FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Quality of Service for cloud services using control theory

Quality of service for VoIP cloud services using PI Controller based on Horizontal Scalability

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering and Internet Computing

eingereicht von

Ing. Oliver Schaumüller, BSc
Matrikelnummer 0726262

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof Ivona Brandić

Wien, 12. Oktober 2018

Oliver Schaumüller	Ivona Brandić

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Quality of Service for cloud services using control theory

Quality of service for VoIP cloud services using PI Controller based on Horizontal Scalability

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Ing. Oliver Schaumüller, BSc
Registration Number 0726262

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof Ivona Brandić

Vienna, 12th October, 2018

Oliver Schaumüller   Ivona Brandić

Erklärung zur Verfassung der Arbeit

Ing. Oliver Schaumüller, BSc
Pfingswiese 19, 2011 Sierndorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. Oktober 2018

Oliver Schaumüller

Danksagung

Vielen Dank an meine Familie für ihre Unterstützung in jeder Situation, meine Freunde für deren Geduld und meiner Betreuerin für ihre Unterstützung.

Acknowledgements

Many thanks to my family for assistance in all manner of situations, to my friends for their patience and support and my advisor for her helpful mentoring.

Kurzfassung

Horizontales Elasizitätsmanagement ist eines der zentralen Kernkonzepte, welches Clouds im Gegensatz zu herkömmlichen Rechenzentren unterscheidet. Dies erlaubt es autonom auf Änderungen der Anforderungen zu reagieren, wodurch Nachfrage basierend scheinbar endlos Ressourcen zur Verfügung gestellt werden können.

Das Datenvolumen von VoIP Netzwerken steigt jährlich, da höhere Bandbreiten verfügbar sind. Dieser Effekt wird vor allem durch mobile Geräte wie Smartphones getrieben, welche Applikationen wie Skype, Whatsapp, Viber oder Facetime ermöglichen und die der Anwender griffbereit ständig bei sich hat. Service Provider müssen sich der ständig ändernden Auslastung stellen und ihre Services dementsprechend designen. Besondere Ereignisse wie Silvester, Katastrophen oder weltweite Sportevents lassen die Auslastung kurzfristig hochschnellen.

Um mit den Anforderungen fertig zu werden, setzen die Service Provider auf horizontale Skalierbarkeit. Daher beschäftigt sich diese Arbeit mit Methoden der Regelungstechnik, um diese Prozesse zu steuern. Als Steuerungsgrößen werden qualitative Größen aus der VoIP Welt verwendet. Die Arbeit versucht zu beantworten, ob klassische Methoden der Regelungstechnik für die Steuerung von Clouds geeignet subd.

In der Arbeit wird Anhand eines in R simulierten Modells die Steuerung eines solchen Prozesses simuliert und gezeigt wie auf Störgrößen reagiert werden kann.

Abstract

Horizontal elasticity management is one of the core concepts which distinguishes clouds from common data centers. This concept allows to react on changes of the requirements easily on demand. Therefore, the impression of infinite resources arises.

The data volume of VoIP networks increases year by year driven by mobile devices such as smartphones. These devices enable applications like Skype, Whatsapp, Viber, Facetime and others to be carried in the user's pocket. Service providers face fluctuation in demand of their services and design their applications for those purposes. Especially events like New Year's Eve, catastrophes or worldwide sport events let the demand skyrocket in a short time window.

To handle the gap in demand, cloud providers need to scale their applications autonomously. As a result, this work is dedicated to control theory methods to control such processes. As a control variable, qualitative metrics of VoIP technology will be used. This work's purpose is to answer the question if classical control theory can be used to accomplish the task of horizontal elasticity in clouds.

A model simulated in R will show how such control loops react to disturbances.

Contents

CHAPTER 1

Introduction

Cloud Computing has changed the industry tremendously and allows service providers to elastically scale their services. One of the key services in every user's life is communication. Whether it is a skype call, a conference call at work, remote support, a teaching lesson on Chinese or even computer games where people play in teams, all these conversations are operated by modern VoIP technology.
All those services have in common that they are cloud services provided for users. Users expect these services to function with proper performance and speech quality.

Since the term cloud computing has been heavily used in marketing campaigns, this work extends the definition from Martin Sölkner's diploma thesis [Sö] as follows:

> A cloud represents $n \geq 2$ physical machines (PM) which are connected over a network with each other. Virtual Machines (VM) are hosted on these PMs and simulate hardware for users. Many VMs can run on one PM. It is possible to execute software applications on a VM.
>
> A cloud service is a software application that is distributed on several VMs but acts transparent to users.

As shown in figure 1.1, if fixed resources are available, costs are always the same whether used or not. In the second picture in figure 1.1, a virtualized environment with horizontal scalability shows that the costs are rising and falling with the workload.

> Even though virtualization needs additional resources, it still pays off due to gained flexibility in terms of scalability.

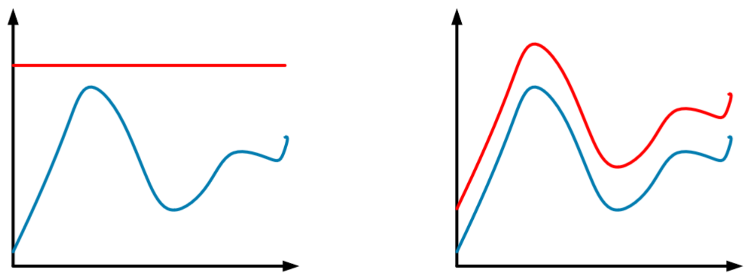


Figure 1.1: Dynamic resource allocation in cloud infrastructure

In case of higher customer demand, the service provider can dynamically adjust to new demand on computing resources. As an example for this work shall serve a VoIP service provider such as Skype. VoIP service providers provide their users communication solutions according to their demand. It is in the service provider's interest to maintain as little resources as possible due to the higher costs involved. If the service provider had to maintain the resources for peak times, a lot of computing resources would remain unused. This would lead to higher costs for Skype as well for their customers.

1.1 Motivation

In fact, modern Voice over IP (VoIP) service providers face fluctuation in phone calls every day and on popular events the number of phone calls peaks rapidly and decreases at the same pace. Furthermore, Mobile traffic forecasts 2010-2020 report [20111] indicate that mobile traffic will double until 2020 as can be seen in figure 2.1. As a direct implication, the report [20111] also indicates that the traffic of VoIP service will grow proportionally to those numbers.

Mobile traffic is predicted to double until 2020. The number of calls changes dramatically over a day. As a result, service providers are highly interested to dynamically adjust to the workload.

The report's forecast states: *It is inevitable that mobile VoIP will be adopted on mass scale in the next ten years and will trigger increases in the mobile voice traffic. This is mainly due to attractive pricing of international calls.* A second figure from the report (figure 1.3) shows that the number of mobile VoIP users is also constantly growing. This trend is a consequence of sinking hardware costs and the fact that mobile phones have more capabilities than ever before. Today's smartphones have enough power to perform tasks like playing games, browsing the internet, perform video conferences and many more for which a computer was needed before. In their work *Modeling Call Arrivals on VoIP Networks as Linear Gaussian Process under Heavy Traffic Condition* Ajarmeh et al. [AAY11] studied the number of calls received during a typical week on one tandem

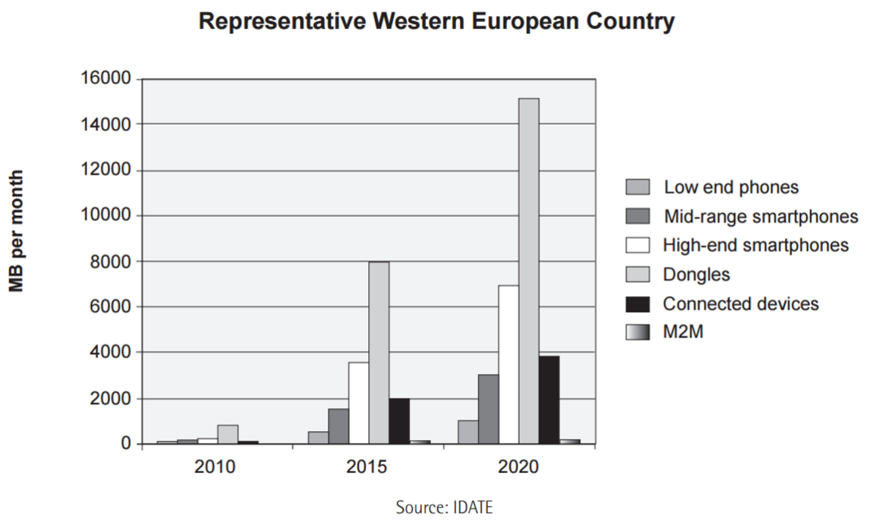


Figure 1.2: Monthly traffic per device until 2020

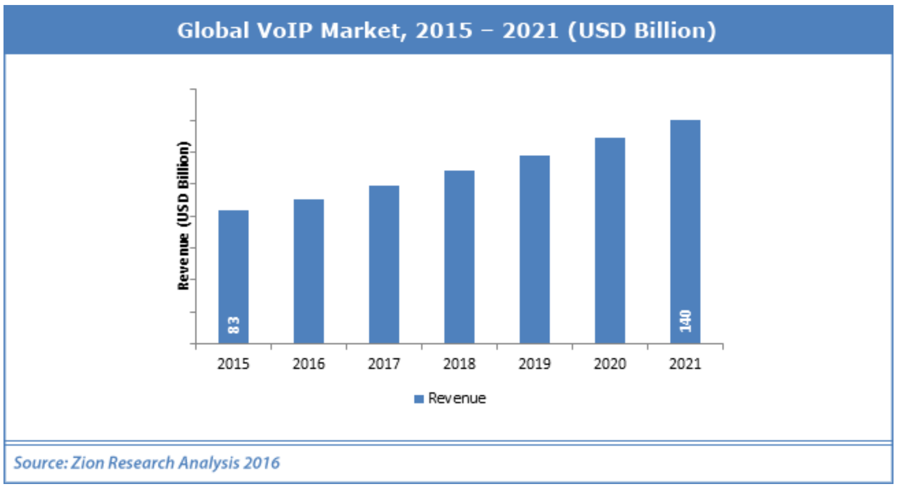


Figure 1.3: mobile VoIP market

central office in the network. In Figure 1.4 the line graph shows that the number of calls is alternating periodically with every day. The vertical axis illustrates the number of calls and the horizontal axis represents the days of a week. Ajarmeh et al. observed that during evenings the number of calls is generally higher while in the early morning it is usually lower. Furthermore, on Saturdays and Sundays there are less calls in general.

As a consequence, in times when the number of calls decreases less computational power is needed.

Another interesting case are special events that drive the workload to rise extremely.

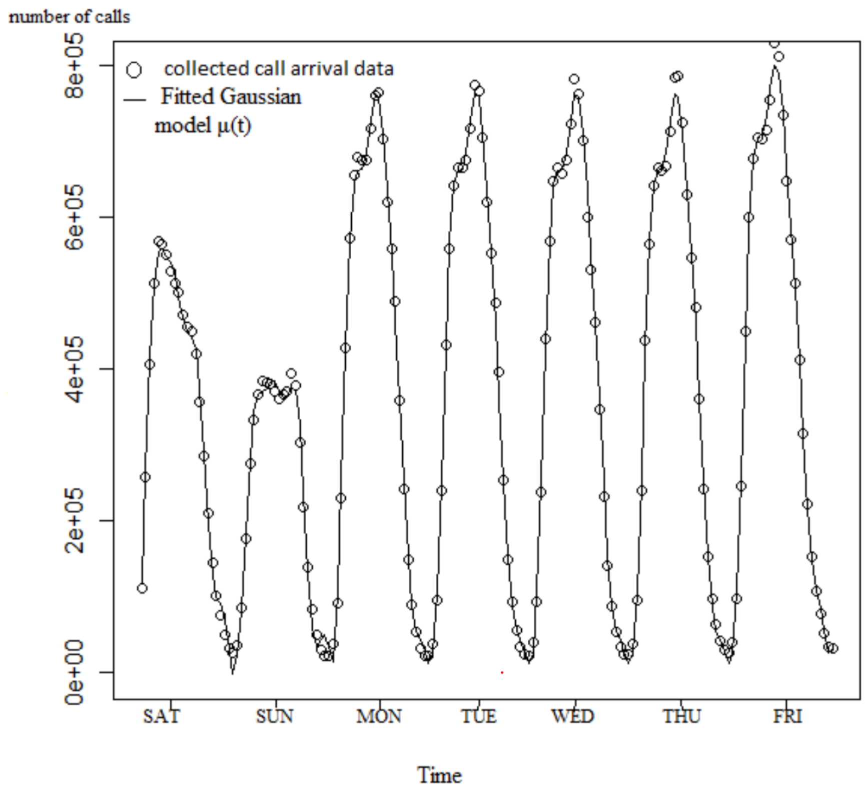


Figure 1. Fitted Gaussian model $\hat{\mu}(t)$against collected data

Figure 1.4: Workload for VoIP calls during a week

For example, it is a natural effect that on special occasions like New Year's Eve or huge sport events the number of calls sharply increases. People all over the world want to communicate with each other during this event. Of course such events have an impact on the infrastructure of a service provider.

Summing up, each New Year's Eve the number of calls is peaking.

Since daily fluctuation as well as predicted or unpredicted changes on the workload occur, service providers like Skype or Whatsapp have a natural interest in automatic algorithms providing users capacity on demand.

1.2 Problem Statement

On the one hand, a service provider wants to deliver its customers the best quality possible, but on the other hand, a minimum number of virtual machines should be used.

Therefore, they have to define service level agreements (SLA) that define the quality or resources they are willing to deliver. As a consequence, a service provider has to implement a controller which starts and stops virtual machines on demand.

Taking the demand for such an algorithm into account, this work shows how to use control theory in the context of informatics as a tool for horizontal scalability.

This work applies horizontal elasticity using closed control loops to cloud computing on the example of a VoIP service. A special focus is on long time delays in such control loops, since they cause instability insuch systems. Methods to prevent such instability are compared.

The application that runs a VoIP service is called a Private Branch Exchange (PBX). A PBX is a telephone switching system for handling multiple telephone lines and connecting them with a Public Switching Telephony Network (PSTN). In the chosen example, one can assume that the PBX is installed on virtual machines. This work uses FusionPBX as a PBX since it is based on Freeswitch and offers a web interface for managing the service. Additionally, it records calls and delivers a data basis for further analysis of the quality of each single call. FusionPBX runs perfectly on Linux and can also run in a docker container which would be an ideal prerequisite for usage of a cloud provider.
In order to implement such a controller, the service provider also needs to monitor the quality of calls for each virtual machine. This work shows how to obtain the needed data from FusionPBX.
Since the aim of this work is to show horizontal scaling, a load balancing service for all incoming calls is needed. Load balancing is implemented using a session boarder controller (SBC) with a round-robin distribution algorithm. The SBC separates the public and private network, like a firewall for the VoIP service, and standardizes the SIP and RTP streams, which are received from various clients.
For the sake of simplicity, one can assume that the network bandwidth is infinite and that the session boarder controller has enough resources at all levels of workload. In a real-world example this is not the case, but these assumptions makes the design of the controller much easier and allow for a focus on the scientific problem.

However, the number of PBX servers should be dynamically adjusted to required workload. Therefore, a method or algorithm is needed to accomplish this task.

1.3 Aim of the work

This section describes the aim of this work consisting of some important criteria finally resulting in the research question.

Application of control theory in cloud computing

The objective of the work is to show that control theory is applicable in computer sciences. Since control theory is used in electrical engineering for applications such as controlling the room temperature, there is already a large variety of scientific studies. As a result, control theory is well studied and accepted among the scientific community.
Cloud computing on the other hand is a comparably young research field, which has developed a high demand in controlling system parameters to increase the performance of services. The demand on controlling algorithms is due to the increasing flexibility and scalability in this field. Therefore, this work verifies if approaches known from electrical engineering can be adapted to cloud computing, using a large variety of already existing approaches.

Application of classical control theory in cloud computing
The previous paragraph has already described that control theory historically provides a huge vareiety of approaches. This paragraph elaborates this even more by exclusively narrowing down to classical control theory. Classical control theory was used for decades in electrical engineering, and since there is no need of using a microprocessor, the controllers only consist of discrete electrical elements built from resistors, capacitors or electrical coils (inductivities). Those elements were available long before the invention of semiconductors and were used to build PID controllers. Nowadays there are computers available which do not suffer restrictions that have been present in the past.
PID is a combination of proportional, integrative and derivate controller components. Due to their simple design, this work tries to use this controller type to control the quality of VoIP calls.

