Matthias Clara, BSc

Model Predictive Control for Networked Control Systems

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Electrical Engineering

submitted to

Graz University of Technology

Supervisor:

Ass.Prof. Dipl.-Ing. Dr.techn. Martin Steinberger

Institute of Automation and Control

Graz, December 23, 2020

AFFIDAVIT

“I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.
The text document uploaded to TUGRAZonline is identical to the present master’s thesis.”

Signature

Date

ACKNOWLEDGMENTS

I would like to thank Martin Steinberger for being my helpful advisor during the whole working period of my master's thesis and for introducing me into the area of Networked Control Systems. I couldn't have asked for a better person.
Special thanks are also given to my family for their financial support throughout my studies and for their superior education that helped me become who I am today.

Matthias Clara
Graz, December 23, 2020

ABSTRACT

Networked Control Systems (NCS) have been one of the main research areas for the past years. In such systems, data may be randomly delayed or even lost during the transmission. Some methods based on Model Predictive Control (MPC) are considered in order to overcome the so-called network-induced constraints. This thesis is focused on the controller design and on the performance analysis within the framework of data packet dropouts and data packet disorder. Both sensor-to-controller and controller-to-actuator network-induced delays are assumed to be upper and lower bounded and equal to an integer multiple of the sampling period. In the beginning of this thesis, an overview of the topic NCSs is given. The next part focuses on the conventional MPC and on the modified MPC approach which can be used in a NCS. The last part of this thesis includes a simulation example and a practical implementation in order to show the feasibility and efficiency of the considered methods.

KURZFASSUNG

Der weitverbreitete Einsatz von vernetzten Regelungssystemen führt zu einem grundlegenden Wandel im Bereich der Regelungstechnik. In solchen Systemen ist die Ankunftszeit der Datenpakete beim Adressaten unbekannt und ein möglicher Datenverlust kann auftreten. Im Zuge dieser Masterarbeit werden Methoden basierend auf die modellprädiktive Regelung (MPC) gezeigt, um die netzwerkinduzierte Totzeiten und die Datenpacketverluste in einem vernetzten Regelungssystem zu kompensieren. Es wird angenommen, dass die Verzögerungen vom Sensor zum Regler und die Verzögerungen vom Regler zum Aktuator beschränkt sind und einem ganzzahligen Vielfachen der Abtastperiode entsprechen. Im ersten Abschnitt dieser Arbeit wird ein allgemeiner Überblick über das Thema vernetzte Regelung gegeben. Des Weiteren werden die konventionelle modellprädiktive Regelung sowie einige modifizierte Versionen davon genauer betrachtet um Netzwerkeffekte berücksichtigen zu können. Im abschließenden Teil dieser Arbeit wird die Effizienz und die Anwendbarkeit der gezeigten Methoden anhand eines konkreten Simulationsbeispiels sowie eines Experiments überprüft.

Contents

Symbols and Acronyms

NCS	Networked Control Systems
MPC	Model Predictive Control
PID	Proportional-Integral-Derivative Controller
CAN	Controller Area Network
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
FTP	File Transfer Protocol
RTD, RTT	Round-Trip Delay, Round-Trip Time
LMI	Linear Matrix Inequality
$SISO$	Single Input Single Output
$MIMO$	Multiple Input Multiple Output
$RHOC$	Receding Horizon Optimal Control
CAS	Control Action Selector
HIL	Hardware-in-the-Loop
$\tau_{c,k}$	Controller calculating delay
$\tau'_{c,k}$	Extended controller calculating delay
$\tau_{sc,k}$	Network-induced delay in the backward channel
$\tau_{ca,k}$	Network-induced delay in the forward channel
τ_k	Round-Trip Delay
h	Sampling period
A_c, B_c, C_c, D_c	Continuous time system dynamic matrix, input matrix, output matrix and direct feedthrough
A, B, C, D	Discrete time system dynamic matrix, input matrix, output matrix and direct feedthrough
n	Number of states
m	Number of inputs
p	Number of outputs
I	Identity matrix
N_c, N_p	Control and prediction horizon
Q, R	Weighting matrices
F, G, H	Matrices for the predicted outputs of the system
F_x, G_x, H_x	Matrices for the predicted states of the system
W, L, M	Matrices for the constraints
J	Cost function
$t_{stamp,back}$	Time stamp in the backward channel
$t_{stamp,for}$	Time stamp in the forward channel
$t_{stamp,start}$	Time stamp for the controller calculating delay
t_{now}	Actual time instant
$U_{k-\tau_{sc,k}\|k-\tau_{sc,k}}$	Entire Control sequence based on information up to time $k - \tau_{sc,k}$
$U_{k\|k-\tau_{sc,k}}$	Reduced control sequence based on information up to time $k - \tau_{sc,k}$
$U_{k\|k-\tau_{sc,k}-\tau_{c,k}}$	Reduced control sequence based on information up to time $k - \tau_{sc,k} - \tau_{c,k}$
$n_{d,sc}$	Upper bounds of consecutive packet drop in the backward channel
$n_{d,ca}$	Upper bounds of consecutive packet drop in the forward channel

1 Introduction

Nowadays, with the development of large-scale or complex industrial systems, there is a need of using distributed actuators, sensors and control devices. In order to reduce high cost and complexity of communication links, shared communication media are used, by which a huge amount of information is sensed, processed and transmitted. Such a control system, where the control loop is closed through a real-time communication network, is referred to as Networked Control System (NCS) [11, 14, 39, 40].

NCSs have made it possible that a large number of actuators, sensors and controllers can be interconnected over the network to interact with the physical environment. Such systems have been applied in a broad range of areas such as remote surgery, automated highway systems, mobile sensor networks and unmanned aerial vehicles [21]. This kind of control system involves controlling a plant from a remote location through a communication channel and brings new functionalities, such as reduced system wiring, low cost, decreased system complexity and simple system maintenance and diagnosis [5, 21]. The advantages brought by NCSs however do not come at no cost. An important basis of conventional control systems is that the data exchange among the control components are lossless. In NCSs, the data has to be transmitted through the communication network, which means that perfect data exchange among the control components is essentially unavailable. This imperfection introduces the so-called communication constraints to the control system, which include, e.g., network-induced delays (the delays occur in transmitting the feedback and control information), data packet dropouts (packets may get lost during the transmission), data packet disorder (the order of the sent packets may be changed), time synchronization problems (different control components may work based on different clocks), and so on. These communication constraints present great challenges for the controller design and can degrade the control performance and even cause instability of the feedback loop.

One of the main problems in NCSs is the design of control schemes accounting for the absence of feedback and control information for possibly long time intervals. Many approaches based on different ideas have been introduced. Depending on the assumptions about the network connection, stochastic approaches as, e.g., in [25, 26], or methods only assuming the boundedness of delays are employed [27]. Buffers are utilized in order to handle time varying delays in [28, 29]. In [30], a version of a filtered Smith predictor is proposed that allows to robustly stabilize uncertain unstable plants with constant time delays. The problem of the design of robust H_∞ controllers for uncertain NCSs with the effects of networked-induced delays and data packet dropouts has been considered in [31]. The controller design and the stability analysis based on over-approximation techniques and linear matrix inequalities (LMIs) are presented in [27, 32]. An MPC strategy for multivariable plants was shown in [33], where the sensor-to-controller delays where described by stochastic and deterministic quantities, but the controller-to-actuator delays were assumed to be known and fixed. In [34], the MPC method was used to compensate the packet dropouts at the sensor-to-controller side, while zero control was applied when the control packet was lost. In [1, 35, 36], modified MPC methods were introduced to compensate the delayed or missing control signals.

Model Predictive Control (MPC) provides a set of future control sequences at one time instant, which means that the future values of the control input can be predicted [1].

Because of this feature, the MPC approach can be used to compensate the communication constraints in NCSs. In this work it's assumed that the network-induced delays are upper and lower bounded. Furthermore, it's also assumed that the time-delays are equal to an integer multiple of the sampling time. The design problem of the closed-loop NCS in the presence of communication delays and data packet dropouts is illustrated and several compensation methods are presented. In order to avoid data packet disordering, an active compensation scheme is proposed, which requires the usage of the time stamp technique and a comparison rule defined at the actuator side. With this method, the effect of data packet disorder can effectively be eliminated. A simulation example and a practical implementation are given in order to show the feasibility and efficiency of the proposed methods.

The goal of this thesis is to investigate how the communication constraints affect the control performance of the control system. Furthermore, it's also shown how the MPC deals with the communication delays in both forward and backward channel. However, the stability is not taken into consideration and no statement about the stability analysis is made.

2 Overview of Networked Control Systems

Over the past years, communication and computing technologies have improved tremendously. Consequently, the field of control engineering itself has also advanced towards networked control. In general, a Networked Control System (NCS) is a control system, where the control loop is closed through a communication network. The defining feature of a NCS is that signals are exchanged among the system components in the form of data packages through a shared communication media. Such kind of systems provide many benefits, but they also raise several challenges which have to be overcome. In this section, the common characteristics which are shared by many NCSs in many application domains are investigated. Figure 1 shows a typical structure of a NCS.

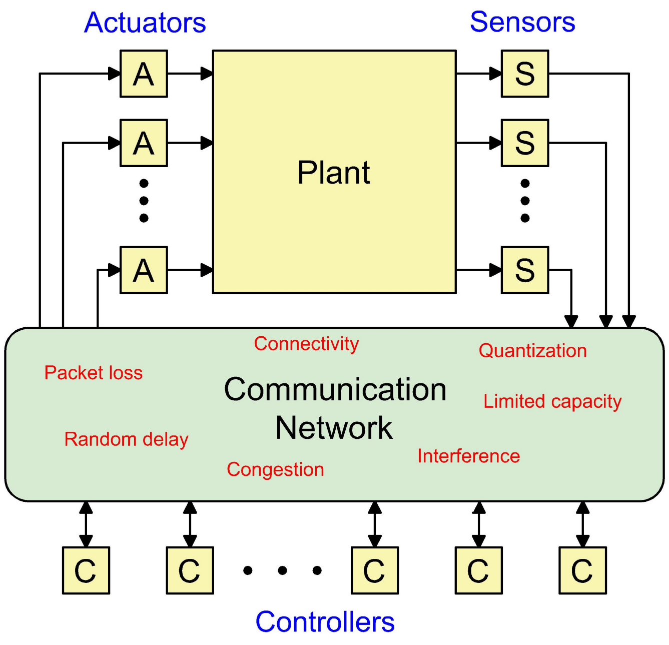

Figure 1: Typical structure of a NCS.

2.1 Advantages and Applications of Networked Control Systems

For many years now, data networking systems have been applied in military and industrial applications, such as aircraft [21], intelligent traffic control systems [11], factory automation [21], remote surgery [14], and so forth. Connecting the control system components via a shared network can effectively reduce the complexity of systems, without a big economical investment. NCSs reduce unnecessary wiring and they allow data to be shared efficiently [5]. One of the biggest advantages of NCSs is scalability. It's easy to add or remove control components, such as sensors, actuators or controllers, which are connected through a network at different locations, without having to heavily change the whole system. Furthermore, NCSs simplify the system maintenance and diagnosis [21]. These systems are becoming more and more important today and they have a lot of potential in applications like space explorations [21], domestic robots [14], and so on.

2.2 Components of Networked Control Systems

A NCS consists of five basic elements:

- **The plant**, which has to be controlled.
- **The sensor/s**, to collect the information.
- **The actuator/s**, to apply the control commands.
- **The controller/s**, to provide decisions and commands.
- **The communication network**, to enable exchange of information.

An essential part of a control system is the acquisition of information from the environment. Sensor data can be in any form starting from small numbers representing weight, temperature, pressure, etc. or in chunk form such as arrays, images or videos [21]. The potential application of large-scale networked systems is constantly growing. In NCSs the relevant data is collected using distributed sensors to study the system under control. The sensors should be cheap, reliable and energy efficient and they should also be easily added or removed from a system.

For many years now, different control strategies, starting from PID control, optimal control, adaptive control, robust control and so on, have been studied when using NCSs [7, 21]. Applying all these control strategies over a network becomes a challenging task, because of the communication constraints that may occur (see figure 1).

When choosing the communication or data transfer type, security, reliability and availability are the main areas to focus on. The choice of the network depends obviously on the application to be served. Control Area Network (CAN) is a serial, asynchronous, multi-master communication protocol mostly used for applications needing high data rates of up to 1Mbps [21]. CAN is applied in industrial and automotive applications. The internet is one of the most used choices for many applications where the plant and the controller are far away from each other.
The generalized NCS structure with all his components is shown in figure 2.

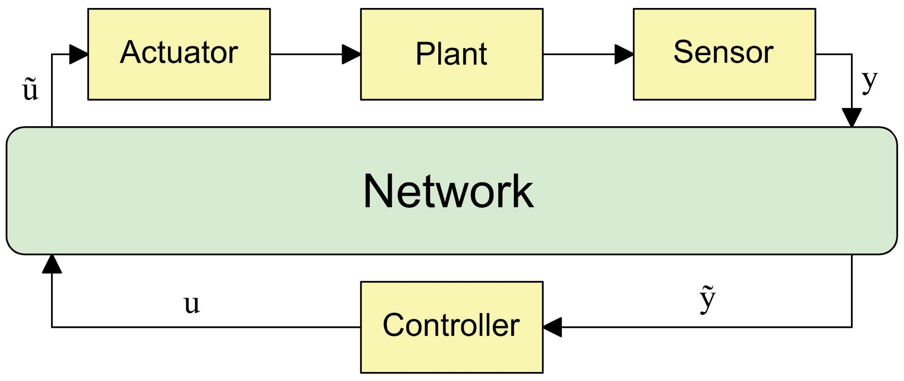

Figure 2: Block diagram of a NCS.

The signals y and u are the plant output and the control signal respectively. The signals $\tilde{y}$ and $\tilde{u}$ are the plant output and the control signal transmitted through the communication network. From a general perspective of system structure, NCSs may contain two different structures: the *direct structure*, shown in figure 2 and the *hierarchical structure*, shown in figure 3 [7, 11].

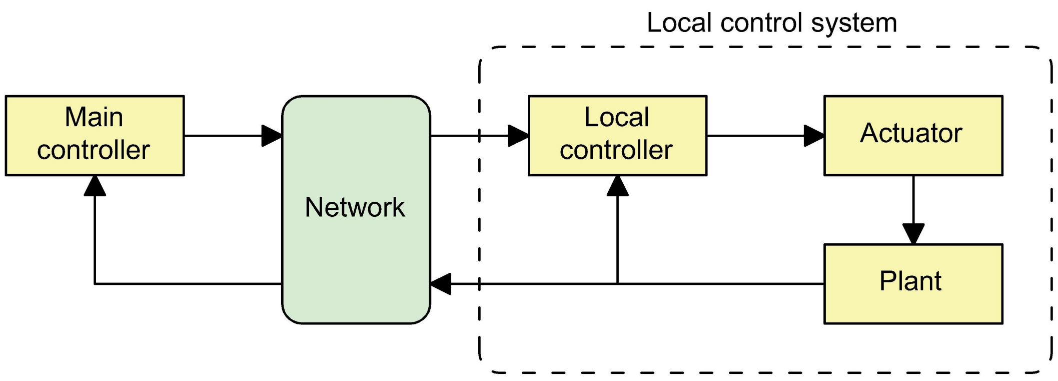

Figure 3: NCS in the hierarchical structure.

