradually. The bit rates were as follows: $\{100, 150, 200, 250\}$ kbit/s. The audio of the video is removed so that the during the experiments only the video channel is streamed. Accordingly, RTP maintains only one jitter buffer. The length of the videos are 9’ 56” and 10’ 53” for BBB and ED, respectively. Both movies were played six times for each bit rate setting. In total this yields 124’ 54” of traces for a single video bit rate. All bit rates were tested both on WLAN and W-CDMA, i.e., 499’ 36” of traces per technology, and 999’ 12” in total.

\(^1\)The Android OS, based on Linux, uses a default MTU of 1500 inherent to the Ethernet v2 standard. This includes however the IP and transport layer header. We must also subtract the tunnel and measurement header size (see further). In total this yields 1440 available bytes for RTP.
Table 5.1: Summary of the fixed and variable parameters during the video streaming experiments

<table>
<thead>
<tr>
<th>State</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>movie resolution</td>
<td>$172 \times 144$ pixels</td>
</tr>
<tr>
<td></td>
<td>video coding</td>
<td>H.264/MPEG-4 AVC</td>
</tr>
<tr>
<td></td>
<td>audio coding</td>
<td>no audio</td>
</tr>
<tr>
<td>variable</td>
<td>LMA</td>
<td>$n \in {1, \ldots, 16}$</td>
</tr>
<tr>
<td>variable</td>
<td>EWMA</td>
<td>$(1 - \beta) \in {0.06, 0.12, \ldots, 0.96}$</td>
</tr>
<tr>
<td></td>
<td>video bit rate</td>
<td>${100, 150, 200, 250}$ kbit/s</td>
</tr>
</tbody>
</table>

The measurement data is smoothed before statistical methods are applied. Different smoothing coefficients for both LMA and EWMA are considered to identify the optimal smoothing coefficient. The $n$ and $\beta$ smoothing coefficient of LMA and EWMA were varied, over 16 steps. $n$ is incremented from 1 to 16 and $(1 - \beta)$ is incremented from 0.06 to 0.96 in steps of 0.06.

The technical parameter settings of the experiments as discussed above are summarized in table 5.1. The table lists the fixed parameters and provides also a summary of the parameters that are varied throughout the experiments.

## 5.2 Experiment setup

The experiments described in section 5.1.1 are conducted in a real-life environment. Blekinge Institute of Technology (BTH)’s Wireless Fidelity (WiFi) infrastructure and a popular Swedish 3G operator are used as testbed.

During the experiments videos are streamed to a handset running Google’s Android platform 1.6. Android hosts the RTP implementation: CORE/8.000.1.1 OpenCORE/2.06 by PacketVideo. PacketVideo [88], member of the Open Handset Alliance, powers the world’s leading mobile entertainment services for major companies around the globe. PacketVideo claims to be embedded in more than 260 million devices [88]. This includes the Android 1.6 handsets that are used for this study. On Android 1.6 the RTP code is easily accessible, but later versions of Android have a more integrated RTP implementation. The source code of PacketVideo’s RTP implementation is available along with Google’s Android source code. Hence, the possibility does exist to peak into its implementation and amend the code for specific purposes, such as logging. For the streaming video server the DSS framework is used, which RTP protocol implementation is compatible with the Android device’s RTP implementation.
Reliable UDP is not enabled during the DSS and Android negotiations.

A DSS streaming server running on Ubuntu 8.10 with Linux kernel version 2.6.27 and a 10/100 Mbps Network Interface Cards (NICs) is used to stream the videos to the Internet. The media is transmitted over WLAN or W-CDMA to the Android mobile handset. The experiments are conducted during the working days of the week. The 3G operator’s service performance is not time-dependent [7]. However, it is known that the WLAN’s performance depends upon the load because of the shared medium properties. To have homogeneous performance across all measurements, the operation time of our experiments are constrained to workdays between 10 am to 4 pm. It is assumed that then the load on the network is more or less constant.

5.2.1 Timestamp tracking

The departure and arrival timestamps, and the sequence numbers were recorded for each packet during the streaming of the videos. These fields are available in the RTP header. RFC 3550 writes the following about the header timestamp:

…the RTP timestamp represents the instant when the first data in the packet was sampled … the clock frequency is dependent on the format of data carried as payload and is specified in the profile or payload format specification that defines the format … [104]. Thus, the resolution of the timestamp in RTP packets is not necessarily a tenfold of a second. It is a predefined value in the application profile or payload format specification of RTP. The clock resolution of the videos in the experiment is 90 kHz, which is a common sampling rate for H.264 encoded videos over RTP [122]. The clock resolution was verified via MP4BOX. Furthermore, the RTP timestamps in the header correspond to the sampling of the video and are not necessarily related to the departure of the packets in question.

This imposes answering the question whether the RTP timestamps are appropriate to be used for accurate end-to-end network measurement purposes or not. If not, additional packets or encapsulation is necessary to obtain a more accurate departure timestamp of each packet. Though, the RTP RFC specifies the use of these timestamps for delay jitter calculations. The question is how the accuracy of end-to-end measurements is affected by relying on the RTP timestamps. To answer this question a simple experiment was set up that compares the inter-departure time of the packets in a video stream with the inter-sample time of the sample enclosed by the packets. Both values can be derived from tcpdump logs by measuring the departure time of the packets and logging its RTP timestamp. We stream from our DSS to a random computer over an optimal connection three different bit rates: 100, 500 and 1000 kbps, and record their traces at the DSS. Then, the inter-sampling times and inter-departure time are selected, converted to the same time unit and subtracted.

Figure 5.1 shows the histogram of the difference in sampling time of sam-
5.3 Validation and verification

The algorithms for the assessment of the QoS metrics discussed in chapter 3 are implemented in C++. To validate and verify the correct functioning, the algorithms are analyzed via shaped dummy traffic. Via the ping program we create traffic between a sender and a receiver with packets of incrementing size and decreasing inter-arrival times. The sender and receiver are directly connected via the switch of a Linksys WRT-54GL. With this setup, nearly optimal QoS conditions are obtained. Via ping a default round-trip time (RTT) is measured for

![Figure 5.1: Difference between the sampling time of video frames and its associated departure time when being transmitted over the Internet for streaming purposes. (bin size = 2 ms)](image-url)
Table 5.2: Summary of the simulation settings for the validation and verification of the QoS metrics assessment algorithms.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ping</td>
<td>inter-arrival time {0.1, 0.07, 0.05, 0.04} s</td>
</tr>
<tr>
<td></td>
<td>ICMP payload {80, 256, 512, 1024, 1500} bytes</td>
</tr>
<tr>
<td>netem</td>
<td>delay variation {1, 2, 5, 10, 20, 50, 100, 200, 500, 1000} ms</td>
</tr>
<tr>
<td></td>
<td>drop rate {0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50} %</td>
</tr>
<tr>
<td></td>
<td>reordering extend {30, 35, 40, 50, 55, 75, 100, 150, 460} ms</td>
</tr>
<tr>
<td>algorithm</td>
<td>buffer size {1, 4, 8, 16} packets</td>
</tr>
<tr>
<td></td>
<td>meas. interval {0.5, 1, 2, 4, 16} s</td>
</tr>
</tbody>
</table>

different bit rates around 0.4 ms, with an average standard deviation of 30 µs and 0% packet loss and reordering. Both sender and receiver run a version of Linux OS with netem enabled. netem [48] is a network emulator that allows for shaping traffic which passes through any network interface in the Linux OS. Only the outgoing traffic from the sender is shaped via netem in the setup.

The ping data is send through a UDP-tunnel which header is extended with a departure timestamp and sequence number [52]. The same tunnel is used for the video streaming experiments. These fields are recorded upon arrival of packets and then processed with the QoS algorithms to obtain an estimate of the QoS metrics. These estimates are then compared to estimations from ping and from tcpdump. It is assumed that these tools produce estimates close to the true value of the QoS metrics.

Via netem packet loss, reordering, and one-way delay variation is introduced on the uplink from the sender. Table 5.2 provides an overview of the settings used for validation and verification. The settings of ping, netem, and the QoS assessment algorithms are varied during the simulation to analyze the performance under different circumstances. For ping, the inter-arrival time of the dummy traffic is varied to simulate different packet rates; from 10 to 25 packets per second. Also, the payload of the ping packets is changed. The size of the payloads listed in Table 5.2 added with the Internet Control Message Protocol (ICMP), IP and UDP header lengths² yield a total packet size of {8, 184, 440, 952, 1428} bytes. During the variation of the ping parameters the settings of netem are also changed. We opted to simulate reordering via the delay variation setting. This explains the millisecond unit in Table 5.2 for reordering extend. When the delay variation grows packets may be reordered as the delay

²The dummy traffic is sent through a UDP-tunnel. Hence the total packet size includes an extra IP, UDP and measurement header besides the ICMP header and payload.
variations overgrows the inter-departure time. To prevent this effect during the study of delay variation, the packets were scheduled in a *pfifo* queue. The *pfifo* queue makes sure the packets are transmitted according to the First In, First Out (FIFO) principle or in other words, in sequence.

Dummy traffic is generated with *ping* and shaped with *netem* given the settings from Table 5.2. The data is recorded at the receiver’s side and the log of the received packets is processed with the QoS assessment algorithms. The algorithms produce estimates per time-interval. To distinguish reordering and packet loss a packet buffer is used as described in section 3.3. The attention is given to what implication the length of the time-interval has on the QoS metric estimates. The effects of the packet buffer length is examined as well.

The efficiency of the algorithms is measured as follows:

\[
\text{Eff } \theta_e = \frac{\theta_m}{\theta_0},
\]

where \(\theta_m\) is the estimated value of a parameter via the measurement algorithms and \(\theta_0\) is the estimated value of the reference tool. It is assumed that \(\theta_0\) is very close to the real value of \(\theta\). \(\text{Eff } \theta_e\) is then a measure on how close the algorithms are to the real value of \(\theta\).

### 5.3.1 Delay variation

When *ping* is terminated, the program summarizes some statistics of the RTTs including the variance of the measured RTTs. The variance in *ping* is calculated according to equation (3.4). However, to compare the *ping* measure to the \(D_J\) estimate we need to adjust the measure. Assuming that the links in the experiment are symmetric and identical then the *ping* packets traverse four links, two uplinks to the switch and two downlinks from the switch. Each link, assumed uncorrelated, adds variation to the OWD delay. According to the Bienaymé equality [71] each link contributes an equal amount of variation. Furthermore, the variation introduced by *netem* is only added on the uplink from the sender. The QoS algorithm only measures the variation from sender to receiver, while *ping* measures the variation on the return path too. Thus, the estimates of our algorithm will be biased compared to the RTTs of *ping*. The average variation induced by the other links must be subtracted from *ping*’s RTT estimate to allow for proper comparison with our estimates. The bias is calculated to be 22.5 \(\mu s\). This value is very small, though it is observed that for the smaller delay variations an increased efficiency of a few percent is obtained.

Figure 5.2 shows the efficiency of the \(D_J\) estimate from the measurement algorithm compared to unbiased estimates of *ping*. The effect on the delay variation estimates was analyzed under changing inter-arrival times and packet sizes of the transmitted data. The buffer lengths and measurement time interval for the measurement algorithm was also varied. We observed that the
inter-arrival, time packet size and the buffer length does not have an effect on
the delay estimates. It is only by changing the measurement time interval that
different results are produced.