Selection of appropriate metrics
Another aspect of this work is to analyze which metric is suitable as a controlled variable for the proposed problem statement. Therefore, this work covers the different types of metrics and goes into depth on analyzing Voice over IP (VoIP) protocols and codecs. It is expected that this analysis leads to a variety of possible metrics. Eventually, this should lead to a representative choice of metrics from which a suitable one is selected. The well-chosen metric is then verified if it is easy to measure, which is exemplified on an open-source PBX. In summary, the requirements for the controlled variable are, that it is easy to measure and represents directly or indirectly the current quality of the service.

Configuration of the selected controller
Additionally, this work defines a methodology of how to configure such a controller considering influences like delays, which easily cause instability in controllers. This is mandatory since this is a well-known phenomenon in closed loops using classical controllers. Therefore, multiple methods of achieving a stable configuration are introduced describing their aspects and for which applications they are usually used. In a second step, the methods are compared with each other and a suitable method is selected. The reasons for the selections are explained to further justify the decision.

Critical Evaluation
Finally, the selected controller is evaluated on several properties. Also, the work compares different approaches and analyzes their applicability for the topic.
The primary focus is on the following properties:

Applicability: The work shall show that classical control theory can be applied in computer sciences.

Flexibility and Adaptability: The provided solution shall be flexible and easily adapted to other applications in cloud computing

Horizontal Scalability: The solution shall enable the controlled system to perform horizontal scalability.

Identification of metrics: Concrete metrics shall be suggested for the application of cloud telecommunication.

Cost-effectiveness: The provided solution shall be cost-effective.

Consistency and Stability: Stability is an important property of closed looped control loops. The work shows that the suggested solution is stable.

Research Question
Considering all subsidiary topics, the aim of the present work is to answer the following fundamental research question:
Is classical control theory using PID controllers applicable to control the quality of calls in Voice over IP services?

1.4 Methodological approach

As a methodological approach, this work is based on case based evidence. The problem is analyzed through the following steps:

- First, an introduction on control theory is given, which covers closed control loops, PID controllers and more advanced approaches such as the Smith Predictor. Also, methods to configure and tune controllers are presented to the reader. Furthermore, the design problems when using classical controllers are discussed. Also, state-of-the-art approaches like the MAPE (Monitor, Analyze, Plan, Execute) control loops, a mechanism providing self-adaption for controller, in chapter 2.1.3 are compared.

- After a brief introduction on metrics, a selection of usable metrics for the provided problem is given. The metrics are selected in order to provide the best possible speech quality for the user. This may be achieved by using direct or indirect metrics.

- To provide an example on how such a service can be built, a VoIP service is proposed. Requirements from users and service providers are formulated as service level objectives.
- In the next step, a controller is selected based on its properties, so it is suitable for the task. An analysis of delays and their impact on the controlled systems is examined. After showing if delays due to starting a virtual machine lead to instability of the controlled system, the controller is tuned using a methodology for designing controller with long time delays. Eventually, the working solution is presented.
- Finally, an evaluation of the solution is performed. The architecture of the test environment is explained. Next, a complexity analysis on the algorithm of the controller is performed, to examine the runtime of the same. Last but not least, a comparison to other controller types as a basis for further comparison with future controller implementations is presented.

1.5 Structure of the work

This work is structured into the following parts:

1.5.1 Introduction

The first part is an introduction to the work and gives the reader an insight into the field of science that is discussed in this work. The Introduction is structured the following: The first chapter provides the motivation, explainging why this work is important for the field of science. Then a problem statement is given which provides a scientific problem definition based on a concrete example for better understanding.

1.5.2 State of the art / analysis of existing approaches

The second part introduces the reader to the key concepts of control theory. It briefly introduces closed control loops to the reader. The section also overviews the most common controller types used in classical control theory such as P, I, D, PID controller. Since those controllers are vulnerable to instabilities, this chapter presents those issues. However, a way how to configure such controllers is introduced in the methodology chapter 3.

The next part of the chapter covers the Cloud Service Metric Model which, is a way to capture information and describe and understand metrics. Then categories of metrics are covered which have been standardized already and are followed by domain specific metrics of VoIP Services, e.g.: the Perceptual Evaluation of Speech Quality (PESQ), E-Model and the Mean Opinion Score. Additionally, topics like cost efficiency of cloud services are covered.
Furthermore, a brief introduction on SIP, SDP and RTP is given in order to provide

the basic concepts of modern Voice over IP (VoIP) technology. This is then used to discuss several metrics. In the chapter about metrics basic terms around metrics in cloud services are explained. At this point the work introduces domain specific performance metrics for Voice over IP and covers the Perceptual Evaluation of Speech Quality (PESQ), Mean Opinion Score (MOS) as well as the E-model. Finally, the E-Model is introduced, showing how to compute the MOS factor with an algorithm.

1.5.3 Methodology

The chapter on methodology 3 presents the reader on the one hand commonly used tools and concepts and languages. This covers R as a programming language up to the open source software like the PBX software. On the other hand, this chapter also covers the methods and concepts for designing and configuring controllers. This includes methods like the Ziegler-Nichols closed loop method, the Chien, Hrones, Reswick tuning method. Finally, the Smith Predictor as a compensator for delays is introduced.

1.5.4 Suggested Solution

In this chapter 4 the suggested solution to the scientific problem is presented based on real world requirements. Firstly, the architecture of the proposed VoIP cloud service is presented describing all components used. In the next step all Service Level Objectives are defined and an utility and a cost function are proposed. Afterwards the requirements on the VoIP cloud services are formalized. Last but not least, the design of the controller is shown using the Nichols-Ziegler method.

1.5.5 Evaluation

Finally, the proposed solution is evaluated in chapter 5. In a first step, the built test environment is described and suggests how to fetch the MOS values from the database of the PBXs. This is followed up by showing the difference between the initially used PID controller and the newly designed PI controller. Additionally, the source code is presented and discussed. Next the efficiency of the controller is shown by suggesting a formula to calculate the same. The result ensures comparability with other controllers and tuning methods and eventually the controller is compared with other controller types. Last but not least, a full complexity analysis is performed on the source code.

CHAPTER 2

State of the art / analysis of existing approaches

This chapter gives an overview of the field control theory and classical controllers, shows modern architectural patterns like the MAPE pattern, shows possible metrics for our problem definition and compares all these approaches with respect to the goals of this work.

2.1 Control Theory

The following section gives the reader a brief introduction to control theory that is used for this work. First, how to classify controllers is described and then closed control loops are covered in particular. Then the basic classical controllers are briefly explained, and for which use cases they are useful in general. As a key point, this section also covers the impact of large delays in closed loops and their impact on stability criteria.

Controllers are divided into two groups (please refer to figure 2.1): PIDT (PID and time delay) controllers and modern controllers. PIDT controllers are well-researched and have been applied to various analog and digital applications like controlling temperatures, controlling the production of iron and many more. Modern controllers on the other hand are used for more advanced problems when PIDT controller reach a certain complexity or limit. Furthermore, they can deal with probabilistic variables and are not bound to absolute values.

2.1.1 Closed Control Loop

A traditional way to use a controller is in a closed control loop as shown in figure 2.2. The user defines a set point variable to tell the controller what the desired level should

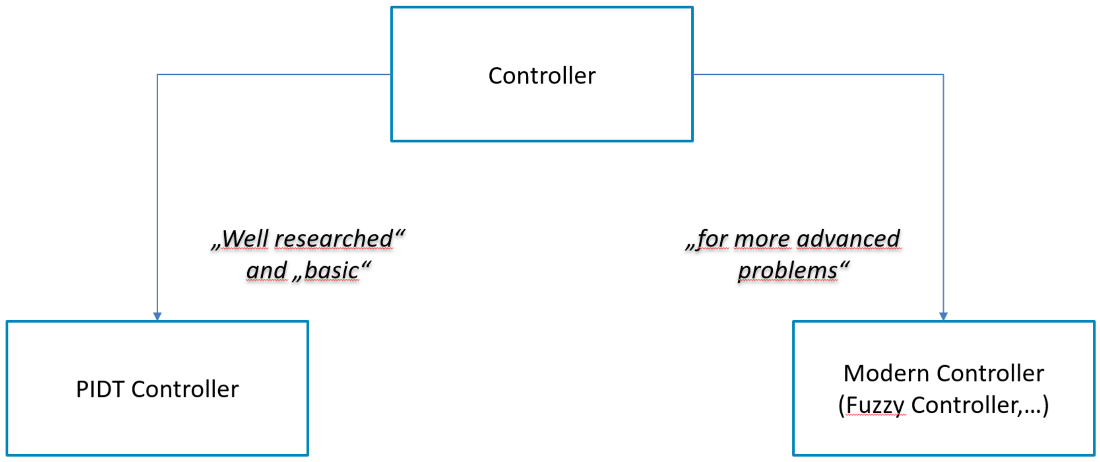

Controller
„Well researched“ and „basic“
„for more advanced problems“
PIDT Controller
Modern Controller (Fuzzy Controller,...)

Figure 2.1: controller types

be. In their book [PAC06] the authors define a set point as follows:

Set Point
The set point is a value for a process variable that is desired to be maintained.

For example, if a process temperature needs to be within 5 degree Celsius of 100 degrees Celsius, then the set point is 100 degrees Celsius. In this work, the set point variable is represented by the desired quality of the call.

Error
The error is obtained by subtracting the current value (process value) which is measured from the controlled system.

In their book[PAC06] the authors define process value as follows: *A process variable is a condition of the process fluid (a liquid or gas) that can change the manufacturing process in some way. In the example of a person sitting by the fire, the process variable was temperature.*

The error is used put into the controller which reacts on this input and informs the actuator on which action to perform on the controlled system. This is represented by the utilization. The controlled system certainly changes its behavior on performed action. This can happen in a linear or non-linear manner.

Additionally, disturbances can change the behavior of the controlled system. Disturbances in this example are the number of calls the service must handle. Also, it is worth to mention that it is possible that the controlled system is changing its process value with a delay. The controller shall be designed to consider occurring disturbances and delays equally.

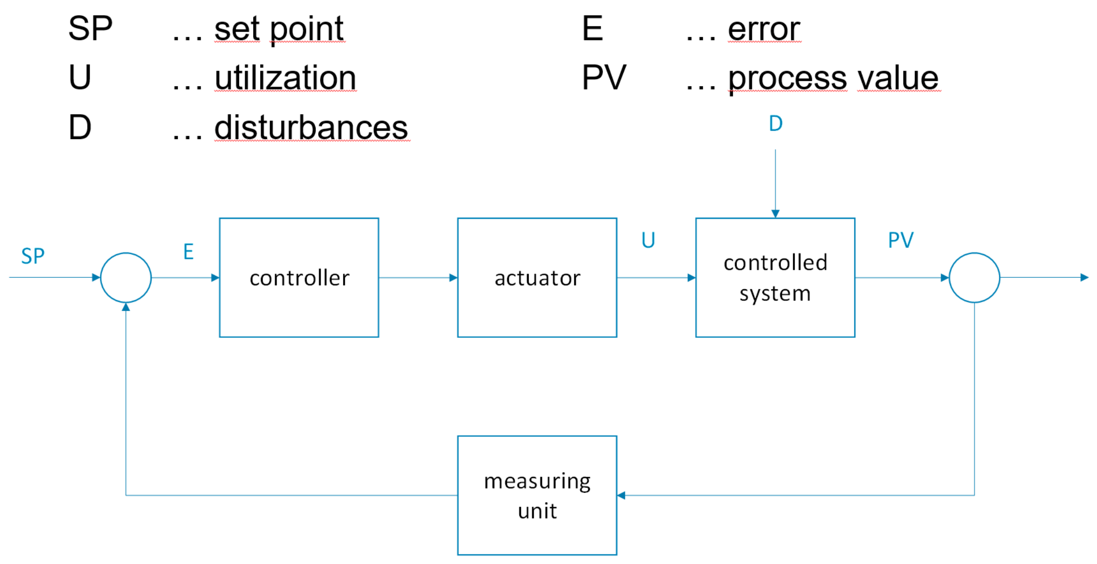


Figure 2.2: closed control loop

2.1.2 Classical Controllers

This chapter presents different classical controllers, as they are used later in this work to design a controller, which is able to handle delays occurring due to the startup time of the VMs.

All classical controllers are composed of 3 parameters: P, I and D. P stands for proportional, I for integrative and D for derivative. These basic controller components are explained in this section with the help of bode plots expressing the frequency response of a system.

A bode plot is a combination of a bode magnitude plot, expressing the gain on a signal, and a bode phase plot, expressing the phase shift on a signal. The y-scale of a bode magnitude plot is in decibel (dB) and the bode phase diagram is in degrees while the x-scale represents the frequency using a logarithmic scale.

Furthermore, this work introduces the PID controller which is a combination of those 3 parameters. The impact on controllers of large delays is covered, as well how PI controllers are used to minimize delays like VM start up time.
A proportional or P controller gives a linear feedback to a measured variable. Haager[Haa03] defines a P Controller as follows: *The output value $x_a(t)$ of a P controller is proportional to the input value $x_e(t)$.*

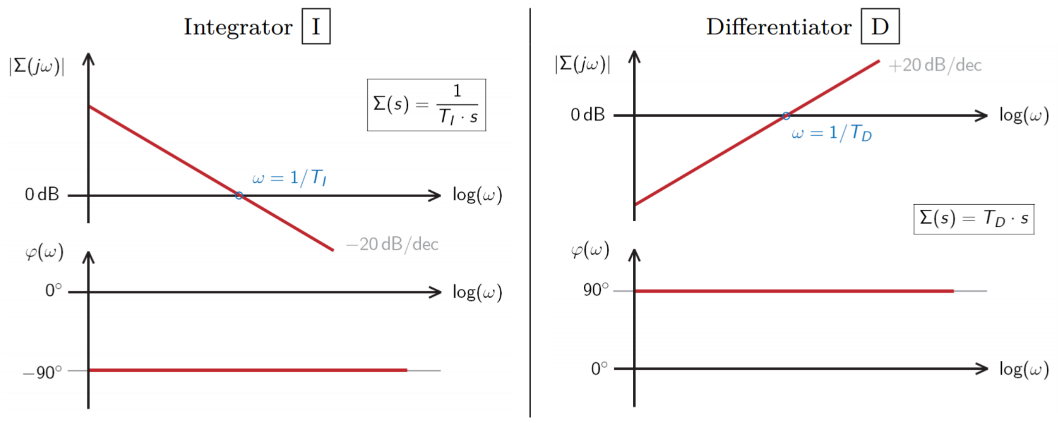


Figure 2.3: I and D controller bode diagram

Frequency Response:
$x_a(t) = k * x_e(t)$
$X_a(s) = k * X_e(s)$
Gain: $G(s) = k$

An integrative controllers' feedback is not only according to the error but also regarding to time. Haager[Haa03] defines an I controller as follows: *The output value $x_a(t)$ of a P controller is proportional by the time integral to the input value $x_e(t)$.*

Frequency Response:
$x_a(t) = k_I * \int_0^t x_e(\tau)$
$X_a(s) = \frac{k_I}{s} * X_e(s)$
Gain: $G(s) = \frac{k_I}{s}$

A derivate or D controller's feedback reduces the short time disturbances. Therefore, it reduces overshoots, but in reality, a pure D controller is not able to flatten the process value to the set point. Nevertheless, if applied on small errors and a high D value leads to overshooting. As a result, it should be carefully used since it can lead to instabilities in the system. If applied on oscillations and the oscillations decrease, the system is considered as stable, if increased as unstable. If the oscillations sustain the system is referred to as marginally stable.
In his book, Haager[Haa03] also points out that in practice the D proportion is not a clean differentiator.
Frequency Response:
$x_a(t) = k_D * \frac{x_e}{dt}$
$X_a(s) = k_D * X_e(s)$
Gain:$G(s) = k_D * s$

A PID Controller combines all the above parameters of P, I and D controllers and are typically used for industrial applications. As a result, there it has been the standard industrial controller since many years and is used in many different industries like Pharmaceutical, Satellite, Oil and Gas, Cement, and Power (like heat control of ovens).
The disadvantage of this controller type is clearly that it cannot adapt to changing parameters. This is due to the fact that it has neither knowledge of the process nor any implementation of self-adapting algorithms.