In the hierarchical structure a local controller is present on the plant side and the communication network is used to close the loop between the main controller and the local system [7]. This type of approach can be seen as a hierarchical combination of the direct structured NCS and a conventional local control system. Most available papers on NCSs have focused on the direct structure, which is also the case in this thesis.

2.3 Packet-based Data Transmission

In NCSs, data is stored in packets and then transmitted through the communication network. The information is split up into similar structures of data before transmission, called packets, which are reassembled to the original data chunk once they reach their destination [7].

The most commonly used protocols for sending packets over a network are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Both protocols build on top of the IP protocol. In other words, all the packets are sent to an IP address, regardless of whether the TCP or the UDP is used. Generally, TCP is more suitable for exchanging larger files while UDP is best for simple messages. The main differences between these two protocols are illustrated in table 1 [18].

For the TCP, the communicating devices have to establish a connection before transmitting data and they must close the connection after the transmission. The TCP never concludes a transmission until all the data has been correctly transmitted. It stores data in a send buffer and receives data in a receiver buffer. TCP uses a flow control mechanism that ensures a sender is not overwhelming a receiver by sending too many packets at once. This protocol is best suited for applications that need high reliability and where transmission time is relatively less critical, e.g. File Transfer Protocol (FTP), email, and so on [18].

The UDP is a connection-less protocol, which means that no connection between sender and receiver has to be established before sending the data. UDP doesn't provide flow control. It continuously sends datagrams to the recipients whether they receive them or not. With the usage of this protocol, packets may get lost and they may be delivered out of order. UDP is best suited for applications that require speed and efficiency, e.g., online games, live broadcasts, streaming videos, and so on [18].

Transmission Control Protocol (TCP)	**User Datagram Protocol (UDP)**
Connection-oriented protocol.	Datagram oriented protocol.
Does error checking and also makes error recovery.	Performs error checking, but it discards erroneous packets.
Retransmission of lost data packets is possible.	Retransmission of lost packets is not possible.
Is reliable as it guarantees delivery of data.	The delivery of data cannot be guaranteed.
Data packet disorder cannot occur.	Data packet disorder can occur.
20-80 bytes variable length header.	8 bytes fixed length header.
Slower, as it requires more resources.	Faster as error recovery is not attempted.
Less time efficient.	Simpler and time efficient.

Table 1: Differences between TCP and UDP.

As far as NCSs is concerned, UDP is used in most applications (and also in this thesis) due to the real-time requirement and the robustness of the control system. As a result, the effect of data packet dropout and data packet disorder has to be explicitly considered.

2.4 Control Challenges

Due to the usage of channels with a limited bandwidth, random transmission delays may occur in the forward and/or in the backward channel. The size of the data packets can be fixed but it is usually limited only to the maximum capacity [7]. In some cases, e.g., if more than one sensor is used, the packet size is too low to carry all the information, therefore, the data can be sent within more than a single packet (data fragmentation). The transmission delay is the time from the beginning of the transmission of the first data packet to the time instant when the last part of the data has been received. If at least one packet may be lost during the transmission, all the information will be unrecoverable.

Overfilled transmission buffers and random delays can be a source of another important issue, which is data packet disorder. Furthermore, in NCSs the limited bandwidth of the network generates a situation, where a smaller sampling period may not lead to a better system performance, which is normally the case for a sampled data system.

2.4.1 Network-induced Delay

The main reason of problems, which can degrade the control performance in NCSs is a limited capacity of the communication media [4]. Due to this limitation, random transmission delays can occur. A delay in a control loop typically affects the performance and the stability of the control system. In certain circumstances, the delay may be so large, that it can be treated as a data lost. Figure 4 shows the structure of a typical NCS, including the random network-induced delays.

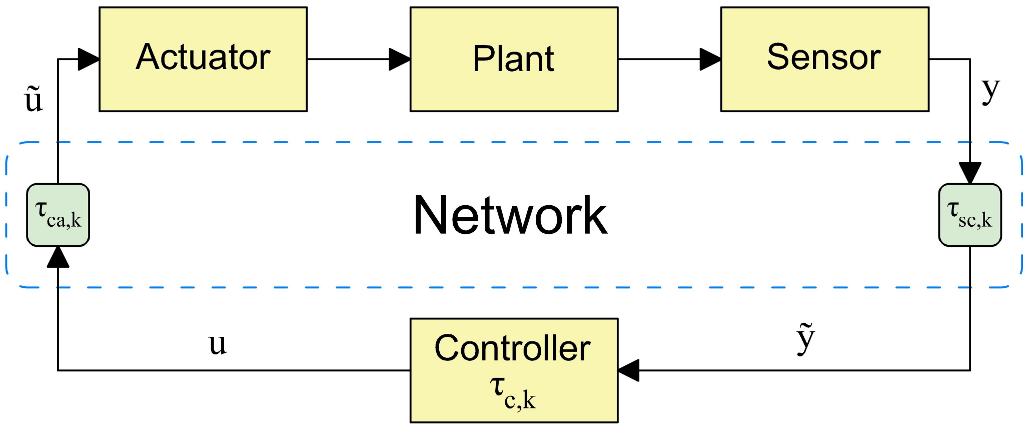

Figure 4: Structure of a NCS with the network-induced delays.

There are three types of network-induced delays:

- $\tau_{sc,k}$: sensor-to-controller delay (backward channel delay)
- $\tau_{ca,k}$: controller-to-actuator delay (forward channel delay)
- $\tau_{c,k}$: controller calculating delay

All these delays are non-deterministic [9]. Very often, the controller calculating delay $\tau_{c,k}$ is negligibly small compared to the other two delays. Depending on the type of communication network being used in NCSs, the characteristics of the network-induced delays vary as the follow [7]:

(i) **Random access networks** (e.g. CAN or Ethernet) are random and unbounded delays.

(ii) **Cyclic service networks** (e.g. Token-Bus) are bounded delays, which can often be assumed as constant.

(iii) **Priority order networks** (e.g. DeviceNet) are bounded delays for the data packets with higher priority and unbounded delays for those with lower priority.

The delays $\tau_{sc,k}$ and $\tau_{ca,k}$ may have different characteristics, however, in most cases they are not treated separately (only for static controllers) and only the so-called Round-Trip Delay (RTD), often also referred to as Round-Trip Time (RTT), is of interest [7, 11].

$$\tau_k = \tau_{sc,k} + \tau_{ca,k} + \tau_{c,k} \tag{1}$$

In this work it's assumed that the network-induced delays are equal to an integer multiple of the sampling period h. Furthermore, it's also assumed that the delays are upper and lower bounded such as

$$\begin{aligned} \tau_{sc,min} \leq \tau_{sc,k} \leq \tau_{sc,max} \\ \tau_{ca,min} \leq \tau_{ca,k} \leq \tau_{ca,max}. \end{aligned} \tag{2}$$

Note that $\tau_{sc,k}$, $\tau_{ca,k}$ and $\tau_{c,k}$ are integer numbers representing the multiple of the sampling period. This means that the actual network-induced time-delays can be computed by multiplying $\tau_{sc,k}$, $\tau_{ca,k}$ and $\tau_{c,k}$ with the sampling period h.

2.4.2 Data Packet Dropout

Data transmission errors in communication networks are inevitable. A significant difference between NCSs and conventional control systems is the possibility, that data may not reach the wanted destination. This phenomenon is often referred to as *data packet dropout*. In figure 5 the structure of a NCS with the data packet dropout problem is shown.

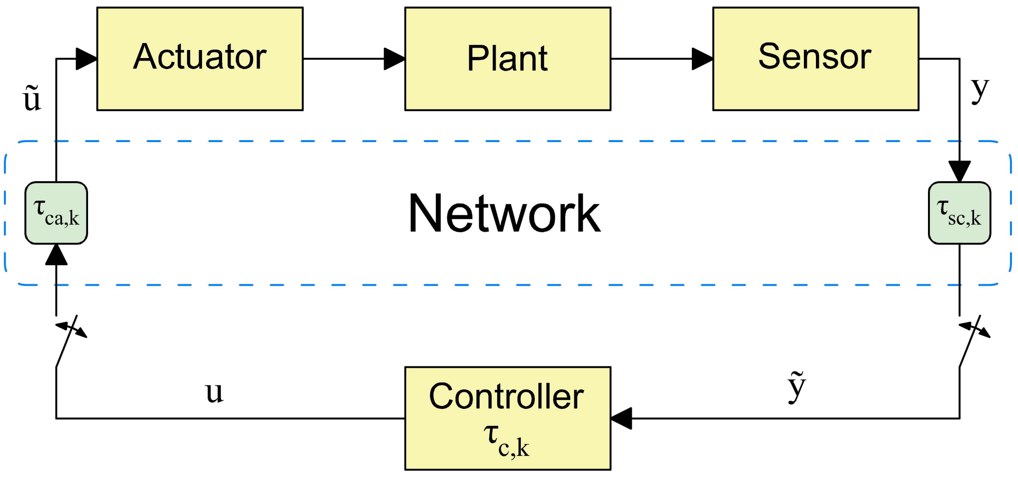

Figure 5: Structure of a NCS with the network-induced delays.

Data packet dropouts can occur either in the backward or in the forward channel and make either the sensing data or the control signals unusable. Usually, packet dropouts result from transmission errors in physical network links, which is far more common in wireless than wired networks, or from buffer overflows due to congestion.

As already mentioned in section 2.3, data packet dropouts cannot occur in the case where the TCP is used, since all the data will be transmitted successfully.

2.4.3 Data Packet Disorder

In communication networks, different data packets suffer different delays, which means that a data packet sent earlier may arrive at the destination later than a data packet which has been sent later [11, 12]. This phenomenon is referred to as *data packet disorder*.

A typical data transmission process in NCSs is illustrated in figure 6. Let's assume that a sensor sends its sampled data every h seconds. Due to the arbitrary network-induced delays, the sampled data packet sent at time instant t_{k-1} may arrive at the controller side later than its subsequent data packet sent at time instant t_k.

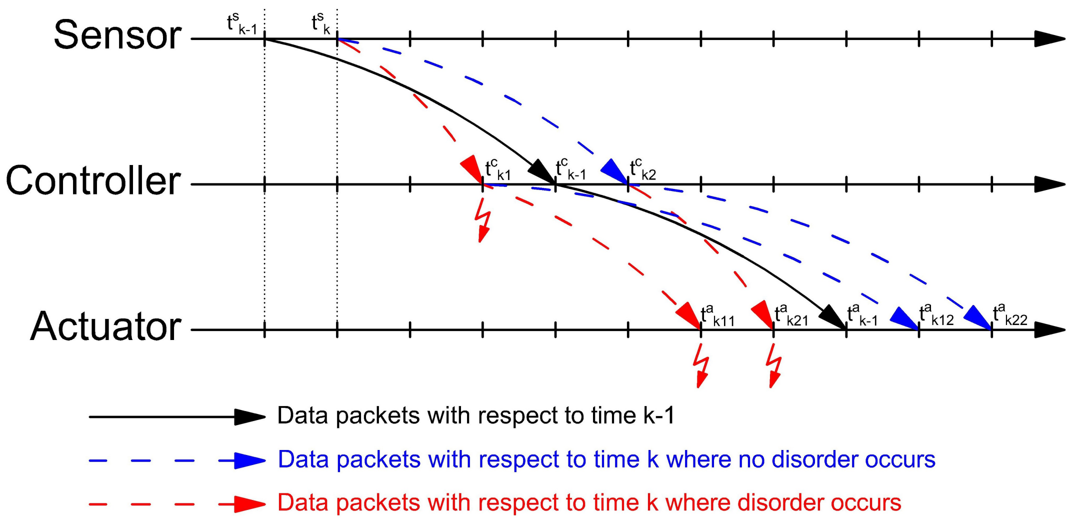
Sensor
Controller
Actuator
t^s_{k-1}
t^s_k
t^c_{k1}
t^c_{k-1}
t^c_{k2}
t^a_{k11}
t^a_{k21}
t^a_{k-1}
t^a_{k12}
t^a_{k22}
Data packets with respect to time k-1
Data packets with respect to time k where no disorder occurs
Data packets with respect to time k where disorder occurs

Figure 6: Typical data transmission process in a NCS.

Remark. *Given a constant sampling period h and variable network-induced delays, data packet disorder occurs if and only if*

$$\Delta\tau_m = \tau_{max} - \tau_{min} > 1 \tag{3}$$

where τ_{max} and τ_{min} are upper and lower bounds of the RTD [6].

2.4.4 Single and Multiple-packet Transmission

In the case where, e.g., multiple sensors and/or multiple actuators are used (MIMO-systems), data may be delayed or lost only in a single link. The controller has to wait for the arrival of all the sensing data packets before it is able to compute the upcoming control actions. If only one sensing data packet is lost, all the others have to be discarded due to insufficiency of information. Such a technique is often referred to as *multi-packet transmission* [7, 11].

Another possibility is to send the sensing data of multiple steps via a single data packet over the network. The packet size used in NCSs can be very large compared to the data size needed to encode a single step of sensing data. Such a technique is often referred to as *single-packet transmission* [7].

2.4.5 Sampling Period

In NCSs, a smaller sampling period may not result into a better system performance [7]. This phenomenon happens because if the sampling period is too small, too much sensing data will be produced. This leads to an overload of the network and it causes congestion, which may increase the chance of longer delays and data packet dropouts. The relation between the sampling period, the network congestion and the system performance is shown in figure 7.

