Delft University of Technology
Master's Thesis in Embedded Systems

How can I touch you?

Setting the Yardstick: A Quantitative Metric for *Effectively* Measuring Tactile Internet

Joseph Verburg

Embedded Networked Systems

TU Delft Delft University of Technology

How can I touch you?
Setting the Yardstick: A Quantitative Metric for *Effectively* Measuring Tactile Internet

Master's Thesis in Embedded Systems

Embedded and Networked Systems Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Joseph Verburg
tudelft@josephverburg.com

August 21, 2019

Author
Joseph Verburg (tudelft@josephverburg.com)

Title
How can I touch you?

MSc presentation
28th of august 2019

Abstract

Tactile Internet (TI) is, the next frontier of connectivity development where not only information but interaction can be exchanged digitally. Much effort in the field of TI has been spent towards developing the necessary technologies needed to transmit interactions over the internet in a naturally perceived way. However not much has been done on improving the evaluation of these TI technologies, that is, Mean Opinion Score score or Peak Signal to Noise Ratio measures for audio-video applications. Existing measures and measuring methods only give coarse-grained indications of performance and lack the depth for good comparison. By using Dynamic Time Warping as starting point we propose our new measures Effective Time & Value -Offset that fundamentally changes the way to evaluate performance.

We verify our measures by showing it performs significantly better compared to existing work by using data from a new testbed designed to allow testing in stringent network conditions and to be usable for many future works.

Our contributions will push forward TI development allowing both incremental and novel developments to be tested more easily and with improved verification.

Preface

After just shy of two years of master study i will be finalizing my studies in Embedded Systems. This thesis is the last report i will be writing towards finishing my master studies.

After finishing both “Real-time Systems” and “Embedded Systems Laboratory” courses i became interested in what the then called “Embedded Software” department could have in possible Thesis projects for me. After several talks with different people of the department i found to have the best connection with my now supervisor Dr. Ir. R.V. Prasad (VP). From his topics TI really popped out, the promise of providing real remote interaction sounded as something that should be pursued. So that is what i chose for. Dtarting what i thought to be a year of doing my thesis in a relaxed manner. But already from the start VP had other plans and was able to challenge me by asking me to improve my grades to be eligible for Cum Laude. Which i did by improving four grades by on average 1.5 points, which was a very fulfilling experience in itself.

After the first month of literature research i was joined by VP’s PHD-er Ir. Kees Kroep, he initially focused on developing TI solutions. From the literature research it became clear that there was a clear lack in ways to properly measure TI. This sparked heavy brainstorm sessions between me and Kees on how to actually measure the TI solutions coined so far. We came up with a fundamental new way of looking at errors by splitting it into two orthogonal challenges. We then used this new concept to design our measures. After completing the first version of our measures we had a great lack in proper test data. This motivated the development of our testbed. While developing the testbed we used initial data and the finding of Dynamic Time Warping (DTW) to significantly improve our measures. The final result is a high quality testbed usable for our current and future work and a high quality measures usable for testing a wide variety of solutions.

A paper has been written on this work and has been submitted to INFOCOM 2020. The paper is 10 pages long and carries the title “Setting the Yardstick: A Quantitative Metric for *Effectively* Measuring Tactile Internet”.

Acknowledgments

During my thesis I received support of many amazing and kind people. I want to thank my fellow researcher and daily supervisor Ir. Kees Kroep for all the fun times we had drinking coffee, late nights (and early mornings), brainstorming and building our ideas. I want to thank Dr. Ir. Vijay Rao and Dr. Ir. Vineet Gokhale for all their feedback and fun sessions. I want to thank my supervisor Dr. Ir. R.V. Prasad for all inspiring feedback both spiritually and technically.

I want to also thank my girlfriend Jennie Christiaanse for all her support and kind and supporting words. I want to thank my mother Annemarie Kemp for calling multiple times a week to support me in my efforts.

List of Abbreviations

5G	Fifth generation of mobile networks
API	Application Program Interface
AWGN	Additive White Gaussian Noise
CC	Cross Correlation
CDF	Cumulative Distribution Function
DTW	Dynamic Time Warping
E2E	end-to-end
ETO	Effective Time-Offset
ETVO	Effective Time & Value -Offset
EVO	Effective Value-Offset
HMI	Human to Machine Interface
HPWPSNR	Haptic Perceptually Weighted Peak Signal to Noise Ratio
JND	Just Noticeable Difference
MOS	Mean Opinion Score
MRS	Mean Ranking Score
MSE	Mean Square Error
HSSIM	Haptic Structurally SIMilarity
OS	Operating System
PMSE	Perceptual Mean Square Error
PSNR	Peak Signal to Noise Ratio

QoE	Quality of Experience
QoS	Quality of Service
RMSE	Root Mean Square Error
SCTP	Stream Control Transmission Protocol
SNR	Signal to Noise Ratio
SSIM	Structurally SIMilarity
TI	Tactile Internet
TCP	Transmission Control Protocol
TSC	TimeStamp Counter register
UDP	User Datagram Protocol
URLLC	Ultra Reliable Low Latency Communication

Contents

Chapter 1

Introduction

1.1 Tactile Internet

Tactile Internet (TI) the field of techniques and systems enabling remote interaction of humans and machines. The TI goal is to be able to transport touch and interaction over long distances in a naturally perceived way. This allows interaction with people, environments and machines without the need of physical presence. This allows humans to interact in environments which was previously not possible or safe to do so. Situations like disaster relief, waste cleanup, physical labor and more can have human interaction without the risk to the human performing said interaction.

The introduction of transmitted interaction, also called human-in-the-loop communication creates new much higher demands for the communication such as sub-10 ms end-to-end (E2E) latency and reliability requirements of up to five 9's. These requirements originate from high responsiveness and sensitivity of human touch.

These requirements drive the innovation towards Ultra Reliable Low Latency Communication (URLLC). For example the limits of new techniques like the Fifth generation of mobile networks (5G) are being pushed by these requirements. This in turn allows other fields like automotive driving to benefit of the existence of URLLC infrastructure.

In parallel to the developments made in telecommunications TI also fueled advancements in sensors, displays and communication protocols. To make these advancements in TI proper evaluation is essential. So far evaluation for the most part has been done using Quality of Experience (QoE) and Quality of Service (QoS). Where QoE is very time consuming and QoS is often limited to network performance. There have been attempts to solve this by tying QoE to quantitative measures by creating so called perceived measures which we will discuss in Section 2.3.2. However these have in common that they assume unrealistically ideal circumstances like zero delay and/or loss. What we saw from all measures that the focus is always on one aspect, either value-offset or time-offset. With time-offset we mean shift in signals caused by time differences between the two signals. With value-offset we mean all differences measured in changes in signal value like error.

To improve upon this a new measure is needed that takes both time-offset and value-offset into consideration showing that these are two orthogonal peaces of information that both should considered when evaluating TI solutions. We identify DTW as a measure that handles both time-offset and value-offset and use it as starting point for developing our measures. Besides a new measure a testbed is needed where realistic situations containing both time-varying delay and loss can be simulated in which you implement TI solutions.

1.2 Research goals

The goal of our research is to forward the field of TI. To achieve this we create the following three main goals:

- A consistent method of measuring TI solutions. Current measuring of TI solutions do not employ a singular method of measuring making it difficult to develop, optimize and compare different TI solutions.
- A good quantitative measure that is employable in most situations. Current measures are only usable under certain conditions greatly limiting their usability as a good performance measure.
- A testbed with all critical components of a TI system.

We achieve these goals by introducing a better measurement method, two new measures and a new testbed to test them.

1.3 Thesis Organization

This Thesis is organized into six chapters. In Chapter 2 we delve deeper into TI and explain what the applications of TI are and the work already done in measuring TI systems. In Chapter 3 we introduce our measures and it's precursor. In Chapter 4 we introduce our testbed and it's precursor. In Chapter 5 we evaluate the performance of our testbed and use it to evaluate the performance of our measures. Finally in Chapter 6 we discuss the things learned, conclude the work we have done and future work.

Chapter 2

Related work

Since the inception of the internet we have conquered the transmission of text, audio and video. The next frontier is the transmission of touch. This is the field of TI [1]. Transmission of touch is expected to revolutionize the way we interact with each other and physical environments. Introducing the transmission of touch to the wider public comes with great challenges. It is shown by Kandel et al. that when touch is involved a human is much more sensitive, a human touch can perceive changes up to 1 kHz [2]. While our sensitivity to for example audio and video is much lower where we can tolerate delays of up to 80 ms [3]. This means that slight variations or error can have a large impact on the experience of the interaction.

Because touch is more sensitive, below 10 ms E2E delay and reliability of at least five 9's is needed [1]. Achieving these constraints is not possible using existing commodity communication protocols like TCP or UDP, where the first incurs high delays when loss is introduced and the latter has no way of recovering lost data [4, 5]. Since the inception of TI extensive effort has been spent on improving communication as can be seen by the existing works [6–8]. As we will see in the following Sections many advancements have been made, however there is still room for extensive improvements.

However, TI already has applications with positive societal impact with much more applications to come.

2.1 Tactile Internet Applications

The application of TI is tele-operated systems. A tele-operation is a session wherein a tele-operator interacts with a remote physical environment through the use of Human to Machine Interface (HMI) on the local side and physical effectors on the remote side. A good example of this is a tele-operator using a HMI and a robotic arm as effector, the HMI in this situation is both responsible for recording the position as well as giving feedback to the tele-operator. A visual representation can be seen in Figure 2.1. In this

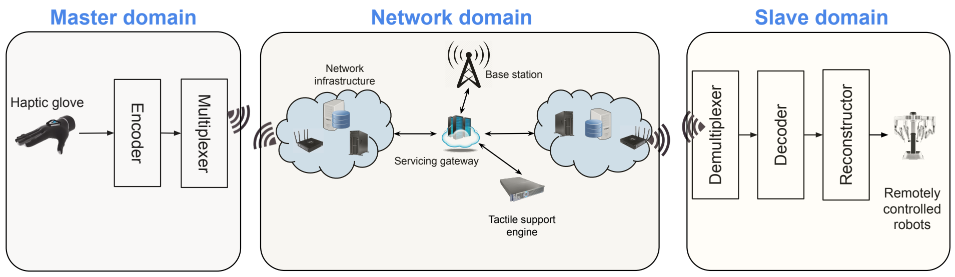

Figure 2.1: Typical domains and their components present in a TI session.

Source: Vijay Rao

work we take this example as a typical TI session.

2.1.1 Construction

In the field of construction tele-operation can be used to easily transport experienced operators to a construction site without needing the operator to physically anywhere near to the construction site. A good example can be seen of the tele-operation of an excavator digging in South Korea tele-operated from Germany [9,10]. This can be extended even further into tele-operation in construction situations where it would not be safe for a human to be, like construction of high rise buildings, underground excavation and more. An example of this can be seen in a unmanned Coal mining robot presented in Huh et al. [11].

2.1.2 Remote Surgery

A well-known example of tele-operation can be found in the medical field. The tele-operation of medical operations, especially complex surgery requiring a high degree of expertise and experience [12]. Tele-operation of surgery allows for the transport of surgeons, actions of qualified surgeons without the inherent time cost, financial cost and risk for either surgeon or patient. This enables also the transport of surgeons to area's of the world that previously would not be able to get access to surgeons, because they are too costly or unwilling to live there. A good example of where tele-surgery could be beneficial is for initiatives like Mercy Ships where only the equipment would need to be shipped and any surgeon could participate without the inherent risk of such area's [13]. Besides preventing the need to transport surgeons, tele-surgery also has the benefit of enhancing the view and interaction of the surgeon with information and processing that is normally not possible. Current solutions include multi-angle view, real-time hand tremor reduction and overlaying heat and other information on video streams [14].

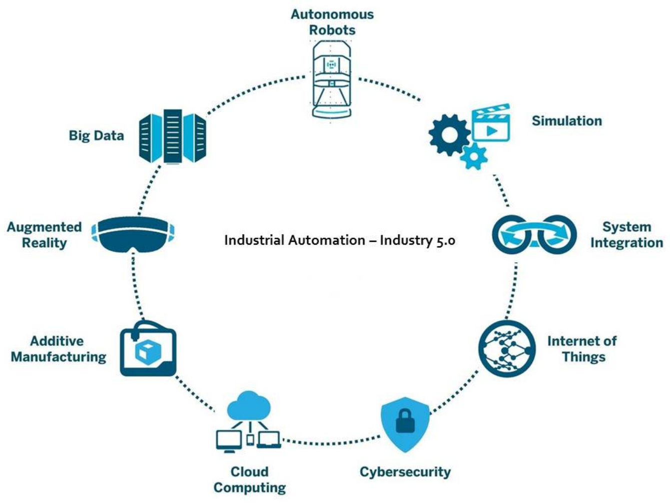

Figure 2.2: All technologies involved in Industry 5.0, most of which benefit from improved communication reliability and low latency due to TI research.

Source: Gem Newz [16]

2.1.3 Disaster Relief

Another area of great interest for tele-operation is disaster relief robots. Remote controlled robots can provide a critical role in providing disaster relief in situations when the risk has not subsided for rescue workers. Clean-up of toxic or nuclear waste where it is too hazardous for a human to come anywhere near the disaster zone. This has been extensively used in the ongoing effort in the cleanup of the Fukushima disaster [15].

While all these area's already give great examples of the usability of current TI technologies, we envision it going much further. We envision the possibility to give real touch enabled interactions over long distances. This can revolutionize human interaction any will change our daily lives.

2.2 Tactile Internet benefits in other fields

The innovations within TI have a beneficial interplay with many other fields. The promise of providing URLLC and real remote physical interaction fuels innovations in the other topics. In the following Subsections we describe the main fields that benefit from TI research.

2.2.1 Industry 5.0

Industry 5.0 has the aim is to enable more communication between devices and humans [17]. To allow a new advent of productivity and flexibility in

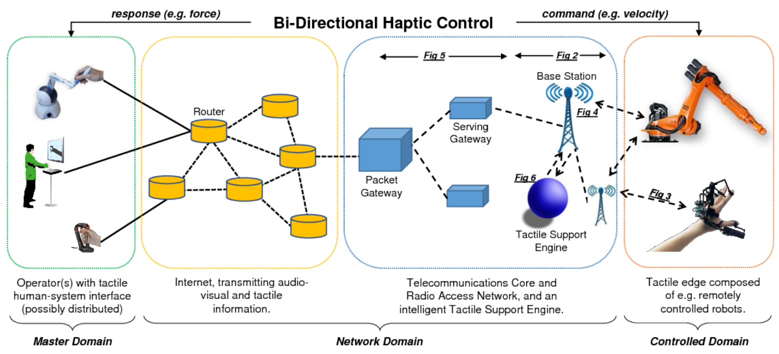
response (e.g. force)
Bi-Directional Haptic Control
command (e.g. velocity)
Router
Fig 5
Fig 2
Base Station
Fig 4
Serving Gateway
Packet Gateway
Fig 6
Fig 3
Tactile Support Engine
Operator(s) with tactile human-system interface (possibly distributed)
Internet, transmitting audio-visual and tactile information.
Telecommunications Core and Radio Access Network, and an intelligent Tactile Support Engine.
Tactile edge composed of e.g. remotely controlled robots.
Master Domain
Network Domain
Controlled Domain

Figure 2.3: A situation where 5G can be used to transmit interaction, this shows that 5G can also be used to power TI applications.