A PID controller consists of proportional, integral and differential components.
$F_R(s) = k_R(1 + \frac{1}{sT_N} + sT_V) = k_R + \frac{k_I}{s} + k_D s$

PID controller are well researched and often used in industrial applications. One common problem with such controllers is stability when confronted with delays in the observed system. Those constellations can lead to oscillations in closed loops. The larger the delay gets the more unstable the loop becomes. Furthermore, this effect is increased by higher values for K_p. Figure 2.4 demonstrates this fact by showing a PID controller with different values for K_p. On the y-axis the amplitude is depicted, and the x-axis depicts the time in seconds. The higher the value for K_p the higher the overshooting is.

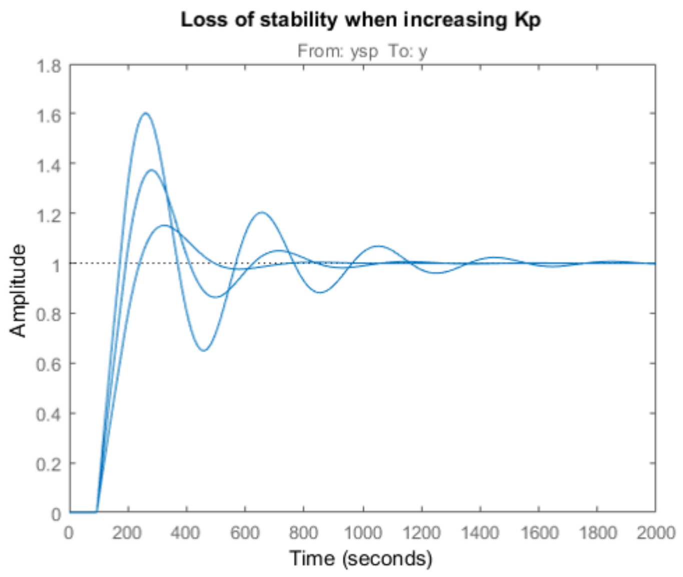


Figure 2.4: Loss of stability

In their work, Zhong et.al. [ZXJ00] describe time delay filter based on a PI filter with guaranteed stability since time delays quickly lead to instability of control loops when they get too large.
To achieve this criteria of stability for linear, time-invariant (LTI) systems, they have

used and described the Nyquist stability criterion. The Nyquist criterion suggests that for a system to be stable, the number of closed-loop poles in the right half plane must be zero. One advantage of the Nyquist Criterion is that it can be determined by using a Nyquist plot and does not require to calculate all points of the diagram. Figure 2.5 shows an example of a Nyquist diagram. The x-axis of the diagram represents the real part and the y-axis the imaginary part. The diagram is drawn by calculating the frequency response from $\omega = 0 to \omega = \infty$.

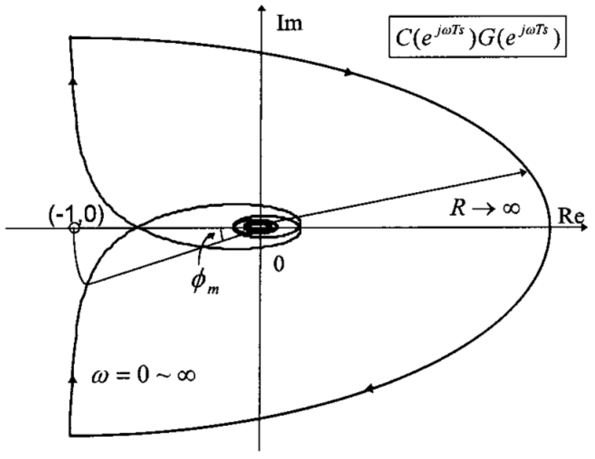


Figure 2.5: Nyquist Curve

After drawing the diagram, it can be examined as follows:

The number of closed-loop poles in the right half plane, Z
= The number of rotations about -1 of the mapping through G(s)H(s), N
+ The number of open-loop poles in the right half plane, P

For a system to be stable, Z must be zero.

To cover these issues with instabilities, a PI controller can be used. The PI controller is a PID controller with the D property set to zero and combines the speed of the P controller and the accuracy of an I controller. It is usually used to deal with delays (dead times) in control loops.

Frequency Response:

$F_R(s) = k_R(1 + \frac{1}{sT_N})$

2.1.3 Monitor, Analyze, Plan, Execute (MAPE)

The MAPE control loop was first introduced by IBM and provides self-adaption mechanism for control loops. Abuseta et al. describe in their work [YA15] the MAPE loop. The MAPE loop basically consists of the activities monitor, analyze, plan and execute. A knowledge management component controls these activities using a policy engine and the stored system state. Sensors are used to collect data that represents the current state of the system and its surrounding environment. This data is necessary for the adaptive process and is therefore aggregated and saved for future reference. The data is later used to create models of past and current states and can be used to deduct trends. Effectors apply corrective changes to the managed resource.

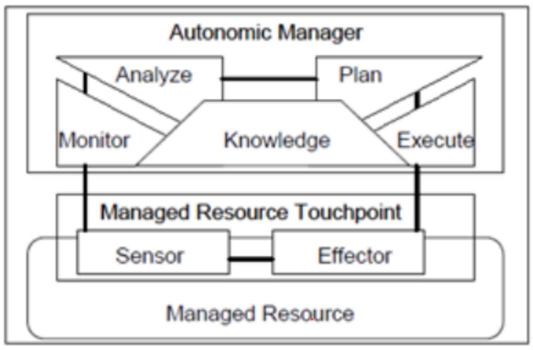


Fig. 1. Autonomic computing control loop [16]

Figure 2.6: MAPE loop

2.2 Metrics

Metrics play an important role in monitoring cloud services and in making decisions (e.g.: in a MAPE loop) to improve them. In order to decide which metrics are used in this work, an overview is given on the classified metrics and how metrics and service level objectives are managed. Furthermore, this section introduces the cloud service metric model which helps to define proper metrics.

The National Institute of Standards in the US[Pri15] defines in their document based on ISO/IEC 15939:2007 the following terms:

Cloud Service Level Objective (SLO)[CSI14]: *Target for a given attribute of a cloud service that can be expressed quantitatively or qualitatively.*

Measurement:*A set of operations having the object of assigning a Measurement Result.*

Metric: *A standard of measurement that defines the conditions and the rules for performing the measurement and for understanding the results of a measurement.*

CSIG[CSI14] expresses this definition in a similar way: *A metric is a defined measurement method and measurement scale, which is used in relation to a quantitative service level objective.*

Measurement Result: *Value that expresses a qualitative or quantitative assessment of a property of an entity.*

Furthermore, the National Institute of Standards describes the importance of metrics by showing the relationship between a metric and a property. A property of a cloud service represents a characteristic of the cloud service. When metrics are observed, they result in measurement results. Those results can then provide knowledge about the service that is observed and represents an abstract representation of the property.

Additionally, the document points out that metrics on the service interface are used to

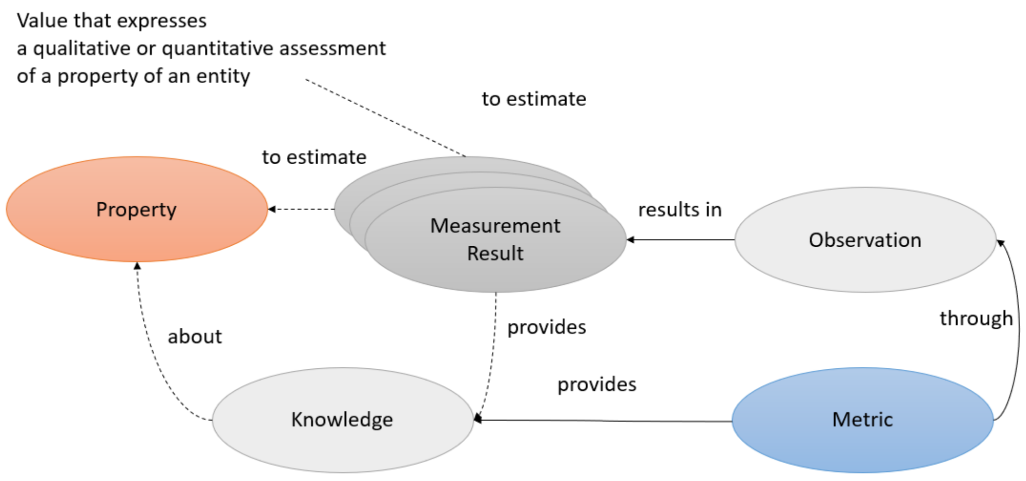


Figure 2.7: Metric and Property[Pri15]

establish service agreements (SA). Service agreements are a binding commitment between the provider and the customer to ensure agreed upon properties. It contains a description about the service, the rights and duties of the provider and the customer and essential information related to the measurement of different cloud service aspects.

The definition of metrics is an essential part of Service Level Agreements (SLA) and Service Level Objectives (SLO). The metrics are useful tools to measure the performance of the services of the provider for monitoring purposes, as well as for remediation (e.g. financial)

As a result, the usage of standardized metric sets allows easier comparison between services.
The Cloud Select Industry Group (CSIG)[CSI14] describes in their work principles for the development of such service level agreements. Furthermore, the author[Pri15] points out that stakeholders need to have a way to understand, assess, compare, combine and make decisions about cloud service properties. A scenario describes a particular use case which helps to handle those cloud service properties. A scenario represents a particular use case

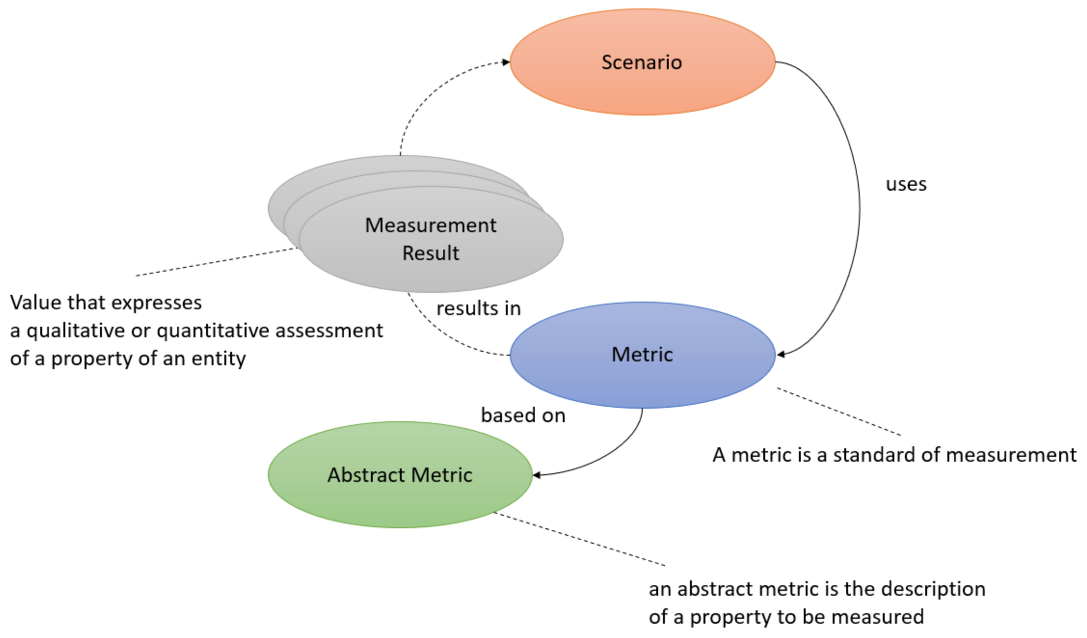

Scenario
uses
Measurement Result
Value that expresses a qualitative or quantitative assessment of a property of an entity
results in
Metric
based on
Abstract Metric
A metric is a standard of measurement
an abstract metric is the description of a property to be measured

Figure 2.8: Scenario and Metric[Pri15]

that uses metrics resulting in measurement results as a basis for decisions. Stakeholders like cloud customer or cloud provider define the scenarios for which the metric is needed. The scenario represents:

- Expectations of an underlying business process
- How the metrics are used to support the process

- What acceptable levels of the measured properties are

For this purpose the National Institute of Standards[Pri15] introduced **Cloud Service Metric Model (CSM)**. The purpose of the CSM model is capturing information needed to describe and understand a metric. The model is used to gain knowledge about the metrics and measuring cloud service properties.

Cloud Service Metric Model (CSM) The CSM is used to gain knowledge about metrics and measuring cloud service properties. Below find the basic terms used in CSMs.

- **Cloud Service Metric Model (CSM)**: The description and definition of a standard of measurement (e.g. metric for customer response time)
- **CSM Context**: The addition of the context of the standard of measurement (e.g. objectives and applicability conditions of the customer response time metric)
- **CSM Observation**: The use of the standard of measurement to define observations (e.g. the observation of response time property based on the customer response time metric)
- **CSM Scenario**: The use of the standard of measurement in a scenario (e.g. the selection and use of the customer response time metric in an SLA scenario) – CSM Scenario

Figure 2.9 shows that the CSM distinguishes between concrete metric definitions and abstract metric definitions. Abstract metric definitions represent the model for a category of metrics (e.g. Service Availability). Concrete metric definitions are specific instances of the abstract metrics. The figure shows furthermore, that abstract metrics consist of parameter definitions and rule definitions. These definitions have as counterparts in the concrete metric definition metric parameter and metric rules. While the parameter definitions and the rule definition describe what the rules and parameters are about, the metric parameter and the metric rule constrain the metric (e.g. Availability).

Figure 2.10 visualizes the process used to define metrics using the CSM model. The CSM defines the core components of the model. Next the abstract metrics are defined. In the last step, concrete implementation metrics are created using the metric parameter instances and metric rules of Block 2 in figure 2.9. Eventually, the National Institute of Standards[Pri15] concludes that metrics are a critical aspect of the selection, operation and use of cloud services. Metrics allow stakeholders to gain better understanding of the behavior of cloud service properties through consistent, reproduce-able and repeatable observations.
The Cloud Select Industry Group (CSIG)[CSI14], which is a working group set up by

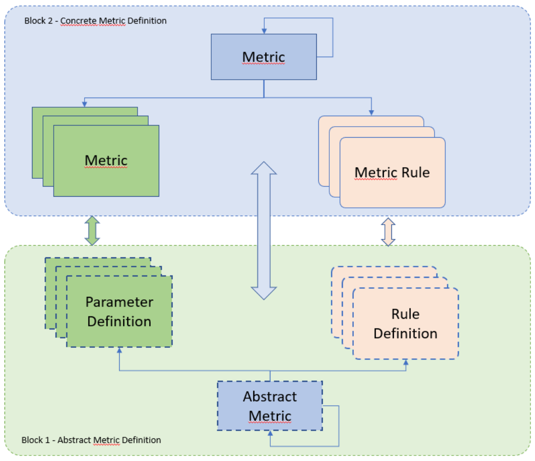


Figure 2.9: CSM Definition Blocks[Pri15]

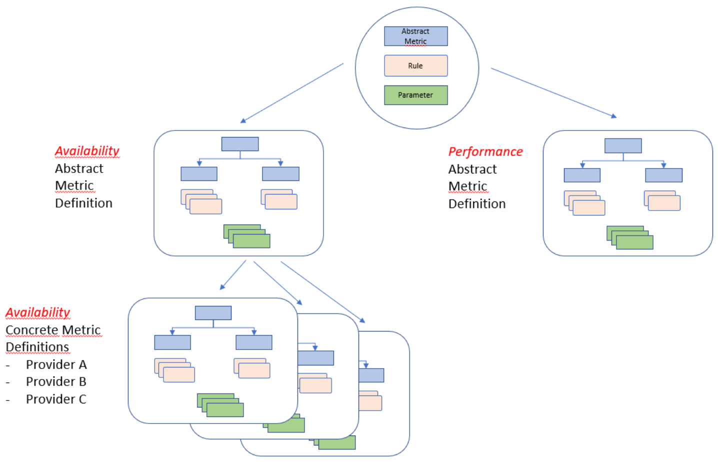


Figure 2.10: CSM Definition Process[Pri15]

the European Union, is composed of representatives from European and multinational industry, public administrations and other. In their work, [CSI14] CSIG summarized principles of service level agreement standards for cloud computing, Cloud SLA vocabulary, Performance service level objectives such as availability and response time and many more. Their aim is to standardize aspects of SLAs to improve clarity in the cloud computing market.

The above mentioned metrics have been later further summarized by Bardsiri et. al [BH14] into the categories of Performance Metrics, Economic Metrics, Security Metrics and General Metrics. While economic metrics purpose is to compare the costs of services, security metrics deal with information security and privacy. The group general metrics consists of CSIG [CSI14] defined properties. *Performance Metrics* indicate the efficiency of a service. There are many different metrics cloud providers can choose from to monitor performance. Nevertheless, many aspects of the performance characteristics can be linked to the features communication, computation, memory and time.