Figure 5.2(a) shows the efficiency of the delay estimates for different packet
inter-arrival times averaged per packet size, the algorithm’s buffer size is set to
4 packets and the measurement time interval to 1 s. For delay variation values
between 30 ms and 100 ms an efficiency of more than 90% is observed. Outside
of this interval, the accuracy of the algorithm is decreasing.

The efficiency of the delay estimates is shown in Figure 5.2(b) for differ-
ent measurement time intervals averaged per buffer size, the inter-arrival time
is set to 0.05 s and the packet size to 512 bytes. It is clearly observed that a
larger measurement time interval increases the efficiency. The optimal region
lies between 3 ms and 90 ms. In this region the 1, 2 and 4 s time interval has

Figure 5.2: Efficiency of the delay variation estimation for the measurement
algorithms compared to the results of ping.
an efficiency larger than 90%. The 0.5 s time interval seems inflated. The reason for this could be that the variance computation contained in the definition of $D_J$ becomes unstable due to the low sample count. On the other side, the time interval of 16 s shows results that seem to over estimate the $D_J$. The 1 s interval is thus a good solution in-between the two extremes.

5.3.2 Packet loss

The Packet Loss ($P_L$) is shown in Figure 5.3. Similar to the delay variation estimation we did not notice any consistent performance change with varying packet sizes or measurement algorithm settings. In both Figures 5.3(a) and 5.3(b) it is observed that the estimates of the loss is nearly optimal, i.e. close to the estimates of ping. The packet loss estimation’s efficiency is up to 97%. The larger efficiency deviations start from 10% drop loss, apart from the 0.07 s
5.3.3 Reordering

The reordering was derived from `tcpdump` traces and calculated with the same definition as the measurement algorithm. The difference in this case is that the results of `tcpdump` are calculated over the whole trace whereas our measurement algorithm measures per time-interval. The measured reordering counts are added up at the end of the trace. No consistent performance variation was noticed for changing packet sizes, hence the measurements were averaged over the packet sizes. The measurement time interval size did not seem to affect the reordering estimation.

The estimation efficiency of the reordering is shown in Figure 5.4. The data points are plotted for a measurement time interval of 1 s. The figure shows the estimates for four different buffer lengths. It is observed that for a buffer length of 1 packet the reordering estimate is very poor. The estimates improves for a buffer length of 4 and 8 packets. The buffer length of 16 packets is very reliable with regards to the reordering rate. The reason for this behavior is as follows. The buffer is used to distinguish packet loss from reordering effects as discussed in section 3.3. When a packet does not arrive within the arrival of some packet inter-arrival time.
number of consecutive packets, it is considered lost. Thus many packets are considered lost instead of reordered for a buffer length of 1. It is observed that a buffer length of 4 to 16 yields much better results. Though, the 4 packet buffer length scenario is still affected at a reordering rate of 50%. The buffer length can be considered to be a trade-off between accuracy and timely reordering statistics.

5.4 Summary

In this chapter the experimental setup to collect the measurements was explored. The setup consists of a streaming server that streams videos to a mobile video player over wireless Internet access technologies including WLAN and W-CDMA. The videos are streamed with varying bit rates. Via a simple experiment it was observed that using the RTP timestamp as departure time of packets yield small biases. This is due to the fact that the RTP timestamps refer to the sampling times of the video frames and not necessarily to the departure time of the frames. Therefore it was decided to stream the data through a UDP tunnel and use the tunnels timestamps to obtain more accurate end-to-end measurements.

Furthermore, the efficiency of the measurement algorithm was analyzed to estimate the QoS metrics of concern. It was observed that mainly the length of the time-interval, by which measurement results are reported, has an effect on the delay variation. The estimation of the packet loss rate is fairly stable for different loss rate settings. The accuracy of the reordering rate is shown to be depended on the buffer length. The buffer length however, is a trade-off between accuracy and timely reordering and loss rate statistics. Based on these observations, the optimal time-interval was measured to be 1 s for a buffer length of 4 packets. The influence of different packet sizes and packet inter-arrival times were also investigated but no noticeable effects on the QoS metrics estimators’ efficiency were identified.
Chapter 6

Multidimensional QoS Analysis

I will tell you why; so shall my anticipation prevent your discovery . . .

The Tragedy of Hamlet, Prince of Denmark – William Shakespeare

This chapter is about the analysis of the QoS measurements collected during the experiments. The goal is to identify the manifestation of jitter buffer starvations via analysis on QoS metrics, i.e., via the Mahalanobis distance and SVM. The analysis is done without prior knowledge of the video content. Optimal settings of the methods are looked for and the results are discussed.

6.1 Exploring the QoS metrics

Each measured QoS metric has its own significance and shows a different side of the video transmission process. Before matching jitter buffer states to network level events, it is needed to have a closer look at the behavior of each of the QoS metrics itself. It is especially interesting to compare the behavior of the metrics over the different wireless technologies measured, i.e., W-CDMA and WLAN. The distribution of the metrics is important to study as normality is a basic assumption in many mathematical procedures. Here, it is useful to look at the shape and the symmetry of the distribution, and identify if any tails or heads are present. Depending on this analysis the metrics may be transformed before being used in the methods in our analysis. It is also useful to determine whether a given QoS metric contributes information to the analysis or if it merely adds noise.
Table 6.1: Recorded number of jitter buffer starvations for the W-CDMA and WLAN measurements under the streaming of different average video bit rates.

<table>
<thead>
<tr>
<th>Bit Rate (kbit/s)</th>
<th>Elephant’s Dream</th>
<th>Big Bug Bunny</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>150</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>250</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.2: Mean Delay Jitter ($D_J$) for the W-CDMA and WLAN measurements under the streaming of different average video bit rates.

<table>
<thead>
<tr>
<th>Bit Rate (kbit/s)</th>
<th>Elephant’s Dream</th>
<th>Big Bug Bunny</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3.151 (ms)</td>
<td>3.248 (ms)</td>
</tr>
<tr>
<td>150</td>
<td>3.231 (ms)</td>
<td>3.291 (ms)</td>
</tr>
<tr>
<td>200</td>
<td>3.236 (ms)</td>
<td>3.388 (ms)</td>
</tr>
<tr>
<td>250</td>
<td>3.495 (ms)</td>
<td>3.571 (ms)</td>
</tr>
</tbody>
</table>

Table 6.1 shows the amount of instances where an empty jitter buffer was recorded during the experiments. The empty jitter buffer can last for multiple time-intervals. For both the bit rates 200 and 250 kbit/s 14 jitter buffer starvations were recorded. 9 starvations were recorded for WLAN and 19 for W-CDMA. The videos with a bit rate of 100 and 150 kbit/s did not exhibit any jitter buffer starvations during streaming. Though, the 150 kbit/s measurements did have some close encounters. In total, the 28 jitter buffer starvations were sampled for 88 times. There are 88 time-intervals or QoS metrics samples that can be related to jitter buffer starvations.

### 6.1.1 Delay variation

The Delay Jitter ($D_J$) is the standard deviation of the unsynchronized One-Way Delay (OWD) as per equations (3.5). Table 6.2 shows $D_J$ for various settings in the experiments. The median $D_J$ varies around 3 ms to 6 ms and exhibits a small upwards trend for increasing bit rates. The $D_J$ is for the W-CDMA case slightly higher than in the WLAN case. The variance of the $D_J$, computed with the MAD, ranges between 20% to 30% of the median for WLAN and 35% to 40% for the W-CDMA measurements. In absolute values, the variance for W-CDMA is about twice as high was for WLAN.

Figure 6.1 shows the distribution of $D_J$. On the lefthand side the PDF is
Figure 6.1: Distribution of the Delay Jitter ($D_J$) for measurements over both W-CDMA and WLAN networks. The two movies Elephants Dream (ED) and Big Bug Bunny (BBB) are plotted separately. On the lefthand side the PDF is plotted (bin size = 1 ms), on the righthand side the CCDF is shown. Values larger than 15 ms are omitted from the histogram.

plotted, on the right hand side the Complementary Cumulative Distribution Function (CCDF) is shown. The histogram shows that the distribution of $D_J$ for WLAN has a higher peak and it is less wide than the W-CDMA distribution. Thus the distribution of WLAN has a smaller variance compared to the W-CDMA case. Also, the center of the W-CDMA distribution’s body is shifted to the right, which means $D_J$ is on average larger for W-CDMA measurements than for WLAN. These observations are also visible in Table 6.2. The difference in $D_J$ between the BBB and the ED video is not large. $D_J$ in both technologies exhibits a long tail that goes up to 100 ms. It is observed that the tail for the W-CDMA measurements is heavier than in the WLAN measurements.

The W-CDMA measurements exhibit a larger center, variance, and tail compared to the WLAN measurements. This is an indicator that one should make a distinction between the wireless technologies when treating the $D_J$ in our multidimensional analysis, e.g., to reduce noise.

### 6.1.2 Zero Throughput Time

The Zero Throughput Time ($T_Z$) is defined to be the maximum inter-packet arrival time measured per time interval. Figure 6.2 shows the distribution of $T_Z$ for both W-CDMA and WLAN. No distinction is made between the movies as their difference in distribution is negligible. It is observed that the distribution of $T_Z$ is remarkable different for W-CDMA and WLAN. The $T_Z$ in the W-CDMA case exhibits clear peaks with about 10 ms interval. For the WLAN case three
large peaks are observed at 40 ms, 50 ms, and 80 ms. The peaks in the W-CDMA measurements are a result of the packet scheduling. The Transmission Time Interval (TTI)\(^1\) of W-CDMA is a multiple of 10 ms [21]. In contrast to W-CDMA, the WLAN distribution is apparent continuous as seen in Figure 6.2. The time slot duration in WLAN are defined in microseconds, e.g., for IEEE 802.11g the time slot duration is set to 9 µs or 20 µs [54]. The measurement module has a resolution of 1 ms, thus the fine granularity of the WLAN time slots is aggregated into a less accurate millisecond scale. The tail of \(T_Z\) is not visible in Figure 6.2, instead Figure 6.3 shows the CCDF of \(T_Z\) for both W-CDMA and WLAN. The tails go up to 10 s, this means that in these cases no data arrived to the jitter buffer for about 10 s. Both graphs show a steep drop in probability around 100 ms, after which large values manifest up till 5 to 10 s. In previous research these large values were valuable indicators for user experience degradation [27].

\(T_Z\) shows a complete different picture for the WLAN and W-CDMA case. Based on this observation, and similar to the conclusions of \(D_J\), \(T_Z\) suggest to separate the QoS analysis between the wireless technologies. The values for the average \(T_Z\) and its variance are not presented given that the distributions of \(T_Z\) are multimodal. In such cases, moments do not yield estimations with practical significance.

\(^1\)The Transmission Interval Time (TTI) in W-CDMA is the time it takes to transmit a transport block over the wireless radio interface [21]. In contrast, HSDPA employs a 2 ms TTI.
6.1 Exploring the QoS metrics

Figure 6.3: CCDF of Zero Throughput Time ($T_Z$) the WLAN (left) and W-CDMA measurements (right).

Table 6.3: Distribution of Clumping (CL) for WLAN and W-CDMA measurements. Low clumping values are observed for both cases.