Source: 5G-Enabled Tactile Internet [18]

products and production. Where Industry 4.0 aimed to improve industry by incorporating the innovations from AI. Industry 5.0 aims to improve upon this even further by using the innovations of TI in both communication and robotics. An consumer example of Industry 5.0 can be found in the medical field where users of medical devices like insulin pumps and insulin sensors get insight into their own data via apps. An production example of Industry 5.0 can be found in the greater production performance by using real-time data from in field products and production facilities by connecting both the product and production facilities.

2.2.2 Fifth generation of mobile networks

The development of 5G is heavily influenced by developments in TI. Requiring URLLC for TI created a demand during the development of 5G to greatly improve latency and reliability guarantees within 5G with relation to the previous generation of mobile networks (4G) [19, 20]. Where 5G is now enabling 1 ms latency compared the >50 ms delay of 4G [21]. The interplay between 5G and TI works two ways. For example the creation of new TI applications inspired by 5G as shown by the developments done by Pandi et al. where they used edge computing together with 5G to enable very low latency multiplayer gaming experiences [22]. That Miao et al. demonstrate by developing a remote surgery architecture using 5G and TI technologies [23].

2.3 Tactile Internet measuring

While the field of TI is quickly developing there is so far little effort has been put into streamlining the efforts. By streamlining we mean the fact that most of the work is difficult diagnose and compare and thus to iterate on. This difficulty stems from the lack of good measures that are universally applicable. This creates a situation where most works do not share common measurement methodologies thus making comparison between the works difficult. In the following Subsections we delve into three categories of existing measures. For each measure we explain their workings and examine why they are not sufficient to measure TI systems.

2.3.1 Subjective measuring

Measuring using subjective measurements is being used in TI research to determine the QoE of different TI systems. Some examples can be found in Yuan et al. testing the QoE of a sensory system and Brandi et al. testing a network coding technique [24, 25]. Using statistical tools like Mean Opinion Score (MOS) or Mean Ranking Score (MRS) it is possible to obtain results that have some stability. However, after these improvements the measurements contain quite some noise and are very hard to reproduce, requiring significant time investment for basic verification.

Furthermore doing subjective measurements is very resource, time and potentially financially intensive. Also much care has to be taken to produce a TI session with sufficiently rotates different scenarios such that a subjects increasing experience with the system does not influence scenario results too much.

We conclude that subjective measurements are not a efficient way to develop TI systems.

2.3.2 Deadband measuring

Deadband based measuring is based on the concept that human perception is relative. Originally coined through Weber's Law by Ernst Weber in 1834, which describes that the Just Noticeable Difference (JND) of an sensory impulse is proportional to the original impulse as shown in Equation (2.1). Jones et al. later verified that indeed human perception follows Weber's Law [26].

$$\frac{\Delta I}{I} = k \Rightarrow \Delta I = kI \tag{2.1}$$

ΔI describes the JND, the smallest amount of change to an sensory impulse that can be perceived. I is the original sensory impulse. k the constant describing the relation between the JND and the original impulse. Generally k is between 5 % and 15 % depending on which part of the body the sensory impulse is received [27].

The concept of sensory deadband is used extensively in TI research to reduce the amount information that needs to be transmitted, by only transmitting when the JND threshold has been reached. As shown by the work done by Hirshe et al. where deadband is used to decrease the amount of packets needed to transmit kinaesthetic information [28]. They achieve a packet rate reduction of up to 89 % without significantly impacting the experience. In the work by Hinterseer et al. they extend this approach by representing kinaesthetic information as a vector in 3D space and applying the deadband as a spherical boundary instead of a per coordinate deadband [27].

The use of deadband however, introduces a new problem, traditional measures like Root Mean Square Error (RMSE) and Signal to Noise Ratio (SNR) greatly increase when deadband is used. The increase in RMSE due to deadband data reduction is however not in line with the actual experience of an tele-operator. To remedy this, kinaesthetic measures focused on perception were introduced.

Haptic Perceptually Weighted Peak Signal to Noise Ratio

The work by Sakr et al. introduces Haptic Perceptually Weighted Peak Signal to Noise Ratio (HPWPSNR) [29]. HPWPSNR uses the concept of JND to reduce the influence of any errors below the JND threshold. HPWPSNR is described in Equation (2.2) and (2.3).

$$\text{HPWPSNR} = 10\log_{10}\frac{||x_{max}-x_{min}||^2}{\text{MSE}\cdot\text{HPW}} \tag{2.2}$$

$$\text{HPW} = \frac{1}{N}\sum_{i=0}^{N-1}\begin{cases} C & \text{if } |x_i - y_i| \leq \text{JND}(x_i) \\ \alpha\cdot(|x_i-y_i| - \text{JND}(x_i)) + C & \text{otherwise}\end{cases} \tag{2.3}$$

x_{max} and x_{min} describe the maximum and minimum value of the kinaesthetic attribute respectively. MSE is the mean square error between the input (x) and output (y). JND(x_i) is the JND of input value i defined as JND(x) $= k \cdot x$ where k is the perception deadband, often 5 %. C is a constant that signifies errors below the perception threshold, value of 1 is given as default. α is an penalty factor for values above the JND threshold, value of 1 is given as default.

Perceptual Mean Square Error

Chaidhari et al. propose an alternative to HPWPSNR called Perceptual Mean Square Error (PMSE) [30]. PMSE is based on the hypothesized idea that the perception of the brain varies logarithmically with size of an impulse. PMSE is described in Equation (2.4).

$$\text{PMSE} = \frac{c^2}{N}\sum_{i=0}^{N-1}(\ln\frac{x(i)}{y(i)})^2 \tag{2.4}$$

c is a scaling constant typically determined experimentally. $x(i)$ and $y(i)$ are the i-th input and output sample respectively.

Haptic Structurally SIMilarity

Hassen et al. propose an improvement to the previous two works: Haptic Structurally SIMilarity (HSSIM) [31]. HSSIM is based on the Structurally SIMilarity (SSIM) index, the SSIM index an method of image quality assessment. The introduction of the SSIM index greatly improved the image quality assessment compared to Mean Square Error (MSE) [32]. The goal of HSSIM is to use the same method as SSIM and apply it to kinaesthetic signals. This is achieved by relating the luminance, contrast and structure of SSIM to force strength, distinguishable force difference and temporal neighbourhood of a signal respectively. HSSIM uses an alternative to JND to take into account perceivability of certain signals and signal changes. This alternative is based on Steven power law, which describes that human perception can be described as a power law [33].

$$x_p = k \cdot x^b \qquad , \qquad y_p = k \cdot y^b \tag{2.5}$$

x_p and y_p are the perceived input and output signal modified with Steven's power law. k is a scaling constant, default of 1.4. b the power exponent, default of 1.8 based on research in [33].

$$l(x_p, y_p) = \frac{2\mu_{x_p}\mu_{y_p} + C_1}{\mu_{x_p}^2\mu_{y_p}^2 + C_1} \tag{2.6}$$

$$c(x_p, y_p) = \frac{2\sigma_{x_p}\sigma_{y_p} + C_2}{\sigma_{x_p}^2\sigma_{y_p}^2 + C_2} \tag{2.7}$$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{2.8}$$

The above three equations describe the luminance (l), contrast (c) and structure (s) respectively. μ_{x_p}andμ_{y_p} are local means of x_p and y_p. $\sigma_{x_p}, \sigma_{y_p}, \sigma_xand\sigma_y$ are local standard deviations of their respective signals. σ_{xy} is the local correlation of x and y. C_1, C_2, C_3 are positive stabilizing constants. The local distribution is determined using a window size of 600 to 1200 samples.

$$\text{HSSIM} = \frac{1}{N}\sum_{i=0}^{N-1}(l(x_p(i), y_p(i))^{\alpha} \cdot c(x_p(i), y_p(i))^{\beta} \cdot s(x(i), y(i))^{\gamma})^r \tag{2.9}$$

α, β and γ are tuning parameters used to tune the importance of each component, default of 1, 1 and 0.5 respectively. r is the power of the Minkowski pooling, default of 8 is used.

Concluding

The above three described measures introduce good methodologies to incorporate human error perception into their measures, making them usable to determine the QoE of a TI session. However none of the three measures have considered delay present in a typical TI session. This results in that when a typical TI sessions is considered where some delay (1 ms to 10 ms) is present all three measures are expected to deteriorate while the experience is not impacted.

The work of Hassen et al. compares the performance of HSSIM, PMSE and HPWPSNR [31]. They demonstrate best performance for their measure HSSIM. Their comparison is well constructed using various statistically sound metrics. However HSSIM only works for force feedback while the deadband reduction for positional information is equally important, thus falling short of PMSE that is usable for both positional and force information. Moreover the implementation and computation complexity of HSSIM is much higher then HPWPSNR and PMSE.

But what all three measures have in common is that only value-offset is considered an delay is assumed to be not present, however, in a typical TI session delay is an important component that influences system performance. Moreover the improvements are also presented in packet rate reduction which is counter intuitive to the assumption that delay is not present.

2.3.3 Network measuring

Using network statistics is widely used method to quantitatively measure networked TI sessions. In the work by Eid et al. in developing Admux, a multiplexing scheme for combining tactile, audio and video jitter and delay was used to to show the improvements by the system [34]. In the work by Gokhale et al. in developing a kinaesthetic congestion control scheme delay was used to test the performance of the scheme [35]. In the work by Cizmeci et al. in developing an adaptive kinaesthetic encoding scheme Delay is used to show improved stability in changing network situations [36]. What all these works have in common is that network delay is used as measure while omitting the value-offset. It is assumed that this is matched perfectly, while this is unlikely to be the case.

If we just look at the network statistics besides packet delay, packet loss and packet rate are also relevant for the performance. Depending on the other protocols deployed packet loss can have detrimental impact on QoE. For example deadband protocols as introduced in Section 2.3.2, they greatly reduce packet rate but this makes any packet loss have extra impact. Moreover, any changes in packet rate may also impact both loss and delay, this should at least should be covered for a good network based measurement.

When looking outside of the scope of just the network there are also other components influencing delay present in an TI session. For some solutions it might be enough to just look at network statistic. However, this is not always the case and the other components that introduce delay must be considered.

The primary other components present in a typical TI session are listed below:

- **Hardware communication**. Communication between a device and your machine takes time. For example an Serial bus operating at 115200 bits/sec will take 3.3 ms to transmit one sample for a six degree of freedom device (assuming 64 bit floating points).
- **Filtering**. Signal filtering to reduce noise or improve stability could introduce delay. For example a filter with a 2 sample delay at 1 kHz introduces 2 ms delay.
- **Packetization**. Converting raw data into packets introduces delays. Also packetization strategies that include deadband reduction or prediction can introduce even more delays.
- **OS Scheduling**. Scheduling delays because the OS scheduler may not give your program all the time it needs to process information or may not give it time exactly when there is new data for your program to process.

All these components can introduce noticeable delay into a TI session which in current work is being neglected.

Besides delay, error also needs to be considered. Currently when delay is used error is always ignored, while the delaying or reordering by delay and/or jitter may introduce considerable signal error.

2.3.4 Concluding

What all three measuring categories have in common is that the performance is measured using a single measure. This essentially collapses the fine grained performance information into a single value. From the usage of value-offset in Section 2.3.2 and time-offset (delay) in Section 2.3.3 we can see there more then just either value-offset or time-offset. Especially because one may impact the other, effects that are currently completely ignored by existing measure options.

2.4 Tactile Internet testbed

While a good measure is important in testing TI sessions, the usage of an good testbed is equally important. In the field of TI there is only one usable

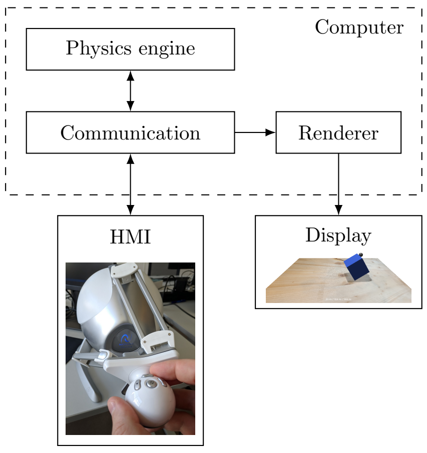

Figure 2.4: Schematic overview of the testbed presented in Bhardwaj et al. [37]

testbed for testing TI sessions. This testbed is introduced in the work by Bhardwaj et al. [37]. It uses a HMI to interact with a virtually simulated environment. A schematic overview of the testbed can be seen in Figure 2.4. In the testbed 3D positions are read from the HMI at a rate of 1 kHz and then scaled to the workspace (device radius of 4 cm is converted to 130 cm in the virtual space). The position of the spherical effector in the virtual environment is updated and another 1 ms step of the physics simulation is calculated. The forces present on the effector in the virtual environment are then scaled and transmitted back to the HMI. In parallel to these physics updates the graphics renderer uses the information from virtual environment to render an image to the display.

The testbed uses the Chai3D engine to simulate and render [38]. The advantage of using Chai3D is the native support for several fairly affordable HMIs. The HMI used is a Novint Falcon, a HMI with 3 degrees of freedom (x, y and z) and gives kinaesthetic feedback with 3 degrees of force feedback up to 8 N [39]. Using this device, the testbed is able to replicate close to real physical interactions within the virtual environment.

Chapter 3

Effective Time & Value -Offset

In this chapter we introduce our measures and their origin. To be able to properly use our measures we introduce our own measurement method. We describe measures usable within our method and examine why they are not suitable as good TI measures. Afterwards we explain our new measures that solve these shortcomings.

All measures discussed in Section 2.3 collapse the error either into the time or value domain. However, both domains need to be considered simultaneously. To do this first the time-offset has to be calculated after which the signal has to be corrected for the time-offset. After this a signal remains with only value-offsets.