Communication represents all metrics that are related to data transfer between internal service instances and external consumers. Typical metrics would be Packet Loss Frequency, Mean Opinion Score, PESQ, etc. *Computation* denotes all computational tasks processed in the cloud. *Memory* describes metrics for the use of temporarily stored information. *Time* denotes the completion of a task within time.

Performance Metrics describe the efficiency of a service and can be grouped into the features communication, computation, memory and time.

In conclusion, it is a very crucial decision which metrics are used in a service. The decision has to be made in the best interest of cloud service providers and cloud consumers.

2.2.1 Domain specific performance metrics in VoIP

This section focusses on the group of performance metrics and more specifically on the feature of **communication** with a focus on VoIP related metrics.

As the ITU describes in its ITU-T Recommendation P.862 [ITU01], **Perceptual Evaluation of Speech Quality (PESQ)** is a successor of PSQM. PESQ compares an original signal X(t) with a degraded signal Y(t) that is the result of passing X(t) through a communication system. ITU points out that the PSQM method as described in ITU-T P.861 (February 1998), was only recommended for use in assessing speech codecs, and was not able to take proper account of filtering, variable delay, and short localized distortions. PESQ addresses those issues with using transfer function equalization, time alignment, and a new algorithm for averaging distortions over time.

Another metric that can be measured is the R-Value or E-Model. The E-model is a network planning tool used in the design of networks for carrying voice applications and can be calculated in software. Perlicki [Per02] describes in his work how the R-Value is computed using the following formulae:

E-Model:

$$R = R_o - I_s - I_d - I_e + A$$

The term R_o is the signal-to-noise ratio (received speech level relative to circuit and acoustic noise). I_s represents all impairments which occur more or less simultaneously with the voice signal, such as: too loud speech level, non-optimum side tone, quantization noise, etc. All delays and echo effects are summed up in the delay impairment factor I_d. The term I_e represents impairments which are caused by low bit-rate codecs used in special equipment. Furthermore, this factor is used to consider the effects of packet loss. The last term A, short for advantage of access, describes all effects that compensate bad quality such as convenience or connections in hard to reach areas.

One of the most important metrics regarding the measurement of VoIP networks is the **Mean Opinion Score (MOS)**. It is usually obtained by asking users directly after an ended phone call to vote the call in a scale from 1 to 5. VOIP calls often are in the 3.5 to 4.2 range[voi]. The ITU P.861 (PSQM)[p86a] and P.862[p86b] standards explain how to calculate MOS scores from an E-Model.

Mean Opinion Score (MOS)
The MOS value is a subjective quality rating for calls and is only perceptible by humans. The values usually go from 1 to 5 where 1 is bad and 5 is excellent speech quality.

$\frac{\sum_{n=1}^{N} R_n}{N}$

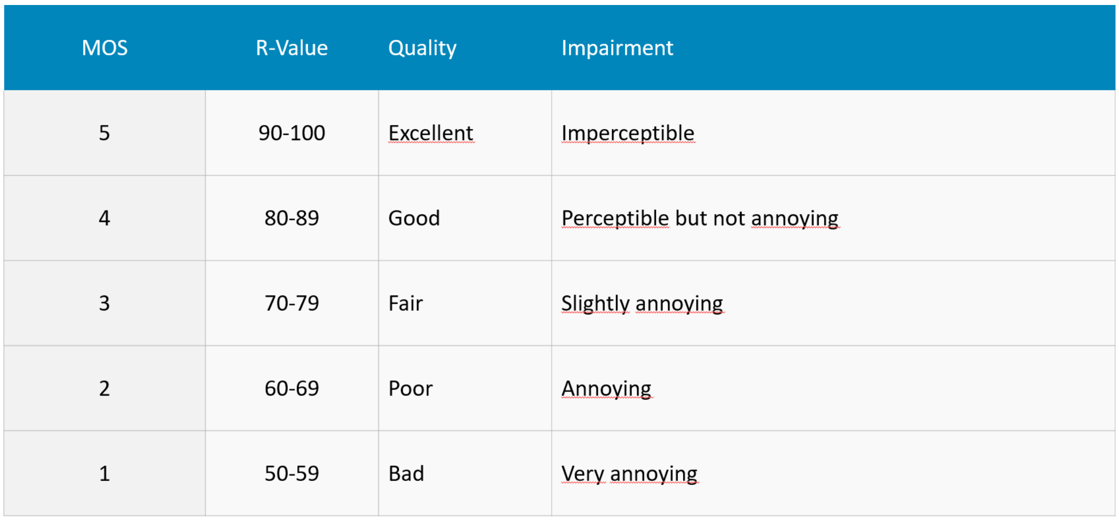
R_n .. rating number n
N .. number of ratings

Often this measurement is used by VoIP providers e.g. Skype to survey users' opinion on the quality of calls. The **Predicted MOS Value (PMOS Value)** values can be determined from R-value by using a mapping from the R-value to the MOS value. This mapped MOS value is called the Predicted MOS value (PMOS value).

2.3 Cost-Efficient Utilization in cloud services

This section shall review the current algorithms and strategies for cost-efficient utilization in cloud services.

According to Maurer et al. [MM11] the computation of resource allocation in clouds are based not only on functional requirements, but also on non-functional requirements. The authors state that those non-functional requirements are quality of service requirements and are expressed and negotiated by means of Service Level Agreements (SLAs). Such SLAs are usually managed using **SLA templates.** Risch et al. [MR09] defines SLA templates as SLA popular formats, comprising elements such as names of trading parties, names of SLA attributes, measurement metrics, and attribute values. In their work, Maurer et al. define a method for adaptive SLA mapping shown on a use case of medical image service.



MOS	R-Value	Quality	Impairment
5	90-100	Excellent	Imperceptible
4	80-89	Good	Perceptible but not annoying
3	70-79	Fair	Slightly annoying
2	60-69	Poor	Annoying
1	50-59	Bad	Very annoying

Figure 2.11: MOS and R value

Therefore, an initial template is created. Service providers can assign their services to a

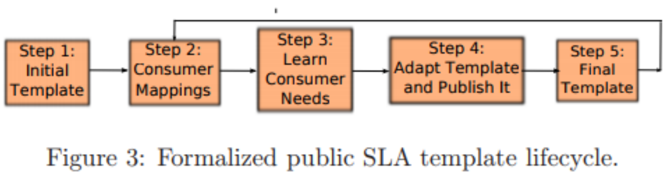

Step 1: Initial Template
Step 2: Consumer Mappings
Step 3: Learn Consumer Needs
Step 4: Adapt Template and Publish It
Step 5: Final Template
Figure 3: Formalized public SLA template lifecycle.

Figure 2.12: SLA Lifecycle[MM11]

particular public SLA template. Based on those mappings a service consumer is able to select a public SLA template, and if the public template differs from the private one, the consumer can make adaptions by mapping for the differences identified. Later, an adaption process is identifying similar templates. Similar templates might be adapted, or subgroups may be created. After the new templates are generated, old ones are deleted. Based on the assumption that the market demands are not changing eventually a final template is created.
This life cycle is depicted in figure 2.12 but is described in more detail in Cost-Efficient Utilization of Public SLA Templates in Autonomic Cloud Markets[IB11].
Furthermore, Maurer et al. define several adaption methods to select attribute names for the SLAs.
The **maximum method** automatically selects the SLA attribute name which has the highest number of mappings. The authors explain that the **threshold method** introduces a threshold value. If a particular attribute name is used more than this threshold and has the highest count, then the attribute is selected. The **Maximum-Percentage-Change** method is structured in two steps. In the first step, the attribute name is chosen

by the maximum method, and in the second step, the attribute name is changed if and only if the percentage difference is between the highest count attribute and the currently selected attribute name exceeds a predefined threshold.

2.3.1 Utility and Cost Models

In order to make the costs and benefits of applications measurable, a utility and a cost model are needed. Maurer et al. state that both, the cost and the utility function take attributes of the customer's SLA template and the public SLA templates as input.
As an example, the authors introduce a utility model that assumes an increase in benefit if an attribute of both templates is identical. On the other side, the cost model captures the effort of changing an SLA mapping.

Utility Model
Mosa et al. [MP16] define in their work a utility function used for utility based resource allocation:

$$Utility(a,t) = Income(a,t) - (EstimatedEnergyCost(a,t) + EstimatedViolationCost(a,t) + PDMCost(a,t))$$

The variable a in this equation is a map representing VMs-to-PMs assignment and t is the assignment of time period. $Income(a,t)$ is the total income from hosting a customer's VMs, $EstimatedEnergyCost(a,t)$ are the estimated energy costs as a result of the assignment. $EstimatedViolationCost(a,t)$ represents the cost of SLA violation due to the over-utilization of the hosting PMs, a cost that is calculated based on the number of VMs in violation. PDMCost(a,t) represents the violation cost of the performance degradation due to the migration (PDM) of VMs among PMs.
The authors provide algorithms calculating the mentioned costs. Furthermore, a genetic algorithm is proposed for finding the VM-to-PM assignment that maximizes the utility. Zhang et al. [ZLZ15] describe that Cloud Service Provider offer reduced resources for illegal economic benefit. As a conclusion in their work they propose a framework2.13 for measuring the performance of cloud applications. Firstly, the runtime of the cloud application is measured by labeling parts of the application. The historical data is then used to determine the most frequent parts of the application.

As shown in figure 2.13, the inserted labels along with a small data set deliver together with the historical observations from the cloud a local prediction model of the performance. A cost function is then used for each hot spot block to obtain the worst case performance model. The performance model is constructed from a performance skeleton tree which is used to model relationships between the most frequent blocks. As can be seen in figure 2.14 each node in the tree represents a block. For an input x, the performance costs to each node are calculated. The performance skeleton tree can be obtained by using static source code analysis. Emeakaroha et al. [VCER10] describe a

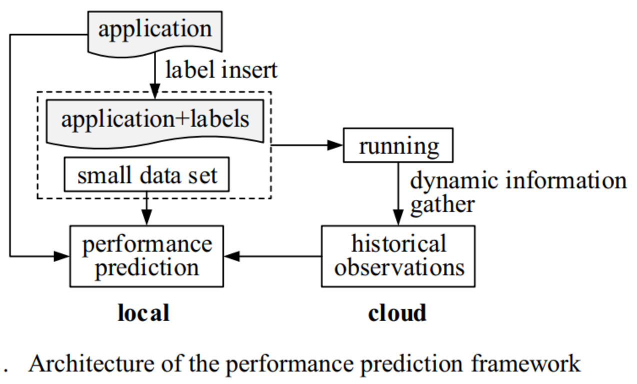


Figure 2.13: Architecture performance prediction framework[ZLZ15]

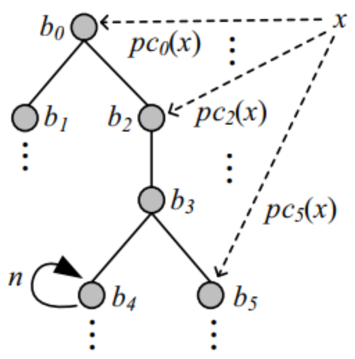


Figure 2.14: typical performance skeleton tree[ZLZ15]

similar approach in their work. DeSVi is an architecture for detecting SLA violations through resource monitoring in Cloud computing infrastructures. The framework is measuring low level resources and is mapping them to high level SLAs using knowledge-based SLA with a cased based reasoning approach.

The achieved results show, based on an image rendering service, that the architecture can monitor and prevent SLA violations considering different costs, measurement intervals, and heterogeneous workloads. [SC15]

2.4 Voice over IP (VoIP)

Telecommunication plays an important role in our every days life and connects people all over the world. Voice over IP clearly has revolutionized the way people communicate nowadays. Whether by phone or computer, one is able to talk to people living in far distant locations. Therefore, this section aims to give a brief introduction to the most common VoIP protocols. This is required to prepare the reader on which metrics may be used in such a system. First an introduction on the SIP protocol is given since it is the most common VoIP protocol used. Secondly, the Real Time Protocol (RTP) is covered since it is used for transmitting the media data over the network. That includes audio as well as video.

2.4.1 SIP Protocol

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as internet telephone calls, and was defined by Rosenberg et al. in RFC 3261[JR02]. SIP can also invite participants to already existing sessions, such as multicast conferences. Media such as audio or video can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility. As a result, users can maintain a single externally visible identifier regardless of their network location. Since SIP is easier than e.g.: H.323 it is nowadays more commonly used in VoIP applications.

The SIP standard [JR02] covers the following aspects of communications:

- **User location**: Definition of the end point (e.g. extension) to be used for communication.
- **User availability**: Availability of the called party to engage in communications.
- **User capabilities**: Definition of the media protocol and media parameters to be used.
- **Session setup**: *Ringing*, establishment of session parameters at both called and calling party.
- **Session management**: This includes transfer and termination of sessions, as well as modifying session parameters, and invoking services.

Figure 2.15 shows a simple SIP communication between two parties Alice and Bob over two proxies (atlanta.com and biloxi.com). When Alice tries to call Bob an INVITE is sent to the atlanta proxy. The proxy in atlanta sends an INVITE to the proxy biloxi.com and also responds with a trying to Alice' soft phone. The biloxi.com proxy send an INVITE to Bob's SIP Phone and a trying to the proxy in atlanta.

Bobs SIP Phone returns a RINGING message to biloxy.com. Biloxy.com itself sends a RINGING message to atlanta.com and atlanta.com sends a Ringing message to Alice soft phone.

When Bob answers the call his SIP phone returns an OK message to biloxy.com which sends itself an OK message to atlanta.com. Atlanta.com sends itself an OK message to Alice's soft phone. In the OK message Bob's SIP phone sends connection data like supported codecs that are used later for the media session.
Alice's soft phone then returns an ACK message to Bob's SIP phone. The result of this messages is that both parties can establish a media session which is mostly done by using Real time transport protocol (RTP).

Eventually Bob finishes the call and his SIP phone sends a BYE message to Alice. Alice's soft phone returns with an OK message then.

2.4.2 Session Description Protocol (SDP)

SDP is defined by Handley et al. [JR06] in RFC 4566. The SDP protocol defines the following parameter groups:

- Media and Transport Information
- Timing Information
- Private Sessions
- Obtaining Further Information about a Session
- Categorization
- Internationalization

Media and transport information refers to information like codec, transport address which is used to establish e.g. an RTP connection transferring the G.711 codec. The term timing information describes two parameters, the time since when a session is active and the repeat times. Parameters used in private sessions are used to give important information regarding the encryption of the connection. This at least contains the encryption key for the session. The encryption key is later used to encrypt the media transferred.

The following excerpt of parameters shows the introduced variable groups like they would occur in the SDP protocol:

```
Session description
        v=  (protocol version)
        o=  (originator and session identifier)
        s=  (session name)
        i=* (session information)
        u=* (URI of description)
        e=* (email address)
```

```
                     atlanta.com  . . . biloxi.com
                 .      proxy              proxy     .
               .                                       .
       Alice's  . . . . . . . . . . . . . . . . . . . .  Bob's
      softphone                                        SIP Phone
         |                |                |                |
         |    INVITE F1   |                |                |
         |--------------->|    INVITE F2   |                |
         |  100 Trying F3 |--------------->|    INVITE F4   |
         |<---------------|  100 Trying F5 |--------------->|
         |                |<-------------- | 180 Ringing F6 |
         |                | 180 Ringing F7 |<---------------|
         | 180 Ringing F8 |<---------------|     200 OK F9  |
         |<---------------|    200 OK F10  |<---------------|
         |    200 OK F11  |<---------------|                |
         |<---------------|                |                |
         |                       ACK F12                    |
         |------------------------------------------------->|
         |                   Media Session                  |
         |<================================================>|
         |                       BYE F13                    |
         |<-------------------------------------------------|
         |                     200 OK F14                   |
         |------------------------------------------------->|
         |                                                  |

            Figure 1: SIP session setup example with SIP trapezoid

      INVITE sip:bob@biloxi.com SIP/2.0
      Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
      Max-Forwards: 70
      To: Bob <sip:bob@biloxi.com>
      From: Alice <sip:alice@atlanta.com>;tag=1928301774
      Call-ID: a84b4c76e66710@pc33.atlanta.com
      CSeq: 314159 INVITE
      Contact: <sip:alice@pc33.atlanta.com>
      Content-Type: application/sdp
      Content-Length: 142

      (Alice's SDP not shown)
```

Figure 2.15: simple example of the SIP protocol as presented in RFC3261[JR02]

```
            p=* (phone number)
            c=* (connection information)
            b=* (zero or more bandwidth information lines)
            One or more time descriptions ("t=" and "r=" lines; see below)
            z=* (time zone adjustments)
            k=* (encryption key)
            a=* (zero or more session attribute lines)
            Zero or more media descriptions

         Time description
            t=  (time the session is active)
            r=* (zero or more repeat times)

         Media description, if present
            m=  (media name and transport address)
            i=* (media title)
            c=* (connection information)
            b=* (zero or more bandwidth information lines)
            k=* (encryption key)
            a=* (zero or more media attribute lines)
```

2.4.3 Real-time Transport Protocol (RTP)

As defined in RFC 3550[HS03] by Schulzrinne, et al., 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. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and mixers.