<table>
<thead>
<tr>
<th></th>
<th>WLAN</th>
<th>W-CDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Density</td>
<td>0.481</td>
<td>0.517</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0.001</td>
<td>0.000</td>
<td>0.005</td>
</tr>
<tr>
<td>0.13</td>
<td>0.001</td>
<td></td>
</tr>
</tbody>
</table>

6.1.3 Clumping rate

Clumping (CL) is the effect where packets arrive with shorter inter-interval times than they were originally sent. Practically, the CL is measured as the maximum number of consecutive packets with the same timestamp. The maximum CL per time interval is recorded, and the resulting distribution is shown in Table 6.3. All measurements per technology were combined as the distribution for the different bit rates and videos didn’t show major differences. A CL equal to one implies that all packets during the time interval of concern arrived at different time instances with a resolution of 1 ms. CL equal to two indicates the arrival of two packets within one millisecond, and so forth. Table 6.3 shows that the behavior of CL in W-CDMA and WLAN is different. Packets arriving over the W-CDMA network often arrive in pairs, sometimes in batches of three, and a small fraction arrives in batches of four or alone. In the WLAN case, packets arrive mostly alone or in pairs with more or less the same probability. Rarely packets arrive in batches of three.

Given that the CL doesn’t show a large spread, its variance will be small. This property is not of great use for the purposes of the analysis, which focuses on extreme behavior, in other words on outlying values.
Table 6.4: Total number of measured packet losses for both the W-CDMA and WLAN networks under video streaming of different bit rates.

<table>
<thead>
<tr>
<th></th>
<th>Elephants Dream</th>
<th></th>
<th>Big Bug Bunny</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bit rate (kbit/s)</td>
<td></td>
<td>bit rate (kbit/s)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>150</td>
<td>200</td>
<td>250</td>
</tr>
<tr>
<td>WLAN</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>W-CDMA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

6.1.4 Packet Loss and Inter-loss Distance

Packet Loss (PL) rate was defined in section 3.2.3 as the number of packets lost per time interval. The time interval for the measurements was set to 1 s. The number of packet losses is shown for each wireless technology and video bit rate in Table 6.4. The packet losses are validated via the PVPlayer’s RTP summary, which is generated at the end of each complete video transmission. PL is very low. For the W-CDMA network this is expected. The low value of PL for WLAN indicates that the shared wireless medium was not much populated by other users or no large traffic volumes were sent.

The total number of packet losses at the W-CDMA measurements is 5 and 2 at the WLAN measurements. The number of jitter buffer starvation occurrences measured is about three times as high. It is unlikely that a few (consecutive) packet losses can cause a jitter buffer to exhaust. One may therefore conclude that the PL metric in this study will not be of great use in the analysis of video buffer starvations.

The Inverse Inter-loss Distance (IILD) is related to the PL as the distances in time between consecutive packet losses. The occurrence of packet losses, as shown in Table 6.4, is very low. Accordingly, the number of IILD values is at least as low. Moreover, the packet losses occurred in different runs of the videos and the packet loss count is not transferred to the next run. IILD needs at least two consecutive packet losses in the same run in order to be computed. As a result, there are very few estimations of the IILD and the metric will, just like PL, most likely not contribute much information to the QoS analysis.

6.1.5 Reordering

Packets are considered reordered when they do not arrive in the sequence as they were initially sent. Table 6.5 shows the frequency of the measured number of reordered packets. All measurements per technology were combined as the distribution for the different bit rates and videos didn’t show major differences. It is clear that in the W-CDMA network reordering occurs more often than in the WLAN case. It is also observed that the reordering occurrences are seemingly
not consistent for varying bit rates.

### 6.1.6 Packet Arrival Rate

The Packet Arrival Rate (PAR) must be treated carefully in the measurement analysis because PAR is dependent on the VBR video content. PAR is also influenced by traffic and congestion control on the application or transport layer. The latter is an interesting effect that can provide us some indication about the QoD of a media stream. However, separating the effects of congestion control and VBR content is not straightforward without prior knowledge of the video content. This is mainly because RTP imposes its own packet sequence numbering over the video file’s original frame sequence. Nevertheless it is useful to track PAR, because the jitter buffer content is not replenished when the PAR drops to zero. In such an extreme situation, troubles may arise if PAR does not stabilize before the jitter buffer starves.

Figure 6.4 shows the histogram of PAR for different bit rates. The measurements for the different wireless technologies are combined as no major difference are noticeable. PAR in the four bit rates has a major peak around 80 packets/s and range from 40 to 120 packets/s. Only the 150 kbit/s bit rate exhibits some deviating behavior, which shows also a peak around 60 packet/s and accordingly a smaller peak around 80 packets/s. What is not easily visible in the histogram is that the bit rates 200 kbit/s and 250 kbit/s also have some small population below 10 packet/s. In such cases the arrival of packets to the mobile device is very low. This can have three causes: the network is temporarily congested, many packets were dropped, or at that particular time the streamed media content was coded with a low bit rate. The second case is here very unlikely as it was shown before that the packet loss rate is very low. The last case may happen, e.g., when the a video contains a few consecutive frames with few motion.
6.2 Multidimensional Analysis

The previous section focussed on the behavior of each QoS metric separately. The interest was to assess the possible contribution of the QoS metrics to the analysis. The rationale is that poor network conditions are manifested in extreme behavior of QoS metrics. Metrics with a large variance are then more likely to be descriptive metrics. Moreover, outlying values may sometimes only be revealed from a multidimensional perspective. In this respect, it is less likely that PL and IILD would contribute to the analysis. This is based on their low variance properties, and also their low number of occurrences. TZ, DJ, and PAR showed a much larger variance and exhibited some sort of tail. As a result, these metrics are better candidates to be useful in the analysis. Accordingly, RR and CL metrics did not show large variations but did occur in large numbers. Their contribution to the analysis is yet to be investigated. The individual analysis of the QoS metrics also pointed out that the distributions of the metrics are notably different for WLAN and W-CDMA. Consequently, in the following we will treat the analysis of the QoS metrics separately for the wireless technologies.

The dichotomous classifiers based on the Mahalanobis distance and Support Vector Machines (SVMs) divides the sample space, i.e., the measurements, in two disjoint subsets. One subset containing all instances that the classifier believes to be related to jitter buffer exhaustions and the rest in the other subset. The classification of the classifier will not 100% accurate. Thus some mea-
measurements may be misclassified. To distinguish the classified instances, they are labeled as true positive (TP), true negative (TN), false positive (FP) or false negative (FN). True positives are measurements classified as jitter buffer starvations, which are indeed jitter buffer starvations. False positives are measurements that are linked to jitter buffer starvations, which in reality are not related. Similarly, false negatives are measurements that are considered not to be jitter buffer starvations whereas they actually are starvations. True negatives are measurements classified not indicating jitter buffer starvations which are indeed not linked to starvations. It is understood that an optimal classifier generates high numbers of true positives and true negatives, and low numbers of false negatives and false negatives.

For each method, i.e., the Mahalanobis distance and Support Vector Machines (SVMs), the precision is calculated as well as the recall under the different simulation settings. The precision is the number of jitter buffer starvations occurrences that are classified correctly by the hypothesis. The precision of a hypothesis $h$ with respect to a target function $f$ is defined to be:

$$\text{precision}(h) = \frac{1}{|TP \cup FN|} \sum_{x \in TP \cup FN} \delta(f(x), h(x)), \quad (6.1)$$

where $|TP \cup FN|$ is the cardinality or number of measurements in the subset containing all true positives (TP) and false negatives (FN), $\delta(\cdot, \cdot)$ is the zero-one loss function that is 0 if $f(x) \neq h(x)$, and 1 otherwise. The precision is calculated over the subset containing all true positives and false positives. The recall of a hypothesis $h$ with respect to a target function $f$ is defined to be:

$$\text{recall}(h) = \frac{1}{|TP \cup FP|} \sum_{x \in TP \cup FP} \delta(f(x), h(x)), \quad (6.2)$$

where $|TP \cup FP|$ is the cardinality of the subset containing all true positives (TP) and false positives (FP), $\delta(\cdot, \cdot)$ is a zero-one loss function that is 1 if $f(x) = h(x)$, and 0 otherwise.

Cross validation is applied to get an estimate of the efficiency of the classifier. A stratified $k$-fold cross-validation is used. In a $k$-fold cross-validation the sample space is randomly divided in $k$ training sets and $k$ disjoint test sets. The performance of the classifier computed over the training set is measured over the test set. Each of the $k$ test sets is used once for testing and the remaining data as the training set. A fourfold cross-validation is used in the following and it is repeated 25 times. In total one-hundred estimates of the classifier are obtained. The precision and recall on the different iterations are averaged to produce an overall classification estimate [124]. Furthermore, the dichotomous classes in our measurements are unbalanced, i.e., only 0.15% of the measurements are linked to jitter buffer exhaustions. In such cases stratified sampling is
advised. Stratified sampling ensures that the class distribution from the whole data set is preserved in the test and training set [74].

### 6.2.1 Performance of the classification methods

Equation (4.14) in section 4.2.1 defines the Mahalanobis distance. Mahalanobis distance is similar to Euclidian distance with the difference that the inverse covariance matrix is used in the distance calculation. This results in a distance measure that describes the probability of a point within a multivariate distribution. The variance of the metrics is an important aspect in the Mahalanobis distance calculation. This implies that only variables with a defined variance larger than zero can be used. \( P_t \) and IILD can not be used in this aspect as their variances are close to zero. The consequence is that these parameters are not considered for the Mahalanobis distance case.

Before the measurements are processed by the Mahalanobis distance and SVM, the data should be transformed, i.e., normalized, or centered and autoscaled. If data is not normalized then for distance calculations, the metric with the largest average value can mask the fluctuations of the variables with smaller average and variance. In the case of SVM, better performance is achieved on scaled data. Therefore the measurements \( x_{\text{norm}} \) of metric \( X \) are normalized as:

\[
x_{\text{norm}} = \frac{x - \mu(X)}{\sigma(X)},
\]

where \( \mu \) and \( \sigma \) are the center and the variance, respectively, of the metric \( X \). For the Mahalanobis distance the center and the variance are computed with robust estimators, which are the median and MAD, respectively. Smoothing and averaging have an effect on the center and variance of a distribution. Consequently, the data is normalized after the smoothing process.

For each simulation setup the optimized cutoff (OC) for the Mahalanobis distance was calculated. The OC (eq. (4.19)) aims to equalize the number of FPs and FN s. This classification process was repeated for different smoothing settings (as specified in section 5.1.1) and over different subsets of the measurements. Accordingly, the precision and recall was calculated for each cutoff distance. However, given that the optimized cutoff aims to equalize FP and FN, the precision and recall will yield the same result (see equations (6.1) and (6.2)). Similarly, SVM is applied to the measurements of all simulation settings and the resulting precision and recall are computed. For the SVM case, the values of precision and recall may differ.