3.1 Blackbox measuring

The first goals as described in Section 1.2 is the lack of a consistent measuring method within current work. To solve this we introduce the blackbox measuring method. The method is straightforward, we take the whole system

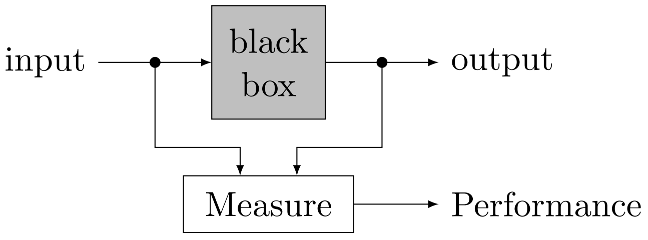

Figure 3.1: High-level overview of the blackbox methodology to illustrate that no information is needed from the tested system just input and output.

and it's properties as a blackbox, hence the name. We then only measure at the input and output and use that information to determine the performance of the system present in the blackbox. The Blackbox method is illustrated in Figure 3.1. The advantage of this approach is that no information of the system is needed to be able to determine it's performance.

3.2 Discrete signals and discretization

For the following sections it is important that the used signals are properly defined. We define two discrete signals, one ideal signal denoted g and one non ideal signal denoted f. If not otherwise specified g and f both have equal length N. For systems that assume the input is ideal, g and f can be seen as input and output respectively. However, we explicitly do not use them in this way because the input may contain noise and the output may be actually the ideal signal.

However, not all signals can be measured perfectly discrete. To solve this we discretize our timestamped signal using the below described Matlab code. Here T contains the timestamps, X contains the data values and Y is the discrete version of X. We define the timescale we want to discretize to to be the same as the timescale of the timestamps. For our purposes that is the millisecond timescale.

```
% starting time, timestamp do not have to start at zero
T_start = min(T);
% Initialize discrete signal
Y = zeroes(max(T) - T_start, 1);
idx = 1; % Starting index

for i=1:size(T)
    % Find the index of the previous non discrete sample
    % before the time point we want to calculate
    while T(idx + 1) < T_start + i
        idx = idx + 1;
    end
    % determine the velocity between previous and next
    % non discrete sample
    velocity = (X(idx + 1) - X(idx)) / (T(idx+1) - T(idx));
    % calculate discrete sample using calculated velocity
    % and elapsed time since previous non discrete sample
    Y(i) = X(idx) + velocity * (T_start + i - T(idx));
end
```

This Matlab code interpolates missing discrete values using the previous and next non discrete sample for a given discrete time point.

3.3 Fixed time-offset as static improvement

The initial solution to calculate the time-offset is to use a fixed time-offset. We implement this by calculating the Euclidean distance between the input and output signal and taking the offset that has minimal distance. Essentially finding the n such that minimum value is obtained, as described by Equation (3.1).

$$\min_{n \in \mathbb{Z}} \sum_{i=0}^{N-1} |f[i] - g[i-n]| \tag{3.1}$$

n is the amount of samples time-offset. Thus this gives us $\varepsilon_{\text{time}} = n$. Using this we can calculate the value-offset of each sample using:

$$\varepsilon_{\text{value}}[n] = |g[n] - f[n + \varepsilon_{\text{time}}]| \tag{3.2}$$

To handle the boundaries we define the following: $0 \leq n + \varepsilon_{\text{time}} \leq N - 1$. While the calculation using fixed time-offset is very straightforward it has one huge drawback, it cannot cope with any changes in time-offset.

3.4 Dynamic Time Warping as dynamic improvement

A dynamic solution for time-offset estimation is DTW [40]. Created to improve voice recognition of spoken words. The goal of DTW is to find a *warp path* such that the Euclidean distance between two signals is minimized. With *warp path* (w) we mean the path that connects the two signals with minimal Euclidean distance. The *warp path* is a list of tuples (i, j) that connects two points between the two signals. The length (K) of the warp path is bound by N and $2N - 1$. The *warp path* has one restriction to preserve time continuity, one sample of a signal may not be mapped to a point that is before mapped points of previous signal samples, thus $w(n) < w(n + 1)$ which means that every next tuple in the *warp path* must have at least one of the two values increase thus $w(n, 0) > w(n + 1, 0)$ or $w(n, 1) > w(n + 1, 1)$. For the boundary conditions the following is defined: $w(0) = (0, 0)$ and $w(N - 1) = (N - 1, N - 1)$. To describe what DTW is trying to solve we can use Equation (3.3) for the best possible warp path.

$$DTW(g, f) = \min_{w} \sum_{n=0}^{K-1} |g[w(k, 1)] - f[w(k, 2)]| \tag{3.3}$$

3.4.1 Implementation

The calculation of DTW goes in three steps.

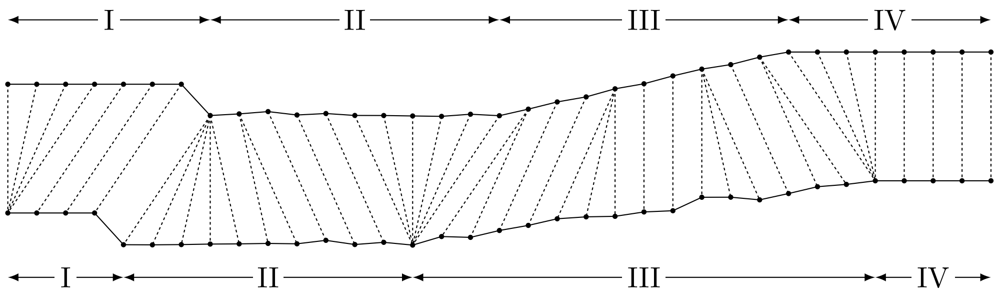

Figure 3.2: An illustrative example of how DTW behaves when aligning two signals, the two solid lines indicate two discrete time series and the dashed lines indicate how DTW aligns the samples

1. Calculate a distance matrix **A** of size NxN using

$$\mathbf{A}[i,j] = |g[i] - f[j]| \tag{3.4}$$

2. Calculate a cumulative distance matrix **C** using

$$\mathbf{C}[i,j] = \mathbf{A}[i,j] + \min \begin{cases} \mathbf{C}[i-1,j] \\ \mathbf{C}[i,j-1] \\ \mathbf{C}[i-1,j-1] \end{cases} \tag{3.5}$$

3. Backtrack through **C** to calculate the warp path using

$$\mathbf{C}[w(k-1,0),w(k-1,1)] = \min \begin{cases} \mathbf{C}[w(k,0)-1,w(k,1)] \\ \mathbf{C}[w(k,0),w(k,1)-1] \\ \mathbf{C}[w(k,0)-1,w(k,1)-1] \end{cases} \tag{3.6}$$

   This means that the time-offset increases, decreases or stay the same, which is equivalent to moving left, moving down and moving diagonally down respectively.

3.4.2 Shortcomings

While DTW offers considerable improvements compared to Cross Correlation, it does not offer a good solution to calculate time-offset. This stems from the fact that the objective of DTW is to minimize Euclidean distance and not offer time-offset information. To achieve minimal Euclidean distance it tries to resolve every error as time-offset. This means that the *warp path* that DTW produces is not accurate as time-offset indicator. We identify four shortcomings of DTW that need to be solved to make it usable as time-offset. Too illustrate these problems we use Figure 3.2.

Start and finish behavior

Due to the boundary conditions of DTW it has to start at zero time-offset and go back to zero offset at the end. This means that DTW has to assume the signals start and finish with zero time-offset, which in nearly all situations is incorrect. This creates a situation where at the start and finish DTW has to sacrifice value-offset to catch up and let go respectively of the signals behavior. This is especially bad in signals that have a large start or finish time-offset. For example, if a signal is perfectly replicated but with a large constant time-offset, DTW will incorrectly conclude large signal mismatch while even fixed time-offset will come to the correct conclusion that the signals match perfectly. This can also be seen at the start and finish of the signals in Figure 3.2, there is an start and finish time-offset of 3 samples but DTW has to first jump by 3 samples and in the end has condense back the same 3 samples.

Erratic decision making

When two low velocity signals are compared which have minute variations relative to each other DTW tends to create very erratic time-offset changes. This is due to the fact that DTW optimizes the remaining value-offset even when that remaining value-offset is infinitesimal. An example of this can be seen in Segment II of Figure 3.2 where both signals have low velocity and experience a small amount of noise. DTW decides to significantly increase and decrease time-offset just to optimize for the noise completely ignoring the broader signal.

Insensitive to shape

DTWs design is to optimize for minimal Euclidean Distance no matter the cost in the time domain. This means that the *warp path* is designed to change frequently. This is useful when comparing two different people saying the same word but with a different pitch, for DTW these two signals are very similar. However, for TI applications a different velocity of a signal might have a completely different result. For example when looking at Segment III of Figure 3.2, the top signal is moving at a higher speed than the bottom signal. However, for DTW the signals have zero value-offset because every signal sample can be matched by adjusting the time-offset.

Counter intuitive stalemate resolution

When there are multiple *warp path*'s that yield the same Euclidian distance DTW makes a decision that is consistently counter intuitive. For example looking at the transition of Segment III into Segment IV of Figure 3.2 DTW reduces the time-offset to zero to align the signals for the change that has yet to come. This means that DTW pre-aligns the signals to accommodate for changes in the future, which is counter intuitive. A more logical choice would be to wait until the time-offset would actually need to be changed, the moment the change happens not earlier.

3.5 Defining Effective Time-Offset

In the previous Section we discussed DTW and it's shortcomings when using it as time-offset, in this Section we are gonna show how DTW can be used create a measure that is usable as time-offset by solving these shortcomings. Thus we use DTW as starting point to introduce Effective Time-Offset (ETO) and Effective Value-Offset (EVO). ETO is defined as the estimated time-offset between a signal (f) and the ideal version of that signal (g), this means that $f[n]$ is mapped onto $g[n - ETO[n]]$. This is our replacement of DTWs *warp path*. EVO is defined as the value-offset between the time-offset corrected version of f and the ideal signal (g). In the following Subsections we explain how we solve the problems that exist in DTW to create our two new measures for the TI.

3.5.1 Removing boundary conditions

As discussed in Section 3.4.2 DTW creates unneeded error during the begin and end of a signal. The root cause of this is because DTW needs to compare two signals of equal length and give a difference score. To resolve this we define the maximum allowed time-offset for ETO as M samples. With this change in definition we also introduce four important changes:

1. Change the length of the ideal signal g to be $N + M$ instead of N. This is needed to deal with possible starting time-offset.

2. Change the size of the cumulative distance matrix $\mathbf{C}$ to MxN, this implements the limit of M time-offset. This changes the cumulative distance matrix to:

$$\mathbf{C}[i,j] = \Lambda(i,j) + \min \begin{cases} \mathbf{C}[i-1,j] \\ \mathbf{C}[i,j-1] \\ \mathbf{C}[i-1,j-1] \end{cases} \tag{3.7}$$

$$\Lambda(i,j) = |f[i] - g[i - j + M]| \tag{3.8}$$

   $C[i,j]$ was originally defined as the distance between $g[i]$ and $f[j]$ this now changes to the distance between $f[i]$ and g with a time-offset of j.

3. Set the first column of the cumulative distance matrix ($\mathbf{C}$) to 0 ensuring that the correct initial delay is chosen during the backtracking step of the algorithm.
$$\mathbf{C}(0,*) = [0]^M$$

4. Set the last value of ETO to be the index of the cheapest value of the last column of $\mathbf{C}$.
$$ETO[N-1] = i \in [0,N) | \min\{\mathbf{C}(N-1,i)\}$$

These changes ensure that at the start and end of f the time-offset of 0 no longer needs to be enforced and thus the correct time-offset can be chosen. This has the added benefit that it limits the time-offset search space to M samples instead of DTWs N samples, thus not exponentially growing the search space with signal length. For those with previous experience with DTW, this is a similar complexity improvement as introduce by Sakoe et al. [41].

3.5.2 Reduce erratic decision making

As discussed in Section 3.4.2 DTW has very erratic time-offset changes when dealing with low velocity sections of non ideal signals. To combat this we introduce P_{step} penalty proportional to the amount by which the time-offset changes. This means that the cumulative distance matrix calculation changes to the following:

$$\mathbf{C}[i,j] = \Lambda(i,j) + \min \begin{cases} \mathbf{C}[i-1,j] \\ \mathbf{C}[i,j-1] + P_{step} \\ \mathbf{C}[i-1,j-1] + P_{step} \end{cases} \tag{3.9}$$

P_{step} suppresses the magnitude of adjustment to the time-offset. This also makes ETO noise resilient.

How P_{step} should be chosen depends on the desired behavior, however we found that setting P_{step} to 5 % of the maximum signal value produces a stable time-offset result. 5 % was chosen based on the concept that most deadband implementations found that 5 % is a good trade off between reduction and still reasonable performance, this seems to hold as well for our algorithm.

3.5.3 Creating context awareness

As discussed in Section 3.4.2 DTW has low awareness for shape because it matches on value and thus can warp a signal significantly to match it. The previously introduced P_{step} does not address this because for large changes this is a penalty that does not influence the decision and has to be payed anyway. Thus still resulting in many time-offset changes. To resolve this we introduce P_{change}, a penalty that has to be payed when a change occurs, thus every time $ETO[n]$ changes. While it is an similar change as P_{step}, the implementation much more complex. To implement this we have to define what a change is. We define a change by splitting changes into an increase or decrease of time-offset.

- An *increase* in time-offset is gradual, time-offset cannot progress more then time can. This means that the largest increase of time-offset can be a diagonal line. Thus we define an increasing change as one or more directly following diagonal steps. We can calculate this using Equation (3.10).

$$\begin{aligned} C_{\nearrow}[i,j] = \min_{k \in \mathbb{N}^+} \Big\{ & \mathbf{C}[i-k, j-k] + P_{step} + P_{change} + \\ & \sum_{l=1}^{k-1} \Lambda(i-l, j+l) + P_{step} \Big\} \end{aligned} \tag{3.10}$$

  k is the amount of diagonal upward steps made in this single change. All the $k-1$ steps in between have to be payed via the difference and step penalty but only one change penalty needs to be payed.

- An *decrease* in time-offset can happen instantaneously, suddenly new information can be present decreasing time-offset instantaneously. Thus we define an decreasing change as one or more vertical downward steps. We can calculate this using Equation (3.11).

$$\begin{aligned} C_{\downarrow}[i,j] = \min_{k \in \mathbb{N}^+} \Big\{ & \mathbf{C}[i, j+k] + P_{step} + P_{change} + \\ & \sum_{l=1}^{k-1} \Lambda(i, j-l) + P_{step} \Big\} \end{aligned} \tag{3.11}$$

  k is the amount of vertical downward steps made in this single change. Again all the intermediary steps have to be counted.

- For uniformity we also change the way we write forward changes into Equation (3.12).

$$C_{\rightarrow}[i,j] = \mathbf{C}[i-1, j] \tag{3.12}$$

Combining Equation (3.10) to (3.12) into the cumulative distance matrix ($\mathbf{C}$) gives us Equation (3.13).

$$\mathbf{C}[i,j] = \Lambda(i,j) + \min \big\{ C_{\rightarrow}[i,j], C_{\downarrow}[i,j], C_{\nearrow}[i,j] \big\}. \tag{3.13}$$

This successfully introduces P_{change} into the algorithm, giving us control over how many adjustments are made to the time-offset instead of many frequent changes like DTW. How P_{change} should be chosen depends on the desired behavior, however we found that setting P_{change} to $2P_{step}$ has the most realistic effect.