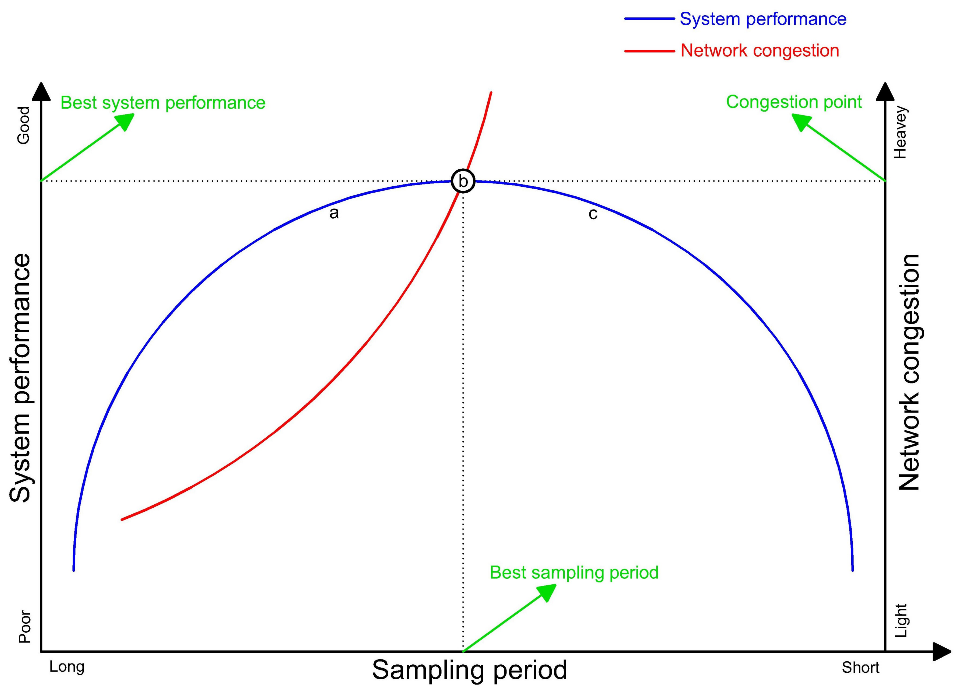

Figure 7: Relation between sampling period, network congestion and system performance.

When the sampling period decreases (a to b), the system performance gets better, since the network congestion does not appear until point b. If the sampling period decreases even more (b to c), the system performance decreases as well, due to network congestion. As shown in figure 7, there is an optimal sampling period (point b), which leads to the best possible system performance.

2.5 Simulation of Variable Time Delays

There are several techniques to accurately simulate the networked-induced delays. Very often, the time-varying delays is assumed to be lower and upper bounded, which is also done in this thesis. Furthermore, the delays are assumed to be equal to an integer multiple of the sampling time h, since all the components work with the same sampling rate. This last assumption simplifies the implementation substantially. In this thesis, all the simulations are performed with Matlab and Simulink. In [16] it has been shown, why the Simulink built-in blocks *Variable Integer Delay* and *Variable Time Delay* shouldn't be used to simulate a packet-based time-varying transmission, since the resulting data heavily deviates from the accurate result.

Let's consider a system consisting of a continuous-time plant and a discrete-time controller that are connected with two communication channels. The feedback information is stored in packets and sent through the communication media to the controller with a sampling time of $h = 1s$. The transmission is affected by an arbitrary variable time delay, which is shown in figure 8.

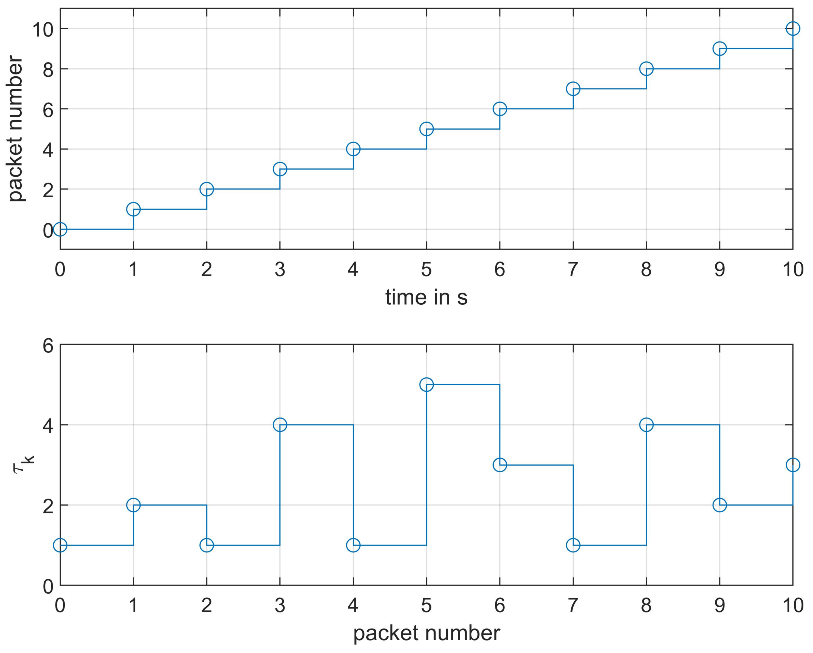

Figure 8: Packets transmitted through the channel (top). Respective delays (bottom).

Packet 0 is delayed by 1s, packet 1 is delayed by 2s, packet 2 is delayed by 1s, and so on. Figure 9 shows why the Simulink build-in blocks shouldn't be used in NCSs. The resulting data packets of both blocks are the same, but they strongly deviate from the accurate result, because the specified delay is related to the sampling instances and not to the individual packets in the networked control loop. In [16], a possible extension of the Simulink built-in blocks *Variable Integer Delay* and *Variable Time Delay* is presented. It may happen, that several packets arrive at the destination at the same time (see figure 10 at time instance $t = 3s$). In this scenario, the most recent packet is used and the older one is simply be discarded. Sometimes, in the case where newer packets are available, it makes sense to skip older packets. In Algorithm 1 the pseudo code of the packet-based transmission, in the case where older packets don't get skipped, is illustrated. Whereas Algorithm 2 shows the pseudo code of the packet-based transmission in the case where older packets get skipped.

Algorithm 1: Packet-based transmission in the case where older packets don't get skipped.

```
Initialize packet buffer;
while simulation is running do
    Subtract $h$ from all delays in the packet buffer;
    if a packet has been sent then
        Get actual packet delay (in multiples of $h$);
        Write actual packet delay at first position in the buffer;
    end
    Choose most recent packet with delay 0 (from the buffer);
    Write the chosen packet at the output;
end
```

Algorithm 2: Packet-based transmission in the case where older packets get skipped.

```
Initialize packet buffer;
Initialize $packet_{old} = 0$;
while simulation is running do
    Subtract $h$ from all delays in the packet buffer;
    if a packet has been sent then
        Get actual packet delay (in multiples of $h$);
        Write actual packet delay at first position in the buffer;
    end
    Choose most recent packet with delay 0 (from the buffer);
    if chosen packet number $> packet_{old}$ then
        Write chosen packet at the output;
        Store the packet number of the chosen packet into $packet_{old}$;
    else
        Write $packet_{old}$ at the output;
    end
end
```

Note that the packet number in algorithm 2 is simply the position of the packet in the packet buffer. In order to be able to store the packets and their respective delays, the so-called *persistent variables* are used. Persistent variables are local to the function in which they are declared, yet their values are retained in memory between calls to the function [20]. The simulated results of the extended version of the Simulink block can be seen in figure 10.

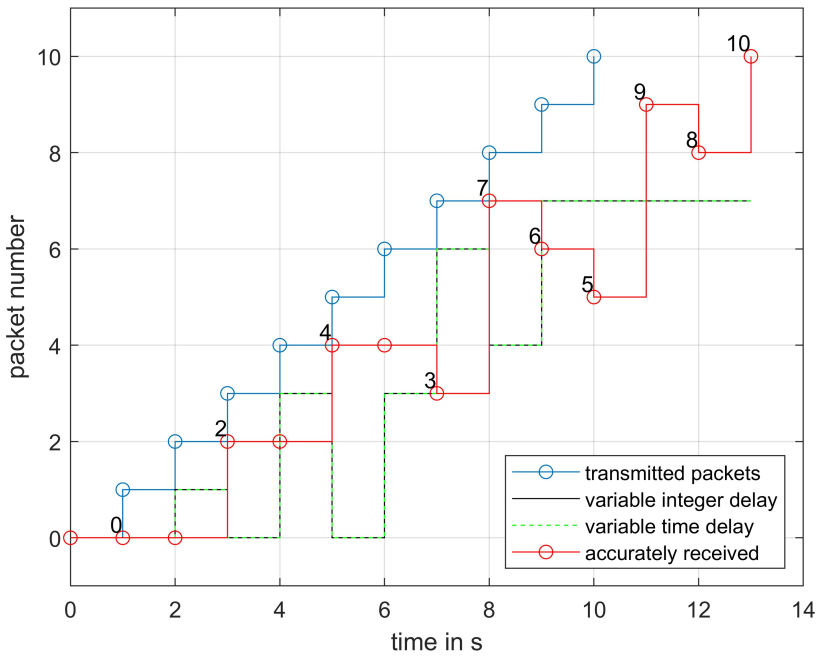

Figure 9: Results using the Simulink built-in blocks.

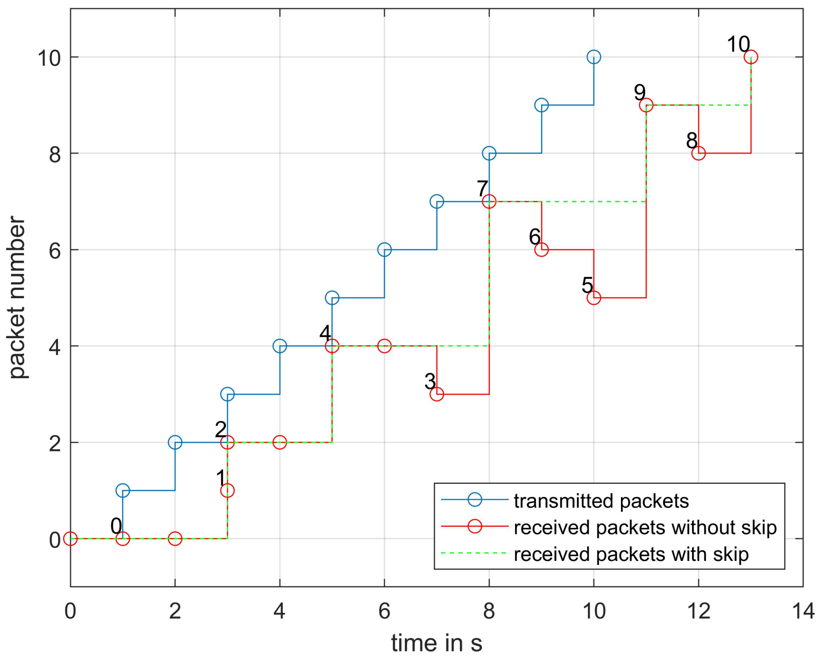

Figure 10: Results using the extended version of the Simulink built-in block *Variable Integer Delay*.

3 Classical Model Predictive Control

Model Predictive Control (MPC) is an advanced control method that is used to handle control problems while systems having input, state and output constraints [5]. It is an algorithm based on the Receding Horizon Optimal Control (RHOC) methodology [3]. The MPC uses a model of the system to make predictions about the system's future behavior. It solves an online optimization problem to find the optimal control action that drives the predicted output to the reference signal. It can also incorporate future reference information into the control problem to improve the control performance [10].

The basic MPC concept can be summarized as follows: the control action is obtained at each time instant based on the control process model predicting either the output or the states of the process over a fixed number of future time instants, known as the prediction horizon N_p, with the current plant state as the initial condition [4, 5]. The result is a sequence of $N_c - 1$ predicted control values, where N_c is the so-called control horizon. Only the first value from the sequence is applied to the process. At the next time instant, a new control sequence is calculated using current process state values as initial conditions. Figure 11 shows the structure of the MPC approach, where $\hat{u}_{k+i}$ and $\hat{y}_{k+i}$ are the predicted values of the plant input and plant output respectively.

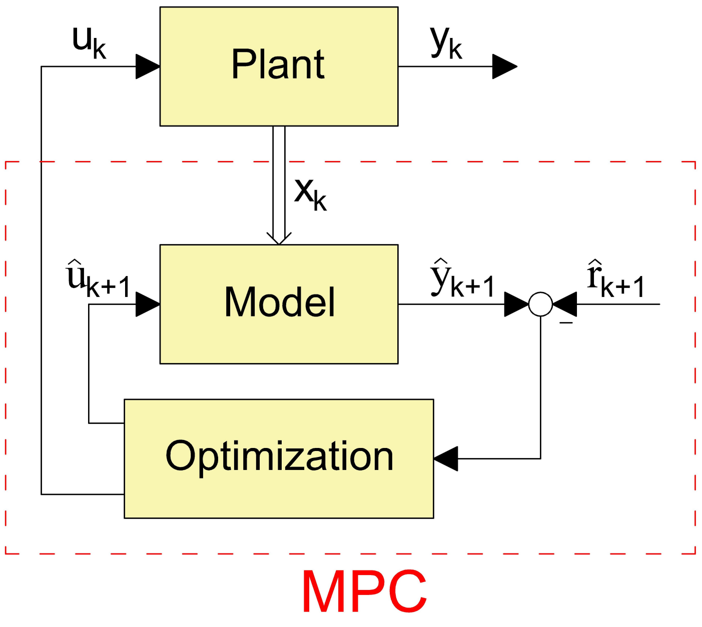

Figure 11: Structure of the MPC.

The MPC offers several important advantages such as [10, 17]:

(1) Constraints on the state, input and output of the system can be imposed (hard and soft constraints).

(2) It's easy to implement for MIMO systems.

(3) The process model captures the static and dynamic interactions between input, output and disturbance variables.

(4) No integrator is needed. $\Rightarrow$ no windup effect

As usual, the advantages brought by the MPC do not come at no cost. Some disadvantages of the MPC approach are [17]:

(1) Fast optimization algorithms should be used, since the optimization is performed online.

(2) It relies on fast hardware.

(3) Depends on the process dynamics.

(4) Challenging for nonlinear systems.

In MPC applications, the output variables are often referred to as *controlled variables*, while the input variables are also called *manipulated variables* [19]. While MPC is suitable to almost any kind of problem, it displays its main strength when applied to problems with [19]:

- A large number of manipulated and controlled variables.
- Constraints imposed on both the manipulated and controlled variables.
- Time delays.

The actual output y, the predicted output $\hat{y}$ and the manipulated input u for a SISO system are illustrated in figure 12.