RTP Fixed Header Fields

The RTP protocol's header looks like the following:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number
|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          timestamp
|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier
|
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers
|
|                             ....
|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The RTP proptocol's header consists of the following fields:

2.4.4 OPUS Codec

The OPUS codec is a modern codec especially designed for Voice over IP application and delivers audio with very high quality which is not only suitable for speech but also music applications. One advantage of the OPUS codec is, that it can be used within different sample rates to adapt automatically to network bandwidths.

In RFC 6716 [JV12] Valin et al. define the OPUS codec as a real-time interactive audio codec especially designed for interactive audio applications(VoIP), videoconferencing, etc. It is composed of a layer based on Linear Prediction (LP) and a layer which uses the Modified Discrete Cosine Transform (MDCT) which are used to achieve good compression for both speech and music. The main idea behind using two layers, is that Linear Prediction(such as Code-Excited Linear Prediction, or CELP) is more suitable at handling low frequency applications like speech than transform (e.g., MDCT) domain techniques, while it is vice versa for music and higher speech frequencies. As a conclusion, if both techniques are used in separate layers not only the codec is able to operate in a much wider frequency range but also can achieve better quality combining both.
Since OPUS scales from a very low bitrate with 6 kbit/s up to a very high with 510 kbit/s, it offers different audio bandwidths like narrowband, mediumband, wideband, super-wideband and fullband which are summarized in table 2.2. As a result, the codec can be used for various applications such as speech, music or video applications.
Moreover, the codec can seamlessly switch between different operating modes at any given time e.g. from 6 kbit/s narrowband mono speech to 510 kbit/s fullband stereo music, with algorithmic delays ranging from 5 ms to 65.2 ms. Furthermore, the LP layer, the MDCT layer, or both, may be activated or deactivated. This feature of dynamically adjusting is helpful as in Voice over IP applications one can never know if there is a variation of network bandwidth due to fluctuation in users that are using the service. It also provides great flexibility when adapting to varying content during a session. This may also be important when handling critical communication.

Field	**Description**
version (V): 2 bits	This field identifies the version of RTP. Nowadays this value is usually 2.
padding (P): 1 bit	If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload.
extension (X): 1 bit	If the extension bit is set, the fixed header MUST be followed by exactly one header extension.
CSRC count (CC): 4 bits	The interpretation of the marker is defined by a profile. It is intended to allow significant events such as frame boundaries to be marked in the packet stream.
marker (M): 1 bit	The interpretation of the marker is defined by a profile. It is intended to allow significant events such as frame boundaries to be marked in the packet stream.
payload type (PT): 7 bits	This field identifies the format of the RTP payload and how the application interprets the content e.g. a specific codec like OPUS.
sequence number: 16 bits	The sequence number increments by one for each packet sent and may be used by the receiver to detect pcaket loss and to restore or predict lost packets.
timestamp: 32 bits	The timestamp is used to allow jitter calculations. The jitter buffer is used to order a few packets if they have not been received in their original order.
SSRC: 32 bits	The SSRC are identifies the synchronization source. It is possible that there exist multiple sources within an RTP session.
CSRC list: 0 to 15 items, 32 bits each	This represents all contributing sources. This field is used when multiple sources are mixed into one stream together and allow identification of the speaker.

Table 2.1: RTP header fields [HS03]

Abbreviation	Audio Bandwidth	Sample Rate (Effective)
NB (narrow-band)	4 kHz	8 kHz
MB (medium-band)	6 kHz	12 kHz
WB (wide-band)	8 kHz	16 kHz
SWB (super-wideband)	12 kHz	24 kHz
FB (fullband)	20 kHz	48 kHz

Table 2.2: Opus codecs bandwidths

This abilities are due to the fact that Opus codec is a combination of two industry proven codecs. The one codec is the SILK codec and the other one is the CELT codec. Firstly, the SILK codec, which was developed by the Skype team, is based on Linear Prediction and is Efficient for voice application and provides 8 kHz of audio bandwidth. Secondly, the CELT codec which was developed by Xiph.Org which is based on Perceptual transform (MDCT) codec. Due to its high bandwidth CELT is a good choice for universal audio/music.

However, combining the SILK codec and the CELT codec enables to dynamically adjust to different bandwidths even better. This procedure is referred to as the hybrid mode. It enables the use OPUS in a variety of applications and makes the connection even more robust to disturbances in connections.

As can be seen in figure 2.16 any given audio input In with 48kHz is encoded in both SILK and CELT encoder. The input for the silk encoder is cut down to 16kHz before encoded, while on the second path the CELT encoder receives a dynamic range of the input signal. Both signals are then mixed together and transmitted to the receiver. The signal is decoded by the SILK decoder and transformed from 16kHz to 48kHz and is then mixed with the output of the CELT decoder. This algorithm enables the codec to provide a better quality since it is always able to fill up gaps where packets got lost with the lower band version of the signal. While in situations when the connection is reliable and with a good bandwidth, the codec delivers up to full band quality which is comparable to CDs. In order to improve the internet robustness, Opus codec provides various features.

- Forward Error Correction (FEC)[JS15] is a mechanism against packet loss. Packets that are suspected to contain important speech information, like onsets or transients, are encoded at a lower bitrate. Later this re-encoded information is added to a subsequent packet.
- Discontinuous Transmission (DTX) reduces packet rate during silence. This decreases the traffic that needs to be transmitted in order to reproduce the audio signal.

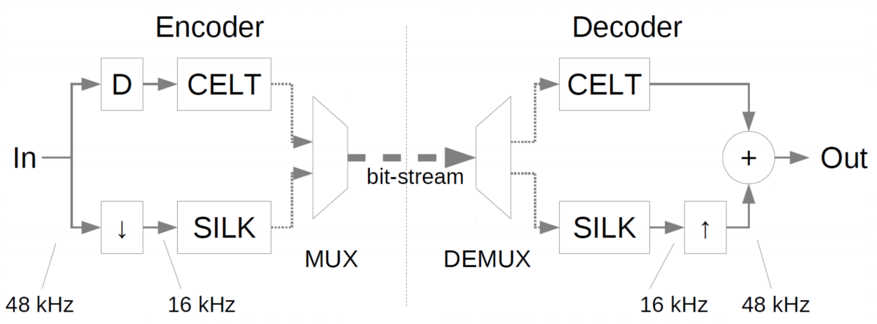


Figure 2.16: OPUS hybrid mode[JV12]

- Opus codec operates more efficient when using variable bitrates. On relatively slow connections bitrate is lowered which decreases the quality but avoids packet loss.

Summing up, all these features make OPUS codec the state-of-the-art VoIP codec since it is more reliable as classical codecs which have not been specifically designed for VoIP purposes. This has also been proven in various comparisons to other codecs.

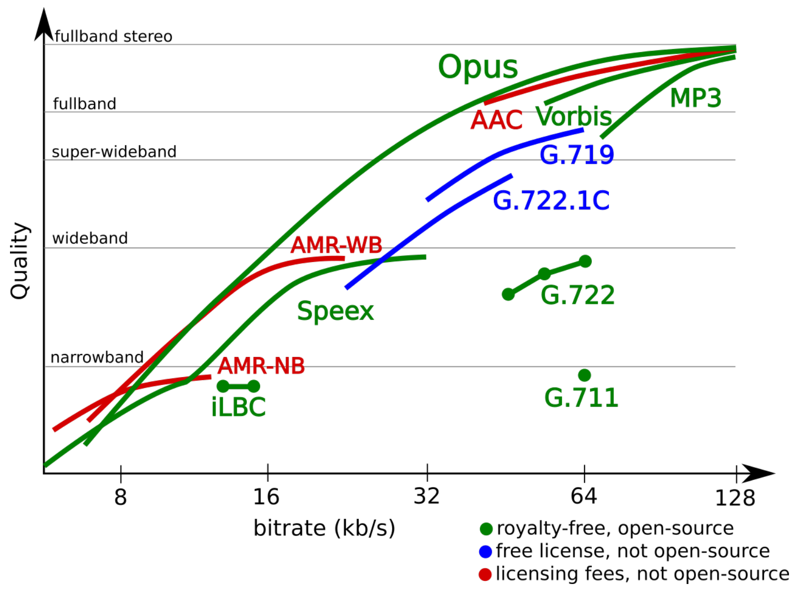


Figure 2.17: Comparison of quality and bitrate of various codecs.

One example is shown in figure 2.17, a comparison of quality and bitrate with other codecs, the Opus codec has at most bitrates the best quality results. While On the y-axis the quality of the codec, on the x-axis the bitrate is shown. The different codecs itself are depicted in different colors: green for royalty-free open source codecs, blue for free

license non-open source codecs and red for non-open source codecs which require to pay a licensing fee.
As depicted in the figure, it is the codec that covers the most possible bitrates, while other codecs (e.g. G.722 or G.711) that are still used in professional applications only cover a very small range. Another important argument for the codec is, that it is a royalty-free open source codec which allows usage without any additional costs when used in a project.

As a conclusion, Opus is the most advanced VoIP codec and is used in most of future applications. This results in applications that are more resilient to disturbances and have better voice quality.

CHAPTER 3

Methodology

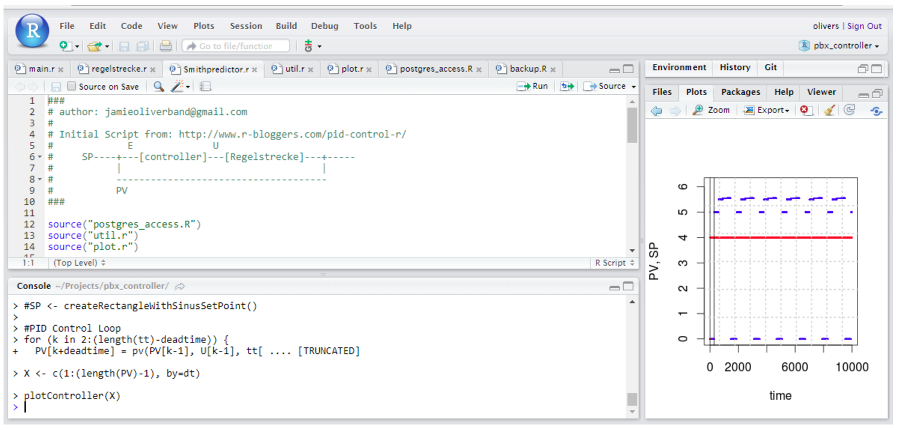
This chapter on methodology (chapter 3) presents the reader, on the one hand, used tools and concepts and languages. This covers R as a programming language up to the open source software like the PBX software. On the over hand, this chapter additionally covers the methods and concepts for designing and configuring controllers. This includes methods like the Ziegler-Nichols closed loop method, and the Chien, Hrones, Reswick tuning method. Last but not least, it also introduces the Smith Predictor as a compensator for delays.

3.1 Programming languages and used software

The following section describes the tools that have been used to successfully accomplish this work.

R[rpr] is a statistical programming language and includes an integrated suite of software facilities for data manipulation, calculation and graphical display.
It includes an effective data handling and storage facility, a suite of operators for calculations on arrays, in particular matrices, a large, coherent, integrated collection of intermediate tools for data analysis, graphical facilities for data analysis and display either on-screen or on hardcopy, and a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities. R was used to simulate the control loops in this work. All source code was written in *.R files. R Studio[rst] is a development environment for R. R Studio comes in two alternatives, the web-based variant and the Win32 executable version. In this case, the web-based version has been used.
As can be observed in figure 3.1, R Studio offers a console for executing *.r files and a plots view to show the plots created during the execution. Furthermore, R Studio offers the ability to debug code and also delivers integration with GIT. For this work, R Studio

has been used to save the progress on a Gitlab based GIT repository. FusionPbx[fus]


File Edit Code View Plots Session Build Debug Tools Help
olivers | Sign Out
pbx_controller
main.r regelstrecke.r Smithpredictor.r util.r plot.r postgres_access.R backup.R
Source on Save
Run
Source
Environment History Git
Files Plots Packages Help Viewer
Zoom Export
PV, SP
time
0 2000 6000 10000
0 1 2 3 4 5 6

```
###
# author: jamieoliverband@gmail.com
#
# Initial Script from: http://www.r-bloggers.com/pid-control-r/
#             E               U
#     SP----+---[controller]---[Regelstrecke]---+-----
#           |                                    |
#           -------------------------------------
#           PV
###

source("postgres_access.R")
source("util.r")
source("plot.r")
```

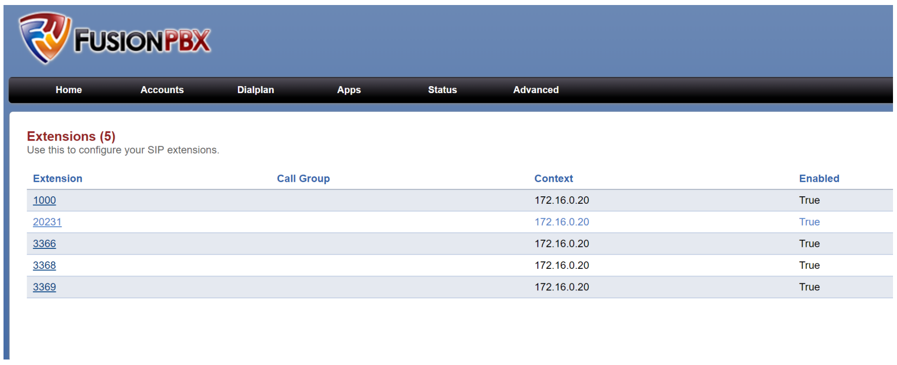
```
Console ~/Projects/pbx_controller/
> #SP <- createRectangleWithSinusSetPoint()
>
> #PID Control Loop
> for (k in 2:(length(tt)-deadtime)) {
+   PV[k+deadtime] = pv(PV[k-1], U[k-1], tt[ .... [TRUNCATED]

> X <- c(1:(length(PV)-1), by=dt)

> plotController(X)
>
```

Figure 3.1: rstudio Webinterface

is an open source PBX which was used as part of the test environment simulating the cloud service PBX. It was selected because it stores all data which is useful to obtain the PMOS value of calls. The product uses freeswitch[fre], an open source IP-based telephony switch, and PostgreSql as a database. Freeswitch is designed to route and interconnect popular communication protocols using audio, video or text. PostgreSQL[pos] is an open source database management system (DBMS) with a strong reputation for reliability, data integrity, and correctness. The DBMS is platform independent and therefore runs on major operating system such as Windows, Linux and Unix and uses Sequel Query Language (SQL)[sql] as a query language. To generate test calls, Linphone, an open source SIP phone[lin], was used.



Extensions (5)
Use this to configure your SIP extensions.

Extension	Call Group	Context	Enabled
1000		172.16.0.20	True
20231		172.16.0.20	True
3366		172.16.0.20	True
3368		172.16.0.20	True
3369		172.16.0.20	True

Figure 3.2: FusionPbx Webinterface

3.2 Methodologies for Configuring Controllers

As covered in chapter 2.1, PID controllers may tend to instability when used with systems providing large delays. To show how to configure such controllers, this section introduces a variety of methods.

In his work, Yoney[YOU07] evaluates the following methods: Ziegler-Nichols Open Loop method, the CHR method for 0% overshoot, and the Ziegler-Nichols Closed Loop method.

In 1942 Ziegler and Nichols proposed in their work[ZN42] *optimum settings for automatic control loops* two different methods for tuning P-, PI- and PID controllers. One method is the open loop method which would require removing the feedback of the control loop to configure the controller appropriately. The method that is covered by this work is the closed loop method which can be used in closed loops.
In his article, Haugen[Hau10] summarized the closed loop method of Ziegler and Nichols. He points out that Ziegler-Nichols' closed loop method can be applied only to processes having a time delay or having dynamics of order higher than 3.