3.5.4 Improved stalemate resolution

As discussed in Section 3.4.2 DTW tends to anticipate on future changes and prematurely changes the time-offset to accommodate them. This way of handling future changes is not realistic, instead changes should be handled when they happen and not earlier. The previously introduced penalties address the magnitude and frequency of adjustments to the time-offset, but not the timing of when these changes occur. To address this a final penalty is added P_{lazy}. This penalty introduces laziness into our algorithm, postponing a change until it is more costly to wait then P_{lazy}. The penalty is introduced differently compared to P_{change} in that it is only included in the final value. This means that P_{lazy} does not influence the conditions on which the cumulative distribution matrix are determined only the resulting value. This results in the cumulative distribution matrix ($\mathbf{C}$) being calculated as shown in Equation (3.14).

$$\mathbf{C}[i,j] = \Lambda(i,j) + \begin{cases} C_{\rightarrow}[i,j] & \text{if } C_{\rightarrow}[i,j] < \min\Big\{C_{\downarrow}[i,j], C_{\nearrow}[i,j]\Big\}, \\ C_{\downarrow}[i,j] + P_{lazy} & \text{if } C_{\downarrow}[i,j] < \min\Big\{C_{\rightarrow}[i,j], C_{\nearrow}[i,j]\Big\}, \\ C_{\nearrow}[i,j] + P_{lazy} & \text{otherwise} \end{cases} \tag{3.14}$$

This change introduces laziness which makes the timing of the algorithm more intuitive. How P_{lazy} should be chosen depends on the desired behavior, however we found that setting $P_{lazy} = 0.5 \cdot P_{change}$ strikes a good balance between being too lazy and skipping over changes or making changes too early.

3.5.5 Concluding

The introduction of the boundary conditions and the three penalties creates a measure that produces much more intuitive time-offset and value-offset compared to DTW. These changes do increase the value-offset, however that is for the better because the value-offset is now realistic instead of the over optimized value-offsets of DTW.

3.6 Defining Effective Value-Offset

In the implementation of DTW the value-offset is added together to produce a single measure of the distance between the two signals. For EVO we are interested in value-offset as an time series, this allows to be able to identify value-offset introducing portions of the signals. Moreover, it also allows other further measures to be applied on top on the value-offset information. Calculation of EVO is non trivial because when the time-offset changes a sample from the ideal or non ideal signal may be used to calculate multiple value-offsets. This results in two problematic cases:

1. When time-offset *increases* multiple samples of f should be calculated against a single sample of g.
2. When time-offset *decreases* multiple samples of g should be calculated against a single sample of f.

However this first case is not relevant for the calculation as we calculate EVO from the perspective of the non ideal signal (f). To cover this second case we need to calculate all intermediary errors and add them together. This results in the formula for EVO being Equation (3.15). The first case is covered using the fact that $ETO[i]$ will increase and thus every sample of f is correctly calculated against the same value of g. The second case is covered by summing all the intermediary value-offsets together onto the sample before the time-offset decrease.

$$\mathrm{EVO}[i] = \begin{cases} \sum_{j=\mathrm{ETO}[i+1]}^{\mathrm{ETO}[i]} \Lambda(i,j) & \text{if } \mathrm{ETO}[i] > \mathrm{ETO}[i+1], \\ \Lambda(i, \mathrm{ETO}[i]) & \text{otherwise.} \end{cases} \tag{3.15}$$

3.7 Implementation of Effective Time-Offset

Using the mathematical description as explained in Section 3.5 we propose an algorithm for calculating ETO. The implementation of ETO is more complex due to the addition of P_{change}, increasing the amount of considered options from 3 to M for every cell of the cumulative distance matrix. Additionally due to the fact multiple steps are considered when calculating a cell of **C** a simple backtracking algorithm won't work anymore because **C** does not contain the amount and direction of steps that need to be taken.

To solve both these problems we introduce an alternative to the cumulative distance matrix, the direction matrix **D**. Every cell of **D** contains the amount of steps either up or down that the back tracker needs to take to reach the next sample. This gives use the following three options:

- **Downward steps**. This is a negative value indicating the amount of diagonal downward steps that need to be taken. This is the inverse of the original cumulative distance matrix because we are storing values for the back tracker so they need to direct a path backwards not forwards.
- **Upward steps**. This is positive value indicating the amount of vertical upward steps that need to be taken. This is again the inverse, because we are tracking backwards.
- **Horizontal steps**. This is indicated as the value 0.

While the introduction of **D** solves these problems we cannot completely remove **C** because we still need to keep track of the cumulative distance.

However we can reduce this to just $2M + 1$ values since D contains the information to do the back tracking and thus only the cumulative distance of the previous values are needed. $2M + 1$ because M values of the previous column, M values for the diagonal value tracking and a value for the vertical change. This reduces the memory cost of the cumulative values from dynamic signal dependant to fixed memory cost dependant only on M. The previous column values are stored in $C_{\rightarrow}$, these are needed to store the necessary information needed for the first option in Equation (3.14). Diagonal values in $C_{\nearrow}$ and indices in $idx_{\nearrow}$ are needed to store the starting cost and index respectively of a diagonal change as seen in Equation (3.10). Vertical value in $C_{\downarrow}$ and index in $idx_{\downarrow}$ are needed to store the starting cost and index of a vertical change as seen in Equation (3.11). Our proposed algorithm of populating $\mathbf{D}$ is visualized in Figure 3.3. Finally using $\mathbf{D}$ back tracking becomes fairly trivial, our implementation is visualized in Figure 3.4. Using the resulting ETO, EVO can be calculated using Equation (3.15).

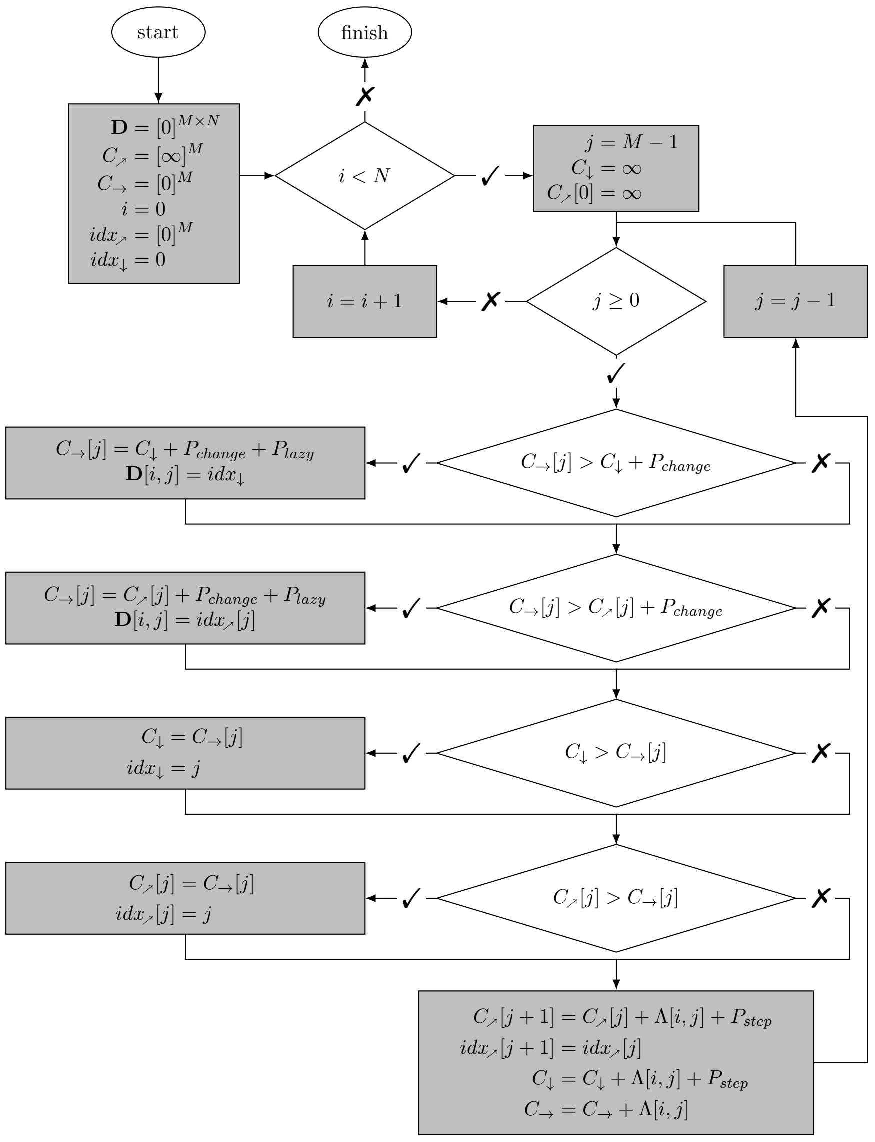
start
finish
$\mathbf{D} = [0]^{M\times N}$
$C_{\nearrow} = [\infty]^M$
$C_{\rightarrow} = [0]^M$
$i = 0$
$idx_{\nearrow} = [0]^M$
$idx_{\downarrow} = 0$
$i < N$
✓
✗
$j = M - 1$
$C_{\downarrow} = \infty$
$C_{\nearrow}[0] = \infty$
$i = i + 1$
$j \geq 0$
$j = j - 1$
$C_{\rightarrow}[j] = C_{\downarrow} + P_{change} + P_{lazy}$
$\mathbf{D}[i,j] = idx_{\downarrow}$
$C_{\rightarrow}[j] > C_{\downarrow} + P_{change}$
$C_{\rightarrow}[j] = C_{\nearrow}[j] + P_{change} + P_{lazy}$
$\mathbf{D}[i,j] = idx_{\nearrow}[j]$
$C_{\rightarrow}[j] > C_{\nearrow}[j] + P_{change}$
$C_{\downarrow} = C_{\rightarrow}[j]$
$idx_{\downarrow} = j$
$C_{\downarrow} > C_{\rightarrow}[j]$
$C_{\nearrow}[j] = C_{\rightarrow}[j]$
$idx_{\nearrow}[j] = j$
$C_{\nearrow}[j] > C_{\rightarrow}[j]$
$C_{\nearrow}[j+1] = C_{\nearrow}[j] + \Lambda[i,j] + P_{step}$
$idx_{\nearrow}[j+1] = idx_{\nearrow}[j]$
$C_{\downarrow} = C_{\downarrow} + \Lambda[i,j] + P_{step}$
$C_{\rightarrow} = C_{\rightarrow} + \Lambda[i,j]$

Figure 3.3: Flowchart of the algorithm that finds the optimal way of traversing the time offset, given the constraints specified for ETO.

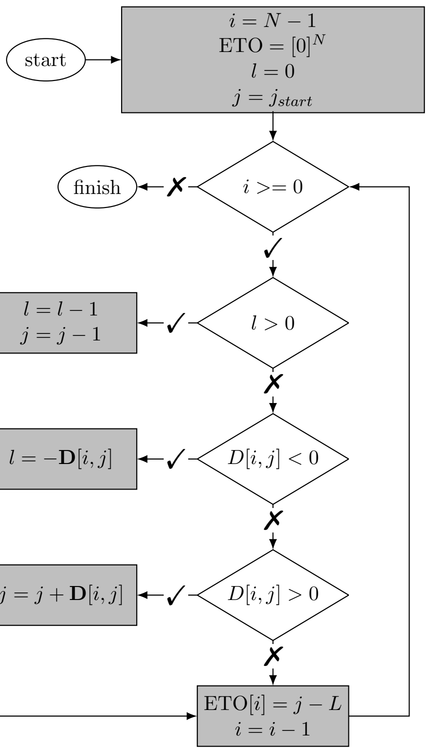

Figure 3.4: Flowchart of the backtracking algorithm used to extract the ETO from direction matrix **D**.

Chapter 4

Tactile Internet Testbed

To be able to test, verify and compare TI solutions a good testbed is needed that produces typical TI sessions. In this Chapter we first elaborate on the requirements for a good TI testbed. Afterwards we apply these requirements to a TI testbed presented in Section 2.4 and analyze how well it meets the requirements. Using the requirements and lessons learned from the existing testbed we design an improved architecture. We explain our implementation of this architecture including code details. Finally, we demonstrate how protocols can be implemented.

4.1 Requirements

To produce a typical TI session (as described in Section 2.1) several components are needed. A testbed needs to be able to support multiple protocols without impacting the whole testbed. To clearly differentiate between the different required components we distinguish the following four main testbed components.

1. **Human to Machine Interface (HMI)**. Producing position data is done using a HMI because this matches a real world TI session. The position data produced when a human is operating a HMI has interplay with the feedback received from the HMI. This is hard to replicate without a HMI.

2. **Networking**. To be able to test real world scenario's, networking needs to be part of the testbed. Moreover this allows the use of a variety of existing network simulators without needing to modify the testbed when changing the simulation.

3. **Feedback enabled interaction**. Being able to physically interact remotely is an component of TI. Introducing the feedback is essential to having actual interaction in a TI session. Moreover it has been proven that adding feedback improves the operator performance [42].

4. **Modulair**. The testbed must be modulair in such a way that different protocols can be tested without affecting the behavior of the rest of the testbed. Moreover, this allows others to easily implement protocols without needing to understand the whole testbed codebase.

4.2 Standard testbed

The testbed from existing work as described in Section 2.4 uses a program to simulate a physical environment. The testbed is able to produce a very convincing experience providing a good TI session. This is thanks to the use of the Chai3D engine, an engine that is capable of driving a physics interaction at a high refresh rate. However, the testbed does not contain networking, so this missing component needs to be added for it to qualify our requirements. Furthermore after code inspection, the current implementation is not modular. It is possible to implement other protocols but there is no Application Program Interface (API) or code present to do this in a consistent manner.

4.3 Networked testbed architecture

The design of our testbed is inspired by the Standard testbed. From this testbed we adapted the same concept of a virtual environment with a renderer and a physics engine, where the latter provides a close-to-reality solution in simulating a realistic experience.

To add the second testbed component, the positional information of the HMI has to to travel over a network connection before being processed in a virtual environment. Similar for the feedback information it first needs to travel over a network connection before being transmitted to the HMI for display to the tele-operator. To achieve this we introduce a second machine which runs a second virtual environment responsible for the physics engine. We remove the physics engine in the original virtual environment, making it purely responsible for containing the information the renderer needs to render to the display. The communication between the two sides works by communicating the positional information from the HMI to the second machine. The second machine then computes the feedback and virtual information and communicates this back to the original computer which uses this information to update the feedback to the operator and to update virtual environment used for rendering. By rendering the virtual environment this way also has an added benefit of not needing to transmit any video. For clarity we label the original computer "Master" en the new computer "Slave".