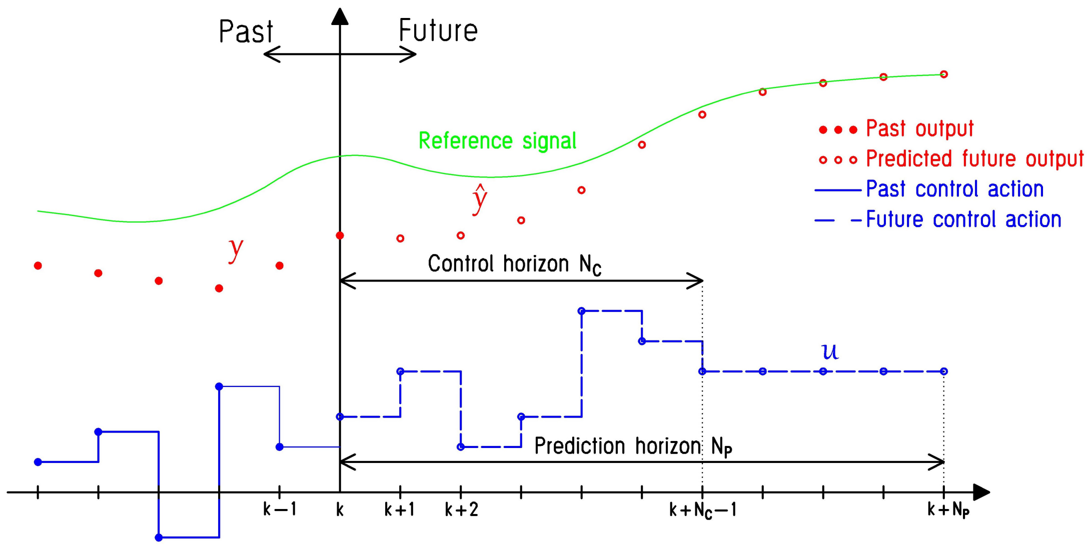

Figure 12: Basic concept for MPC.

Due to the ability to pre-compute future control actions, the MPC approach provides a possible implementation for NCSs, with the advantage of being able to deal with delays and random data packet loss. Therefore, the MPC should be developed by considering constraints and data packet dropouts.

In this thesis, the Matlab toolbox for optimization modeling called YALMIP is used [37]. This toolbox makes development of optimization problems extremely simple. The *qpoases* solver is used to solve the optimization problem [38].

3.1 Design of the Controller

The plant of a linear, time-invariant, time-continuous MIMO system is given as

$$\begin{aligned} \frac{dx(t)}{dt} &= A_c x(t) + B_c u(t) \\ y(t) &= C_c x(t) \end{aligned} \tag{4}$$

with states $x(t) \in \mathbb{R}^n$, input $u(t) \in \mathbb{R}^m$, output $y(t) \in \mathbb{R}^p$ and constant matrices $A_c \in \mathbb{R}^{n\times n}$, $B_c \in \mathbb{R}^{n\times m}$ and $C_c \in \mathbb{R}^{m\times n}$. With the usage of a sample-and-hold element, the equivalent discrete-time model can be written as

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{5}$$

with $A = e^{A_c h}$, $B = \int_0^h e^{A_c(h-\tau)} B_c d\tau$, $C = C_c$ and where $h \in \mathbb{R}^+$ is the sampling period.

3.1.1 Predictive Model

The model representation in (5) is further modified using a new formulation by incorporating

$$u_k = u_{k-1} + \Delta u_k, \tag{6}$$

where u_{k-1} is the control input at time instant $k-1$ and Δu_k is the difference between the control input at time instant k and the control input at time instant $k-1$ [3, 17]. As it's shown in figure 12, in the case where $N_c < N_p$ the following equation has to hold:

$$\hat{u}_{k+i} = \hat{u}_{k+N_c-1} \quad \text{for} \quad N_c \leq i \leq N_p \tag{7}$$

The predictive model of the output in matrix notation results to

$$\begin{aligned} \bar{y}_{k+1} = \begin{bmatrix} \hat{y}_{k+1} \\ \vdots \\ \hat{y}_{k+N_p} \end{bmatrix} &= Fx_k + Gu_{k-1} + H\Delta\bar{u}_k \\ &= \bar{g}_k + H\Delta\bar{u}_k \end{aligned} \tag{8}$$

with

$$\Delta\bar{u}_k = \begin{bmatrix} \Delta\hat{u}_k \\ \vdots \\ \Delta\hat{u}_{k+N_c-1} \end{bmatrix}, \qquad F = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix}, \qquad G = \begin{bmatrix} CB \\ C(A+I)B \\ \vdots \\ C(A^{N_p-1} + \ldots + A + I)B \end{bmatrix},$$

$$H = \begin{bmatrix} CB & 0 & \cdots & 0 \\ C(A+I)B & CB & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ C(A^{N_c-1} + \ldots + A + I)B & C(A^{N_c-2} + \ldots + A + I)B & \cdots & CB \\ \vdots & \vdots & \ddots & \vdots \\ C(A^{N_p-1} + \ldots + A + I)B & C(A^{N_p-2} + \ldots + A + I)B & \cdots & C(A^{N_p-N_c} + \ldots + A + I)B \end{bmatrix},$$

where $F \in \mathbb{R}^{pN_p \times n}$, $G \in \mathbb{R}^{pN_p \times m}$ and $H \in \mathbb{R}^{pN_p \times mN_c}$ are the matrices used to specify the predicted output of the system.

Another possibility would be to predict the states of the system. The predicted states result to

$$\begin{aligned} \hat{\bar{x}}_{k+1} &= F_x x_k + G_x u_{k-1} + H_x \Delta\bar{u}_k \\ &= \bar{g}_{x,k} + H_x \Delta\bar{u}_k \end{aligned} \tag{9}$$

where F_x, G_x and H_x are the matrices used to specify the predicted states of the system. In order to compute the predictive model of the states, the matrix C has simply to be replaced with the identity matrix I.

3.1.2 Cost Function

The proposed MPC approach is formulated based on a quadratic cost function given as

$$J = \sum_{i=1}^{N_p} (\hat{y}_{k+i} - r_{k+i})^T Q_i (\hat{y}_{k+i} - r_{k+i}) + \sum_{i=1}^{N_c} \Delta\hat{u}_{k+i-1}^T R_i \Delta\hat{u}_{k+i-1}, \tag{10}$$

with the weighting matrices $Q_i \in \mathbb{R}^{pN_p \times pN_p}$ and $R_i \in \mathbb{R}^{mN_c \times mN_c}$ [17]. At this point, the question is how to choose the weighting matrices Q_i and R_i.

- Q_i and R_i should be symmetric (non-symmetric part has no influence)
- $Q_i < 0$, $R_i > 0$: The cost function J may be small, but the error can be large.
- $Q_i < 0$, $R_i < 0$: The optimal solution diverges towards infinity. The optimization problem is unbounded, if no further constraints are used.
- $Q_i > 0$, $R_i > 0$: Is a conservative choice.
- $Q_i \geq 0$, $R_i \geq 0$, $H^TQH + R > 0$: The optimization problem is convex, which means that the solution is for sure a global minimum.

The so-called error vector is introduced to the calculations such as

$$\bar{e}_k = \bar{g}_k - \bar{r}_{k+1}. \tag{11}$$

Equation (10) is further modified and the quadratic cost function in matrix notation can be written as

$$J(\Delta\bar{u}_k) = \Delta\bar{u}_k^T (H^T Q H + R) \Delta\bar{u}_k + 2\Delta\bar{u}_k^T H^T Q \bar{e}_k + \bar{e}_k^T Q \bar{e}_k,$$

where $\Delta\bar{u}_k$ is the optimization variable.

As mentioned before, in this thesis a quadratic cost function is used. Some alternative cost functions are, e.g.,

$$\Rightarrow \quad J = \sum_{i=1}^{N_p} \hat{x}_{k+i}^T Q_i \hat{x}_{k+i} + \sum_{i=1}^{N_c} \hat{u}_{k+i-1}^T R_i \hat{u}_{k+i-1} \tag{12}$$

$$\Rightarrow \quad J = \sum_{i=1}^{N_p} |q_{k+i}^T \hat{x}_{k+i}| + \sum_{i=1}^{N_c} |a_{k+i}^T \hat{u}_{k+i}| \tag{13}$$

The structure of the MPC feedback loop is shown in figure 13.

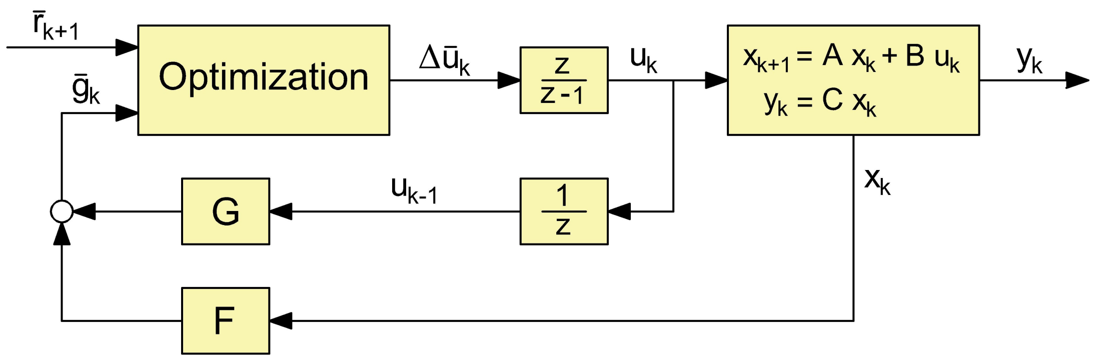

Figure 13: Structure of the classical MPC.

3.1.3 Specification of the Reference Signal

The predicted reference signal $\bar{r}_{k+1} \in \mathbb{R}^{pN_p}$ has to be applied to the system. Let's assume that $p = 3$, which means that the system has three outputs that need to be controlled. Figure 14 shows the basic concept for the specification of the reference signal over the entire simulation time [17].

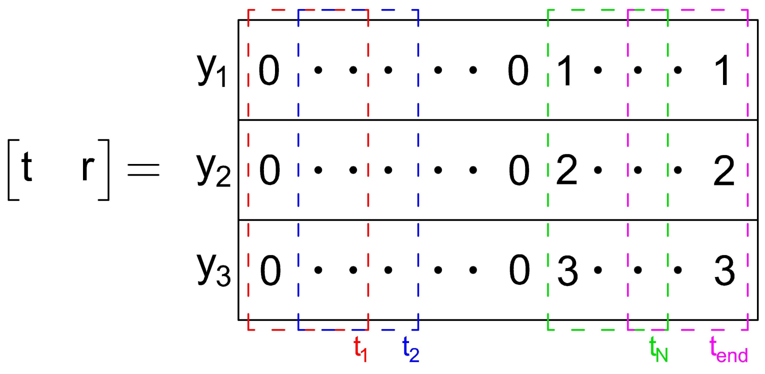
$\begin{bmatrix} t & r \end{bmatrix} =$
y_1 0 · · · · · 0 1 · · · 1
y_2 0 · · · · · 0 2 · · · 2
y_3 0 · · · · · 0 3 · · · 3
t_1
t_2
t_N
t_{end}

Figure 14: Specification of the reference signal over the entire simulation time in the case where $p = 3$.

At each time instant, N_p predicted values of the reference signal are applied to the system. A timeseries object is created in Matlab and with the usage of the *From Workspace* block, the reference signal can be sent to Simulink. There are two methods that can be used for the specification of the reference signal.

- Without preview: With this method no preview of the reference signal is given.

$$\begin{bmatrix} t & r \end{bmatrix} = \begin{bmatrix} t1 & 000 & 000 & 000 & \dots & 000 \\ t2 & 000 & 000 & 000 & \dots & 000 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{N-1} & 000 & 000 & 000 & \dots & 000 \\ t_N & 123 & 123 & 123 & \dots & 123 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{end} & 123 & 123 & 123 & \dots & 123 \end{bmatrix} \tag{14}$$

  Whenever the values in the first column change, the entire row will change as well. With this method, the controller reacts later, because the future values of the reference signals are unknown.

- With preview: With this method, a preview of the reference signal is given.

$$\begin{bmatrix} t & r \end{bmatrix} = \begin{bmatrix} t1 & 000 & 000 & 000 & \dots & 000 & 000 \\ t2 & 000 & 000 & 000 & \dots & 000 & 000 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{N-1} & 000 & 000 & 000 & \dots & 000 & 000 \\ t_N & 000 & 000 & 000 & \dots & 000 & 123 \\ t_{N+1} & 000 & 000 & 000 & \dots & 123 & 123 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{end} & 123 & 123 & 123 & \dots & 123 & 123 \end{bmatrix} \tag{15}$$

  Whenever new information comes in, the values in the last column change and all the others are shifted to the left by one. The values stored in the first column are simply being discarded. With this method, the controller reacts earlier, because the future N_p values of the reference signal are known.

With the usage of the second method, a better control performance can be guaranteed, because the controller reacts earlier whenever the reference signal changes. However, since the main topic of this work is to investigate how the network affects the control performance, the first method is used instead.

3.1.4 Constraints

As mentioned before, one of the biggest advantages of the MPC is that the constraints on the state, input and output of the system can easily be imposed [17].

- Constraints on the input:

$$u_{min} \leq u_k \leq u_{max} \quad \forall\, k \tag{16}$$

  Equation (16) is further modified to

$$\begin{aligned} & u_{min} \leq u_k = u_{k-1} + \Delta\hat{u}_k \leq u_{max} \\ & u_{min} \leq u_{k+1} = u_k + \Delta\hat{u}_{k+1} = u_{k-1} + \Delta\hat{u}_k + \Delta\hat{u}_{k+1} \leq u_{max} \\ & \qquad \vdots \\ & u_{min} \leq u_{k+N_c-1} = u_{k-1} + \Delta\hat{u}_k + \Delta\hat{u}_{k+1} + \ldots + \Delta\hat{u}_{k+N_c-1} \leq u_{max} \end{aligned}$$

The constraints on the input of the system can be specified in matrix notation as

$$\bar{u}_{min} \leq Lu_{k-1} + M\Delta\bar{u}_k \leq \bar{u}_{max}, \tag{17}$$

where $\bar{u}_{min} \in \mathbb{R}^{mN_c}$, $\bar{u}_{max} \in \mathbb{R}^{mN_c}$, $L \in \mathbb{R}^{mN_c \times m}$, $M \in \mathbb{R}^{mN_c \times mN_c}$ and $\Delta\bar{u}_k \in \mathbb{R}^{mN_c}$ are given as

$$\bar{u}_{min} = \begin{bmatrix} u_{min} \\ \vdots \\ u_{min} \end{bmatrix}, \qquad L = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}, \qquad M = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I \end{bmatrix},$$
$$\bar{u}_{max} = \begin{bmatrix} u_{max} \\ \vdots \\ u_{max} \end{bmatrix} \quad \text{and} \quad \Delta\bar{u}_k = \begin{bmatrix} \Delta\hat{u}_k \\ \vdots \\ \Delta\hat{u}_{k+N_c-1} \end{bmatrix}.$$

- Constraints on the change of the input:

$$\Delta u_{min} \leq \Delta u_k \leq \Delta u_{max} \quad \forall\, k \tag{18}$$

Equation (18) is further modified to

$$\begin{aligned} \Delta u_{min} &\leq \Delta\hat{u}_k \leq \Delta u_{max} \\ \Delta u_{min} &\leq \Delta\hat{u}_{k+1} \leq \Delta u_{max} \\ &\vdots \\ \Delta u_{min} &\leq \Delta\hat{u}_{k+N_c-1} \leq \Delta u_{max} \end{aligned}$$