The Tables 6.6, 6.7, and 6.8 show the performance results of the classification methods over the whole data set. The precision and recall of the Mahalanobis distance for both WLAN and W-CDMA are given in Table 6.6. The different scenarios are ranked according to their recall, the thirty-four best results are
<table>
<thead>
<tr>
<th>Scenario</th>
<th>SM</th>
<th>n/α</th>
<th>PR</th>
<th>RE</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ewma</td>
<td>0.06</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>ewma</td>
<td>0.12</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>ewma</td>
<td>0.18</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>smo</td>
<td>1</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>smo</td>
<td>0.06</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>smo</td>
<td>0.12</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>smo</td>
<td>0.18</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>smo</td>
<td>0.06</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>smo</td>
<td>0.12</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>smo</td>
<td>0.18</td>
<td>63.64</td>
<td>63.64</td>
<td>28</td>
<td>16</td>
<td>28933</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>smo</td>
<td>1</td>
<td>63.64</td>
<td>62.22</td>
<td>28</td>
<td>17</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>ewma</td>
<td>0.24</td>
<td>63.64</td>
<td>62.22</td>
<td>28</td>
<td>17</td>
<td>28932</td>
<td>16</td>
</tr>
<tr>
<td>13</td>
<td>smo</td>
<td>1</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28930</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>21</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>22</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>23</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>24</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>25</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>26</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>27</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>28</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>29</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>31</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>33</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
<tr>
<td>34</td>
<td>smo</td>
<td>0.24</td>
<td>63.64</td>
<td>60.87</td>
<td>28</td>
<td>18</td>
<td>28931</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 6.6: Precision (PR %) and recall (RE %) ranking of the Mahalanobis distance for W-CDMA and WLAN, smoothed with LMA and EWMA. sm. is the smoothing method applied, n and β are the smoothing constants of LMA and EWMA, respectively, (α = 1 − β).
Table 6: Precision (PR) and recall (RE) rankings of SVR for W-CDMA, smoothed (sm.) with LMA (α) and W-CDMA (γ).}

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Preci. (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W-CDMA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Preci. (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W-CDMA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Sm.</th>
<th>PR</th>
<th>RE</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
<th>Scenario</th>
<th>Sm.</th>
<th>PR</th>
<th>RE</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DjPAR</td>
<td>sa</td>
<td>63.64</td>
<td>93.33</td>
<td>28</td>
<td>2</td>
<td>28986</td>
<td>16</td>
<td>RqPAR</td>
<td>sa</td>
<td>3</td>
<td>40.91</td>
<td>100</td>
<td>18</td>
<td>0</td>
<td>28987</td>
</tr>
<tr>
<td>DjPAR</td>
<td>ew</td>
<td>0.06</td>
<td>63.64</td>
<td>93.33</td>
<td>28</td>
<td>2</td>
<td>28987</td>
<td>16</td>
<td>PAR,CL</td>
<td>sa</td>
<td>3</td>
<td>40.91</td>
<td>100</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>DjRgPAR</td>
<td>sa</td>
<td>1</td>
<td>63.64</td>
<td>93.33</td>
<td>28</td>
<td>2</td>
<td>28986</td>
<td>16</td>
<td>RqPAR,CL</td>
<td>ew</td>
<td>0.48</td>
<td>38.64</td>
<td>100</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>DjRgPAR</td>
<td>ew</td>
<td>0.06</td>
<td>63.64</td>
<td>93.33</td>
<td>28</td>
<td>2</td>
<td>28987</td>
<td>16</td>
<td>RqPAR,CL</td>
<td>ew</td>
<td>0.48</td>
<td>38.64</td>
<td>100</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>DjPAR,CL</td>
<td>sa</td>
<td>1</td>
<td>63.64</td>
<td>93.33</td>
<td>28</td>
<td>2</td>
<td>28986</td>
<td>16</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.54</td>
<td>34.09</td>
<td>100</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>DjPAR,CL</td>
<td>ew</td>
<td>0.06</td>
<td>63.64</td>
<td>93.33</td>
<td>28</td>
<td>2</td>
<td>28987</td>
<td>16</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.54</td>
<td>34.09</td>
<td>100</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>RqPAR</td>
<td>sa</td>
<td>1</td>
<td>63.64</td>
<td>93.33</td>
<td>27</td>
<td>1</td>
<td>28987</td>
<td>17</td>
<td>RqPAR,CL</td>
<td>sa</td>
<td>1</td>
<td>63.64</td>
<td>96.43</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>PAR,CL</td>
<td>sa</td>
<td>1</td>
<td>63.64</td>
<td>96.43</td>
<td>27</td>
<td>1</td>
<td>28987</td>
<td>17</td>
<td>PAR,CL</td>
<td>sa</td>
<td>1</td>
<td>63.64</td>
<td>96.43</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>DjPAR,Tz</td>
<td>ew</td>
<td>0.24</td>
<td>63.64</td>
<td>93.1</td>
<td>27</td>
<td>2</td>
<td>28987</td>
<td>17</td>
<td>RqPAR,CL</td>
<td>sa</td>
<td>1</td>
<td>63.64</td>
<td>96.43</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>DjPAR,Tz</td>
<td>ew</td>
<td>0.3</td>
<td>63.64</td>
<td>93.1</td>
<td>27</td>
<td>2</td>
<td>28987</td>
<td>17</td>
<td>DzPAR,Tz</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>DjRgPAR,Tz</td>
<td>ew</td>
<td>0.24</td>
<td>63.64</td>
<td>93.1</td>
<td>27</td>
<td>2</td>
<td>28987</td>
<td>17</td>
<td>DjRgPAR,Tz</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>DjRgPAR,Tz</td>
<td>ew</td>
<td>0.3</td>
<td>63.64</td>
<td>93.1</td>
<td>27</td>
<td>2</td>
<td>28987</td>
<td>17</td>
<td>DjRgPAR,Tz</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>DjPAR,CL</td>
<td>ew</td>
<td>0.24</td>
<td>63.64</td>
<td>93.1</td>
<td>27</td>
<td>2</td>
<td>28987</td>
<td>17</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.12</td>
<td>56.82</td>
<td>96.15</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>DjPAR,CL</td>
<td>ew</td>
<td>0.3</td>
<td>63.64</td>
<td>93.1</td>
<td>27</td>
<td>2</td>
<td>28987</td>
<td>17</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.12</td>
<td>56.82</td>
<td>96.15</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>DjRgPAR,CL</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
<td>28987</td>
<td>18</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.06</td>
<td>56.82</td>
<td>96.15</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>DjRgPAR,CL</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
<td>28987</td>
<td>18</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.06</td>
<td>56.82</td>
<td>96.15</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR,Tz</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
<td>28987</td>
<td>18</td>
<td>RqPAR,CL</td>
<td>ew</td>
<td>0.12</td>
<td>56.82</td>
<td>96.15</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR,Tz</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
<td>28987</td>
<td>18</td>
<td>RqPAR,CL</td>
<td>ew</td>
<td>0.12</td>
<td>56.82</td>
<td>96.15</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR,CL</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>96.3</td>
<td>26</td>
<td>1</td>
<td>28987</td>
<td>18</td>
<td>DzPAR</td>
<td>sa</td>
<td>2</td>
<td>54.55</td>
<td>96</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR,CL</td>
<td>ew</td>
<td>0.12</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>DzPAR,CL</td>
<td>sa</td>
<td>2</td>
<td>54.55</td>
<td>96</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>DjPAR</td>
<td>ew</td>
<td>0.18</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>DzPAR,CL</td>
<td>sa</td>
<td>2</td>
<td>54.55</td>
<td>96</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR</td>
<td>ew</td>
<td>0.24</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>DzPAR,CL</td>
<td>sa</td>
<td>2</td>
<td>54.55</td>
<td>96</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>RqPAR</td>
<td>ew</td>
<td>0.18</td>
<td>52.27</td>
<td>95.83</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR</td>
<td>ew</td>
<td>0.18</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.18</td>
<td>52.27</td>
<td>95.83</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>DzPAR</td>
<td>ew</td>
<td>0.18</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.18</td>
<td>52.27</td>
<td>95.83</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>DjRgPAR</td>
<td>ew</td>
<td>0.24</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>RqPAR,CL</td>
<td>ew</td>
<td>0.24</td>
<td>52.27</td>
<td>95.83</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>DjRgPAR</td>
<td>ew</td>
<td>0.3</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>RqPAR,CL</td>
<td>ew</td>
<td>0.24</td>
<td>52.27</td>
<td>95.83</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>DjRgPAR</td>
<td>ew</td>
<td>0.36</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>RqPAR,CL</td>
<td>ew</td>
<td>0.3</td>
<td>47.73</td>
<td>95.45</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>DjPAR,Tz</td>
<td>ew</td>
<td>0.12</td>
<td>59.09</td>
<td>92.86</td>
<td>26</td>
<td>2</td>
<td>28987</td>
<td>18</td>
<td>PAR,CL</td>
<td>ew</td>
<td>0.3</td>
<td>47.73</td>
<td>95.45</td>
<td>21</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.8: Precision (PR %) and recall (RE %) ranking of SVM for WLAN, smoothed (sm.) with LMA (sa) and EWMA (ew), $\alpha = 1 - \beta$. 

6.2 Multidimensional Analysis
shown. Table 6.7 shows the results of SVM for the W-CDMA data set. In the left part of the table, the results are ordered based on the precision while in the right part the data is ordered based on the recall. Table 6.8 shows the same results but then for the WLAN data set. Scenarios yielding a precision or a recall lower than 10% are not included in the listings.

The maximum precision and recall for the Mahalanobis distance classification method is achieved for the scenarios including $D_J$, $T_Z$, PAR, and CL. For both technologies the combination $\{T_Z, PAR\}$ performs best. $T_Z$ is the only parameter present in all the top ranked scenarios. For the W-CDMA case the smoothing method EWMA is most present, in the case of WLAN both EWMA and LMA are about equally represented. The smoothing coefficients for both methods are low. The coefficients in the W-CDMA case are on the average a step lower compared to the WLAN coefficients. The case where $n = 1$ in LMA yields the original data (not smoothed). $1 - \beta = 0.06$ is the smallest smoothing coefficient used for EWMA, and results in a data set close to the original one. This indicates that the QoS data doesn’t necessarily needs to be smooth to yield good jitter buffer starvation predictions. The precision and recall is slightly higher for WLAN than for the W-CDMA case. Both wireless technologies show maximum performance measures of about 65%. This means that 65% of the jitter buffer exhaustions were predicted correctly with the Mahalanobis distance classification method. Consequently, 35% of the identified starvations are actually no indicators thereof.

The precision and recall for SVM is not equal as in the Mahalanobis distance case. Table 6.7 shows the SVM ranking of the scenarios for the W-CDMA data set. It is observed that the highest ranked scenarios have small smoothing coefficients, i.e., $n \leq 2$ for LMA and $1 - \beta \leq 0.18$ for EWMA. This indicates that the QoS measurements does not necessarily need to be smoothed or averaged at the preprocessing stage of the analysis. The most reoccurring metric in all scenarios is $T_Z$ for the precision ranking, and PAR and $T_Z$ for the recall ranking. For a multitude of scenarios the classification accuracy reaches almost 50%. These scenarios were not smoothed (LMA: $n = 1$) or slightly smoothed (EWMA: $1 - \beta = 0.06$). The recall reaches in two cases almost 90%, this means that if the SVM classifies a measurements as an exhaustion, there is 90% chance that it is indeed an exhaustion. This is a good figure, though, in relation to the precision, this only applies to 40% of measured jitter buffer exhaustions.

Table 6.8 shows the SVM results for the WLAN dataset. It is observed that the composition of the scenarios are less complex, in the sense of the number of QoS parameters, compared to the W-CDMA case. In both the precision and recall ranking, the PAR is most present. Smaller smoothing constants score better rankings for the precision, but we see more higher smoothing coefficients compared to W-CDMA. The recall shows that the smoothing constants $n = 3$ for LMA and $1 - \beta \approx 0.50$ yield an index of 100%. This means that the mea-
measurements classified as exhaustions do not contain any false positives. A recall of 100% cannot be better, but the trade-off is that the corresponding precision is only as high as 40%. We can thus only classify 40% of measurements with 100% certainty, given the specific scenario and smoothing settings.

The next section analyses the cross-validation results, performance differences between WLAN and W-CDMA, as well as between the Mahalanobis distance and SVM are addressed. Also, the performance of scenarios containing the parameters $P_L$, $C_L$, and $R_R$ are analyzed, to learn whether they indeed contribute to the QoD definition.