To fulfill the fourth testbed requirement the communication is split up into two sides with each 3 parts. A Packetizer and Depacketizer responsible for

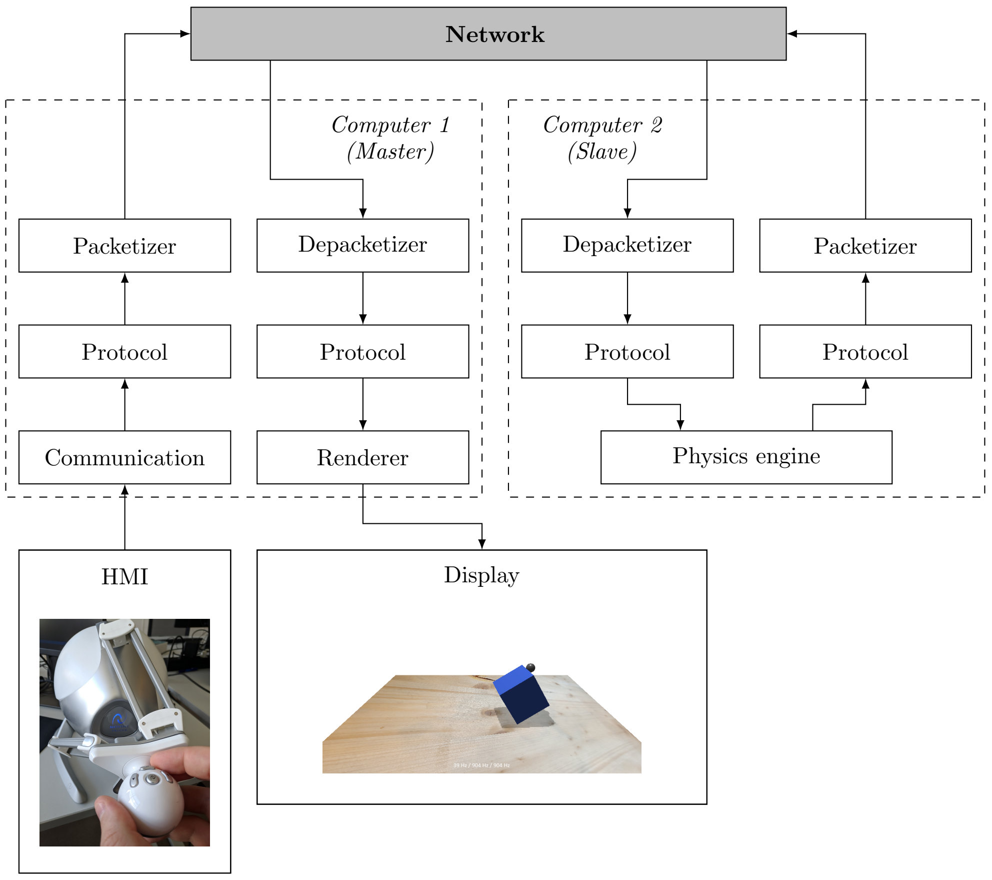

Figure 4.1: Schematic overview of our testbed architecture.

writing to and reading from the network respectively. A Protocol component containing the logic of the currently tested protocol.

All these changes result in a new architecture visualized in Figure 4.1.

4.4 Testbed implementation

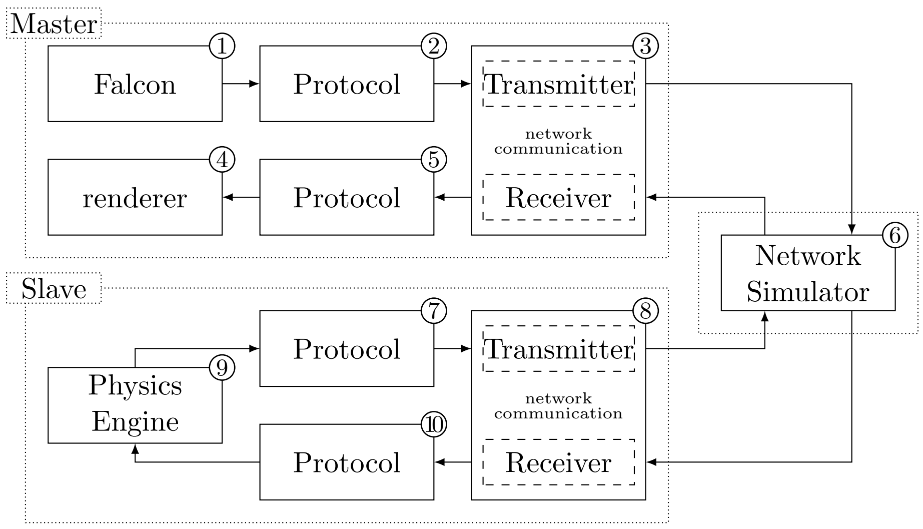

Figure 4.2: Overview of the components in our testbed.

To implement this new architecture we first create an high level implementation overview shown in Figure 4.2. Our implementation contains six components.

- **Physics Engine**. The physics engine handles converting position information into virtual interactions resulting in feedback and virtual object information.
- **Renderer**. The renderer is responsible for rendering virtual object information to a display.
- **Network communication**. The network communication handles reading and writing from the underlying Operating System (OS) network sockets.
- **Protocol**. The protocol reads data from and writes data to the HMI and network communication component. Depending on the employed protocol some read data may be ignored or changed. This is the only component that changes when a different protocol is tested.
- **Application Main**. The application main instantiates all components and ties them together.

- **Logging**. The logging is responsible of writing the logged data to file.

The testbed is implemented using 3 machines as shown by the dotted boxes in Figure 4.2. Two Windows machines running the Master and Slave side. One Linux Machine running the Netem Network Simulator, block 6 in Figure 4.2.

In the following subsections we go into depth on how all the components are implemented and what their APIs look like.

4.4.1 Physics engine

For the implementation of the physics engine we use the same engine as in the standard testbed: Chai3D. There were several reasons for doings this, we list them below.

- **HMI support**. Chai3D supports various HMIs that are fairly affordable and that offer a decent feature set. Currently 8 device types are supported and support for more devices can easily be added using the available API of the engine [38].

- **Fast physics**. The physics engine of Chai3D is optimized to provide feedback for HMIs, which require much higher update frequencies then normal for physics engines (1 kHz versus 20 Hz to 60 Hz). The cause for this is that most physics engines are built for the use in games where an low update rate does not impact the experience.

- **Developed for kinaesthetic applications**. Chai3D is developed with kinaesthetic feedback in mind which reflects in various parts playing nice even when updating at high frequencies. For example updating the position of objects at a high rate does not trigger complex render logic, this is only done when a new graphics frame needs to be rendered.

To simulate the interaction of the HMI with the virtual objects the position of the HMI is simulated as a small sphere hereafter called effector that interacts with the virtual environment, as shown in Figure 4.3.

We implemented the physics simulation using an update loop that is run at 1 kHz. Every step of the simulation we calculate 1 ms of physics interactions. We then record the force present on the effector and the positions of the virtual objects. The recorded information is then returned to the Protocol as `ToolFeedbackInformation` object. This object contains the following information:

- **numDofs**. The number of `double` values present in the **dofs** array.

- **numFeedbacks**. The number of `double` values present in the **feedbacks** array.

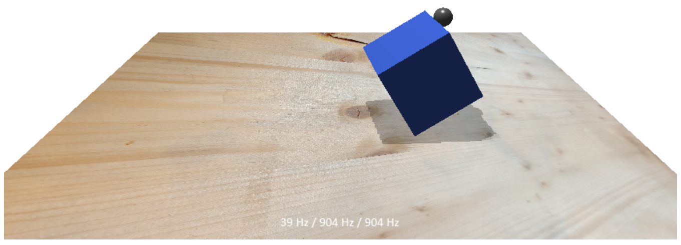

Figure 4.3: Snapshot of the rendered virtual environment, showing the effector as a dark gray sphere.

- **originalTimestamp**. The timestamp (µs) of when the **dofs** were recorded at the Master side. Stored as an unsigned 64 bit integer.
- **feedbackTimestamp**. The timestamp (µs) of when the **feedbacks** were recorded at the Slave side. Stored as an unsigned 64 bit integer.
- **dofs**. The degrees of freedom recorded at the Master side used in the calculation of this feedback. Stored as an array of `double` values.
  Position 0 to 2 contain x, y and z.
- **feedbacks**. The information used to represent the feedback. Stored as an array of `double` values.
  Position 0 to 2 contain x, y and z feedback force.
  Position 3 to 5 contain x, y and z feedback torque.
  Position 6 to 17 contains the positions of the virtual objects.

The API of the Physics engine is as follows:

- `ToolFeedbackInformation updateTool(ToolInformation * info);`
  This function takes as input a pointer to `ToolInformation` which is used to update the position of the effector. The function returns a `ToolFeedbackInformation` object calculated using the physics engine.

This API is used by the Protocol to execute a physics simulation step. In every update loop iteration the Protocol executes this function using the `ToolInformation` it wants to apply at that step.

4.4.2 Renderer

The Renderer also uses the Chai3D engine to run the virtual environment on the Master side. The virtual environment is updated by the Protocol when it has feedback information it wants to apply. When the Protocol applies `ToolFeedbackInformation` the Renderer forwards the force feedback to the

HMI and updates the position of all objects in the virtual environment. This is done by the Renderer because the driver of the HMI is managed by the Chai3D engine.

The actual rendering of the virtual environment is done in parallel in a separate thread that updates the visual image at 60 Hz, the refresh frequency of the used display.

Because the Chai3D engine manages the underlying HMI driver the Renderer is also responsible for exposing the readout function of the HMI. It presents these readout's using a `ToolInformation` object. This object contains the following information:

- **numDofs**. The number of `double` values present in the **dofs** array.
- **timestamp**. The timestamp (µs) of when the **dofs** were recorded. Stored as an unsigned 64 bit integer.
- **dofs**. The degrees of freedom read from the HMI. Stored as an array of `double` values.
  Position 0 to 2 contain x, y and z.

The API of the Renderer is as follows:

- `ToolInformation getInformation();`
  This function exposes the readout function of the underlying HMI driver. The function executes a new readout of the device and returns the readout as an `ToolInformation` object. When a new readout is stored into `ToolInformation` object also a timestamp from the globalClock is stored into the same object.
- `void applyFeedback(ToolFeedbackInformation * feedback);`
  This function uses the given feedback to update the force feedback that should be given by the HMI using the underlying HMI driver. It also updates the position of the objects in the virtual environment.

4.4.3 Network communication

To handle transmission of data from and to the network we introduce the network communication component. This is implemented by using two `Packet` object queues and two threads. The two queues are responsible for storing `Packet` objects ready to be transmitted and `Packet` objects that have been received. The two threads are responsible for reading from the socket and putting the raw data into `Packet` objects and storing them in the receive queue and for writing the raw data from `Packet` objects into the Socket. The reason for using these two queues and threads is to decouple the network IO from the rest of the application and thus removing and blocking delay that the network IO may introduce. This blocking delay exist because whenever we call the send function and the send buffer is full we have to wait until there is space for our packet in the send buffer.

For the underlying socket protocol we considered several options, we examine them below.

- **Transmission Control Protocol (TCP)**. The most used protocol for internet based communication, offering reliability and ordering guarantees. The downside of how TCP implements the reliability is that when packets are lost they have to be re-transmitted and the whole communication has to wait because of the guaranteed ordering. Furthermore, TCP is a streaming protocol meaning that data is handled as a stream and the application has no control over how data is divided into packets, this has the downside that packet can be divided illogically increasing delay even further because all parts are needed before the packet can be decoded.
- **Stream Control Transmission Protocol (SCTP)**. A protocol that offers the same reliability guarantees as TCP. SCTP differs from TCP in that is not stream based but message based. However, it still has delay downsides because of the same reliability guarantees as TCP. Furthermore, SCTP is not natively supported on most OSs and often implemented on top of TCP which readds all the problems TCP has.
- **User Datagram Protocol (UDP)**. A protocol that transmits packets without making control decisions. This gives the application full control on how reliability and related delay is managed. It has the downside that protocol offers no guarantees and thus the implementation of such features need to be handled by the application.

We chose to use UDP as it does not introduce extra delay and offers full flexibility to the protocol component to implement reliability and other features.

The `Packet` object used contains the following information:

- **type**. A one byte unsigned integer value from the first byte of the raw packet denoting the type of the packet.
- **length**. A two byte unsigned integer containing the total amount of bytes stored in the **data** array, which contains the original packet data.
- **data**. Raw data stored as an array of unsigned integer one byte values.

The API of the network communication component is as follows:

- `void send(Packet);`
  This function is responsible for queuing the `Packet` object into the internal transmission queue.

- `ConcurrentQueue<Packet> receivedPackets;` This is the queue where received packets are stored.

4.4.4 Protocol

The Protocol exists in two parts, a Master and a Slave side. On both sides the Protocol is responsible (de)packetization and decision making. Whenever new data is available from either the network or the local environment it decides based on it's programmed logic if it should be processed or not. For example because we use the UDP protocol packets may arrive out of order and the protocol drops any packets that arrive after newer packets have already arrived. This how our current protocols deal with out of order information however in the future there may be other protocols that decide to use the old information somehow and is thus the reason that this is the responsibility of the Protocol. Each side has it's own API and logic which we describe below.

Master side

On the Master side the Protocol has two responsibilities. Firstly it has to process `Packet` objects and convert feedback packets into `ToolFeedbackInformation` and call the `applyFeedback` function of the Renderer. Secondly it is responsible for processing information from HMI and encode it into a `Packet` object and place it in the send queue of the Communication component.

The Protocol has an API that is designed in such a way that allows a wide variety of protocols to be implemented. The API of the Master side is as follows:

- `void onPacketReceive(Packet);`
  This function is responsible converting `Packet` objects and deciding if they need to be applied.

- `void onToolInformation(ToolInformation);`
  This function is responsible for processing HMI information and converting them into `Packet` objects and placing them in the send queue of the Communication component.

Slave side

On the Slave side the Protocol has two responsibilities. Firstly it has to process `Packet` objects and convert information packets into `ToolInformation` and store it temporarily in it's local state. Secondly it is responsible for applying the most recent `ToolInformation` to the Physics engine using the `updateTool` function. The resulting feedback needs to be converted into a `Packet` object and placed in the send queue of the Communication component.

The API of the Slave side is as follows:

- `void onPacketReceive(Packet);`
  This function is responsible converting `Packet` objects and deciding if they need to be stored.

- `void onUpdate();`
  This function is responsible for applying the most recent information and converting the resulting feedback into `Packet` objects and placing them in the send queue of the Communication component.