The constraints on the change of the input can be specified in matrix notation as

$$\Delta\bar{u}_{min} \leq \Delta\bar{u}_k \leq \Delta\bar{u}_{max}, \tag{19}$$

where $\Delta\bar{u}_{min} \in \mathbb{R}^{mN_c}$ and $\Delta\bar{u}_{max} \in \mathbb{R}^{mN_c}$ are given as

$$\Delta\bar{u}_{min} = \begin{bmatrix} \Delta u_{min} \\ \vdots \\ \Delta u_{min} \end{bmatrix} \quad \text{and} \quad \Delta\bar{u}_{max} = \begin{bmatrix} \Delta u_{max} \\ \vdots \\ \Delta u_{max} \end{bmatrix}. \tag{20}$$

- Constraints on the states:

$$x_{min} \leq x_k \leq x_{max} \quad \forall\, k \tag{21}$$

Equation (21) is further modified to

$$\begin{aligned} x_{min} &\leq x_k \leq x_{max} \\ x_{min} &\leq \hat{x}_{k+1} \leq x_{max} \\ &\vdots \\ x_{min} &\leq \hat{x}_{k+N_p} \leq x_{max} \end{aligned}$$

The constraints on the states of the system can be specified in matrix notation as

$$\bar{x}_{min} \leq \bar{x}_{k+1} \leq \bar{x}_{max}. \tag{22}$$

By inserting equation (9) into (22) we get

$$\bar{x}_{min} \leq \bar{g}_{x,k} + H_x \Delta\bar{u}_k \leq \bar{x}_{max}, \tag{23}$$

where $\bar{x}_{min} \in \mathbb{R}^{pN_p}$, $\bar{x}_{max} \in \mathbb{R}^{pN_p}$ and $\bar{x}_{k+1} \in \mathbb{R}^{pN_p}$ are given as

$$\bar{x}_{min} = \begin{bmatrix} x_{min} \\ \vdots \\ x_{min} \end{bmatrix}, \qquad \bar{x}_{max} = \begin{bmatrix} x_{max} \\ \vdots \\ x_{max} \end{bmatrix} \quad \text{and} \quad \bar{x}_{k+1} = \begin{bmatrix} \hat{x}_{k+1} \\ \vdots \\ \hat{x}_{k+N_p} \end{bmatrix}.$$

- Constraints on the output:

$$y_{min} \leq y_k \leq y_{max} \quad \forall\, k \tag{24}$$

Equation (24) is further modified to

$$\begin{aligned} y_{min} \leq \hat{y}_{k+1} &\leq y_{max} \\ &\vdots \\ y_{min} \leq \hat{y}_{k+N_p} &\leq y_{max} \end{aligned}$$

The constraints on the output of the system can be specified in matrix notation as

$$\bar{y}_{min} \leq \bar{y}_{k+1} \leq \bar{y}_{max}. \tag{25}$$

By inserting equation (8) into (25) we get

$$\bar{y}_{min} \leq \bar{g}_k + H \Delta\bar{u}_k \leq \bar{y}_{max}, \tag{26}$$

where $\bar{y}_{min} \in \mathbb{R}^{pN_p}$, $\bar{y}_{max} \in \mathbb{R}^{pN_p}$ and $\bar{y}_{k+1} \in \mathbb{R}^{pN_p}$ are given as

$$\bar{y}_{min} = \begin{bmatrix} y_{min} \\ \vdots \\ y_{min} \end{bmatrix}, \qquad \bar{y}_{max} = \begin{bmatrix} y_{max} \\ \vdots \\ y_{max} \end{bmatrix} \quad \text{and} \quad \bar{y}_{k+1} = \begin{bmatrix} \hat{y}_{k+1} \\ \vdots \\ \hat{y}_{k+N_p} \end{bmatrix}.$$

The goal is to combine all the computed constraints into one equation. It results

$$W \Delta\bar{u}_k \leq \bar{w}, \quad \ldots \text{linear inequality constraints} \tag{27}$$

where

$$W = \begin{bmatrix} -M \\ M \\ -I \\ I \\ -H_x \\ H_x \\ -H \\ H \end{bmatrix} \qquad \text{and} \qquad \bar{w} = \begin{bmatrix} -\bar{u}_{min} + Lu_{k-1} \\ \bar{u}_{max} - Lu_{k-1} \\ -\Delta\bar{u}_{min} \\ \Delta\bar{u}_{max} \\ -\bar{x}_{min} + \bar{g}_{x,k} \\ \bar{x}_{max} - \bar{g}_{x,k} \\ -\bar{y}_{min} + \bar{g}_k \\ \bar{y}_{max} - \bar{g}_k \end{bmatrix}$$

are the combination of the constraints represented in matrix notation. The optimization problem results to

$$\begin{aligned} \min_{\Delta\bar{u}_k \in \mathbb{R}^{mN_c}} \quad & \Delta\bar{u}_k^T (H^T Q H + R)\Delta\bar{u}_k + 2\Delta\bar{u}_k^T H^T Q \bar{e}_k \\ \text{subject to} \quad & W\Delta\bar{u}_k \leq \bar{w} \end{aligned} \tag{28}$$

with

$$Q \geq 0, \qquad R \geq 0 \qquad \text{and} \qquad H^T Q H + R > 0.$$

3.1.5 Soft Constraints

There are situations where no feasible solution of the optimization problem can be found, e.g., due to model uncertainties, disturbances, estimator error of the observer, and so on [17]. If this happens, the controller stops the computations. A possible remedy of this problem would be to use the control input from the previous step or to apply the predicted control input from the previous iteration. Those solutions are quiet simple but they can lead to further issues. The best remedy would be to use the so-called *soft constraints*. Soft constraints allow to violate the constraints, but not by much and only if absolutely necessary. An additional optimization variable $\bar{\varepsilon}$ is introduced and the optimization problem results to

$$\begin{aligned} \min_{\substack{\Delta\bar{u}_k \in \mathbb{R}^{mN_c} \\ \bar{\varepsilon} \in \mathbb{R}^{length(\bar{w})}}} \quad & \Delta\bar{u}_k^T (H^T Q H + R)\Delta\bar{u}_k + 2\Delta\bar{u}_k^T H^T Q \bar{e}_k + \rho\|\bar{\varepsilon}\|^2 \\ \text{subject to} \quad & W\Delta\bar{u}_k \leq \bar{w} + \bar{\varepsilon} \\ & \bar{\varepsilon} \geq 0 \end{aligned} \tag{29}$$

where ρ is a tuning parameter. An alternative way to choose the optimization problem would be

$$\begin{aligned} \min_{\substack{\Delta\bar{u}_k \in \mathbb{R}^{mN_c} \\ \bar{\varepsilon} \in \mathbb{R}}} \quad & \Delta\bar{u}_k^T (H^T Q H + R)\Delta\bar{u}_k + 2\Delta\bar{u}_k^T H^T Q \bar{e}_k + \rho\bar{\varepsilon} \\ \text{subject to} \quad & W\Delta\bar{u}_k \leq \bar{w} + V\bar{\varepsilon} \\ & \bar{\varepsilon} \geq 0, \end{aligned} \tag{30}$$

where ρ is again a tuning parameter. The array V is given as

$$V = \begin{bmatrix} \alpha \\ \alpha \\ \vdots \\ \alpha \end{bmatrix} \qquad \text{or} \qquad V = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix}, \tag{31}$$

where N is the length of the vector $\bar{w}$.

4 Model Predictive Control for Networked Control Systems

As it has already been mentioned in section 3, for the classical MPC only the first optimal control move is implemented as the current control law. In the case where a MPC is used for NCSs, not just the first value of the predicted sequence, but the entire predicted control sequence is sent to the actuator side. The basic structure of the MPC approach for NCSs is illustrated in figure 15.

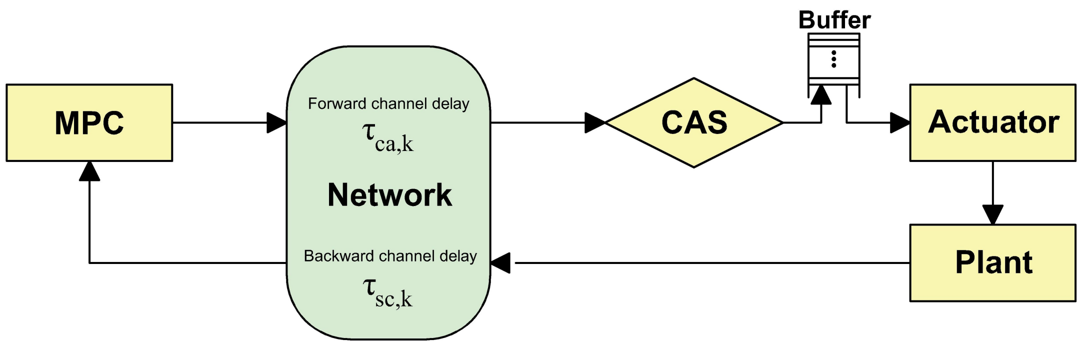

Figure 15: Basic structure of the MPC approach for NCSs.

Note that with this approach the controller calculating delay $\tau_{c,k}$ is assumed to be zero. Later on in this work, an additional approach, where the controller calculating delay is taken into account, is presented as well. The RTD is given as

$$\tau_k = \tau_{sc,k} + \tau_{ca,k}. \tag{32}$$

The working concept of this approach can be explained as follows: the data from the sensors is stored in a packet and sent to the controller side via the shared communication network. The information received by the controller gets delayed by the sensor-to-controller delay $\tau_{sc,k}$. After the MPC finishes the computations, the predicted control sequence is stored in a packet and sent via the communication network to the actuator side. Again, the information reaching the actuator side gets delayed by the controller-to-actuator delay $\tau_{ca,k}$. With the knowledge of both network-induced delays, the Control Action Selector (CAS) has to select the correct control input, which has been predicted by the controller. Whenever a packet reaches the CAS, the most recent packet is stored in a buffer. If two or more packets arrive either at the controller or at the actuator side at the same time due to the random time-delays, the most recent one is used.

In order to understand the basic concept of the MPC approach for NCSs, an example is presented.

Example: Let's assume that the data from the sensors is stored in a packet and send to the controller side via a shared communication media.

x_1
x_2
$\vdots$
x_n

The packet gets delayed by the backward channel delay $\tau_{sc,k}$. After the MPC finishes the computations, the entire predicted control sequence is stored in a packet and sent through the communication network to the actuator side.

$\hat{u}_k$
$\hat{u}_{k+1}$
$\hat{u}_{k+2}$
$\vdots$
$\hat{u}_{k+N_c-1}$

The packet gets delayed by the forward channel delay $\tau_{ca,k}$. Whenever the packet reaches the CAS it will be stored in a buffer. The sampling period in this short example is assumed to be $h = 1s$ and the delays at time instant k (for this specific example) are given as

$$\tau_{sc,k} = 1 \qquad \tau_{ca,k} = 2 \qquad \Rightarrow \qquad \tau_k = \tau_{sc,k} + \tau_{ca,k} = 3.$$

The CAS selects the input of the plant as:

$\hat{u}_k$
$\hat{u}_{k+1}$
$\hat{u}_{k+2}$
$\hat{u}_{k+3}$
$\vdots$
$\hat{u}_{k+N_c-1}$

Let's assume that in the next time instant no packet reaches the actuator side. The last received packet, which has been stored in the buffer, is reused instead. The next predicted control input of the last used control sequence is chosen as the new plant input.

$\hat{u}_k$
$\hat{u}_{k+1}$
$\hat{u}_{k+2}$
$\hat{u}_{k+3}$
$\hat{u}_{k+4}$
$\vdots$
$\hat{u}_{k+N_c-1}$

The same will happen again as long as a new packet reaches the CAS or until the last predicted control input $\hat{u}_{k+N_c-1}$ is reached.

$\hat{u}_k$
$\hat{u}_{k+1}$
$\hat{u}_{k+2}$
$\vdots$
$\hat{u}_{k+N_c-1}$

In this short example it has been assumed that both delays are known by the CAS, without performing any computations. In reality this is obviously not the case.

At this point the question is how to identify the random time-delays in the forward and in the backward channel. As mentioned in section 2.5, the network-induced delays are assumed to be equal to an integer multiple of the sampling time h [16] and both delays have to be bounded such as

$$\begin{aligned} \tau_{sc,min} \leq \tau_{sc,k} \leq \tau_{sc,max} \quad &\Rightarrow \quad 1 \leq \tau_{sc,k} \leq N_c - 2 \\ \tau_{ca,min} \leq \tau_{ca,k} \leq \tau_{ca,max} \quad &\Rightarrow \quad 1 \leq \tau_{ca,k} \leq N_c - 2, \end{aligned} \tag{33}$$

where $\tau_{sc,k}$ and $\tau_{ca,k}$ are the integer multiples of the delays in the forward and backward channel at time instant k respectively. Since it's impossible to have a perfect data transmission over the network, the minimum value of both delays is assumed to be equal to the sampling period h. The sum of both delays $\tau_k = \tau_{sc,k} + \tau_{ca,k}$ (RTD) has also to be bounded such as

$$\tau_{k,min} \leq \tau_k \leq \tau_{k,max} \quad \Rightarrow \quad 2 \leq \tau_k \leq N_c - 1. \tag{34}$$

If equation (34) holds, then the CAS will always be able to find a suitable predicted control signal value for the plant.

Note that the maximum value of both network-induced delays is equal to $N_c - 2$, since the RTD isn't allowed to exceed the given upper bound. This can be explained as follows:

- If $\tau_{sc,k} = 1 \quad \Rightarrow \quad 1 \leq \tau_{ca,k} \leq N_c - 2 \quad \Rightarrow \quad 2 \leq \tau_k \leq N_c - 1 \quad \checkmark$
- If $\tau_{sc,k} = 2 \quad \Rightarrow \quad 1 \leq \tau_{ca,k} \leq N_c - 3 \quad \Rightarrow \quad 2 \leq \tau_k \leq N_c - 1 \quad \checkmark$

$$\vdots$$

- If $\tau_{sc,k} = N_c - 2 \quad \Rightarrow \quad 1 \leq \tau_{ca,k} \leq 1 \quad \Rightarrow \quad 2 \leq \tau_k \leq N_c - 1 \quad \checkmark$

With the usage of time stamps it's possible to identify the transmission delays in the forward and backward channel. The sampled data packets are labeled with a time stamp, which contains the information of the corresponding sampling time instant of the sampled data packet. There are two types of approaches which can be applied to the system: the forward and backward channel delays $\tau_{sc,k}$ and $\tau_{ca,k}$ can be identified separately (packet-based control with time synchronization) or simply the RTD τ_k has to be identified (packet-based control without time synchronization) [7].