### 6.2.2 Ranking of the performance

This section presents the cross-validation performance comparison of the different scenarios and classification methods. A stratified four-fold cross validation is repeated twenty-five times. Under general assumptions, the error in the cross-validation is an estimate for the error over an independent data set [80].

Figures 6.5 and 6.6 show the outcome of the cross-validation with the help of box-plots for the Mahalanobis distance and SVM. The numbers on the x-axis correspond to the scenario rank numbers in Table 6.6, 6.7 and 6.8. Remember that the precision and the recall for the Mahalanobis classifier are equal. For the W-CDMA case (Figure 6.5(a)), three groups of scenarios can be distinguished. The first three best performing scenarios are: \{PAR, $T_Z$; LMA: $n = 1$, \}, \{PAR, $T_Z$, $D_J$; LMA: $n = 1$ \} and \{$D_J,T_Z$; LMA: $n = 1$ \}. Considering the first and forth quantile (the whiskers of the box-plot), their performance is similar. Compared to the other scenarios, the cross-validation results median of these scenarios are higher than the rest. The second group of scenarios perform less than the first and better than the last group, though, the average difference is only 1.02%. The scenarios in the second group have the ranks \{1,2,3,5,6,7,8,9,10,12,14,15,16,17 \}. The third group consist of scenarios that perform on the average 6.11% less than the first. In the WLAN experiments (Figure 6.5(b)) three scenarios stand out again: \{PAR, $T_Z$; LMA: $n = 2$, \}, \{$T_Z$, $C_L$; LMA: $n = 2$ \} and \{PAR, $T_Z$, $C_L$; LMA: $n = 2$ \}. Based on the 10 following scenarios, the first three scenarios perform on the average 3.69% better. The performance of the other scenarios decreases gradually. Comparing the three best rated scenarios of W-CDMA and WLAN, it is observed that the Mahalanobis distance performs on the average 2.60% better in the WLAN environment.

The precision and recall for the Support Vector Machines (SVMs) looks different from the Mahalanobis classifier. The Interquartile range (IQR) and whiskers of the box plots are less long and the values for all scenarios are more consistent. This is particularly the case for the W-CDMA precision (Figure 6.5(c)). All values are located around 45% and the IQR ranges from 0% to 5%. SVM thus produces in this case consistent performance irrespective of the sce-
Figure 6.5: Cross-validation of the Mahalanobis distance for WLAN and W-CDMA, and Support Vector Machine (SVM) for WLAN.
Figure 6.6: Cross-validation of the precision and the recall for Support Vector Machine (SVM) – W-CDMA and WLAN.
nario. The first ten scenarios perform best, among them are: \{PAR,TZ\}, \{PAR,CL\}, and \{PAR,TZ,DJ\}. All the data of the top ranked scenarios here are not smoothed (LMA: \(n = 1\)). The average recall for the W-CDMA case (Figure 6.6(a)) is 78.85\%. No scenario shows a distinct performance difference in the ranking compared to its neighbors. The three top rated scenarios are: \{PAR,TZ\}, \{PAR,TZ,CL\}, and \{RR,PAR,TZ\}. As with the precision, these top rated scenarios are not smoothed either. The recall in SVM is about 15\% higher than for the Mahalanobis distance classifier, this is a better result. The precision and recall are, however, a trade-off within a particular data set. The corresponding precision for the top ranked W-CDMA recall scenarios show values below 50\% (see Table 6.7).

The precision for the highest ranked scenarios in SVM and WLAN are shown in Figure 6.6(b). The precision is on the average 13.60\% higher compared to the W-CDMA SVM case for the best ranked scenarios. The highest ranked scenarios include: \{DPj,PAR\}, \{Dj,RRj,PAR\}, \{DPj,PAR,CL\}, \{DPj,RR\}, and \{PAR,CL\}. As in the other cases, these measurements are not smoothed (LMA: \(n = 1\)). The recall (Figure 6.6(c)) is also higher for the WLAN measurements compared to the W-CDMA case. Some scenarios’ recall is close to 100\% and shows low variance for the cross validation. The last nine scenarios in the Figure show a lower recall (on the average 17.36\% lower) and the IQR is also larger. Among the best ranked scenarios are: \{RR,PAR\}, \{PAR,CL\}, and \{RR,PAR,CL\}. Low smoothing coefficients are performing best; SMA: \(n = 1\) and EWMA: \(1 − \beta = 0.06\).

In Figure 6.7 the influence of the smoothing coefficients is shown on the precision and the recall for both EWMA and LMA. The median precision and recall of the cross-validation is plotted for selected scenarios whose performance are in the top ranking of Tables 6.6 to 6.8. For both smoothing methods a descending trend is visible, indicating that smaller smoothing coefficients yield better results than larger coefficients. Also, the precision and recall for the Mahalanobis distance decays less fast than in the case of SVM. The smoothing coefficient \(1 − \beta < 0.48\) for EWMA in Figure 6.7(a) shows a smaller decreasing trend than for the \(1 − \beta ≥ 0.48\) part. Comparison of the the measurement points’ variance show that the difference in median performance is relatively small. One can concluded that the performance of the precision and recall is equal or slightly decreasing for EWMA smoothing factors \(1 − \beta < 0.48\). Furthermore, the performance for smoothing factors \(1 − \beta ≥ 0.48\) degrades significantly. Scenario 2 is an exception for these observations. In the case of the Mahalanobis distance, the smoothing factors \(1 − \beta ≈ 0.18\) seemingly performs best.

The precision and recall for LMA (Figure 6.7(b)) shows different behavior compared to EWMA. For the scenarios 3, 4 and 5, \(n = 1\) performs best, for scenarios 1, 2 and 6, \(n = 2\) performs best. A downwards trend is visible for all scenarios. For the SVM lines, the performance drops to 0 between \(3 < n < 8\) for all SVM scenarios, the Mahalanobis distance decays less fast and does not hit 0\% precision or recall. The Mahalanobis distance shows a performance peak
6.2 Multidimensional Analysis

Figure 6.7: Effect of the smoothing factor on the precision and recall for EWMA and LMA. The median precision and recall of the cross-validations is plotted. The scenarios cover both SVM and the Mahalanobis distance (MD) over WLAN and W-CDMA. It is observed that the precision and recall yields better results for small smoothing coefficients.
Table 6.9: Presence of the QoS metrics in the top ranked scenarios for the Mahalanobis distance (MD) and SVM.

<table>
<thead>
<tr>
<th></th>
<th>MD</th>
<th></th>
<th></th>
<th>SVM</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W-CDMA</td>
<td>WLAN</td>
<td>W-CDMA</td>
<td>WLAN</td>
<td>W-CDMA</td>
<td>WLAN</td>
</tr>
<tr>
<td></td>
<td>precision</td>
<td>precision</td>
<td>precision</td>
<td>recall</td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>T_Z</td>
<td>34</td>
<td>34</td>
<td>32</td>
<td>30</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>PAR</td>
<td>22</td>
<td>20</td>
<td>27</td>
<td>30</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>D_J</td>
<td>20</td>
<td>9</td>
<td>19</td>
<td>16</td>
<td>31</td>
<td>8</td>
</tr>
<tr>
<td>R_R</td>
<td>–</td>
<td>–</td>
<td>19</td>
<td>16</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>CL</td>
<td>13</td>
<td>20</td>
<td>18</td>
<td>19</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>P_L</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>IILD</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

for \( n = 2 \). Note that LMA \( n = 1 \) yields the original data set. Based on the results for the EWMA and LMA plots, it can be stated that smoothing does not increase notably the performance of the precision and recall for SVM in these measurements. For the Mahalanobis distance, a small smoothing factor LMA: \( n = 2 \) or EWMA: \( 1 - \beta \approx 0.18 \), may yield improved performance.

Table 6.9 shows the presence of the different QoS metrics in the top ranked scenarios of Tables 6.6 to 6.8. Distinct behavioral difference is observed for the Mahalanobis distance and SVM, and W-CDMA and WLAN. For the Mahalanobis distance, scenarios containing \( T_Z \) are most prominent among the top ranked scenarios. PAR is the second most represented metric in W-CDMA and WLAN. CL is useful for WLAN while \( D_J \) is more important in the W-CDMA case. The variance of \( R_R \), \( P_L \), and IILD is too small to be usable by the MCD for the Mahalanobis distance calculation. For \( P_L \) and IILD this is mainly because low number of packet losses we recorded.

For the SVM case, \( R_R \) has shown to be more useful than for the Mahalanobis distance classifier. \( T_Z \) is valuable for the W-CDMA case but much less for WLAN. On the other hand, PAR performs better for the WLAN case than for W-CDMA. The metrics \( D_J \), \( R_R \) and CL are about equally useful for the W-CDMA case. For WLAN, \( D_J \) is more eminent in the precision but does not participate in the recall as much. CL and \( R_R \) are more prominent in the recall than in the precision.

Overall, the ranking of the metrics for the difference classification methods and wireless technologies does not show consistency. This indicates that particular QoS metrics are more useful in certain situations/settings while others are less meaningful. Generally speaking however, \( T_Z \) and PAR are the metrics that are most present in the top ranked scenarios. From this point of view, the \( T_Z \) and PAR metrics are most meaningful in the context of these experiments. \( T_Z \) tells something about the packet inter-arrival time whereas PAR focuses on the packet arrival rate.
6.3 Experiment reflections

The classifier based on the Mahalanobis distance worked well in the previous research to map QoS metrics to user experience [27]. In the analysis of QoD of variable bit rate video, however, the Mahalanobis distance based classifier is less reliable. Nevertheless, the Mahalanobis distance classifier performs marginally better than SVM for the precision. The experiments with the classifier show that it is possible to identify around 65% of the jitter buffer starvations correctly. The rationale that unusual QoS metrics behavior has bad consequences for the jitter buffer only seem to work in about half of the cases. This means that there may be other causes that lead to jitter buffer starvation besides poor network conditions. Furthermore, based on analysis of the Mahalanobis distance it is observed that LMA and EWMA smoothing with low coefficients may help increase the performance of the precision or recall.

SVM was able to present a better recall compared to the Mahalanobis distance classifier. It is shown that SVM can provide a recall close to 100%, but with a poor precision. Similarly to the Mahalanobis distance, it was observed that EWMA nor LMA does not necessarily improve the performance of the system. Moreover, the QoS behavior in WLAN was observed to be less complex than for W-CDMA. And generally speaking, the performance of SVM is better in WLAN than in W-CDMA. Figures 6.8(b) and 6.9(b) shows examples of the classification methods for some of the best performing scenarios, for W-CDMA and WLAN. The Figures are given to increase the understanding of the classification methods and the performance metrics. In each Figure the decision boundary of the Mahalanobis distance classifier and SVM is shown. In Figure 6.8(b) the scenarios \( \{D_J, T_Z\} \) and \( \{PAR, T_Z\} \) are depicted. It is observed that the decision boundaries are similar for both scenarios. From the Figures it is clear that no linear or elliptical decision boundary can be defined that separates the jitter buffer starvations perfectly. This is because the two data sets are not separable, resulting in a source of noise (of FPs and FNs). The measurement points for jitter buffer starvations are mostly located in the area where PAR is low and \( T_Z \) is high. Basically it means that jitter buffer starvations occur when the packet-arrival rate is low and/or the inter-arrival time is high. This is not a surprising result. It is not excluded that other classification methods besides the ones presented here can perform better, maybe by representing the data differently.