According to Ziegler and Nichols[ZN42][Hau10], the stability of the controlled system is OK if the ratio of the amplitudes of subsequent peaks in the same direction is approximately $\frac{1}{4}$ as can be seen in figure 3.3. In his work, [Hau10] Haugen describes the process

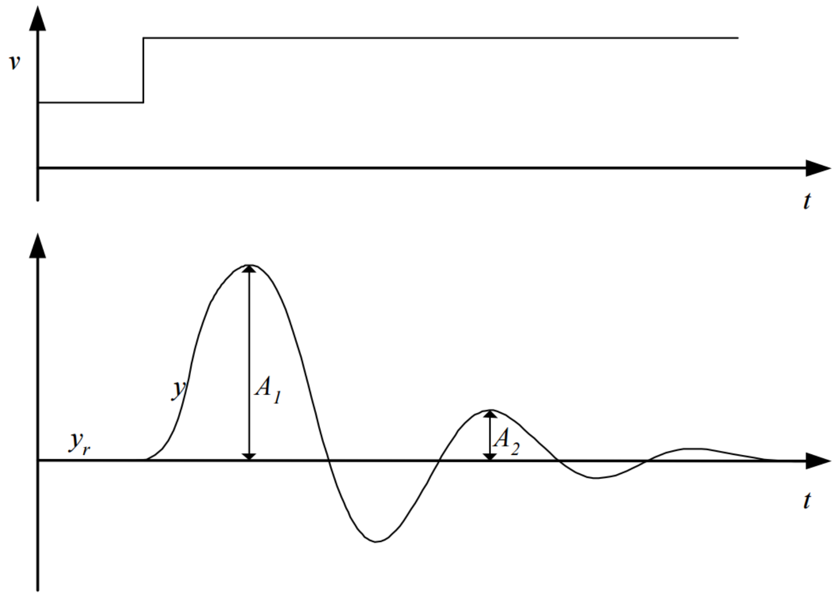


Figure 3.3: If A2/A1 $\approx \frac{1}{4}$ the stability of the system is OK, according to Ziegler and Nichols[Hau10]

of tuning a PID controller in a closed loop.

The tuning procedure is described as follows:

1. The controller should be configured to bring the controlled system to the specified operating point. E.g.: the process variable is approximately equal to the set point.
2. Turn the PID controller into a P controller by setting $T_i = \infty$ and $T_d = 0$. Initially set gain $K_p = 0$. Close the control loop by setting the controller in automatic mode.
3. K_p shall be increased until there are sustained oscillations. (refer to figure:4.4)
4. Measure the ultimate (or critical) period P_u of the sustained oscillations.
5. Calculate the controller parameter values according to table 3.5

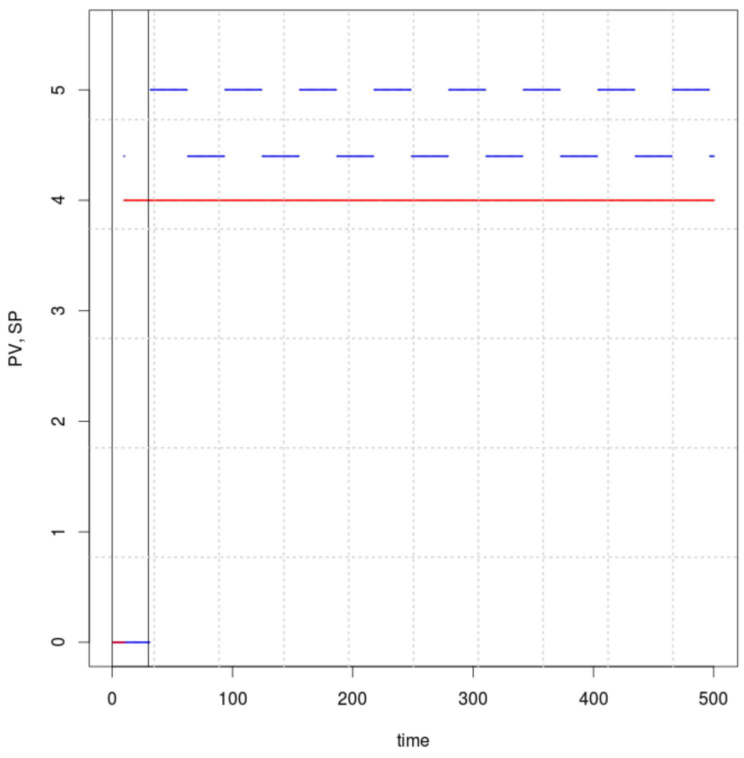


Figure 3.4: sustained oscillations

After configuring the controller parameters, the control loop is ready for use.

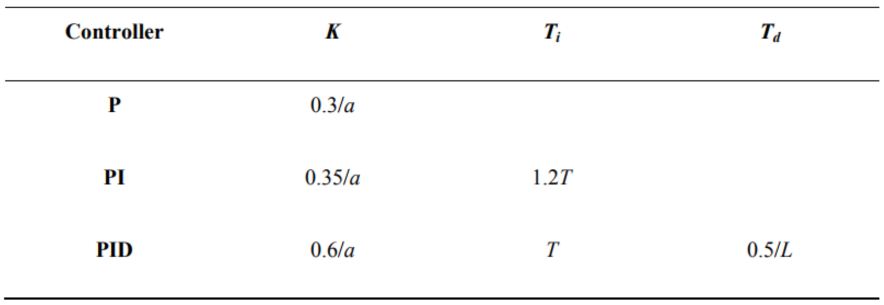
	K_p	T_i	T_d
P controller	$0.5K_{p_u}$	∞	0
PI controller	$0.45K_{p_u}$	$\frac{P_u}{1.2}$	0
PID controller	$0.6K_{p_u}$	$\frac{P_u}{2}$	$\frac{P_u}{8} = \frac{T_i}{4}$

Figure 3.5: Formulas for the controller parameters in the Ziegler-Nichols' closed loop method.[Hau10]

3.2.1 Chien, Hrones, Reswick Tuning Method

In his work, Youney[YOU07] describes the Chien Hrones Reswick (CHR) Tuning Method. The Chien Hrones Reswick tuning method was derived from the classical Ziegler Nichols method. The intention is to obtain the quickest response without overshoot and the quickest response with 20 percent overshoot.

To tune the controller according to the CHR method, the parameters a, L and T have to be determined. The determination of the parameters is similar to those in the Ziegler-Nichols method. The controller parameters can be calculated using the table 3.6 for 0 percent overshoot and table 3.7 for 20 percent overshoot. Furthermore, Youney states that due to some edge cases, Astrom and Hagglund developed a method to solve for T with an open loop step response test. Youney states that there are also many



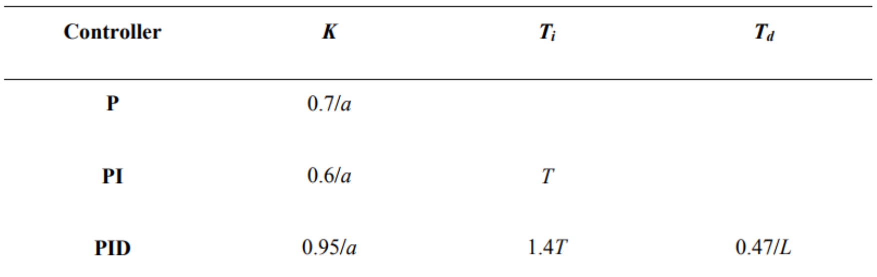
Controller	K	T_i	T_d
P	$0.3/a$		
PI	$0.35/a$	$1.2T$	
PID	$0.6/a$	T	$0.5/L$

Figure 3.6: CHR for 0 % overshoot parameters

other methods that are based on the Ziegler Nichols method. In conclusion, the above methods can be easily used to tune PID controllers with delays. Certainly, one of the downsides is that the methods are not adapting to changes in delays.

3.2.2 Smith Predictor

The Smith predictor is described by Vodencarevic[BMD11] as a dead time (delay) compensator. Vodencarevic states that in practice many control loops are tuned by trial and error procedures and therefore are far from optimal and delivering a bad perfor-



Controller	K	T_i	T_d
P	$0.7/a$		
PI	$0.6/a$	T	
PID	$0.95/a$	$1.4T$	$0.47/L$

Figure 3.7: CHR for 20 % overshoot parameters

mance. According to Daxini et. al [TD15] the first time-delay compensation algorithm was suggested by Otto Smith in 1957. This algorithm is now known as the Smith predictor.

The Smith predictor is composed of two parts: the controller part and the predictor part. The controller part consists of all the components a standard control is built from, while the predictor part consists of a model of the process without time delay and a model of the time delay.
Daxini et. al explain that the comparison between the output of the process $y(t)$ and the model including time delay is the predicted error $\hat{e}_p$. If there are no errors in the modeling and furthermore no disturbances, then the error is zero. Tala et. al 3.9 provided a block

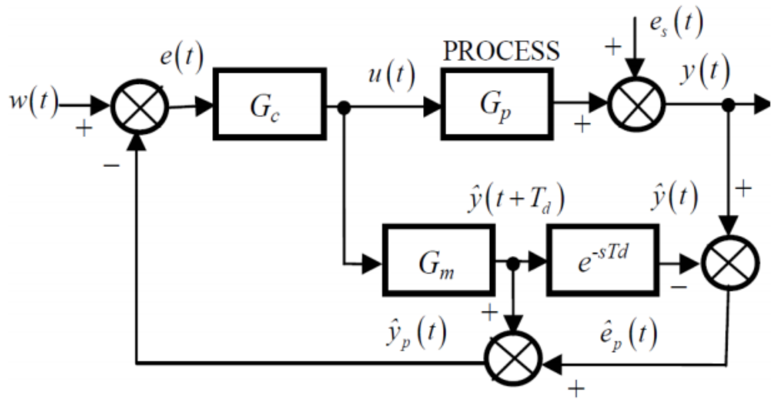


Figure 3.8: block diagram of a smith predictor[TD15]

diagram 3.9 of a Smith predictor that allows for an overview of the controllers' variables. The diagram shows that G_m can be translated logically to gains and time constraints as a model of the controlled system without delays and e^{sT_d} to a model of the delay(dead time). Therefore, if the delay time changes, only the dead time component needs to be adjusted. Also, one can observe that the estimated disturbances and predicted process variable are added to the signal. As a result, if the process is changing over time also the process model has to change. Unfortunately, the complexity of this method and the need to create a model of the controller lead to increased efforts.

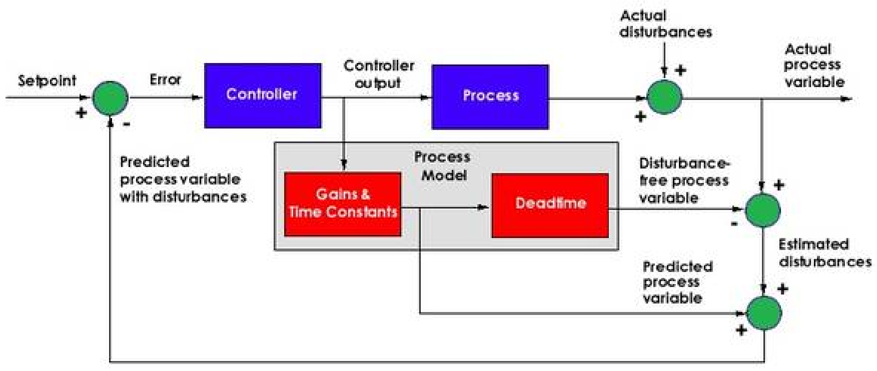


Figure 3.9: block diagram of a Smith predictor[TD15]

CHAPTER 4

Suggested Solution

This chapter shows how to design a controller based on the earlier defined problem. In a first step, the architecture of such a proposed cloud service will be presented. The design is based on the below defined requirements. Then the design of the controller, including the stability and the transfer function, will be discussed Also, the solution will be analyzed and evaluated using a cost function.

4.1 Architecture for a VoIP cloud service

As stated previously, the aim of this work is to apply control theory to informatics. Control theory is a proven scientific method which has been applied to a variety of different use cases in the past, e.g.: heatings, power plants, control of water cleaning facilities and so on.

This work takes this approach and applies it to a VoIP PABX cloud service. A substantial question for a service provider would be *How to scale if there is a sudden increase or decrease of the workload?* Not only does a service provider want to handle peaks or lows in the workload, but also to maintain a certain quality for such a service to increase customer satisfaction on the one side and to be able to monitor the performance of the service. Consequently, the architecture of the VoIP cloud service shall be scalable on demand. Therefore, the solution is modularized into several parts. As can be seen in figure 4.1, such a cloud service consists of several components:

A **client** or agent can be any SIP device like a desk phone, a desktop application a mobile app, etc. The client connects to the cloud service over the internet using either an IP address or a domain name and registers on a so-called **Session Border Controller (SBC)**, which is the entry point for all incoming and outgoing communication. Firstly, the SBC acts as a VoIP firewall since all external agents need to register on it. Secondly, the session border controller is the endpoint for each SIP session to the agent. On the

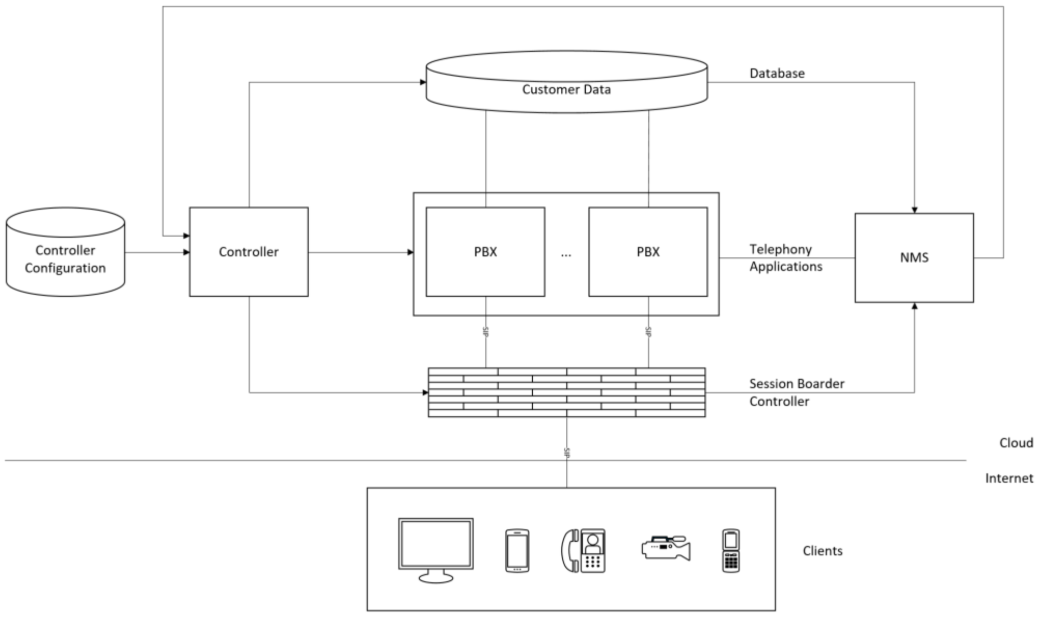


Figure 4.1: Architecture of the proposed cloud service

other side it opens a new connection to the internal network. That way it acts as a transparent layer.
If needed, the session border controller is able to convert non-standard protocol implementations to standardized protocols (protocol normalization). That way the internal network is more resilient to interruptions from the external network.
Furthermore, the session border controller also acts as a load balancer. As a load balancer, the SBC distributes incoming calls to the internal communication network components e.g.: PBX.

A **Session Border Controller (SBC)** is the entry point for all incoming and outgoing communication and:

- acts as a VoIP Firewall and entry point for external SIP agents
- enforces Protocol Normalization
- acts as a Load Balancer

As a result, the SBC should be taken into account for scalability reasons. In the proposed scenario it is therefore assumed that the SIP protocol from the clients is standardized. Consequently, there is no need to perform any protocol conversion. Furthermore, it is assumed that the SBC has enough resources to distribute the calls to the internal network.

In conclusion, there is no need for scalability of the SBC.

The purpose of a **Private Branch Exchange (PBX)** is to establish phone calls between different clients. Those calls can be either audio or video conversations consisting of two or more clients. The performance of such a PBX is depending on RAM, CPU, network bandwidth. These parameters have an impact on the quality of calls and result in better or worse quality of calls (MOS value). To simplify the models, it is expected that all calls are established with the same preconfigured codecs to achieve better comparability. The PBX is establishing calls via SIP and RTP protocol.