4.1 Packet-based Control With Time Synchronization

For the design of the packet-based control with the time synchronization approach, the following assumptions are required.

Assumption 1. *The control components in the NCS are all time-synchronized and the data packets sent from both the sensor and the controller are time-stamped.*

Assumption 2. *The delays in the backward and forward channel are upper bounded with* $\tau_{sc,max}$ *and* $\tau_{ca,max}$ *respectively.*

The structure of the packet-based control approach with time synchronization is illustrated in figure 16 and the timeline of the packet-based approach is shown in figure 17.

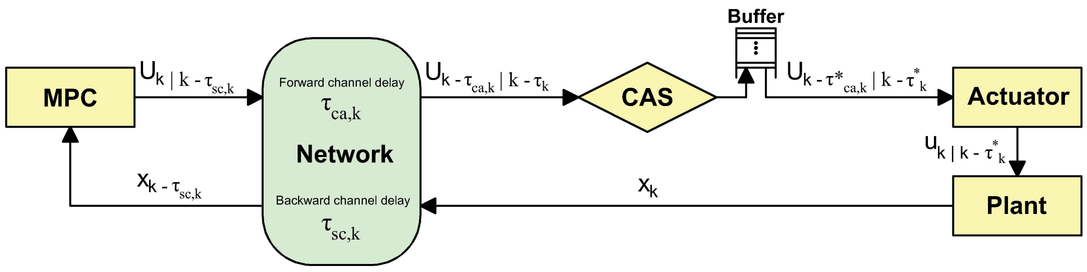

Figure 16: Structure of the packet-based control approach with time synchronization.

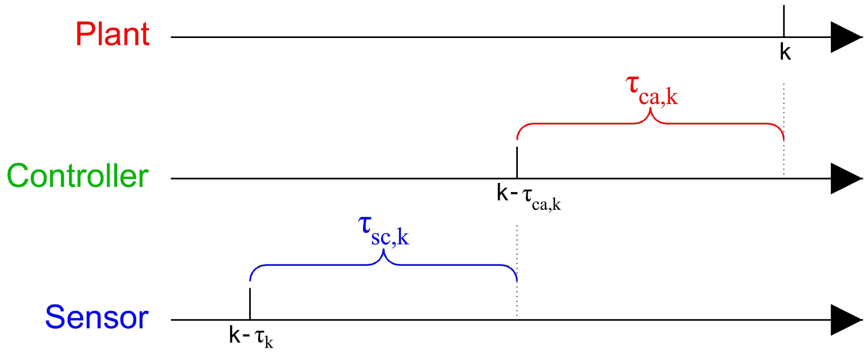

Figure 17: Timeline of the packet-based approach for NCSs.

The working principle of this approach can be explained as the follow: the state vector x_k is stored in a packet together with a time stamp, which contains the time instant of the sampled data packet.

x_1
x_2
$\vdots$
x_n
$t_{stamp,back}$

The packet is transmitted to the controller side and delayed by the sensor-to-controller delay $\tau_{sc,k}$. When the packet reaches the controller, the backward channel delay can be computed as

$$\tau_{sc,k} = \frac{t_{now} - t_{stamp,back}}{h}. \tag{35}$$

Since a sample-and-hold element is used, it's assumed that the measured time t_{now} and the stored time stamp $t_{stamp,back}$ are equal to an integer multiple of the sampling period.
After the MPC finishes the computations, the controller determines a sequence of forward control actions

$$U_{k|k-\tau_{sc,k}} = \begin{bmatrix} \hat{u}_{k|k-\tau_{sc,k}} & \dots & \hat{u}_{k+\tau_{ca,k}|k-\tau_{sc,k}} \end{bmatrix}^T, \tag{36}$$

where $\hat{u}_{k+i|k-\tau_{sc,k}}$ for $i = 0, 1, ..., \tau_{ca,k}$ are the forward control action predictions based on information up to time $k - \tau_{sc,k}$ [13]. In other words, the first $\tau_{sc,k} + 1$ values of the entire resulting predicted control sequence are simply thrown away and the rest is stored in a packet together with a time stamp, which again contains the time instant of the sampled data packet [7].

$\hat{u}_{k+\tau_{sc,k}+1}$
$\hat{u}_{k+\tau_{sc,k}+2}$
$\vdots$
$\hat{u}_{k+N_c-1}$
$t_{stamp,for}$

The packet is transmitted to the CAS on the actuator side and delayed by the controller-to-actuator delay $\tau_{ca,k}$. When the packet reaches the CAS, the forward channel delay can be computed as

$$\tau_{ca,k} = \frac{t_{now} - t_{stamp,for}}{h}. \tag{37}$$

The CAS compares its time stamp with the one already stored in the buffer and only the one with the latest time stamp is saved. With this comparison, data packet disorder can be prevented. Let's denote the forward control sequence already in the CAS and the one just arrived by $U_{k_1-\tau_{ca,k_1}|k_1-\tau_{k_1}}$ and $U_{k_2-\tau_{ca,k_2}|k_2-\tau_{k_2}}$ respectively, then the chosen sequence is determined by the following comparison [7]:

$$U_{k-\tau^*_{ca,k}|k-\tau^*_k} = \begin{cases} U_{k_2-\tau_{ca,k_2}|k_2-\tau_{k_2}} & \text{if} \quad k_1 - \tau_{k_1} < k_2 - \tau_{k_2} \\ U_{k_1-\tau_{ca,k_1}|k_1-\tau_{k_1}} & \text{otherwise} \end{cases} \tag{38}$$

Finally, the CAS has to select the correct value from the predicted control sequence as the new input of the plant.

$\hat{u}_{k+\tau_{sc,k}+1}$
$\hat{u}_{k+\tau_{sc,k}+2}$
$\vdots$
$\hat{u}_{k+\tau_{ca,k}}$
$\vdots$
$\hat{u}_{k+N_c-1}$
$t_{stamp,for}$

The packet containing the state vector is sent at each time instant, however, the packet containing the predicted control sequence is only sent whenever the MPC is active. This approach can be summarized as follows:

Algorithm 3: Packet-based control approach with time synchronization

```
if the controller receives the delayed state data $x_{k-\tau_{sc,k}}$ at time instant $k$, the
 controller then
    Calculates the current backward channel delay $\tau_{sc,k}$;
    Computes the control sequence as in (36);
    Packs $U_{k|k-\tau_{sc,k}}$ and sends it to the CAS in one data packet with the
      corresponding time stamp;
else
    Let $k = k + 1$ and wait for the next time instant;
end
if a data packet arrives at the CAS then
    The forward channel delay $\tau_{ca,k}$ is calculated;
    The CAS updates the control sequence stored in the buffer;
    The new control action is picked out by the CAS and applied to the plant;
end
```

With the usage of the packet-based control approach with time synchronization, the risk of network congestion decreases, since the size of the transmitted packet on the controller side is reduced. However, very often it's only required to identify the RTD τ_k.

4.2 Packet-based Control Without Time Synchronization

The second approach that can be applied to the system is the so-called packet-based control approach without time synchronization. In practice, it's often not required to identify the forward and backward channel delays separately since it's normally the RTD that affects the system performance. The following assumptions are required for this approach.

Assumption 3. *The RTD τ_k is upper bounded by τ_{max}.*

Assumption 4. *The data packets sent from the sensor are time-stamped.*

The structure of the packet-based control approach without time synchronization is illustrated in figure 18.

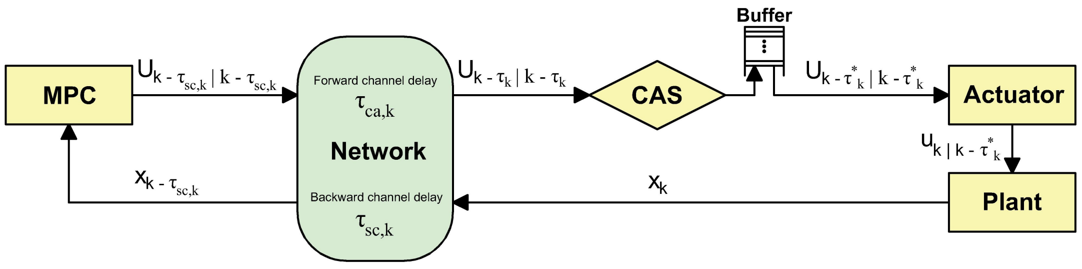

Figure 18: Structure of the packet-based control approach without time synchronization.

The state vector x_k is at each time instant stored in a packet together with a time stamp, which contains the time instant of the sampled data packet.

x_1
x_2
$\vdots$
x_n
$t_{stamp,back}$

The packet is transmitted to the controller side and delayed by the sensor-to-controller delay $\tau_{sc,k}$. When the packet reaches the controller, the backward channel delay doesn't need to be computed.

After the MPC finishes the computations, the controller again determines a sequence of forward control actions such as

$$U_{k-\tau_{sc,k}|k-\tau_{sc,k}} = \begin{bmatrix} \hat{u}_{k-\tau_{sc,k}|k-\tau_{sc,k}} & \cdots & \hat{u}_{k-\tau_{sc,k}+\tau_k|k-\tau_{sc,k}} \end{bmatrix}^T . \tag{39}$$

It has been assumed that the maximum possible value for the RTD is $\tau_k = N_c - 1$. It is noticed that in such a case the backward channel delay $\tau_{sc,k}$ is not required for the controller, since the controller simply produces $N_c - 1$ step forward control actions whenever a data packet containing the feedback information arrives. This relaxation implies that the time synchronization between the controller and the actuator (plant) isn't required anymore.

The entire predicted control sequence and the time stamp of the packet containing the sensing data ($t_{stamp,back}$) are stored in a packet, but this time no time stamp containing the time instant of the sampled data packet sent by the controller ($t_{stamp,for}$) has to be attached.

$\hat{u}_k$
$\hat{u}_{k+1}$
$\vdots$
$\hat{u}_{k+N_c-1}$
$t_{stamp,back}$

The packet is transmitted to the CAS on the actuator side and delayed by the controller-to-actuator delay $\tau_{ca,k}$. When the packet reaches the CAS, the RTD can be computed as

$$\tau_k = \frac{t_{now} - t_{stamp,back}}{h}. \tag{40}$$

The CAS compares its time stamp with the one already stored in the buffer and only the one with the latest time stamp is saved. With this comparison, data packet disorder can be prevented. Finally, the CAS has to select the correct value from the predicted control sequence as the new input of the plant.

$\hat{u}_k$
$\hat{u}_{k+1}$
$\vdots$
$\hat{u}_{k+\tau_k}$
$\vdots$
$\hat{u}_{k+N_c-1}$
$t_{stamp,back}$

The packet containing the sensing data is sent at each time instant, however, the packet sent by the controller is only sent whenever the MPC is active. Algorithm 4 shows a possible implementation of this approach.

Algorithm 4: Packet-based control approach without time synchronization

```
if the controller receives the delayed state data $x_{k-\tau_{sc,k}}$ at time instant $k$, the controller then
    Calculates the control sequence as in (39);
    Packs $U_{k-\tau_{sc,k}|k-\tau_{sc,k}}$ and sends it to the CAS in one data packet;
else
    Let $k = k + 1$ and wait for the next time instant;
end
if a data packet arrives at the CAS then
    The RTD $\tau_k$ is calculated;
    The CAS updates the control sequence stored in the buffer;
    The new control action is picked out by the CAS and applied to the plant;
end
```

The only difference between this approach and the approach presented in section 4.1 is the size of the packet sent by the controller. For the packet-based control without time synchronization approach, the packet size is always the same (maximum), since the controller simply sends all $N_c - 1$ predicted control steps to the actuator side. The risk of network congestion increases, since more information is sent through the network.

4.3 Consideration of the Controller Calculating Delay

As already mentioned in section 2.4.1, very often the controller calculating delay $\tau_{c,k}$ is assumed to be zero, since its value is smaller compared to the other two network-induced delays. This assumption holds for slow systems with a large sampling period. However, neglecting the controller calculating delay can become a problem for fast systems with a small sampling period. It can happen that $\tau_{c,k}$ is larger than the sampling period h. In this case the controller calculating delay can become problematic and it can significantly reduce the control performance. The control approach presented in section 4.1 is further modified such that the controller calculating delay $\tau_{c,k}$ can be compensated. Note that for the packet-based control approach without time synchronization the controller calculating delay is already taken into account, since the time interval between the time instant when the feedback information is sent to the controller side and the time instant when the control signals reach the CAS is measured.

Since $\tau_{c,k}$ is not equal to an integer multiple of the sampling time h, a new delay $\tau'_{c,k}$ is introduced [24].

$$\tau'_{c,k} = \tau_{c,k} + \tau_{wait} \tag{41}$$

Figure 19 shows the timeline of the controller calculating delay in a typical NCS.

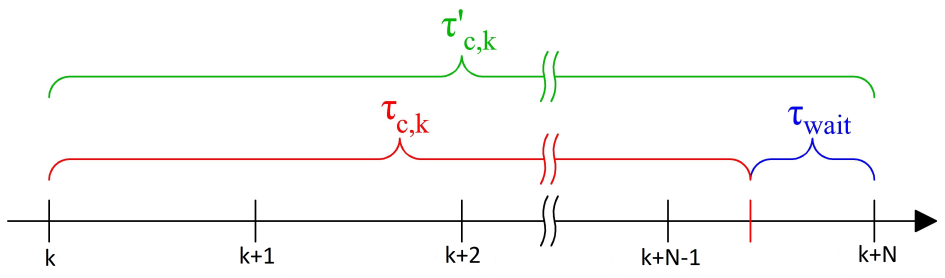

Figure 19: Timeline of the controller calculating delay.

The measured RTD results to

$$\tau_k = \tau_{sc,k} + \tau_{ca,k} + \tau'_{c,k}. \tag{42}$$

For the modified packet-based control approach with time synchronization the controller calculating delay has to be identified. The structure and the timeline of this approach are shown in figure 20 and 21 respectively.

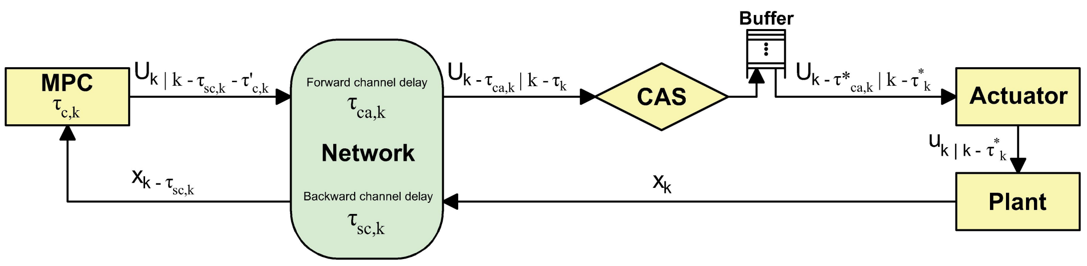

Figure 20: Structure of the modified packet-based control approach with time synchronization.