4.4.5 Logging

To be able to store and later analyze testbed input and output good logging facilities are necessary. To facilitate this a Logging component is implemented. To prevent blocking delay due to disk IO we implement the logging facilities using a `ConcurrentQueue` and a writer thread. This works by placing to be logged information in the queue after which the writer thread every second writes the queue contents to disk. Avoiding blocking disk IO is essential because these delays can easily increase to multiple milliseconds for traditional hard-drives. Moreover, it is efficient to write to disk using large chunks.

The logging of both the input and output samples happen on the Master side of the testbed, this is to follow the blackbox method which is previously explained in Section 3.1. In our situation the whole testbed is the blackbox, the Falcon the input and the display the output.

The input and output are logged to separate files. We do this to avoid needing to link the two asynchronous sample streams at run time and also because input and output samples may not be linkable at all. Input samples are logged as $timestamp; x; y; z;$ where output samples are logged as $timestamp; x; y; z; source\_timestamp$, here $source\_timestamp$ is the input timestamp as logged in the input file.

The API of the Logging component is as follows:

- `void log(string);` This function puts the value into the queue after which it is written to disk by the writer thread. The writer thread automatically adds a newline character after every entry, this is intuitive to how conventionally logging is implemented.

4.4.6 Application Main

The application main is responsible for initializing and connecting all the components.

On both sides on startup a `globalClock` is initialized, this clock used throughout the application as high precision clock for both timing and recording timestamp. To create the timestamps needed for this clock we need a timestamp source that is high enough precision and quick to query. We chose the Time Stamp Counter register (TSC) of the processor (exposed by the *QueryPerformanceCounter* api), which gives us the accuracy and querying speed we need (1 µs precision and below 1 µs query time, TSC provides 333 ns precision and 33 ns query time) [43].

On both sides the main creates a thread to process Packet objects from the receive queue of the Communication component. Using a infinite loop and the `wait_and_pop` function of our `ConcurrentQueue` implementation we remove Packet objects from the receive queue with which we then call the `onPacketReceive` function of the Protocol.

One more thread is used on the Master side to run a loop used to call the `getInformation` function of the Renderer and call the `onToolInformation` function of the Protocol with the acquired `ToolInformation`. This loop is run at 1 kHz using our own loop delay logic. This loop delay works by keeping track of the elapsed time during every loop iteration and sleeping for the amount of time there is left in the 1 ms frame. If by any chance a certain loop iteration overruns the 1 ms the loop catches up by skipping sleeps until it's caught up. This combined creates a stable 1 kHz loop.

One more thread is used on the Slave side using the same loop logic but this time to run the `onUpdate` function of the Protocol at 1 kHz.

4.5 Network simulation

As mentioned before we simulate the network by routing the traffic through a linux machine running Netem as network simulator [44]. The used Linux variant is Ubuntu Server on version 18.04 [45]. We simulate various network situations by introducing delay, jitter and/or loss.

The three machines are connected together using a switch. The Master computer is assigned the ip *192.168.1.12*. The Slave is assigned the ip *192.168.1.11*. The network simulator is assigned the ip *192.168.1.10*.

4.5.1 Installation

To install Netem we need to use the following steps:

- Install the iproute package using the following command:

```
sudo apt-get install iproute
```

- Enable ipv4 forwarding in the linux kernel by editing */etc/sysctl.conf* and uncommenting (remove starting #) the following line:

```
net.ipv4.ip_forward=1
```

Afterwards reboot the system to apply the change.

To setup the routing in the Master and Slave we needed to create a manual route in the Windows routing table. We did this by executing the following commands in an elevated Command Prompt. For the Master computer:

```
route add 192.168.1.11 mask 255.255.255.255 192.168.1.10
```

For the Slave:

```
route add 192.168.1.12 mask 255.255.255.255 192.168.1.10
```

4.5.2 Delay and Jitter

Using Netem we introduce delay with jitter. We specifically chose to use highly correlated jitter to introduce slowly changing fluctuations to resemble long term delay changes also present on normal networks. The following command is used on the network simulator machine to configure the delay:

```
    sudo tc qdisc add dev eno1 root netem delay 30ms 15ms 90%
```

This command introduces 30 ms delay with 15 ms jitter with a correlation factor of 90 %. As mentioned by the Netem documentation the correlation factor is not exactly correlation in the statistical sense, however for our needs it works well. There are more realistic models to simulate the network delay, however they actually produce an easier situation for our measures.

4.5.3 Bursty loss

Using the Gilbert-Elliot model we simulate bursty loss [46].

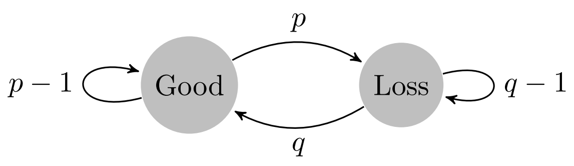

Figure 4.4: Gilbert-Elliot model used to model bursty packet loss.

The Gilbert-Elliot model uses a two state markov chain to simulate bursty packet loss also present in normal networks. The markov chain contains a "Good" and "Bad" state. When in the "Good" state no packets are dropped. When in the "Bad" state all packets are dropped. The chance to go from the "Good" state to the "Bad" state is p. The change to go from the "Bad" state to the "Good" state is q.

We implement this by using the built in support for the Gilbert-Elliot model in Netem. To add Gilbert-Elliot Model based loss we execute the following command on the network simulator machine:

```
sudo tc qdisc add dev eno1 root netem loss gemodel 5% 50%
```

This command introduces the Gilbert-Elliot model with a $p = 0.05$ and $q = 0.50$. These are fairly high values for packet loss however due to the high packet rate of the testbed these values are needed to create noticeable loss bursts. This essentially means we test under unrealistically difficult packet loss, however as will be shown in Section 5.1 our setup still operates without problem.

There are more complex models that produce even more realistic loss as shown in the work by Ellis et al. [47]. However, these models would only make the network conditions easier for our testbed and makes reproducing the setup more difficult as these models would need to be implemented outside of Netem as they are not supported.

4.6 Testbed protocols

To help testing our measure and to test how modular our testbed is we implement several protocols. We describe the implemented protocols below.

4.6.1 Simple protocol

The Simple protocol is a protocol that just forwards packets without making any dropping decisions besides dropping out of order arrived packets. This means the implementation of this is fairly straight forward. The Master side always applies the latest received feedback packets and transmits all recorded `ToolInformation` objects.

4.6.2 Deadband protocol

The Deadband protocol is a protocol based on the deadband concept as described in Section 2.3.2. The idea behind this protocol is that it greatly reduces packet rates because much less packets need to be transmitted. Which in turn can increase reliability. To implement the protocol both the Master and Slave side need to keep track of the last information they transmitted.

The Master side needs to keep track of the last transmitted position (x_{last}), we solve this by storing the last transmitted `ToolInformation` in memory. On every sample the protocol checks if the the current position (x_{cur}) changed beyond the threshold and if so queues a new packet containing the position information for the network communication to handle and records this position as the last transmitted position in memory. The condition that checks if the information changed sufficiently enough is shown in Equation (4.1).

$$|x_{last}[i] - x_{cur}[i]| > \text{deadband}_{\text{position}} \tag{4.1}$$

The Slave side needs to track similarly using `ToolFeedbackInformation`. Moreover, the slave side also needs to transmit the position of virtual objects. Thus for the implementation we also included position of the virtual objects into the deadband checks.

Chapter 5

Performance Analysis

In this Chapter we evaluate the performance of our testbed and our measures Effective Time & Value -Offset (ETVO). We show that the performance of our testbed meets all our requirements and is recommended for developing and testing future TI projects. We also show that our measures ETVO are a huge leap forward compared to the state of the art.

5.1 Testbed

In this Section we characterize the performance of our testbed to verify if the testbed meets the requirements. The characterization can be used so it's properties can be considered when developing with the testbed. Moreover we look at the signals produced when certain network conditions or protocols are introduced.

5.1.1 Latency analysis

A key performance measure of any TI testbed is the amount of base delay they introduce. As described in Section 4.4.5 we log the output containing the original timestamping allowing us to easily calculate per sample delay. We then draw a Cumulative Distribution Function (CDF) of the delay distribution, shown in Figure 5.1. We used two variables:

- **Sample rate**. The frequency at which data from the device is sampled and thus transmitted. The purpose is to see if changing the sample rate would significantly impact the testbed delay.

- **Network presence**. We simulated both with and without the physical presence of the network simulator. This means that without both the Master and Slave are run on the same machine (thus only one machine is used). While when the network is present the simulation is ran on two separate machines for the Master and Slave respectively connected to each other via the network simulator. This allows use to observe the impact of the physical presence of a network and the network simulator.

The results of these simulations can be seen in Figure 5.1. We must note that the left out frequencies (200, 300, 400 and 750 Hz) show similar behavior to those that are shown and thus can be safely not shown here to preserve clarity. When the frequency increases the CDF moves left and thus the median delay decreases. This behavior is to be expected because when the sampling frequency increases packets get pushed out more quickly and the packet inter arrival time decreases. This results in packets with too much delay getting skipped due to the early arrival of later transmitted packets. Furthermore the presence of a network introduces a extra delay while the distribution is mostly unchanged.

Using the data we calculated that 98 % of the packets experience less then 5 ms delay. This does stay below the delay budget of 10 ms. Preferably it should be smaller so that it does not consume half of the delay budget allowing the budget to be used for other purposes, like longer distances between parts of the testbed. However, the current testbed performance is greatly influenced by the use of the Windows OS making it difficult to decrease the max delay. A significant contributor to the E2E delay in our solution is that the underlying OS handling of our thread and network scheduling is not real-time. This is reinforced by the fact that the Windows OS does not offer APIs to do real-time scheduling.

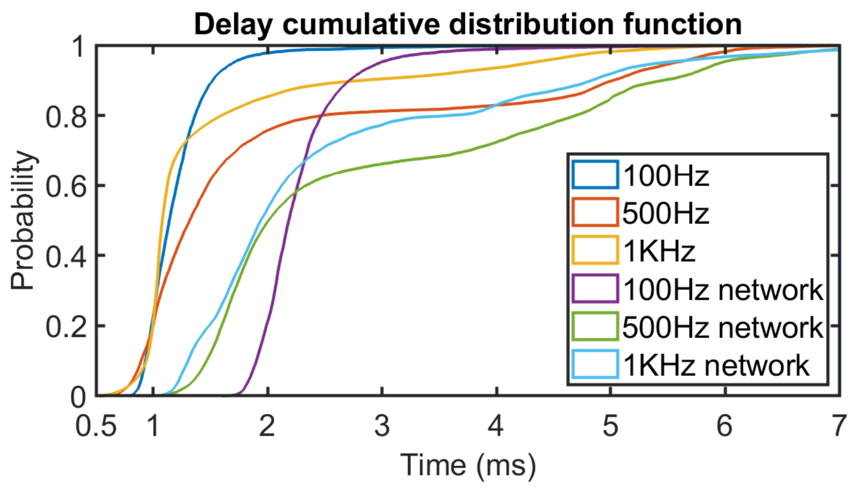

Figure 5.1: Delay CDF comparison for different sampling frequencies and presence of a physical network. Other frequencies were measured but not displayed for clarity.

Parameter	Value	Standard Error
μ	7.6968	0.0032
σ	0.4702	0.0022

Table 5.1: Parameters of the Lognormal fit for delay distribution of the default testbed situation of 1 kHz sample rate and network included.

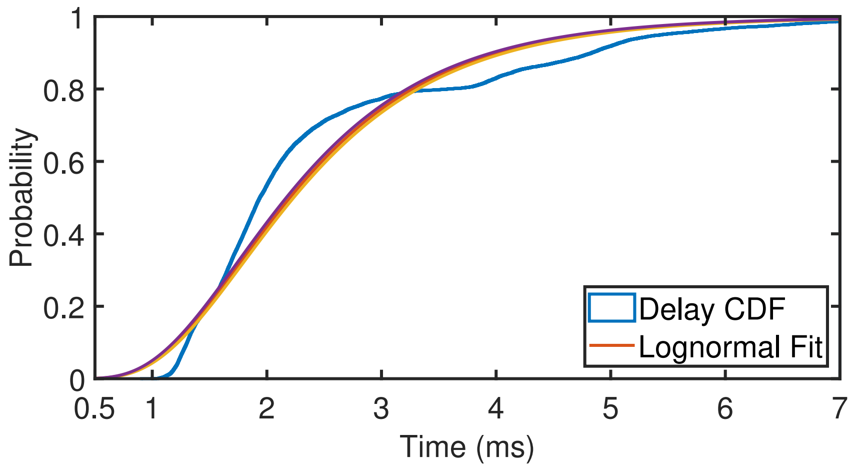

Figure 5.2: Lognormal fit with 99 % confidence interval shown. The default situation of 1 kHz sample rate and network included is fitted.

Latency distribution

Too get a more concrete idea of the behaviour of the testbed delay we fitted several distributions to our delay data. The best fit, the Lognormal fit can be seen in Figure 5.2. The parameters of the Lognormal distribution can be seen in Table 5.1. From the standard error and very tight confidence interval it can be seen that the fit is good. However, visually it can be seen that the fit is not perfect but for this scale of delay values it still is useful tool for making the testbed behavior concrete.

5.1.2 Binning analysis

During testbed development we noticed that the packet arrival was not ideal resulting in packets being skipped due to newer packets also being available simultaneously (binning). After extensive diagnoses we found that there were two causes either packets arriving simultaneously or being handled simultaneously on the Master side. We diagnosed this by logging the processing and arrival timestamp of both usable and unusable packets. The log then showes packets having same arrival and/or processing timestamps. To get a better insight into the magnitude and primary source of this problem we

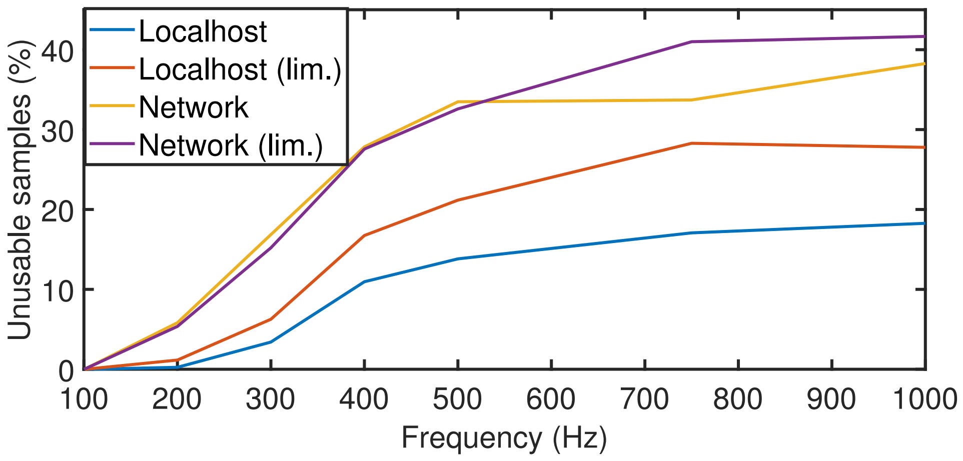

Figure 5.3: Showing the percentage of unusable packets due to binning for different situations and sampling frequencies. we measured in four situations, the inclusion of a network and/or a performance limited system marked "(lim.)".

simulated four situations: with and without the network simulator included and with and without performance limiting the Master side. The results are shown in Figure 5.3. The Figure contains a dip for the "Network" signal, the cause of this is due to unstable behavior of the Windows OS. From this Figure can be seen that both performance and network binning have a considerable impact on the percentage of packets becoming unusable. The main cause is that our used OS is not real-time causing the asynchronous handling of packets to be unaligned creating binning. This is especially clear when loop overruns occur due to high system utilization. With loop overrun we mean that the execution time of a single loop iteration is longer than the allowed 1 ms.