Naturally, a cloud service provided needs some sort of customer data for billing and statistics. Those statistics are used to measure the quality of VoIP calls, but also to keep track of which users are the most active ones and when the most calls are established. A **Network Monitoring System (NMS)** is capable of monitoring certain metrics and SLO objectives such as network parameters. This enables the service provider to better monitor the communication infrastructure. The obtained data is then used predict the state of the PBXs, especially if more computing power is needed or not. The key task of the **controller** is to decide if a new PBX must be started by processing the measured metrics. This must be initiated whenever the PMOS value is falling under a configured level. As a result, whenever the PMOS gives an indication, further VM should be started or shut down. In the edge case of only one PBX running, it is of course not possible to shut down the remaining PBX even if the controller would decide to do so based on the PMOS value input. The controller's configuration parameters are stored in the controller configuration. The configuration can be adjusted to the measured delays.

4.2 Definition of SLOs and Selection of Metrics

To make such a service scaleable, a service provider needs to define a certain SLO and other specific metrics. Several SLOs like availability, response time, CPU utilization or RAM usage would be applicable, but would not be representative of the quality of the phone calls. One of the key aspects of such a service is good quality of the conversations. This means that the users can perform audio or video calls at a certain level of quality that allows them to communicate with each other without any notable interruption. The assumption on the SLO is that if the customer is happy with the service e.g. the quality of the calls is considered as *good*, then the service provider is satisfied as well. Therefore, the PMOS factor should be considered as a metric. As shown in an earlier chapter, the PMOS factor can be observed without manual interaction and is also independent of the underlying hardware.

Since service providers would like to have good quality a PMOS value of 4 is reasonable.

Additionally, the PMOS factor will also be influenced by parameters like CPU or RAM usage. The assumption is that as long as CPU and RAM are in a normal range, the PMOS factor will not change too much from value 4. If CPU and RAM are decreasing and crossing a certain limit, the PMOS factor will consequently decrease.

4.3 Requirements for a VoIP cloud service

Based on the above agreed SLOs and architecture of the service, now more detailed requirements will be summarized and shall serve as an example for further implementations. Since our service needs is scalable, from time to time it will need to start virtual machines to increase the amount of resources used by the service. Unfortunately, this process of setting up such a machine takes some time. As we use an service for providing the virtual machines we can assume that there will be no time delay for starting physical machines. Nevertheless, we still must take into account the time that is needed to start and initialize the virtual machine hosting the PBX. For this reason, we can assume that it will take a maximum of 5 minutes. Since we do a simulation to verify the results, the below table 4.1 shall summarize the most important simulation parameters:

Parameter	Description	Value	Unit
VM start time	Time a virtual machine needs from triggering a start	5	Minutes
desired PMOS	desired PMOS	4	Integer
maximum cycle number	cycles after which the desired value is reached	3	Integer
simulation duration	Time range that is simulated	100	Minutes
simulation step size	granularity of simulation	10	Integer

Table 4.1: Parameters relevant for the controller design

4.4 Design of the controller

This section describes design decisions regarding the controller and justifies these decisions briefly.

One of the key aspects is the control of the scalability which shall be achieved by designing the controller. As one can observe in figure 4.2, the model consists of a controller and one or more PBX that allow calls between different participants to take place. The controller will decide based on information gained from the database of the PBX if the

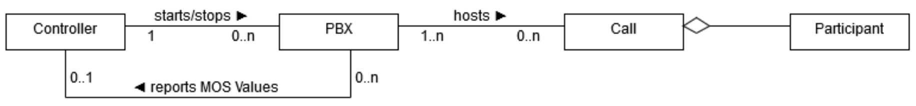


Figure 4.2: Architecture

resources are sufficient or if another PBX needs to be started. Before starting a PBX server, for instance, the server must be created from an image first. That means that system resources need to be allocated, the system image must be copied and booted to finally inject the configuration. As a result, this process will need a certain amount of time to finish. This duration is the delay time of our process. The delay time is crucial as it might cause instability, or even worse, oscillation of the control loop if it gets too long.

Applying this knowledge to classical control theory, as it is used in many other engineering areas, leads us to select a PI controller. PI controller have an advantage over normal PID controller as they are more smoothly adjusting to changes and therefore are more reliable in case of large delays. It is also a classical controller for delays (dead times)

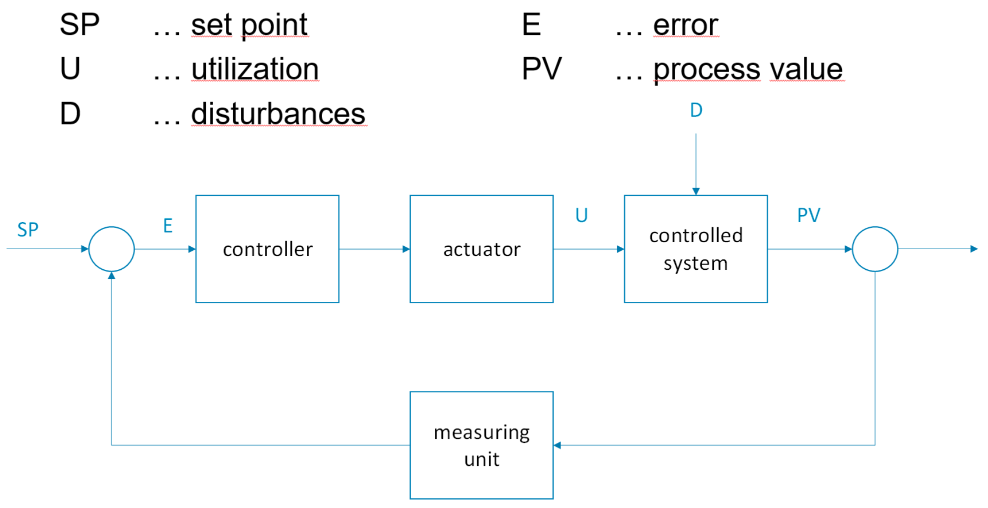


Figure 4.3: Closed loop

or controlled systems with long delays and is actually a special configuration of the PID controller where the D factor is set to zero. As a high D factor makes the controller faster reacting on incoming changes in the error. This property leads to a controller configuration which concludes in a less resilient behavior to delays. Since the stability increases with the value of T_I, it correlates with the delay. The properties of the PI controller enable proper handling of delays and increase the stability of the controller for that purpose.

One of the main goals is to design the controller so it operates in an area where it resilient to the assumed delays. The below section introduces a calculation using Ziegler-Nichols method to calculate the parameters satisfying the stability criteria.

The first step is to configure the controller according to the Ziegler-Nichols tuning table for a PID controller. To achieve a stable configuration of the controller, we first bring the controller into sustained oscillation as described in 3.2. Sustained oscillations are achieved if the oscillation has no attenuation. Applying the Ziegler-Nichols method, we obtain the following controller values:

$K_p = 11$
$T_i = \infty$ (100000.0001)
$T_d = 0$
Figure 4.4 depicts a step function (represented by the red graph) and the actual process

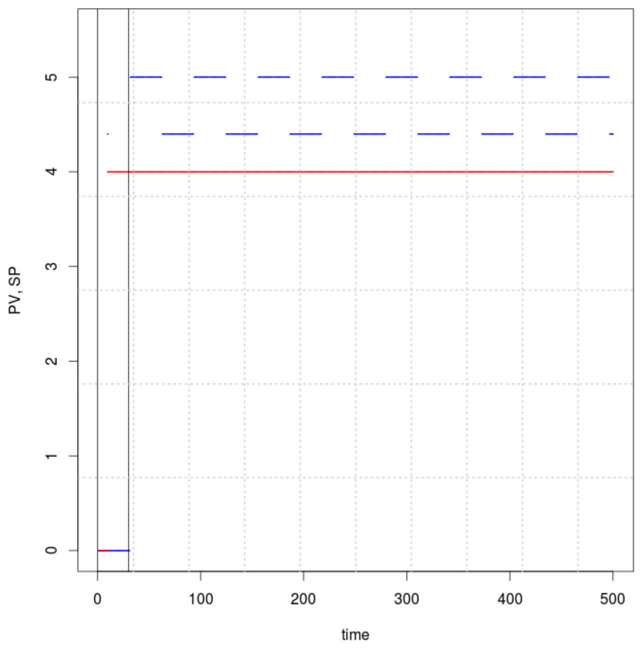


Figure 4.4: sustained oscillations

value (represented by the blue graph) over the time (x-axis). One can clearly see that the process value is oscillating between 5 and around 4 $\frac{1}{2}$ in sustained oscillations. The next step is to derive the values needed for the final configuration of the PI controller.

The ultimate Period $P_u = 600ms$. According to table 3.5 the values for the PI controller are chosen as follows:

$K_p = 0.45 * K_pu = 0.45 * 11 = 4.95$

$T_i = \frac{P_u}{1.2} = \frac{600ms}{1.2} = 500ms$

The resulting configuration enables the controller to handle the specified delays occurring when starting a virtual machine. Finally, the controller is ready for use and can be tested. Further details will be covered later in chapter 5.4.

4.5 Definition of a utility and a cost function

To make the results of the controller measurable, a definition of a utility and a cost function is mandatory. One approach is to measure the difference in the quality of calls. The smaller the area under the desired PMOS factor, the better. Unfortunately, this does not consider energy consumption. The usage of the service depends on how reliable it performs. Therefore, the income of the service provider relies on this variable. Referring to chapter 2.3.1, one can see that the income is basically depending on the usage of the provided service:

$Utility(a, t) = Income(a, t) - EnergyCost(a, t)$

As shown in chapter 2.3.1 the costs can also be splitted into the following parts:

$Utility(a, t) = Income(a, t) - (EstimatedEnergyCost(a, t) + EstimatedViolationCost(a, t) + PDMCost(a, t))$

The above formula uses the variable a as a mapping of VMs to PMs. Since the proposed scenario runs fully on amazon virtual machines, we can assume that all VMs are mapped to one single PM. The price for the energy costs can be estimated by taking an example from the Amazon e2 service[ama].
According to the Freeswitch website there is a FreeSwitch installation in production running on an Amazon EC2 - m1.medium [ec2] virtual machine. This virtual machine can run 250 concurrent calls at 15% of CPU usage with a maximum of 15-20 calls per second. This information shall serve as a reference for this use case.

The estimated violation costs can be deducted by measuring the difference of the actual MOS value to the expected MOS value, if and only if the actual value is smaller than the expected value. The higher the difference, the higher the costs. For real life implementations one must consider how hard such a violation should be punished and how violations should be weighted since there are situations where the energy cost would rise too high. For now this is out of scope and should be covered in succeeding work.

CHAPTER 5

Evaluation

The aim of this chapter is to evaluate the proposed solution. I will give an overview on the two variants of test environments that have been used to capture data and were later used to test the controller simulation. Furthermore, the obtained data is used to compare the proposed solution with the related work. In the end of the chapter, open issues will be discussed.

5.1 Test Environment

The test environment was used to ensure that the assumptions made previously were justified. In the first step shown in figure 5.1, a simple SIP Phone was used to create some calls and enable us to analyze how to extract the MOS values from the PBX's database.

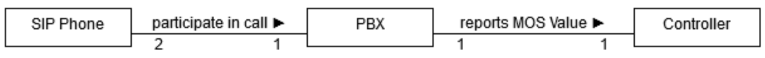


Figure 5.1: creating calls using a SIP Phone

The listing below gives a sense of how easily the PMOS value from fusionpbx is obtained in R. As mentioned in a previous chapter, fusionpbx is based on freeswitch. Fusionpbx uses PostgreSQL as a database and allows to obtain MIN, MAX and AVG PMOS of the calls in a certain time interval by a simple SQL SELECT statement. Tests showed that the interval should be not too small and a small minute interval fits perfectly. In real application, this parameter should be dynamically configured.

```
require("RPostgreSQL")

```

```
3  getPMOSFromFusionPbx <- function(){
4    pw <- {
5      "pwd"
6    }
7
8  drv <- dbDriver("PostgreSQL")
9  con <- dbConnect(drv, dbname = "fusionpbx",
10                  host = "172.16.0.20", port = 5432,
11                  user = "fusionpbx", password = pw)
12
13 rm(pw) # removes the password
14
15 dbExistsTable(con, "v_xml_cdr")
16
17 # query the data from PostgreSQL
18 df_postgres <- dbGetQuery(con,
19 "SELECT␣MIN(rtp_audio_in_mos)␣as␣min,
20 MAX(rtp_audio_in_mos)␣as␣max,␣AVG(rtp_audio_in_mos)␣as␣avg
21 FROM␣v_xml_cdr␣WHERE␣rtp_audio_in_mos␣IS␣NOT␣NULL")
22
23 # AND answer_stamp > (NOW() - INTERVAL '5' MINUTE)")
24
25 # compares the two data.frames
26 identical(df, df_postgres)
27
28 on.exit(dbDisconnect(con))
29 dbUnloadDriver(drv)
30
31 return (df_postgres);
32 }
```

Listing 5.1: postgres_access.R

In line 18 the minimum, the maximum and the average MOS values are extracted.

To create more realistic values of the MOS factor the Voxtronic Call Generator was used. The call generator played a minute-long audio file and was forwarded in a dial plan to a number which created echo. As a result, it could be observed that the resulting MOS factor was usually around 4.

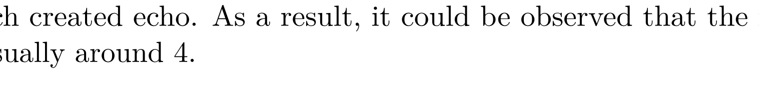


Figure 5.2: Call generator used to create work load on the PBX.

5.2 Analysis of the working solution

The solution has been implemented using an PI controller since it is more resilient to delays than a PID controller. In fact, using a normal PID controller would lead to unpredictive behavior as can be seen in figure 5.3. The controlled variable PV (blue in the figure) and the utility variable U (green in the figure) are not able to follow the set point variable SP (red in the figure). It even appears that there are several points with infinity values, indicated by the ellipses in green and blue.
This behavior can be explained by having a look at the equation of the PID controller. While the P property is time-independent, the I property will just slowly adapt over time to changes. The D component on the other side immediately tries to fix deviations in the processed value based on the deviation of time. Due to the delay the controller is not able to apply any change on the controlled system. In conclusion the PID controller is not usable for our purpose. When compared to a PID controller, the PI controller

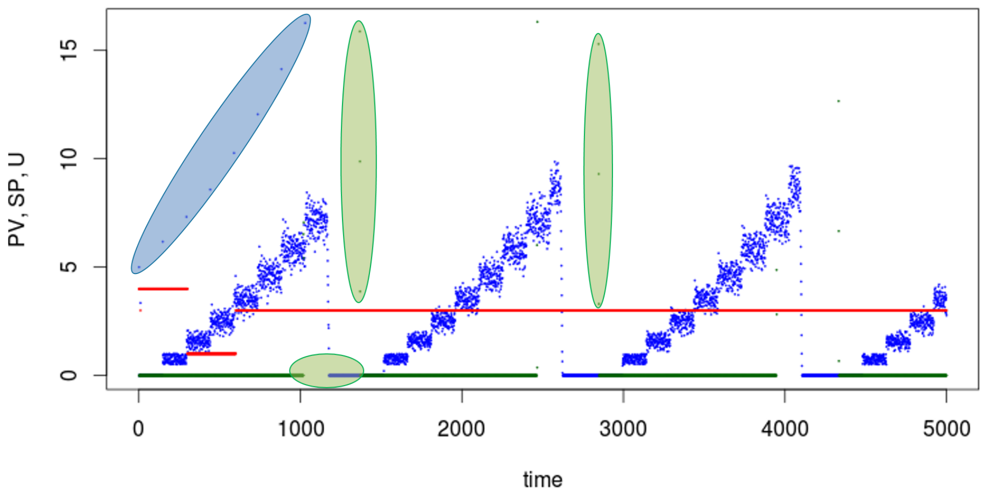


Figure 5.3: PID controller with delay

is much more resilient to long delays. In figure 5.4, one can see the step response of the PI controller for the chosen simulation values. It is obvious that the desired MOS value is achieved within 2 cycles. Afterwards the curve even overshoots for a few cycles. The overshooting depends on the starting value of the set point variable. As a result, the adjustment from zero to 4 the overshooting to be much higher and last longer than the adjustment from 2 to 4. To simulate an PI controller with a PID controller, R was used. The basis for the source was originally from r-bloggers[hbcr16], which showed a basic PID controller implementation. From line 6 to line 8 the controller parameters are configured, these have been changed according to the controller design. Note, that T_d is set to zero as the controller is configured as an PI controller. Line 11 to line 13 define the simulation parameters which are derived from the requirements. dt is the time step. tt is the time vector containing all the time values. Finally, the delay is defined by the