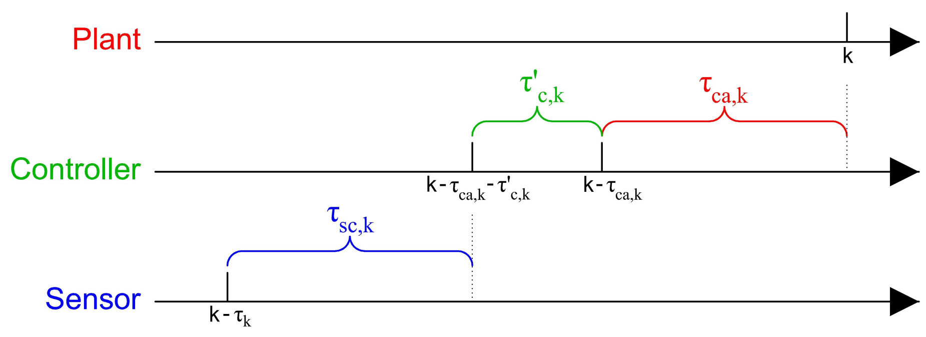

Figure 21: Timeline of the modified packet-based control approach with time synchronization.

The working principle of this approach can be explained as follows. The state vector x_k is stored in a packet together with a time stamp, which contains the time instant of the sampled data packet.

x_1
x_2
$\vdots$
x_n
$t_{stamp,back}$

The packet is transmitted to the controller side and delayed by the sensor-to-controller delay $\tau_{sc,k}$. When the packet reaches the controller, the backward channel delay can be computed as

$$\tau_{sc,k} = \frac{t_{now} - t_{stamp,back}}{h} \tag{43}$$

and the measured time instant is stored in a time stamp such as

$$t_{stamp,start} = t_{now}. \tag{44}$$

After the MPC finishes the computations, the controller calculating delay $\tau'_{c,k}$ can be computed as

$$\tau'_{c,k} = \frac{t_{now} - t_{stamp,start}}{h}. \tag{45}$$

It's assumed that if another packet reaches the controller side while the MPC is still running, the incoming packet is simply discarded. Another possibility would be that whenever a packet reaches the controller while this last one is still running, the computation gets interrupted. The main idea behind this approach is that the controller always uses the most recent information, but this can also lead to a scenario, where the controller is not able to finish the computations, since it gets interrupted all the time. Because of this, it cannot be guaranteed that enough information reaches the CAS and the risk of instability increases drastically.

The predicted sequence of forward control actions results to

$$U_{k|k-\tau_{sc,k}-\tau'_{c,k}} = \begin{bmatrix} \hat{u}_{k|k-\tau_{sc,k}-\tau'_{c,k}} & \cdots & \hat{u}_{k+\tau_{ca,k}|k-\tau_{sc,k}-\tau'_{c,k}} \end{bmatrix}^T, \tag{46}$$

where $\hat{u}_{k+i|k-\tau_{sc,k}-\tau'_{c,k}}$ for $i = 0, 1, ..., \tau_{ca,k}$ are the forward control action predictions based on information up to time $k - \tau_{sc,k} - \tau'_{c,k}$. In other words, the first $\tau_{sc,k} + \tau'_{c,k} + 1$ values of the entire resulting predicted control sequence are simply thrown away and the rest is stored in a packet together with a time stamp, which again contains the time instant of the sampled data packet.

$\hat{u}_{k+\tau_{sc,k}+\tau'_{c,k}+1}$
$\hat{u}_{k+\tau_{sc,k}+\tau'_{c,k}+2}$
$\vdots$
$\hat{u}_{k+N_c-1}$
$t_{stamp,for}$

The packet is transmitted to the CAS on the actuator side and delayed by the controller-to-actuator delay $\tau_{ca,k}$. When the packet reaches the CAS, the forward channel delay can be computed as

$$\tau_{ca,k} = \frac{t_{now} - t_{stamp,for}}{h}. \tag{47}$$

The CAS compares its time stamp with the one already stored in the buffer and only the one with the latest time stamp is saved. With this comparison, data packet disorder can be prevented. Finally, the CAS has to select the correct value from the predicted control sequence as the new input of the plant.

$\hat{u}_{k+\tau_{sc,k}+\tau'_{c,k}+1}$
$\hat{u}_{k+\tau_{sc,k}+\tau'_{c,k}+2}$
$\vdots$
$\hat{u}_{k+\tau_{ca,k}}$
$\vdots$
$\hat{u}_{k+N_c-1}$
$t_{stamp,for}$

This approach is suited for fast systems with small sampling periods. However, this approach is not used very often. Measuring time delays using the time stamp technique takes significant time and a even higher time-delay can be caused [22]. For slow systems this restriction is not relevant, since the time stamp-induced delays are very small compared to the measured network-induced delays. Another restriction is the network congestion. The controller calculating delay is usually very small and NCSs are not suited for systems with such a small sampling period (see section 2.4.5). Furthermore, this approach is quiet complicated and complex to implement.

In algorithm 5 a possible implementation of this approach is presented.

Algorithm 5: Modified packet-based control approach with time synchronization

```
if the controller receives the delayed state data $x_{k-\tau_{sc,k}}$ at time instant $k$, the controller then
    Calculates the current backward channel delay $\tau_{sc,k}$;
    Computes the control sequence as in (46);
    Calculates the controller calculating delay $\tau'_{c,k}$;
    Packs $U_{k|k-\tau_{sc,k}-\tau'_{c,k}}$ and sends it to the CAS in one data packet with the corresponding time stamp;
else
    Let $k = k + 1$ and wait for the next time instant;
end
if a data packet arrives at the CAS then
    The forward channel delay $\tau_{ca,k}$ is calculated;
    The CAS updates the control sequence stored in the buffer;
    The new control action is picked out by the CAS and applied to the plant;
end
```

4.4 Compensation of Data Packet Dropouts

Some packets not only suffer transmission delays but also, even worse, can be lost during the transmission. In fact, there is a challenging issue in NCSs: how to actively compensate for communication delays and overcome data packet dropouts without degrading the control performance. In general, time-delays and packet dropouts are always linked to each other [22]. Most of the control systems are delay-sensitive. In other words, for most of the control systems, out-of-date information is useless and only real-time information should be used. Therefore, a packet with relatively large delay should be classified as a junk information, which is better to be dismissed. In such a scenario, a packet dropout is happened proactively in order to improve the control performance. It's natural to assume that only a finite number of consecutive data dropouts can be tolerated in order to avoid the NCS to become an open loop [2].

In this thesis it's assumed that the network-induced delays and the data packet dropouts are independent from each other. This assumption leads to a model which is obviously defective in terms of authenticity and reliability. In [22], the relation between time-delays and data packet dropouts is taken into consideration and a possible implementation based on the Markov chain approach is presented. With the Traditional Markov Chain approach, the relation between time-delays and data packet dropouts is not considered, since 3 independent Markov chains are used. With the so-called Nested Markov Chain approach, the relation between time-delays and data packet dropouts is taken into consideration.

In order to compensate for data packet dropouts the following assumption is required.

Assumption 5. *The sum of the maximum backward and forward channel delay and the maximum number of continuous data packet dropouts is upper bounded by $\bar{\tau}_{sc}$ ($\bar{\tau}_{ca}$ respectively)*

$$\bar{\tau}_{sc} = \tau_{sc,max} + h \cdot n_{d,sc} \qquad \bar{\tau}_{ca} = \tau_{ca,max} + h \cdot n_{d,ca} \tag{48}$$

where $n_{d,sc}$ and $n_{d,ca}$ are the upper bounds of the number of consecutive data packet dropouts in both channels.

In NCSs, there are three methods which are often used in order to compensate the random data packet dropouts [8].

- **Method 1**: In the case where a data dropout occurs, the control input is set to zero.
- **Method 2**: In the case where a data dropout occurs, the control input keeps the previous value until new data arrives.
- **Method 3**: In the case where a data dropout occurs, the control prediction approach presented in section 4 is used.

5 Two Tank System

In this section, a simulation example is given to show the feasibility and efficiency of the proposed methods. Given is a system consisting of two tanks and a reservoir, as shown in figure 22. With the usage of two pumps, water can be transferred into the tanks and the goal is to control the water-level of the bottom tank.

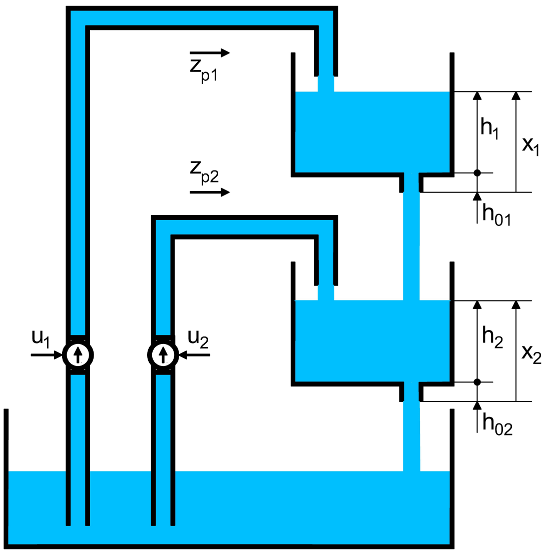

Figure 22: Structure of the two tank system.

5.1 Mathematical Model

The mathematical model for the two tank system is given in form of the following nonlinear differential equations

$$\begin{aligned}\frac{dx_1}{dt} &= f_1(x,u) = -k_1\sqrt{x_1} + z_{p,1}\\ \frac{dx_2}{dt} &= f_2(x,u) = -k_2\sqrt{x_2} + k_1\sqrt{x_1} + z_{p,2}\\ y &= g(x,u) = h_2\\ h_1 &= x_1 - h_{01}\\ h_2 &= x_2 - h_{02}\end{aligned} \tag{49}$$

where the parameters k_1 and k_2 are constant. The state variables x_1 and x_2 are the levels in the top and bottom tank respectively, which are defined as the sum of the length of the drains h_{01}, h_{02} and the water levels h_1, h_2, i.e. $x_i = h_i + h_{0i}$ $(i = 1, 2)$. The output y of the system is defined as the water-level of the bottom tank. The inflow due to pump 1 is symbolized by $z_{p,1}$ and the inflow due to pump 2 by $z_{p,2}$. In this thesis the second pump is not used and the voltage u_2 is set to zero. The nonlinear characteristics of both pumps are given as

$$z_{p,i}(u_i) = \begin{cases}\alpha_i + \sqrt{\beta_i + \gamma_i u_i} & \text{for} \quad \beta_i + \gamma_i u_i > 0\\ 0 & \text{otherwise}\end{cases}. \tag{50}$$

The parameters of the two tank model are given in table 2.

Parameter	**Tank 1**	**Tank 2**
$k_i[cm^{\frac{1}{2}}s^{-1}]$	0.391	0.386
$h_{0i}[cm]$	10.17	10.53
$\alpha_i[cms^{-1}]$	0.055	0.059
$\beta_i[cm^2s^{-2}]$	-3.077	-3.246
$\gamma_i[cm^2s^{-2}V^{-1}]$	3.551	3.61

Table 2: Parameters of the two tank system.

5.1.1 Linearization of the Model

The nonlinear model is linearized in the equilibrium $y_E = 15cm$. It results

$$\begin{aligned} 0 &= f_1(x_E, u_E) = -k_1\sqrt{x_{1E}} + z_{p,1E} \\ 0 &= f_2(x_E, u_E) = -k_2\sqrt{x_{2E}} + k_1\sqrt{x_{1E}} + z_{p,2E} \\ y_E &= g(x_E, u_E) = 15cm \end{aligned} \tag{51}$$

with the equilibrium point

$$x_E = \begin{bmatrix} x_{1,E} \\ x_{2,E} \end{bmatrix} = \begin{bmatrix} \frac{k_2^2 x_{2,E}}{k_1^2} \\ y_E + h_{02} \end{bmatrix}. \tag{52}$$

With the usage of the transformation

$$\begin{aligned} \Delta x &:= x - x_E \quad &\rightarrow \quad x = x_E + \Delta x \\ \Delta u &:= u - u_E \quad &\rightarrow \quad u = u_E + \Delta u \end{aligned} \tag{53}$$

it's possible to compute the nonlinear functions $f(x,u)$ and $h(x,u)$ as a Taylor series for the equilibrium point (x_E, u_E) as the following:

$$\begin{aligned} f_1(x,u) &= \underbrace{f_1(x,u)\Big|_{x_E,u_E}}_{=0} + \frac{\partial f_1(x,u)}{\partial x}\Big|_{x_E,u_E}\Delta x + \frac{\partial f_1(x,u)}{\partial u}\Big|_{x_E,u_E}\Delta u + \cdots \\ f_2(x,u) &= \underbrace{f_2(x,u)\Big|_{x_E,u_E}}_{=0} + \frac{\partial f_2(x,u)}{\partial x}\Big|_{x_E,u_E}\Delta x + \frac{\partial f_2(x,u)}{\partial u}\Big|_{x_E,u_E}\Delta u + \cdots \\ g(x,u) &= \underbrace{g(x,u)\Big|_{x_E,u_E}}_{=y_E} + \frac{\partial g(x,u)}{\partial x}\Big|_{x_E,u_E}\Delta x + \frac{\partial g(x,u)}{\partial u}\Big|_{x_E,u_E}\Delta u + \cdots \end{aligned} \tag{54}$$

Only the linear terms are used, therefore the terms of higher order can be neglected. With the definition

$$\Delta\dot{x} := \frac{d}{dt}(\Delta x) \tag{55}$$

one gets

$$\begin{aligned} \dot{x} = \Delta\dot{x} &\approx \frac{\partial f(x,u)}{\partial x}\Big|_{x_E,u_E}\Delta x + \frac{\partial f(x,u)}{\partial u}\Big|_{x_E,u_E}\Delta u \\ y = y_E &+ \frac{\partial g(x,u)}{\partial x}\Big|_{x_E,u_E}\Delta x + \frac{\partial g(x,u)}{\partial u}\Big|_{x_E,u_E}\Delta u, \end{aligned} \tag{56}$$

with the abbreviations

$$\begin{aligned} A &:= \frac{\partial f(x,u)}{\partial x}\Big|_{x_E,u_E} & b &:= \frac{\partial f(x,u)}{\partial u}\Big|_{x_E,u_E} \\ c^T &:= \frac{\partial g(x,u)}{\partial x}\Big|_{x_E,u_E} & d &:= \frac{\partial g(x,u)}{\partial u}\Big|_{x_E,u_E} \end{aligned} \tag{57}$$

and

$$\Delta y := y - y_E. \tag{58}$$

The linear mathematical model can be described as

$$\begin{aligned} \Delta\dot{x} &= A\Delta x + b\Delta u \\ \Delta y &= c^T\Delta x + d\Delta u. \end{aligned} \tag{59}$$