In both classification methods we noted that observing a selected subset of QoS metrics yields more accurate results compared to analyzing all the available metrics together. The most useful metrics were found to be \( T_Z \) and PAR. These metrics contain most information with the purpose of quantifying QoD. \( D_J \) and CL were useful in particular cases, though the metrics contain more noise compared to \( T_Z \) or PAR. For the \( D_J \), as per equation (3.5), the accuracy increases when the number of packets increase per time interval. For low packet
Figure 6.8: Illustration of the decision boundary of SVM and the Mahalanobis distance (MD) classifier for the scenarios $\{D_J, T_Z\}$ and $\{PAR, T_Z\}$ in W-CDMA (LMA: $n = 1$). Measurements corresponding to jitter buffer starvations are depicted by circles, the others by triangles.
6.3 Experiment reflections

Figure 6.9: Illustration of the decision boundary of the SVM and the Mahalanobis distance (MD) classifier for the scenarios {D_J,T_Z} and {PAR,T_Z} in WLAN (LMA: n = 1, 2). Measurements corresponding to jitter buffer starvations are depicted by circles, the others by a triangles.
numbers $D_I$ might be less stable. For the rest of the metrics, $R_R$ showed to be less useful. $P_L$ and IILD did not contribute to the analysis of our dataset nor to a better definition of the classification models. This was mainly because they did not occur in large numbers in our data set. A dedicated study should be conducted to understand the impact of these metrics on the QoD of VBR video.

An important question is why the Mahalanobis distance classifier failed to perform better. The classifier is designed to identify unusual network behavior, which tentatively may affect the QoD. The main reason for the low precision and recall is the presence of noise in the measurements. After an analysis of the data, and shown in the Figures 6.8 and 6.9, one can find two different sources of noise. The first kind of noise appeared to be coming from very small jitter buffer starvations, time-wise. The duration of these starvation were around one second and often smaller. The QoS metrics show values during these events around the mean/median of their distribution. In other words, these measurement points cannot be distinguished whatsoever from the average metric values. This can be seen in the Figures 6.8(b) and 6.9(b), some measurements indicating jitter buffer exhaustion are located in the point clouds. Most likely these buffer starvations have a minimal effect on the video quality either, as they are very short. The second case of noise was coming from the longer buffer starvations. Suppose that the jitter buffer is starved, after a short time data starts to come in slowly, however, past their playout deadline. What happens then is that the data that missed its deadlines is thrown out from the jitter buffer without consideration of the decoder. Consequently, the QoS metrics indicate fairly normal conditions even though the jitter buffer is in essence empty.

Another question is how can the methods be improved to identify jitter buffer exhaustion more effectively. From the point of view of unusual network behavior, it can be stated that some QoS metric(s) is missing that indicates QoD problems well in the noisy circumstances. On the other hand, there are signs that we are measuring the effects of not only extreme network behavior, but also some artifacts of other processes. In the first case, there is a need to continue the hunt for more descriptive QoS metrics. Or make a clear distinguish between pauses and breaks (see Figure 1.1). This might be easier if the second point is also addressed. During the experiments hardly any packet losses were noticed. When the total number of packets sent per video was analyzed, it was observed that this count varied for the same video. For example, for the 200 kbit/s ED movie a minimum and maximum received number of packets of 45373 and 47978, respectively, were observed, with packet losses close to zero$^2$. The difference in the number of received packets amounts to about 5.5%. This observation is an indication that most likely also congestion and/or traffic control is reflected in the QoD measurements of the experiments, besides un-

\footnote{These values were gathered from the PVPlayer’s statistics after the movie finished playback.}
6.3 Experiment reflections

usual network behavior. Stream thinning was used in this case as the videos were precoded and UDP was used as transport layer protocol\textsuperscript{3}. Furthermore, if the QoS data from the video traces of different runs of the same video are superposed, then some points in the video are identified where jitter buffer starvations are more likely to occur, i.e., content dependency. These regions in the movies where starvations frequently occurred were mostly consecutive still frames, e.g., no motion or black screen\textsuperscript{4}. Still frames in a VBR video induce a short sudden drop in bit rate. This drop in rate affects the packet sizes and not necessarily the packet rate. The reason why the jitter buffer exhausts during these moments is not clear. One possible reason could be that the low bit rates induce spurious timeouts in RTP’s congestion/traffic control algorithm (depending on the implementation). This is a well know effect in TCP’s congestion control [68, 72]. Another possible reason is that the wireless technologies resource scheduling reacts adversely on a sudden drop in bit rate. Some other locations in the movie were also identified where the buffer size shrinks consistently and cause short buffer starvations. These events, however, could not be linked to low bit rates. It is clear that there are still open questions, especially related to the content dependent behavior of the jitter buffer and starvations.

For a W-CDMA networks it is not unusual to have low packet losses. Whereas for a WLAN networks high a packet-loss rate may be expected, for short to long periods of time. The video traces in the presented experiments show a low number of packet losses. The measurements were conducted in a real-live environment and it was shown that jitter buffers can exhaust in the absence of packet loss. The models yielding the best QoD estimates do not incorporate the $P_L$ or ILD metric. In other words, these models are $P_L$ and ILD independent, and could in theory be applicable in WLAN environments with an arbitrary packet loss. It is easily understood however that severe packet losses do have the ability to adversely affect the content of the jitter buffer. Thus the validity of the $P_L$ and ILD independent model need to be verified and adjusted accordingly under WLAN (and W-CDMA) with high packet loss conditions.

\textsuperscript{3}Via the logs of the streaming server these settings were validated.

\textsuperscript{4}Black screens and no motion in movies are used to build up suspense and tension for example.
Chapter 7

Conclusion and Future Work

Far better an approximate answer to the right question, than the exact answer to the wrong question, which can always be made precise . . .

John Wilder Tukey

7.1 Conclusion

The Quality of Delivery (QoD) of variable bit rate videos over a Wireless Local Area Network (WLAN) and Wide-band Code Division Multiple Access (W-CDMA) network was studied. This was done by relating network performance changes to the behavior of the Real-time Transport Protocol (RTP). More specifically, the state of the jitter buffer in RTP, a packet queue before the video decoder, was tracked under changing Quality of Service (QoS) conditions. Almost 1000 minutes of video were streamed in a live environment. Jitter buffer fluctuations and QoS metrics were recorded accordingly. After the acquisition, the QoS measurements were related to jitter buffer exhaustion via two classification methods; a Mahalanobis distance based classifier and Support Vector Machines (SVMs). The Mahalanobis distance classifier’s strength is to identify outlying measurement observations, while SVM was used as a comparative method.

The QoS metrics of interest were: Delay Jitter ($D_j$), Packet Loss ($P_L$), Zero Throughput Time ($T_Z$), Packet Arrival Rate ($PAR$), Reordering Rate ($R_R$), Inter-loss Distance ($ILD$), and Clumping ($CL$). A separate analysis of the QoS metrics revealed that the variance of $P_L$, $ILD$ and $R_R$ were very small. These metrics showed not to be useful for the Mahalanobis distance classifier in the QoD analysis. For SVM, the metrics $P_L$ and $ILD$ didn’t show to be valuable. The $CL$ and $D_j$ metric showed to contain information, though consistent correlation
with the jitter buffer states could not be identified. The metrics that were most
descriptive for the QoD process were PAR and $T_Z$. Both classification methods
showed optimal behavior for these metrics or a combination thereof. It was
shown that measurements with high $T_Z$ and low PAR values compared to their
average, are most likely to cause QoD degradation and hence video quality
impairments.

The performance of the classification methods after smoothing the QoS mea-
surements with Linear Moving Average (LMA) and Exponentially Weighted
Moving Average (EWMA) was studied as well. It was empirically found that
in both for WLAN and W-CDMA networks smoothing does not necessarily im-
prove the performance. In the case of the Mahalanobis distance a small smooth-
ing coefficient of LMA: $n = 2$ or EWMA: $1 − \beta \approx 0.12$ slightly improves the per-
formance. For SVM it was shown that smoothing did not increase the accuracy
of the classification method. For QoE measurements, smoothing of parameters
with large coefficients has shown to improve the accuracy of models. For the
study QoD however, this is not the case.

The precision and recall of the Mahalanobis distance classifier in the best
cases is around 65%. Unfortunately this is not reliable enough for many practi-
cal applications. The Mahalanobis distance classifier performs marginally bet-
ter than SVM for the precision. SVM shows a larger recall. A recall of 95% was
obtained based on $T_Z$, PAR and $D_J$ metrics. This is a good result, however the
trade-off is a relatively small precision.

The use of the Mahalanobis distance as a classifier was motivated by the
assumption that extreme network behavior affects QoD. The assumption did
not show to be yielding accurate results. By studying the data set more closely
it was observed that extreme network behavior indeed manifests and affects
the QoD, but there are artifacts of other processes too. These artifacts create
noise in the data set and hence the Mahalanobis distance performs poorly. SVM
suffers from the same problems as the Mahalanobis distance classifier does.

### 7.2 Future work

Future work includes the research on the sources of noise in the QoS measure-
ments. It was observed that not only extreme network behavior shows to be
harming QoD but also other effects. Candidates to be analyzed are the conges-
tion and traffic control of the streaming server. Traffic and congestion control
is designed to enhance the QoD but there are indications in our data set that at
some points the control mechanisms took wrong actions. Therefore the conges-
tion and traffic control algorithms should be studied more closely. Accordingly,
the congestion control can be amended to reduce the probability of jitter buffer
starvations, hence increasing the QoD. The relation between the control mecha-
nism and the video content also needs to be studied.

Another noise source in the measurements was emanating from content dependent events. It was observed that QoD impairments often manifested at the same time in the video files. The indications are that the streaming server is degradation when specific content patterns occur. The causes of this effect are open to research.

In the classification methods the misclassification costs of a FN and FP was considered equal. In reality, a FN might be deemed more severe then a FP. This can be expressed in assigning different costs for a FP and FN. Changing the cost affects the outcome of the classification methods and the precision and recall will yield different results. Also, to compare the methods and scenarios to each other, statistical significant test can be used. For example, the Friedman test and analysis of variance (ANOVA) are methods that can compare groups of objects at once. Pairwise post-hoc test can then be used further, if the null-hypothesis of the group-methods is rejected. These test can tell us how significant the performance differences actually are. More conventional performance measures, besides the precision and recall, can be used for comparing the performance of classification methods and finding their optimal thresholds. A common measure is the Area Under Curve (AUC) for example. For this measure to function, the classification methods must produce posterior probabilities. SVM can produce posterior probabilities and can also handle different misclassification costs. The Mahalanobis distance classifier however does not support these features yet and should be extended accordingly. The classification costs can be added to equation (4.19), whereas the posterior probabilities could be generated by a properly dimensioned sigmoid function, as the case with SVM. Other transformation methods however might be more appropriate, this should be investigated more closely. Furthermore, statistical significance test based on ranks were not used in this work (e.g. the Friedman test) because the ranks showed many ties. In such cases the test may be biased severely and hence falsely reject or accept the null-hypothesis. The main cause of the many ties is that relatively low number of jitter buffer starvations were recorded. Therefore, the precision and recall can only take a limited number of values if computed over the outcomes of the $k$-fold cross-validation. This can be addressed by measuring more traces and jitter buffer starvations.