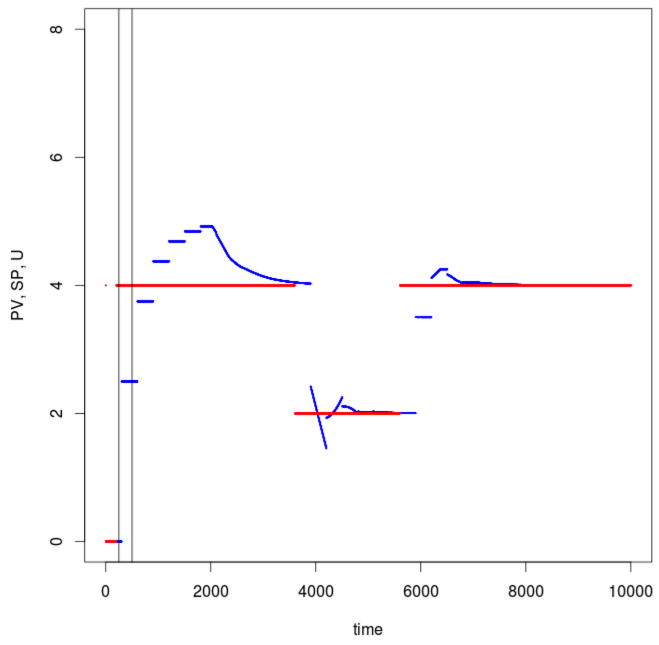


Figure 5.4: PI Controller working

VM start up time.
Eventually, the processed variables are plotted into a diagram 34.

```
source("postgres_access.R")
source("util.r")
source("plot.r")

# controller parameters
Kp = 8                          # proportional gain
Ti = 50                         # integral time
Td = 0                          # derivative time

# simulation parameters
dt = 0.1                         # time step
tt = seq(0, 1000, by=dt) # time vector
deadtime=300 # startup time in seconds of the VM

# initialize the following to a vector of zeros
dy = y1 = OUT = Gp = F = PV = U = E = EI = ED = rep(0, length(tt))
PV[1] = 0 # initial state of the process variable

```

```r
SP <- createRectangleSetPoint()

for (k in 2:(length(tt)-deadtime)) {
  PV[k+deadtime] = pv(PV[k-1], U[k-1], tt[k])

  E[k] = PV[k] - SP[k] # proportional
  EI[k] = EI[k-1] + E[k]*dt  # integral
  ED[k] = (E[k] - E[k-1])/dt # derivative

  U[k] = Kp*(E[k] + (1/Ti)*sum(E*dt) + Td*ED[k])

  if (U[k] < 0) U[k] = 0
}

X <- c(1:(length(PV)-1), by=dt)
plotController(X)
```

Listing 5.2: main.r

Util.r provides some very helpful functions. One of the most important functions is the createRectangleSetPoint function on line 4 which sets the SetPoint variable. The createRectangleWithSinusSetPoint function on line 11 creates a step function that jumps from zero to 4. This function was mainly used to analyze the behaviour of the controller.

```r
1  delayBuffer  = rep(0, length(tt))
2
3  #Setpoint Rectangle
4  createRectangleSetPoint <- function(){
5    SP = rep(0, length(tt))
6    SP[which(tt >= 1)] = 4
7
8    return (SP)
9  }
10
11 createRectangleWithSinusSetPoint <- function(){
12   SP = sin(0.5*tt) + cos(.8*tt) + 0
13   SP[which(tt >= 20)] = 0.3*(sin(0.5*tt) + cos(.8*tt)) + 4
14   SP[which(tt >= 360)] = sin(0.5*tt) + cos(.8*tt) + 1
15
16   return(SP)
17 }
18
19 pv = function(pv.prev, u, tt) {
20   out = (pv.prev*.1 + 4 )  # exponential growth + linear growth
21   out = out - 0.1*u        # the control response
```

```

  out = out + .2*runif(length(tt)) # noise function

  if (out < 0) out = 0

  return(out)
}

Tt = function(pv.prev, u, tt) {
  out = (pv.prev*.1 + 4 )          # exponential growth + linear growth
  out = out - 0.1*u                # the control response

  out = out + .2*runif(length(tt)) # noise function

  if (out < 0) out = 0             # keep values positive
  #if (out >= 5) out = 5 ###eliminate values over 5
  return(out)
}
```

Listing 5.3: Util.r

5.3 Evaluation of Costs

To make the proposed solution comparable to others and also to show the effectiveness, a cost evaluation has to be completed. Therefore, service level agreements must be defined, including which penalties should be imposed if the desired quality is not achieved and which reward should be granted if the desired MOS value is achieved. Furthermore, it should be taken into consideration that each virtual machine consumes energy and therefore costs money. It is obvious that if there are more virtual machines than necessary, costs are effected.
But, since the purpose of this work was solely on the quality of the speech, the costs of energy have not been taken into consideration. Instead of the energy costs the efficiency of the solution should be measured by using the following formula:

Effeciency of the controller:

$$\frac{\sum_{n=1}^{N} SP(n)-PV(n)}{N}$$

Similar to an integrator the above formular sums up the differences between the set point and the processed value at any given time. As a result, the area describes the deviation of the expected curve, comparable with any other controller. Therefore, a smaller deviation means the used algorithm is better than one with a higher deviation.

5.4 Comparison with other Controller Types

In order to make this solution better comparable with future work, this chapter will provide some suitable properties. The suggested solution was compared with a Standard PID controller with the variables used, if no delay occurs in the system and with a Smith Predictor.
All selected controllers will be compared based on the properties presented in table 5.1. The column Parameter is a short description of the property that is evaluated. Secondly, the column labeled as description is a longer description of the property which should also give a sense of how it can be measured. The column named Unit describes the measurement unit of the property.
To summarize the findings, this table is filled out for the applicable controllers and should serve as a comparison to the interested reader for his/her own selections of a controller.

Parameter	Description	Unit
Complexity of the Controller	describes how difficult the controller is to implement	easy, medium, hard
Self-adaptive	describes if the controller adapts to changes in the delay	Yes, No
Deviation from ideal value	Area under which the controlling output is not ideal	Integer

Table 5.1: Framework for comparing controllers

Parameter	Evaluation Result
Complexity of the Controller	medium
Self-adaptive	No
Deviation from ideal value	on long delays unpredictable

Table 5.2: PID Controller

Parameter	Evaluation Result
Complexity of the Controller	hard
Self-adaptive	No
Deviation from ideal value	depends on the accuracy of the model

Table 5.3: Smith Predictor

As can be seen in tables 5.4 and 5.2 above, the PI and PID controller are much easier to configure and to tune compared to a smith predictor. One advantage of the Smith Predictor on the other side is that if the model of the system is clear, the controller is much more accurate. Having said that, the Smith Predictor has two problems that need to be tackled. One problem is, that it requires a significant amount of effort to design

Parameter	Evaluation Result
Complexity of the Controller	medium
Self Adaptive	No
Deviation from ideal value	depends on the tuning method used for configuration of the paramet

Table 5.4: PI Controller configured for no delay

such a smith predictor. When configuring the Smith Predictor during this work, I have struggled with the Smith Predictor quite a while and still it was hard to configure it. The second problem is that if the smith predictors' model is slightly deviating, the controller will generate an output that drives the actual process variable off into oblivion.
Unfortunately, all selected controllers are not self-adaptive. However, this should not result in issues in this case, since the delay times does not vary too much for replicas of the same VM. Due to the lack of the D component in the PI controller, it outperforms the PID controller when it comes to delays.

5.5 Complexity Analysis

This section shall discuss the runtime in terms of a complexity analysis of the proposed solution. Therefore, each function used will be analyzed and provide the complexity in θ notation.

The main program consists of one big for loop containing all calculations done. Furthermore, three other functions can be identified.

The first one is the createRectangleSetPoint() function which is used to set the Set Point variable. Its complexity is $\theta(1)$ since only assignments are processed. The second function is the plotController function which is used to plot the inputs and outputs of the controller. As the runtime strongly depends on the number of input arrays and there size, the complexity can therefore be considered as $\theta(n^2)$ Also notable is the pv() function which calculates the actual value that is available on the controlled system. As there are only assignments in this function, its complexity can be considered as $\theta(1)$.

The main loop consists of different statements and also the pv function which we have shown to have $\theta(1)$ complexity. As a result, the main loop's complexity is $\theta(n)$.

To summarize for all parts of the main program, the complexity is as follows:

$$\theta(mainloop) + \theta(createRectangleSetPoint) + \theta(line\ 33) + \theta(plotController) =$$

$$\theta(n) + \theta(1) + \theta(n) + \theta(n^2) =$$

$\theta(2n + 1 + n^2) =$

$\theta(2n + n^2) = \theta(n^2)$

However, since the plotController function would not be used in a real life version. As a result the complexity of the algorithm would decrease to $\theta(2n)$.

CHAPTER 6

Summary and future work

This work presented how classical control theory can be used in computer science. Based on the example of VoIP services, it showed how a PI controller can be used to achieve this task. The purpose of the controller is to provide service users a constant quality of speech. As a controlled value, the MOS value was selected since it is a standard for monitoring the quality of VoIP networks. It has been shown how this value is extracted from a modern open source PBX by a simple SQL command.

The work describes that starting a new machine leads to a delay in the control loop which is a huge problem for a PID controller. Therefore, the PID controller has been replaced by an PI controller which is more resistant to large delays since it has no derivative component. It was shown that the PI controller could provide stability. The controller values have been selected using the Ziegler-Nichols method which was described in the work.

As a result of the work, it has been shown that the PMOS value is an eligible parameter that can be observed to not only monitor VoIP networks but also to achieve horizontal scalability. As a downside, classical controllers do not adapt automatically to changes in the system e.g. longer starting time of virtual machines. One way to address this would be using modern controllers that are self-adaptive to changes of such values. However, the PI controller would be very simple to implement and functionally a good choice, unless the starting of virtual machine would result in unpredictable startup times.

Future work should focus on the one side on the implementation of a modern controller which is self-adaptive to changes as discussed above. Also, the controller could combine different parameters such as RAM, CPU, etc. to evaluate the controlled system even more accurately. It also might be a bit more preactive rather than waiting for the quality to actually decrease. Furthermore, the solution should be implemented using a modern programming language such as Java or Python with, e.g., a Freeswitch cluster along with the designed controller at a cloud service. This would provide further proof of the simulated data. Another interesting topic is how different types of media could have implications on the choice and complexity of the controller.

Summing all up, this work showed successfully, that the scalability of a VoIP service can be controlled using the quality of calls as a metric for a PI controller.

List of Figures

List of Tables

List of Algorithms

Bibliography

[20111] IDATE & UMTS Forum 2011. Mobile traffic forecasts 2010-2020 report - UMTS Forum Report 44 page 74. Master's thesis, UMTS Forum, 2011.

[AAY11] Imad AL Ajarmeh, Mohamed Amezziane, and James Yu. Modeling call arrivals on voip networks as linear gaussian process under heavy traffic condition. Master's thesis, Los Alamitos, CA, USA, 2011.

[ama] Amazon ec2 pricing. `https://aws.amazon.com/ec2/pricing/on-demand/`. Accessed: 2017-11-04.

[BH14] Amid Khatibi Bardsiri and Seyyed Mohsen Hashemi. Qos metrics for cloud computing services evaluation. *I.J. Intelligent Systems and Applications*, 12:27–33, 2014.

[BMD11] Vladimír Bobál, Radek Matunu, and Petr Dostál. Digital smith predictors-design and simulation study. In *ECMS*, pages 480–486, 2011.

[CSI14] CSIG. Cloud Service Level Agreement Standardisation Guidelines. *CSIG-SLA*, 2014.

[ec2] Real-world results. `https://freeswitch.org/confluence/display/FREESWITCH/Real-world+results`. Accessed: 2017-11-04.

[fre] Freeswitch. `https://freeswitch.org/`. Accessed: 2017-09-30.

[fus] Fusionpbx. `https://www.fusionpbx.com/`. Accessed: 2017-09-30.

[Haa03] Dipl.-Ing. Dr. Wilhelm Haager. *Regelungstechnik*. 2003.

[Hau10] Finn Haugen. Ziegler-nichols' closed-loop method. *TechTeach*, 2010.

[hbcr16] http://www.r-bloggers.com/pid-control r/. PID Control in r, 2016.

[HS03] R. Frederick V. Jacobson H. Schulzrinne, S. Casner. Rfc 3550 - rtp: Real time protocol, 2003.

[IB11] Vincent C. Emeakaroha Ivona Brandic Schahram Dustdar Ivan Breskovic, Michael Maurer. Cost-efficient utilization of public sla templates in autonomic cloud markets. 2011.

[ITU01] ITU. Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. Master's thesis, INTERNATIONAL TELECOMMUNICATION UNION, 2001.

[JR02] G. Camarillo A. Johnston J. Peterson R. Sparks M. Handley E. Schooler J. Rosenberg, H. Schulzrinne. Rfc 3261 - sip: Session initiation protocol, 2002.

[JR06] G. Camarillo A. Johnston J. Peterson R. Sparks M. Handley E. Schooler J. Rosenberg, H. Schulzrinne. Rfc 4566 - sdp: Session description protocol, 2006.

[JS15] JM. Valin J. Spittka, K. Vos. Rfc 7587 - payload format for the opus speech and audio codec, 2015.

[JV12] T. Terriberry JM. Valin, K. Vos. Rfc 6716 - opus audio codec, 2012.

[lin] Linphone. `http://www.linphone.org/`. Accessed: 2017-09-30.

[MM11] Ivona Brandic Jörn Altmann Michael Maurer, Vincent C. Emeakaroha. Cost and benefit of the sla mapping approach for defining standardized goods in cloud computing markets. 2011.

[MP16] Abdelkhalik Mosa and Norman W. Paton. Optimizing virtual machine placement for energy and sla in clouds using utility functions. *Journal of Cloud Computing*, 5(1):17, Oct 2016.

[MR09] J. Altmann. M. Risch, I. Brandic. Using sla mapping to increase market liquidity. 2009.

[p86a] Objective quality measurement of telephone-band (300-3400 hz) speech codecs. `https://www.itu.int/rec/T-REC-P.861/en`. Accessed: 2017-12-20.

[p86b] Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. `https://www.itu.int/rec/T-REC-P.862/en`. Accessed: 2017-12-20.

[PAC06] PAControl.com. *Instrumentation & Control - Process Control Fundamentals*. PAControl, 2006.

[Per02] Krzysztof Perlicki. Simple analysis of the impact of packet loss and delay on voice transmission quality. Master's thesis, NIT, 2002.

[pos] postgresql. https://www.postgresql.org/. Accessed: 2017-09-30.

[Pri15] Penny Pritzker. Cloud computing service metrics description, 2015.

[rpr] r-project. https://www.r-project.org/. Accessed: 2017-09-30.

[rst] rstudio. https://www.rstudio.com/. Accessed: 2017-09-30.

[Sö] Martin Sölkner. *Energieeffizienter Elastizitätsmanager für Clouds.*

[SC15] Sukhpal Singh and Inderveer Chana. Qos-aware autonomic resource management in cloud computing: A systematic review. *ACM Comput. Surv.*, 48(3):42:1–42:46, December 2015.

[sql] Iso/iec 9075-1:2016. https://www.iso.org/standard/63555.html. Accessed: 2017-09-30.

[TD15] Ajay Tala and Bhautik Daxini. Smith predictive control of time-delay processes. *ETCEE–2015*, page 7, 2015.

[VCER10] Marco A. S. Netto Ivona Brandic Vincent C. Emeakaroha, Rodrigo N. Calheiros and Cesar A. F. De Rose2. Desvi: An architecture for detecting sla violations in cloud computing infrastructures. 2010.

[voi] Call quality metrics. https://www.voip-info.org/wiki/view/Call+Quality+Metrics. Accessed: 2017-12-20.

[YA15] Khaled Swesi Yousef Abuseta. Design patterns for self adaptive systems engineering. *International Journal of Software Engineering & Applications (IJSEA)*, 6(4), 2015.

[YOU07] JUSTIN YOUNEY. A COMPARISON AND EVALUATION OF COMMON PID TUNING METHODS . Master's thesis, B.S. Rochester Institute of Technology, 2007.

[ZLZ15] H. Zhang, P. Li, and Z. Zhou. Performance difference prediction in cloud services for sla-based auditing. In *2015 IEEE Symposium on Service-Oriented System Engineering*, pages 253–258, March 2015.

[ZN42] J. G. Ziegler and N. B. Nichols. Optimum Settings for Automatic Controllers. *Transactions of ASME*, 64:759–768, 1942.

[ZXJ00] Qing C Zhong, Jian Y Xie, and Qing Jia. Time delay filter-based deadbeat control of process with dead time. *Industrial & Engineering Chemistry Research*, 39(6):2024–2028, 2000.