5.1.3 Deadband protocol implementation

A requirement of our testbed is that other protocols should be easy to implement without needing to change anything else in the testbed. We successfully implemented the widely used deadband protocol for both position and force. This implemented deadband without making changes to any testbed code. This gives an indication that our modularity is properly designed. We also tested and verified the working of the deadband protocol. A snippet of such a signal is shown in Figure 5.4.

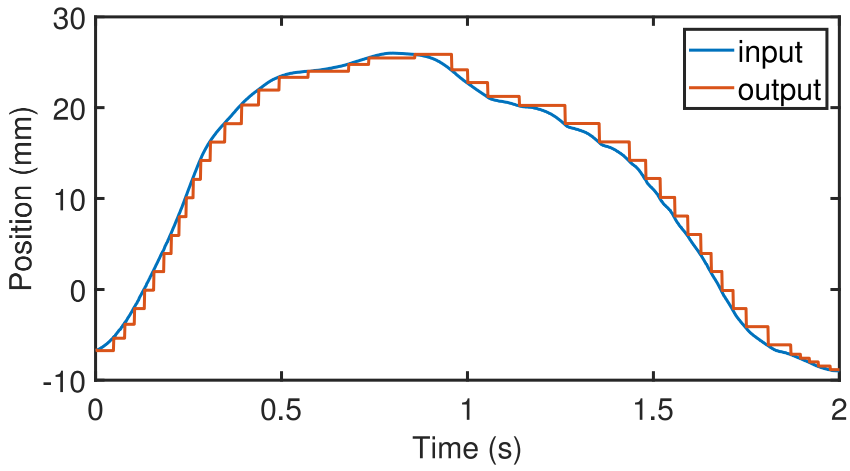

Figure 5.4: Snippet of a signal with both input (without deadband) and output (with 5 % deadband).

5.1.4 Bursty loss simulation

To check that the bursty loss is simulated correctly and has the impact we expect, we simulated several traces using Netem. Deploying the Gilbert-Elliot model for bursty packet loss. A snippet of one of those traces can be seen in Figure 5.5. The behavior is as we expect it would be, when no packets arrive (due to bursty loss) the output does not change. In Figure 2.1 two large burst are marked, where the behavior is clearly visible. This same behavior can be seen for every trace.

5.2 Effective Time & Value -Offset

In this Section we evaluate the performance of our measures ETVO. For ETO we use the penalties as described in Chapter 3. The data is generated while sampling at 1 kHz.

5.2.1 Course-grained comparison

In this Subsection we look at the course-grained performance of our measures compared to existing measures. We show why and how existing measures perform worse compared to our measures.

Existing TI measures

Existing TI measures cannot differentiate between different time-offsets, to illustrate this we created two delayed signals by introducing artificial delay to an existing signal, these signals are shown in Figure 5.6. When calculating

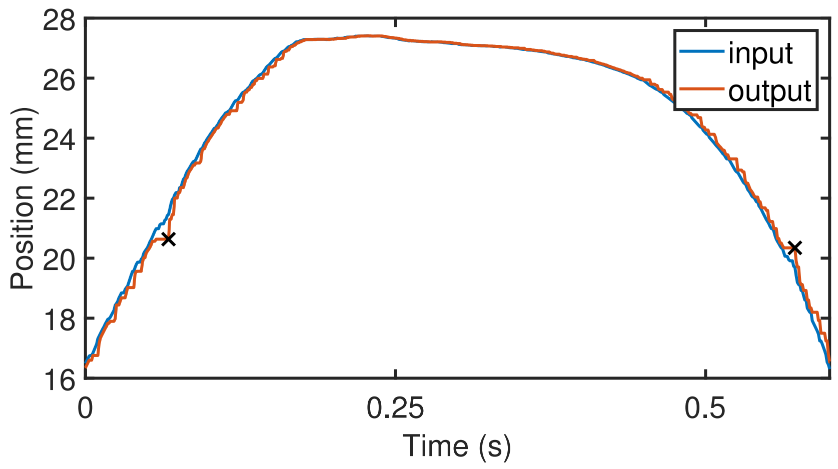

Figure 5.5: Snippet of a signal where the network conditions contain bursty packet loss. Two occasions of bursty packet loss marked with a cross.

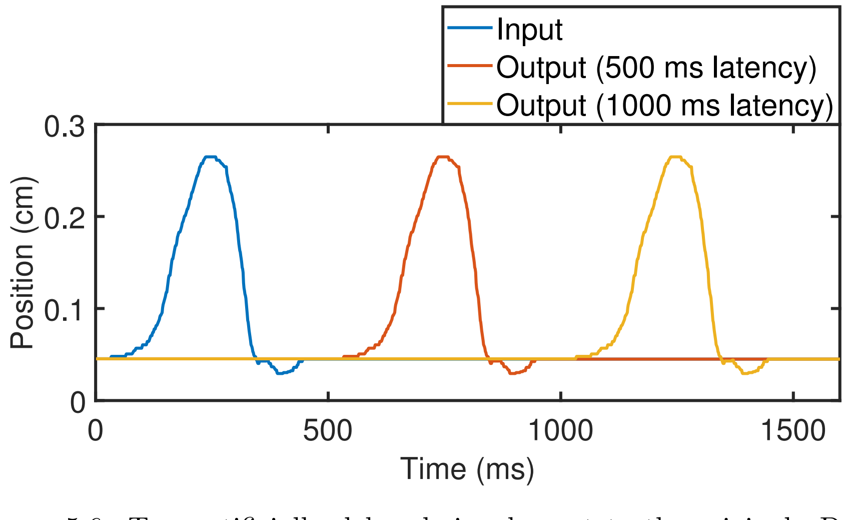

Figure 5.6: Two artificially delayed signals next to the original. Delay of 500 ms and 1000 ms was introduced.

Measure	500 ms	1000 ms
HPWPSNR	9.0141	9.0141
PMSE	0.5506	0.5506
HSSIM	5.4950	5.4524
ETO	500	1000
EVO	0	0

Table 5.2: Calculated measures between the original signal and the 500 ms and 1000 ms delayed versions, which are shown in Figure 5.6

existing measures HSSIM, HPWPSNR and PMSE using the mathematics as explained in Section 2.3.2 between the original signal and the two delayed versions, we can see as shown in Table 5.2 that they cannot correctly identify the different time-offset. It is interesting to note that HSSIM is able to see a small difference, however, this is not due it being able to differentiate between different time-offsets but rather that it is using a very large window size (600 samples) for it's statistical values thus running into boundary problems with the length of our signals (1600 samples). Moreover, our measures is able to perfectly obtain the correct time and value -offset.

Jitter based tests

To test the stability of our measures we do a course-grained analysis in different network conditions. We simulated different network jitter magnitudes to show the influence of every greater changing network conditions. We chose the base network delay of 50 ms to allow the testing of a wide variety of network jitter magnitudes without introducing too much delay so that the interaction becomes unusable. The results of these simulations can be found in Figure 5.7. For RMSE we also show a modified DTW where the boundaries of the calculated value-offset are set to zero, this is to show that DTW can produce the lowest value-offset but is heavily impacted by the boundary conditions. The amount of samples set to zero is only 0.5 % of the total signal length, thus showing the difference in value-offset between the two can be mainly contributed to the boundary conditions.

What is clear from these results is that ETVO is able to produce a near identical value-offset performance compared to DTW while requiring 20 times to 100 times less changes in time-offset to do so. Moreover, only when jitter increases ETO needs to make more changes.

5.2.2 Fine-grained comparison

To show fine-grained performance of ETO we designed four scenario's, showcasing the improved decision making of our measures.

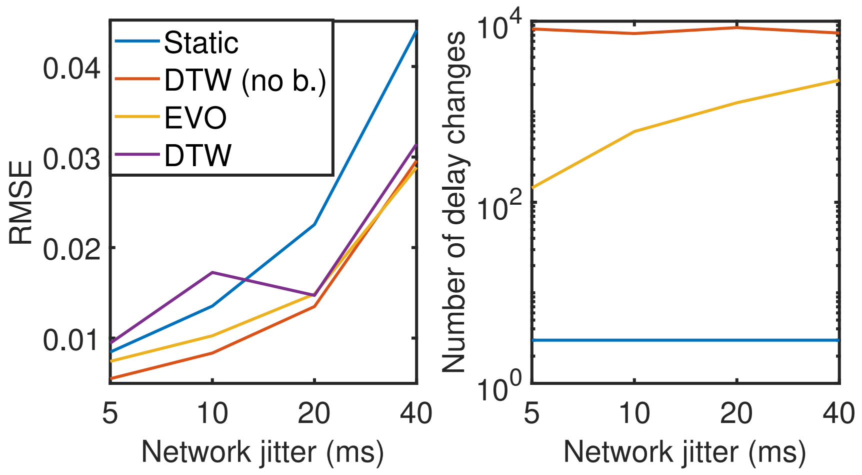

Figure 5.7: Comparison of RMSE and number of changes needed to achieve the RMSE for 50 ms simulated network delay and different simulated network jitters. DTW is shown double for the RMSE graph to show the impact of boundary value-offset. DTW (no b.) and DTW are without and with boundary value-offset respectively. This clearly shows that DTW achieves the lowest RMSE when ignoring boundary problems but requires the highest amount of changes.

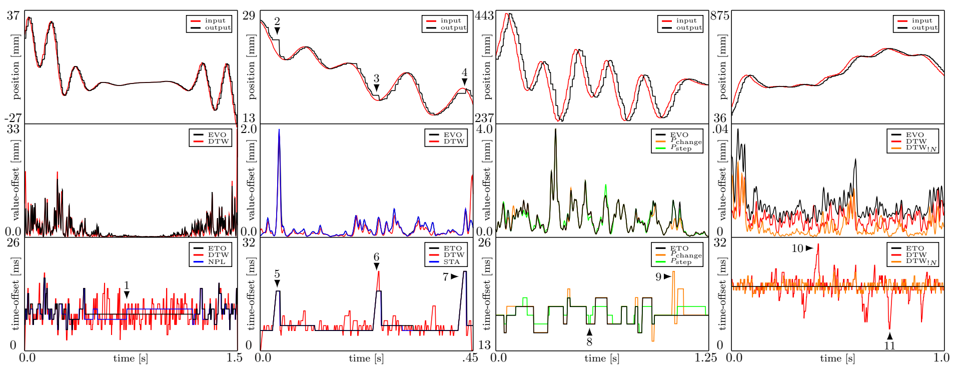

Figure 5.8: Fine-grained behavior of CC, DTW and ETO showed using four scenario's

Context-sensitivity

In this scenario we show that DTW keeps changing time-offset even when there are extremely small gains. While ETO correctly chooses to not change time-offset. Also can be seen that P_{lazy} is especially important in these situations to make sure changes are not made where it does not make sense. This behavior is shown in the first column of Figure 5.8. Here can be seen that DTW makes changes regardless of velocity, while our system decided that it is not worth it in periods of low velocity. The version of our measure with P_{lazy} disabled is marked as NPL. At the marked point 1 you can see that a counter-intuitive change is made when P_{lazy} is absent.

Lack of updates

In this scenario we compare the performance when deadband is introduced. We do this by combining the deadband protocol using a 5 % deadband and bursty loss simulated using the Gilbert-Elliot model. The resulting behavior can be seen in the second column of Figure 5.8. Here can be seen as denoted by 2, 3 and 4 large events happening because of bursty loss causing a big loss in information on the output side. In these three situation our measure correctly changes the time-offset as seen by the peaks at 5, 6 and 7 while not changing the time-offset in between because there the offset is value-offset caused by the deadband protocol.

Comparison between P_{change} and P_{step}

In this scenario we show how both P_{change} and P_{step} are relevant to improving the performance of our measure ETO. In this scenario we keep $P_{lazy} = 0$ to remove it's influence when showing the behavior of the other two penalties. The behavior is shown in the third column of Figure 5.8. Here it can be seen that when only P_{step} is present and P_{change} is removed (as denoted by P_{step}) all changes are made in more frequent smaller steps. Especially present at the mark 8 where multiple smaller steps are used instead of on large step. When only P_{change} is present and P_{step} is removed (as denoted by P_{change}) changes are made in unreasonable large amounts. Especially present a the mark 9 where a very large but short change is made to optimize for a small value-offset.

Effect of noise

In this scenario we show how DTW cannot cope when noise is introduced. We do this by introducing Additive White Gaussian Noise (AWGN) with an SNR of 70 dB comparing that to behavior of DTW when the noise is no present. This can be seen in column four of Figure 5.8. Here the behavior of our measure and DTW with the added noise (marked as DTW) and the DTW behavior without the added noise (marked as $DTW_{!N}$) can be seen. The unreasonable jumps of DTW are increased considerable just by the addition of a small amount of noise, especially visible at point 9 and 10. While our measure correctly keeps reporting an stable time-offset.

Chapter 6

Conclusion and Future Work

6.1 Discussion

To push ETVO further it is important that we study the behaviour of all the parameters especially how each influences the decision making. Right now we only have a rough understanding of how the parameters can be chosen. An improved understanding could be used to design methods or algorithms to improve parameter tuning.

Our measures are currently not linked to the subjective, this means that we cannot yet prove that our time-offset or value-offset matches the actual experience of a user. Thus extensive work still has to be done to test user experience and tune our measure to match that behavior as closely as possible.

We expect our measures to be a significant improvement to measuring, however, we do not yet have the proof to show this is true. There are probably works that can already work with existing measures. To properly prove it we need several TI solutions that are tested with our measures and previous measures to show that our measures significantly improve testing and comparison.

Our measures are purely an offline solution, this means that they cannot be used in online systems to evaluate on the fly. An online solution would allow the measures to be used to calculate system performance and drive decision making.