Once these points are investigated and addressed, one should reconsider the applicability of the Mahalanobis distance classification method in the context of QoD quantification. Moreover, one should look further whether other machine learning methods could be better performing than the Mahalanobis distance classifier or SVM. When the current methods are optimized or another good method is found, the models need to be tested again on the prediction of QoD impairments. An interesting next step is to look into the future and try to predict forthcoming QoD evolvements. But this demands for good models
and methods to describe the QoD process. Once this is found, one can anticipate possible QoD deterioration to maximize the user experience of mobile multimedia.
Appendix A

Code Snippets and Log Files

A.1 UDP tunnel implementation

The User Datagram Protocol (UDP) tunnel used in the measurements was implemented in the Linux kernel. Besides encapsulating data in a UDP header, a measurement header was also added which consists of a 32 bit timestamp. Sequence numbers are also important for QoS measurements, though to save some bytes the sequence number of the Real-time Transport Protocol (RTP) was used. The UDP tunnel is implemented as a virtual interface kernel module, which has the advantage to be loadable during runtime. Kernel level encapsulation is also preferred over application level encapsulation because of performance motivations [79]. The module is based on the IP/IP tunneling available in any Linux distribution. Only a few lines should be altered and added to convert the IP/IP tunnel into a UDP tunnel. The sequel of this section shows the necessary amendments in the source code of the ipip.c file for a Linux 2.6.27 kernel.

The major difference between the UDP and IP/IP tunneled packets is that UDP tunneled packets are caught by the transport layer, the outer UDP header is stripped of, and then sent back to the network layer to be processed again. In contrast, IP/IP packets traverse the network stack in a straight line up and down (with reference to the Open Systems Interconnection (OSI) model). Two functions should be changed to achieve this: ipip_rcv which receives packets and ipip_tunnel_xmit which send packets. Also, a packet catcher listening to some UDP port should be installed that redirects the packets to the ipip_rcv function.

The ipip_tunnel_xmit appends the IP, UDP header and the measurement header. The IP/IP code adds already the IP header, we need to take care of the rest. First the proper Maximum Transmission Unit (MTU) of the packet must be calculated to account for the extra IP, UDP and measurement header.
To catch a packet at an UDP port an in-kernel UDP server can be set up. The ipip_init function is the proper location to initialize a UDP server. This function is called when the module is initially loaded. The custom code for the UDP server looks like follows.

```c
struct ip_tunnel *tunnel = netdev_priv(dev);
if (tiph->frag_off)
    mtu = dst_mtu(&rt->u.dst) - tunnel->hlen;

memset(&socket_address, 0, sizeof(struct sockaddr_in));
socket_address.sin_family = AF_INET;
socket_address.sin_addr.s_addr = htonl(INADDR_ANY);
socket_address.sin_port = htons((unsigned short) port_number_source);

if (sock_create_kern(AF_INET, SOCK_DGRAM, IPPROTO_UDP, &udpsocket) < 0) {
    return -EIO;
}

udpsocket->sk->sk_data_ready = receive_message;

if (udpsocket->ops->bind(udpsocket, (struct sockaddr*)&socket_address, sizeof(struct sockaddr_in))){
    sock_release(udpsocket);
    return -EIO;
}

The following code is added to the source code to catch the packets and redirect them to the ipip_rcv function.

```c
static void receive_message(struct sock *sk, int length){
    read_lock(&sk->sk_callback_lock);
    struct sk_buff *skb;
    while((skb = skb_dequeue(&sk->sk_receive_queue)) != NULL){
        skb_orphan(skb);
        ipip_rcv(skb);
    }
    read_unlock(&sk->sk_callback_lock);
}
```

The ipip_rcv function that strips of the headers must account for the UDP and measurement header as follows.

```c
int offset = sizeof(struct measurement_header) + sizeof(struct udphdr);
...
if (!pskb_may_pull(skb, sizeof(struct perimeter_header) + sizeof(struct udphdr))
    goto drop_nolock;
iph = ip_hdr(skb);
phdr = (struct perimeter_header*) (iph + sizeof(struct udphdr));
...
Then the UDP header needs to be written to the packet. In this case we write our headers after the IP is installed.

```c
udph = ((struct udphdr*)((struct iphdr*)iph+1));
udph->source = htons(port_number_source);
udph->dest = htons(port_number_dest);
udph->len = htons(skb->len - sizeof(struct iphdr));
udph->check = 0;
phdr = ((struct measurement_header*)((struct udphdr*)iph+1));
phdr->time_msec = htonl((this_time.tv_sec*1000) \n + (this_time.tv_usec/1000));
```

Last, one should not forget to set the proper MTU of the virtual interface itself. The hard_header_len and mtu in the function ipip_tunnel_setup should therefore be adjusted as follows:

```c
dev->hard_header_len = LL_MAX_HEADER + sizeof(struct iphdr) \n + sizeof(struct udphdr) + sizeof(struct measurement_header);
dev->mtu = ETH_DATA_LEN - sizeof(struct iphdr) \n - sizeof(struct udphdr) - sizeof(struct measurement_header);
```

### A.2 QoS metrics calculation

The snippet of C++ code below was used to process incoming packets and forms the basis of the computation of the QoS measurement analysis. The QoS metrics included are the reordering rate, packet loss rate, delay jitter, zero throughout time, Inter-loss distance, packet rate, and clumping. The function is expected to be called every time a packet arrives. The parameters of the function are: the sequence number, the departure timestamp from the sender’s side, and the local arrival timestamp. Throughout the source code two predefined values are used, TIME_INTERVAL: the time interval (in milliseconds) over which the QoS parameters are computed, and MAX_REORDERING: the length of the reordering array. Information on the latter is found in section 3.3.

```c
void processPacket(int sequenceNr,
                   unsigned int departureTime,
                   unsigned int arrivalTime)(
    totalPacketCount++;
```
The above snippet process the timestamps and sequence numbers of incoming packets. The piece of code then calls at the appropriate time the functions that handle the calculation of each QoS metric. The QoS metrics are computed as follows.

- **The Delay Jitter ($D_J$)** is updated as follows

```c
void updateDelayJitter(unsigned int departureTime,
                       unsigned int arrivalTime) {
    // Update the delay jitter value.
    if (departureTime != arrivalTime) {
        // Calculate the delay jitter.
        delay_jitter += departureTime - arrivalTime;
    }
```

The code snippet checks if we are still in the same time interval. If not, it prints a summary and resets all metrics. It then creates a new data structure for the arrived packet, registers it with the ZTT algorithm, and updates the CL (Code Snippets). It also checks the sequence number to determine if the packet is reordered or if it's a new reference. The snippet includes logic to handle packet reordering and calculate QoS metrics such as delay jitter.
/Compute the DJ with reference to the absolute reference
unsigned int unsync_departure = \\
  departureTime - delay_Departure_Reference;
unsigned int unsync_arrival = \\
  arrivalTime - delay_Arrival_Reference;
unsigned int inter_packet_delay_variation = 0;
if(unsync_departure > unsync_arrival)
  inter_packet_delay_variation = \\
    unsync_departure - unsync_arrival;
else
  inter_packet_delay_variation = \\
    unsync_arrival - unsync_departure;
if(dj_min > inter_packet_delay_variation)
  dj_min = inter_packet_delay_variation;
delay_Jitter_Sum += inter_packet_delay_variation;
delay_Jitter_Sum_Squared += \\
  pow(inter_packet_delay_variation,2);
delay_Jitter_Count++;
}

The D_J is then calculated as

double calculateDelayVariation(){
  double ipdv_count = (double) delay_Jitter_Count;
double ipdv_sum = (double) delay_Jitter_Sum;
double ipdv_sum_sqrt = (double) delay_Jitter_Sum_Squared;

double first_part = 1/(ipdv_count-1)*ipdv_sum_sqrt;
double second_part = ipdv_count/(ipdv_count-1)* \\
  pow(ipdv_sum/ipdv_count,2);
  return sqrt(first_part-second_part);
}

• Clumping (CL) is updated as follows

void updateCL(unsigned int timestamp){
  if(CLpreviousTimeStamp == timestamp){
    CLcount++;
  }
  if(CLpreviousTimeStamp != timestamp){
    if(CLcount > CLmax){
      CLmax = CLcount;
    }
    CLpreviousTimeStamp = timestamp;
    CLcount = 0;
  }
}

• Zero Throughput Time (TZ) is updated as follows
void updateZTT(unsigned int arrivalTime) {
    if ((previous_Arrival_Time / TIME_INTERVAL) != \\
        arrivalTime / TIME_INTERVAL \\
        && previous_Arrival_Time != 0) {
        zttStandard = arrivalTime - previous_Arrival_Time;
        previous_Arrival_Time = arrivalTime;
    }
}

• Packet Loss (PL) and Inter-loss Distance (ILD) are updated as follows

void recordMissingPackets(int start, int end) {
    int located = 0;
    functionLoss++;
    vector<packet>::iterator iter;
    for (iter = myWaitingPackets.begin();
        iter != myWaitingPackets.end(); ++iter) {
        if (iter->sequenceNr >= (start - (MAX_REORDERING))
            && iter->sequenceNr < (end - (MAX_REORDERING))) {
            located++;
        }
    }
    int losses = (end - start) - located;
    if (totalPacketCount > MAX_REORDERING + 1) {
        packetLoss += losses;
        packetCount += losses;
        if (losses > 0) {
            if (ildLastLoss != -1) {
                ildSum += end - ildLastLoss;
                ildCount++;
            }
            ildLastLoss = end;
        }
    }
}

The ILD is then computed as

double getILD() {
    double outcome = 0;
    if (ildSum != 0 && ildCount != 0) {
        outcome = 1 / (double(ildSum) / double(ildCount));
    }
    return outcome;
}

A.3 Measurement log files

A.3.1 RTP log files

In the RTP log file information about the following events is stored:
1. When a packet arrives to the jitter buffer. This is identified by the lines which second column contains the string PA. The columns represent: log sequence no., log ID, RTP stream Synchronization Source (SSRC), content Multipurpose Internet Mail Extensions (MIME) type, RTP sequence no., arrival timestamp, RTP delay variation, no. of packets in jitter buffer, and packet burst indicator;

2. When a packet is retrieved by the decoder from the jitter buffer. This is identified by the lines which first column contains retrieveElement. The elements in the columns are: log ID, read offset, no. of elements left, last retrieved RTP sequence no.;

3. When a receiver report is sent to the streaming server. This is identified by the lines which second column contains the string RP. The columns represent: log sequence no., log ID, RTP sender SSRC, content MIME type, total packets lost, maximum sequence no. received, RTP delay variation, no. of bytes in jitter buffer, no. of packets in jitter buffer.