In this work we only tested our measures for their use within TI. However, the measures could also be applicable for other purposes possible greatly extending the possible uses of the measures. This should be explored to make sure we are not missing possible uses of our measure.

The current testbed is not built using a real-time OS, this impacts performance and makes synchronizing asynchronous sample streams difficult. This results in the output data needing to be discretized in post processing. To solutions is to move the testbed to a real-time OS, however requiring an almost complete rewrite depending on the real-time OS used.

Currently the testbeds virtual environment is very simple, this makes it difficult to design scenario's for tests. Preferably you would have scenario's of different difficulties so that the difference in performance per scenario can be properly tested.

The purpose of the testbed is to eventually produce datasets that links measurements to QoE. These datasets would allow future developments to be simplified. However, these datasets do not yet exist.

As seen by our latency analysis the testbed incurs delays of up to 5 ms, this means that our testbed uses a considerable amount of delay budget. This limits the flexibility within this budget. Which limits the possible distance parts of the testbed can be placed apart.

During our measurements we saw that limiting the system performance of the Master side impacted the performance of the testbed. This means that running the testbed requires powerful hardware to guarantee quality performance. The reason for this should be investigated and improved to allow the testbed to also be run on less powerful hardware.

We characterized the latency of the testbed using a Lognormal distribution, while this gives a fairly tight fit distribution it can be further improved. Possibly by employing multiple convoluted distributions.

The current device used, the Novint Falcon is not being produced anymore as the original manufacturer went bankrupt some year ago. This means that there is currently only limited supply, solely available via second hand sale. Thus making possible replacement when needed difficult. Currently widely available devices should be investigated for support with the current testbed as possible replacement of the Novint Falcon to ensure the continued use of the testbed if the current device were to become unusable.

6.2 Conclusion

In this work we introduced a new method to test and measure TI technologies. Our solution fundamentally changed the approach to TI measuring by considering time and value -offset as two orthogonal parts of a signals error. We showed that this new approach to TI measuring performs considerable better than DTW and existing TI measures.

Our testbed offers considerable more options to how a TI solution can be tested. We did this by designing a completely new architecture from the ground up and implementing it in a way that produced a testbed that can be employed in wide variety of future research projects. Moreover we also present extensive tests of the testbed, showing the performance and characteristics of the testbed. The testbed also has simple and extendable APIs to make it easily usable in future research projects.

Using the data generated by the testbed we extensively verified the performance and behavior of the proposed measures ETVO. We demonstrated that ETO and EVO have superior performance in solving the stated challenges when compared to existing solutions. We recommend ETVO as most suitable measures to evaluate TI sessions with. We also expect our measures to find use outside of the field of the TI. This is a direct result of our introduced measuring methodology which allows our measure to be employed in any situation where two signals need to be compared.

6.3 Future Work

We delivered considerable measuring and testing improvements. This opens the road to many new possibilities. We envision that this work will significantly help TI to move forward and motivate others to develop and test innovative TI solutions. In this Section we highlight possible ways forward for TI research and propose possible future steps enabled by this work.

6.3.1 Testbed Future Work

We identify some modifications that could make the existing testbed perform better. Below is a non-exhaustive list of possible improvements.

- **Linux support**. Linux offers better schedulers and especially tunability, possibly allowing some of the testbed delay to be reduced by employing the better OS scheduling of Linux. The implementation is challenging because there is no official driver for the current hardware for Linux. There are custom drivers but these have not been updated in a while have to be made compatible with Chai3d.

- **Implement Real-time scheduler**. Implement an own scheduler by implementing it into the testbed. This can possibly be achieved by dedicating a cpu core to the testbed process and using that core continuously to have very reliable loop timing.

- **Decouple driver readout**. Currently the device information is read from inside the main loop to make sure we are working the most recent information. A possible improvement is performing the readout in a separate thread and whenever a data sample is needed request an sample from the readout thread that interpolates using velocity from the samples to bridge the small time gap.

- **Implement multiplexer support**. Extend the protocol implementation to allow also transmitting other information like video and/or audio. Care should be taken that a good API is designed so that easily different multiplexing methodologies can be implemented and tested.

- **Subjective measurements**. Make the testbed ready for subjective measurements. For this tooling is needed that simulates configured scenario's in a way so that subjects can easily perform measurements.

6.3.2 Effective Time & Value -Offset

We identify a set of tasks that could improve the behavior and impact of ETVO considerably. Below is a non-exhaustive list of possible future steps.

- **More sophisticated parameter tuning**. Currently we give a indication of good penalty values to use, however more research and experimentation is required to expand the theory on how to best tune the three penalty parameters.

- **Quality measure**. Using the information extracted from two signals using ETVO design a measure that condenses the quality of the observed into a single grade indicating how good the session is.

- **Coupling with the subjective**. Do subjective measurements and couple the quality of the observed signal to the scores given by subjects and design a metric like R-factor which describes perceived performance of a situation without needing subjective measurements.

- **3 dimensional extension**. By changing positions into 3D coordinates it is possible to extend ETVO into a measure that calculates time and value-offset for 3D coordinates.

- **Real time heuristic solution**. By changing the ETO algorithm it should be possible to create a version that works in real time allowing systems to have real time estimates of time and value-offsets. These values could then be used in the decision making process of the application.

6.3.3 Tactile Internet

We see this work as a new starting point to considerably fuel the development of TI. Providing improvements that make developing for the TI easier while also improving verification.

Bibliography

[1] G. P. Fettweis, “The tactile internet: Applications and challenges,” *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[2] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of neural science.* McGraw-hill New York, 2000, vol. 4.

[3] S. Okamoto, M. Konyo, S. Saga, and S. Tadokoro, “Identification of cutaneous detection thresholds against time-delay stimuli for tactile displays,” in *2008 IEEE International Conference on Robotics and Automation.* IEEE, 2008, pp. 220–225.

[4] M. C. Chan and R. Ramjee, “TCP/IP performance over 3G wireless links with rate and delay variation,” *Wireless Networks*, vol. 11, no. 1-2, pp. 81–97, 2005.

[5] F. Verloy. (2013) Effects of latency and packet loss on tcp throughput. Accessed: 2019-08-12. [Online]. Available: https://filipv.net/2013/06/19/effects-of-latency-and-packet-loss-on-tcp-throughput/

[6] V. Millnert, J. Eker, and E. Bini, “Achieving predictable and low end-to-end latency for a network of smart services,” in *2018 IEEE Global Communications Conference (GLOBECOM).* IEEE, 2018, pp. 1–7.

[7] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, “Federated learning for ultra-reliable low-latency v2v communications,” in *2018 IEEE Global Communications Conference (GLOBECOM).* IEEE, 2018, pp. 1–7.

[8] F. Gringoli, R. Klose, M. Hollick, and N. Ali, “Making wi-fi fit for the tactile internet: Low-latency wi-fi flooding using concurrent transmissions,” in *2018 IEEE International Conference on Communications Workshops (ICC Workshops).* IEEE, 2018, pp. 1–6.

[9] theconstructionindex. (2019) Excavator controlled in germany digs in korea. Accessed: 2019-07-03. [Online]. Available: https://www.theconstructionindex.co.uk/news/view/excavator-controlled-in-germany-digs-in-korea

[10] P. Pratumsuwan, S. Hutamarn, and W. Po-ngaen, “A development of tele-controller for hydraulic excavator,” in *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, July 2012, pp. 2123–2126.

[11] S. Huh, U. Lee, H. Shim, J. Park, and J. Noh, “Development of an unmanned coal mining robot and a tele-operation system,” in *2011 11th International Conference on Control, Automation and Systems*, Oct 2011, pp. 31–35.

[12] G. Fau, T. Matsunaga, and K. Ohnishi, “Development of a five degrees of freedom master/slave robot for tele-operated laparoscopic surgical operations,” in *2014 7th International Conference on Human System Interactions (HSI)*, June 2014, pp. 172–177.

[13] M. Ships. (2019) Mercy ships - international hospital ships. Accessed: 2019-07-04. [Online]. Available: https://mercyships.org/

[14] P. J. Choi, R. J. Oskouian, and R. S. Tubbs, “Telesurgery: Past, Present, and Future,” *Cureus*, vol. 10, no. 5, pp. 1–5, 2018.

[15] R. Becker. (2019) Robot squeezes suspected nuclear fuel debris in fukushima reactor. Accessed: 2019-07-05. [Online]. Available: https://www.theverge.com/2019/2/15/18225233/robot-nuclear-fuel-debris-fukushima-reactor-japan

[16] D. Moore. (2019) Industrial automation — digital manfucturing industry 5.0. Accessed: 2019-08-12. [Online]. Available: https://www.gemnewz.com/2019/07/16/industrial-automation-digital-manfucturing-industry-5-0/

[17] B. Rossi. (2019) What will industry 5.0 mean for manufacturing? Accessed: 2019-07-01. [Online]. Available: https://www.raconteur.net/technology/manufacturing-gets-personal-industry-5-0

[18] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, “5g-enabled tactile internet,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 460–473, March 2016.

[19] A. Aijaz, M. Dohler, A. Hamid Aghvami, V. Friderikos, and M. Frodigh, “Realizing the Tactile Internet: Haptic Communications over Next Generation 5G Cellular Networks,” *IEEE Wireless Communications*, vol. 24, no. 2, pp. 82–89, 2017.

[20] C. Li, C.-P. Li, K. Hosseini, S. B. Lee, J. Jiang, W. Chen, G. Horn, T. Ji, J. E. Smee, and J. Li, “5G-Based Systems Design For Tactile Internet,” *Proceedings of the IEEE*, pp. 1–18, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8452975/

[21] Ofcom. (2015) Ofcom research shows 4g significantly outperforms 3g networks. Accessed: 2019-06-28. [Online]. Available: https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2015/4g-outperforms-3g

[22] S. Pandi, R. S. Schmoll, P. J. Braun, and F. H. Fitzek, "Demonstration of mobile edge cloud for tactile Internet using a 5G gaming application," *2017 14th IEEE Annual Consumer Communications and Networking Conference, CCNC 2017*, pp. 607–608, 2017.

[23] Y. Miao, Y. Jiang, L. Peng, M. S. Hossain, and G. Muhammad, "Telesurgery robot based on 5g tactile internet," *Mobile Networks and Applications*, vol. 23, no. 6, pp. 1645–1654, Dec 2018. [Online]. Available: https://doi.org/10.1007/s11036-018-1110-3

[24] Z. Yuan, S. Chen, G. Ghinea, and G.-M. Muntean, "User quality of experience of mulsemedia applications," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 11, no. 1s, p. 15, 2014.

[25] F. Brandi, J. Kammerl, and E. Steinbach, "Error-resilient perceptual coding for networked haptic interaction," in *Proceedings of the 18th ACM international conference on Multimedia.* ACM, 2010, pp. 351–360.

[26] L. A. Jones and I. W. Hunter, "Human operator perception of mechanical variables and their effects on tracking performance." *ASME DYN SYST CONTROL DIV PUBL DSC., ASME, NEW YORK, NY(USA), 1992,*, vol. 42, pp. 49–53, 1992.

[27] P. Hinterseer, E. Steinbach, and S. Chaudhuri, "Perception-based compression of haptic data streams using kalman filters," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, May 2006, pp. V–V.

[28] S. Hirche, P. Hinterseer, E. Steinbach, and M. Buss, *Network Traffic Reduction in Haptic Telepresence Systems By Deadband Control.* IFAC, 2005, vol. 38, no. 1. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1474667016371804

[29] N. Sakr, N. Georganas, and J. Zhao, "A perceptual quality metric for haptic signals," in *2007 IEEE International Workshop on Haptic, Audio and Visual Environments and Games.* IEEE, 2007, pp. 27–32.

[30] R. Chaudhari, E. Steinbach, and S. Hirche, "Towards an objective quality evaluation framework for haptic data reduction," in *2011 IEEE World Haptics Conference.* IEEE, 2011, pp. 539–544.

[31] R. Hassen and E. Steinbach, "HSSIM: An Objective Haptic Quality Assessment Measure for Force-Feedback Signals," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, may 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8463365/

[32] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[33] S. S. Stevens, "On the psychophysical law." *Psychological review*, vol. 64, no. 3, p. 153, 1957.

[34] M. Eid, J. Cha, and A. El Saddik, "Admux: An adaptive multiplexer for haptic–audio–visual data communication," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 21–31, 2010.

[35] V. Gokhale, J. Nair, and S. Chaudhuri, "Congestion control for network-aware telehaptic communication," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 2, pp. 17:1–17:26, Mar. 2017. [Online]. Available: http://doi.acm.org/10.1145/3052821

[36] B. Cizmeci, X. Xu, R. Chaudhari, C. Bachhuber, N. Alt, and E. Steinbach, "A multiplexing scheme for multimodal teleoperation," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 2, pp. 21:1–21:28, Apr. 2017. [Online]. Available: http://doi.acm.org/10.1145/3063594

[37] A. Bhardwaj, B. Cizmeci, E. Steinbach, Q. Liu, M. Eid, A. E. Saddik, R. Kundu, X. Liu, O. Holland, M. A. Luden, S. Oteafy, and V. Prasad, "A Candidate Hardware and Software Reference Setup for Kinesthetic Codec Standardization," in *2017 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE)*, 2017, pp. 53–58.

[38] (2019) Chai3d - features. Accessed: 2019-07-12. [Online]. Available: https://www.chai3d.org/concept/features

[39] (2015) Novint falcon - product page. Accessed: 2019-07-12. [Online]. Available: https://web.archive.org/web/20150215032400/http://www.novint.com/index.php/products/novintfalcon

[40] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[41] H. Sakoe, S. Chiba, A. Waibel, and K. Lee, “Dynamic programming algorithm optimization for spoken word recognition,” *Readings in speech recognition*, vol. 159, p. 224, 1990.

[42] C. Basdogan, C.-H. Ho, M. A. Srinivasan, and M. Slater, “An experimental study on the role of touch in shared virtual environments,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 7, no. 4, pp. 443–460, 2000.

[43] (2019) Acquiring high resolution time stamps. Accessed: 2019-07-11. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/sysinfo/acquiring-high-resolution-time-stamps

[44] (2019) networking:netem [wiki]. Accessed: 2019-07-19. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[45] U. Foundation. (2019) Download ubuntu server — download — ubuntu. Accessed: 2019-07-24. [Online]. Available: https://ubuntu.com/download/server

[46] E. N. Gilbert, “Capacity of a burst-noise channel,” *The Bell System Technical Journal*, vol. 39, no. 5, pp. 1253–1265, Sep. 1960.

[47] M. Ellis, D. P. Pezaros, T. Kypraios, and C. Perkins, “121. Modelling packet loss in RTP-based streaming video for residential users,” *Proceedings - Conference on Local Computer Networks, LCN*, pp. 220–223, 2012.