All time units are in milliseconds. The three different log events may occur in an arbitrary sequence in the log file. A typical log file looks as follows.

```
316 PA 704341308 RTP/video/H264 25590 4256 3688 270 0
317 PA 704341308 RTP/video/H264 25591 4257 1 271 0
318 PA 704341308 RTP/video/H264 25592 4258 1 272 0
319 PA 704341308 RTP/video/H264 25593 4261 3747 273 0
320 PA 704341308 RTP/video/H264 25594 4262 1 274 0
321 PA 704341308 RTP/video/H264 25595 4263 1 275 0
2460 RP 1326087897 RTP/video/H264 0 25595 1163 0 113314 276
```

A.3.2 Kernel measurement log files

A line is written to the kernel log each time a packet arrives at the tunnel UDP socket. The log commonly looks as follows:

```
[ 2808.680847] packet dep 178019082 arr 155154740 sqnr 34280 ssrc 2041604118
[ 2808.720886] packet dep 178019123 arr 155154780 sqnr 34281 ssrc 2041604118
[ 2808.761718] packet dep 178019158 arr 155154821 sqnr 34282 ssrc 2041604118
[ 2808.800811] packet dep 178019200 arr 155154860 sqnr 34283 ssrc 2041604118
[ 2812.499847] packet dep 178022901 arr 155158559 sqnr 34284 ssrc 2041604118
[ 2812.510955] packet dep 178022901 arr 155158571 sqnr 34285 ssrc 2041604118
[ 2812.512329] packet dep 178022901 arr 155158572 sqnr 34286 ssrc 2041604118
[ 2812.521453] packet dep 178022901 arr 155158581 sqnr 34287 ssrc 2041604118
[ 2812.522766] packet dep 178022901 arr 155158582 sqnr 34288 ssrc 2041604118
```

Here, the value between square brackets is the time of the kernel clock when the packet arrived add the tunnel driver, followed by: packet departure
time and arrival time (both in milliseconds), RTP sequence number, and the RTP stream SSRC. Note that the time stamp between brackets is in essence the same as the arrival time stamp, but then in a larger unit. Thanks to the RTP sequence number in the kernel measurement log file and the RTP log file, both the RTP and kernel log files can be connected.
Appendix B

List of Publications

B.1 List of publications


6. İckin, S., De Vogeleeer, K., Fiedler, M., and Erman, D. On the choice of performance metrics for user-centric seamless communication. In Third Euro-
NF IA.7.5 Workshop on Socio-Economic Issues of Networks of the Future (2010), EuroNF Network of Excellence, pp. 4–5.


Bibliography


[86] Norton, W. B. Video internet: The next wave of massive disruption to the u.s. peering ecosystem. In Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT) (February 2007).


## Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>3G</td>
<td>3rd Generation</td>
</tr>
<tr>
<td>ABC</td>
<td>Always Best Connected</td>
</tr>
<tr>
<td>AMR</td>
<td>Adaptive Multi Rate</td>
</tr>
<tr>
<td>ANOVA</td>
<td>analysis of variance</td>
</tr>
<tr>
<td>AUC</td>
<td>Area Under Curve</td>
</tr>
<tr>
<td>BBB</td>
<td>Big Bug Bunny</td>
</tr>
<tr>
<td>BTH</td>
<td>Blekinge Institute of Technology</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>CCDF</td>
<td>Complementary Cumulative Distribution Function</td>
</tr>
<tr>
<td>CI</td>
<td>Continuity Index</td>
</tr>
<tr>
<td>CL</td>
<td>Clumping</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>Dj</td>
<td>Delay Jitter</td>
</tr>
<tr>
<td>DSS</td>
<td>Darwin Streaming Server</td>
</tr>
<tr>
<td>DV</td>
<td>Delay Variation</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial-of-Service</td>
</tr>
<tr>
<td>ED</td>
<td>Elephants Dream</td>
</tr>
<tr>
<td>EWMA</td>
<td>Exponentially Weighted Moving Average</td>
</tr>
<tr>
<td>FAST-MCD</td>
<td>Fast Minimum Covariance Determinant</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In, First Out</td>
</tr>
<tr>
<td>FN</td>
<td>false negative</td>
</tr>
<tr>
<td>FP</td>
<td>false positive</td>
</tr>
<tr>
<td>FV</td>
<td>Flash Video</td>
</tr>
<tr>
<td>GOP</td>
<td>Group of Pictures</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GSP</td>
<td>Grid Service Providers</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transport Protocol</td>
</tr>
<tr>
<td>HDTV</td>
<td>high-definition television</td>
</tr>
<tr>
<td>HSDPA</td>
<td>High-speed Downlink Packet-data Access</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>i.i.d.</td>
<td>independent and identically distributed</td>
</tr>
<tr>
<td>IILD</td>
<td>Inverse Inter-loss Distance</td>
</tr>
<tr>
<td>ILD</td>
<td>Inter-loss Distance</td>
</tr>
<tr>
<td>IPDV</td>
<td>Inter-packet Delay Variation</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>IQR</td>
<td>Interquartile range</td>
</tr>
<tr>
<td>LMA</td>
<td>Linear Moving Average</td>
</tr>
<tr>
<td>MAD</td>
<td>Mean Absolute Deviation</td>
</tr>
<tr>
<td>MBR</td>
<td>Multiple Bit Rate</td>
</tr>
<tr>
<td>MCD</td>
<td>Minimum Covariance Determinant</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>MIME</td>
<td>Multipurpose Internet Mail Extensions</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimation</td>
</tr>
<tr>
<td>MPEG</td>
<td>Motion Picture Experts Group</td>
</tr>
<tr>
<td>MRA</td>
<td>Multiple Regression Analysis</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum Transmission Unit</td>
</tr>
<tr>
<td>MVE</td>
<td>Minimum Volume Estimator</td>
</tr>
<tr>
<td>MVT</td>
<td>Multivariate Trimming</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>OC</td>
<td>optimized cutoff</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OWD</td>
<td>One-Way Delay</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-Peer</td>
</tr>
<tr>
<td>PAR</td>
<td>Packet Arrival Rate</td>
</tr>
<tr>
<td>PCA</td>
<td>Principle Component Analysis</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PDV</td>
<td>Packet Delay Variation</td>
</tr>
<tr>
<td>PIT</td>
<td>Packet Inter-arrival Time</td>
</tr>
<tr>
<td>PL</td>
<td>Packet Loss</td>
</tr>
<tr>
<td>PSS</td>
<td>Packet-switched Streaming Service</td>
</tr>
<tr>
<td>QCIF</td>
<td>Quarter Common Intermediate Format</td>
</tr>
<tr>
<td>QP</td>
<td>quadratic programming</td>
</tr>
<tr>
<td>QTSS</td>
<td>QuickTime Streaming Server</td>
</tr>
<tr>
<td>QoD</td>
<td>Quality of Delivery</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoP</td>
<td>Quality of Presentation</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Index

3G, 51
3GPP PSS, 18

adaptive streaming, 18
Adobe, 12
Akamai, 12
AMR, 17
arithmetic mean, 36, 37
Arlos, 39
attribute, 21

BBB, 50
Bernoulli process, 29
Bienaymé equality, 55
box-and-wisker, 39
BTH, 51
bursty traffic, 28

CBR, 17, 25
Cl, 15
Cisco Systems, 11
CL, 30, 65, 68, 74, 75
class, 21
clock resolution, 31
computational complexity, 46
covariance matrix, 37, 38
distance
  Euclidean, 40, 42
  Mahalanobis, 3, 4, 40, 42, 46
  Minkowski, 40
Dj, 18, 26, 27, 55, 57, 62, 68, 74, 75, 81
dSS, 13, 50–52

ED, 50
entropy, see information
Erman, 15
Euclidean, see distance
EWMA, 74, 75, 81
false
  negative, 44, 69, 74, 75, 81
  positive, 44, 69, 74, 75, 81
fast streaming, 13, 18
FAST-MCD, 38
feature vector, 21
Fiedler, 39
function
  contrast, 36
  score, 35
Gilbert model, 29
GOP, 17
GPAC, 50
HD, 3, 12, 17
HSDPA, 39, 64
İçkin, 30–32
IETF, 26
IILD, 29, 66, 68
ILD, 29
inter-
  arrival time, 28, 53–56, 58, 59
  departure time, 31, 52
Index

packet arrival, 63
sampling time, 52
IPDV, 26
ITU, 2, 26
jitter buffer, 5, 13
kernel, 91
trick, 46
kurtosis, 31

Links, 53
Linux, 25, 31, 50, 52, 54, 91
LMA, 74, 75, 81
location model, 18
M-estimator, 36, 37
MAD, 38, 39, 70
Mahalanobis, see distance
Markov model, 29
MBR, 17
MCD, 38, 39, 42
measurements
Qo, 91
median, 36
MIME type, 97
Minkowski, see distance
MLE, 35
MP4BOX, 52
MTU, 91
netem, 54, 55
NetFlix, 11
networking index, 11
NIC, 25, 52
Nokia, 2
optimized cutoff, 70
outlier, 4
overbuffering, 13
OWD, 23, 26, 62
P2P, 8, 15
PacketVideo, 32, 51
PAR, 25, 67, 68, 74, 75, 81
PCA, 6
PDV, 26, 27
PERIMETER, 2, 3
ping, 53–55, 57
PIT, 31
P_l, 28, 57, 66, 68
Poison, 14
precision, 5, 69, 78
PSS, 12, 18
QCIF, 50
Quality of
QoD (Delivery), 3–5, 8, 16, 20, 21, 67, 81, 87
QoE (Experience), 2, 3
QoP (Presentation), 3
QoS (Service), 1, 3, 4, 20, 23, 25, 61, 93
Real Networks, 12
real-time, 5, 11, 26, 28, 33, 46
rebuffering, 15
recall, 5, 69, 78
reordering, 58
reordering density, 29, 30
RFC
2326, 12
3357, 28
3393, 26, 32
3550, 12, 26, 28, 32, 52
4737, 29
5481, 26
robust statistics, 4
round-robin, 27
Rousseeuw, 38
R, 30, 58, 66, 68, 78
RTCP, 12, 16
RTP, 12, 13, 26, 28, 32, 33, 49, 51–53, 67, 85, 87, 91
RTSP, 12
RTT, 53, 55
seamless communication, 1
sequence number, 12, 52, 93
Shurygin, 37
smartphone, 1
still frames, 85
stratified cross validation, 69
stream thinning, 17, 85
support vector, 44
SVC, 17
SVM, 4, 44, 47
tcpdump, 52, 54, 58
timestamp, 12, 52, 93
true negative, 69
ture positive, 69
TTI, 64
T_Z, 18, 27, 63, 68, 74, 75, 81
Ubuntu, 52
UDP, 91
Reliable, 13, 17, 52
server, 92
socket, 97
tunnel, 59, 91
user experience, 1–6, 20, 31, 64, 81, 90
Varga, 31
VBR, 8, 17, 25, 67, 85
Verdoy, 27
video quality impairments, 2
video streaming, 11
VoD, 11
VoIP, 13
W-CDMA, 39, 61, 74, 75
web-browsing, 28
WiFi, 51
WLAN, 61, 74, 75
WRT-54GL, 53
YouTube, 11, 13
Zepernick, 2
Several factors affect the perceived quality of a video streamed over a network. Technical aspects related to the codec and the video play an important role besides the delivery of the video content in a timely and error-free manner. If the streamed video frames do not arrive in time, temporal artifacts may become observable during the video playback. This has potentially an adverse effect on the user experience. The study of temporal artifacts during the video playback is the area of concern for Quality of Delivery.

The licentiate work investigates the Quality of Delivery for Variable Bit Rate video sent over wireless technologies. The ability of the network to deliver data to the video’s jitter buffer before its playout deadline is studied. Jitter buffer exhaustions are of particular interest as they tell something about the Quality of Delivery. A number of experiments are conducted where a set of videos with different bit-rates are streamed to a mobile device over a wireless LAN and a W-CDMA network. The video streams are recorded and analyzed based on specific Quality of Service parameters and are related to the states of the jitter buffer. The statistical tools Support Vector Machines and the Mahalanobis distance are applied to the parameters to obtain a model that can classify the jitter buffer states. The performance evaluation indicates that the Mahalanobis distance can classify the jitter buffer state marginally better than the Support Vector Machines. However, the Support Vector Machines can produce more reliable predictions compared to the Mahalanobis distance. It is also observed that the perceived Quality of Delivery is not only affected by the behavior of the wireless networks but also by the behavior of the streaming server. Finally, the ability of the Quality of Service parameters to describe the Quality of Delivery is studied. The results indicate that metrics based on the packet arrival-rate and packet inter-arrival time are most suitable in this particular case.