For the computation of the dynamic matrix A, the so-called Jacobian matrix has to be used

$$\frac{\partial f(x,u)}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -\frac{k_1}{2}\frac{1}{\sqrt{x_1}} & 0 \\ \frac{k_1}{2}\frac{1}{\sqrt{x_1}} & -\frac{k_2}{2}\frac{1}{\sqrt{x_2}} \end{bmatrix}. \tag{60}$$

By inserting the equilibrium point (x_E, u_E), the system matrix A is

$$A = \frac{\partial f(x,u)}{\partial x}\Big|_{x_E,u_E} = \begin{bmatrix} -\frac{k_1}{2}\frac{1}{\sqrt{x_{1,E}}} & 0 \\ \frac{k_1}{2}\frac{1}{\sqrt{x_{1,E}}} & -\frac{k_2}{2}\frac{1}{\sqrt{x_{2,E}}} \end{bmatrix} = \begin{bmatrix} -\frac{k_1}{2}\frac{1}{\sqrt{\frac{k_2^2 x_{2,E}}{k_1^2}}} & 0 \\ \frac{k_1}{2}\frac{1}{\sqrt{\frac{k_2^2 x_{2,E}}{k_1^2}}} & -\frac{k_2}{2}\frac{1}{\sqrt{y_E + h_{02}}} \end{bmatrix}. \tag{61}$$

The same is done for the input vector b

$$b = \frac{\partial f(x,u)}{\partial u}\Big|_{x_E,u_E} = \begin{bmatrix} \frac{\gamma_1}{2}\frac{1}{\sqrt{\beta_1 + \gamma_1 u_{1,E}}} \\ 0 \end{bmatrix}, \tag{62}$$

the output vector c^T

$$c^T = \frac{\partial g(x,u)}{\partial x}\Big|_{x_E,u_E} = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{63}$$

and for the direct feedthrough d

$$d = \frac{\partial g(x,u)}{\partial u}\Big|_{x_E,u_E} = 0. \tag{64}$$

Hence, the linear mathematical model of the two tank system is given as

$$\begin{aligned} \Delta\dot{x} &= \begin{bmatrix} -\frac{k_1}{2}\frac{1}{\sqrt{\frac{k_2^2 x_{2,E}}{k_1^2}}} & 0 \\ \frac{k_1}{2}\frac{1}{\sqrt{\frac{k_2^2 x_{2,E}}{k_1^2}}} & -\frac{k_2}{2}\frac{1}{\sqrt{y_E + h_{02}}} \end{bmatrix}\Delta x + \begin{bmatrix} \frac{\gamma_1}{2}\frac{1}{\sqrt{\beta_1 + \gamma_1 u_{1,E}}} \\ 0 \end{bmatrix}\Delta u \\ \Delta y &= \begin{bmatrix} 0 & 1 \end{bmatrix}\Delta x. \end{aligned} \tag{65}$$

5.2 Simulation using the Linearized Model

First of all, the linearized model, which has been discretized with a sampling period of $h = 0.5s$, is analyzed. A model predictive controller is used to track the desired water level for the bottom tank. The symmetric weighting matrices Q and R are set equal to the identity matrix and the prediction and control horizons are chosen to be

$$N_p = 20 \qquad \text{and} \qquad N_c = 8. \tag{66}$$

In this thesis, only constraints on the input u_1 and the states x_1 and x_2 are imposed. The constraints for the pump voltage u_1 are given as

$$\begin{aligned} u_{min} &\leq u_i \leq u_{max} \\ 0 - u_{1,E} &\leq u_i \leq 5V - u_{1,E} \end{aligned} \tag{67}$$

with

$$u_{min} = \begin{bmatrix} u_{k,min} \\ u_{k+1,min} \\ \vdots \\ u_{k+N_c-1,min} \end{bmatrix} = \begin{bmatrix} -1.8782 \\ -1.8782 \\ \vdots \\ -1.8782 \end{bmatrix} \qquad u_{max} = \begin{bmatrix} u_{k,max} \\ u_{k+1,max} \\ \vdots \\ u_{k+N_c-1,max} \end{bmatrix} = \begin{bmatrix} 3.1218 \\ 3.1218 \\ \vdots \\ 3.1218 \end{bmatrix}. \tag{68}$$

The constraints for the states x_1 and x_2 are given as

$$\begin{aligned} x_{min} &\leq x_i \leq x_{max} \\ h_{0i} - x_{i,E} &\leq x_i \leq 20cm + h_{0i} - x_{i,E} \qquad \text{for} \quad i = 1, 2 \end{aligned} \tag{69}$$

with

$$x_{min} = \begin{bmatrix} x_{1,min} \\ x_{2,min} \\ \vdots \\ x_{N_p-1,min} \\ x_{N_p,min} \end{bmatrix} = \begin{bmatrix} -14.7112 \\ -15 \\ \vdots \\ -14.7112 \\ -15 \end{bmatrix} \qquad x_{max} = \begin{bmatrix} x_{1,max} \\ x_{2,max} \\ \vdots \\ x_{N_p-1,max} \\ x_{N_p,max} \end{bmatrix} = \begin{bmatrix} 5.2888 \\ 5 \\ \vdots \\ 5.2888 \\ 5 \end{bmatrix}. \tag{70}$$

As it has already been shown in section 3.1.5, the combination of the constraints represented in matrix notation is given as

$$W \Delta \bar{u}_k \leq \bar{w}, \tag{71}$$

with

$$W = \begin{bmatrix} -M \\ M \\ -H_x \\ H_x \end{bmatrix} \qquad \text{and} \qquad \bar{w} = \begin{bmatrix} -\bar{u}_{min} + L u_{k-1} \\ \bar{u}_{max} - L u_{k-1} \\ -\bar{x}_{min} + \bar{g}_{x,k} \\ \bar{x}_{max} - \bar{g}_{x,k} \end{bmatrix}. \tag{72}$$

5.2.1 Without Network

Let's assume that the control components are all directly connected to each other, in other words no network is used. The classical MPC approach, which has been presented in section 3, where only the first value from the predicted control sequence is applied to the process, is used. The schematic of the feedback loop using the linearized model is shown in figure 23.

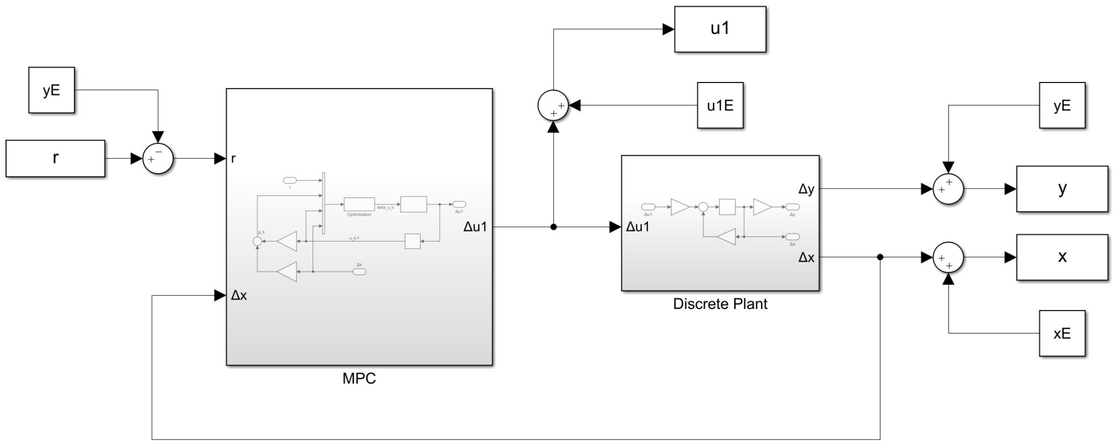

Figure 23: Schematic of the feedback loop with the linearized model in the case where no network is used.

The output of the feedback loop with the linearized model can be seen in figure 24. It's shown that there is almost no overshoot at the output of the linearized model, because no network-induced delays or data packet problems affect the control performance. The water level in the bottom tank takes nearly 40 seconds to reach the equilibrium point $y_E = 15cm$ and nearly 40 seconds to get from the equilibrium point to a height of 5cm.

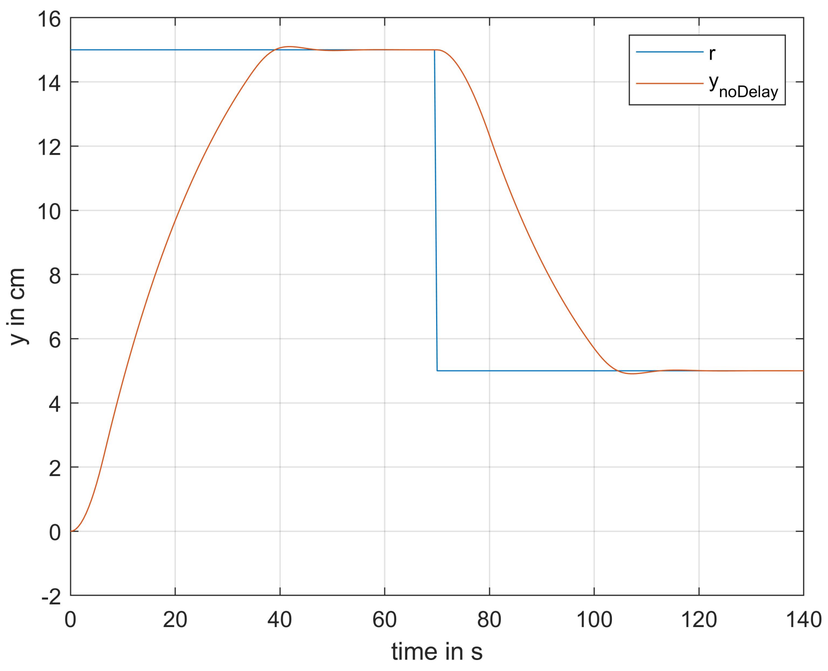

Figure 24: Water level in the bottom tank for the feedback loop with the linearized model without network.

5.2.2 Without Time-delay Compensation

Let's now assume that a network is used for the communication between the control components. This means that random time-delays will affect the transmission of the data packets between the plant and the controller side. For the communication network, an artificial network is created as it has been shown in section 2.5. In order to simplify the simulation, both delays are chosen to be equal to the sampling period such as

$$\tau_{sc,k} = \tau_{ca,k} = 1. \tag{73}$$

Figure 25 shows the behavior of the water level in the bottom tank in the case where no time-delay compensation is applied to the system. It can be seen that the solver is not able to find a solution at time instant $t = 8s$, because the problem is infeasible. The simulation stops running, since the chosen constraints have been exceeded. In figure 26 the states and input of the system are illustrated.

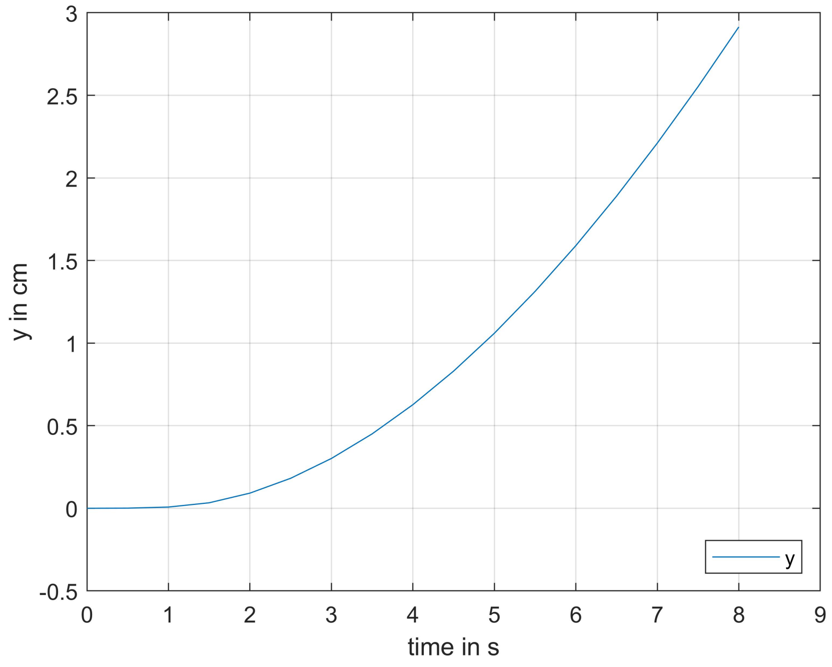

Figure 25: Water level in the bottom tank for the feedback loop with the linearized model without the time-delay compensation.

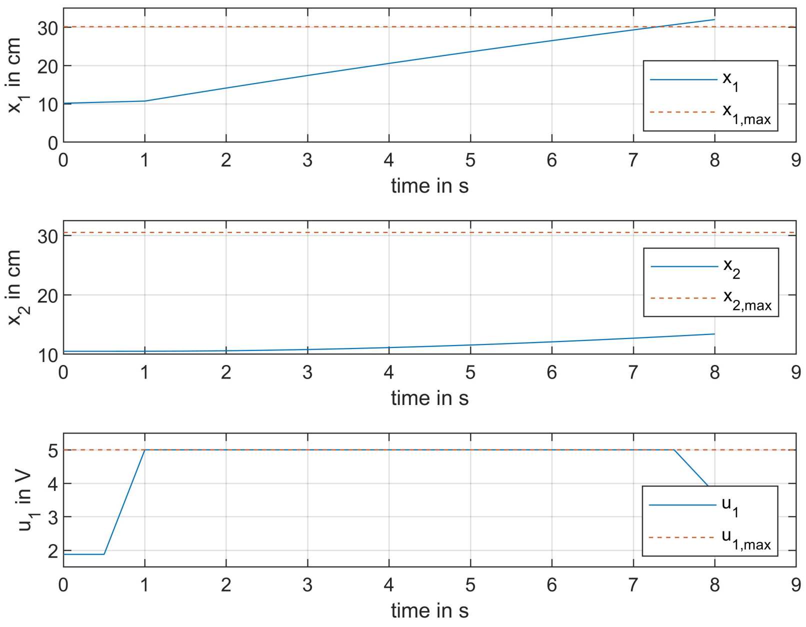

Figure 26: States and input of the system for the feedback loop with the linearized model without the time-delay compensation.

The state x_2 and the input u_1 do not exceed the given constraints. The solver is not able to find a solution without exceeding the imposed constraint of the state x_1 and that's why the simulation stops running.

5.2.3 Time-delay Compensation Without Time Stamps

First of all, it's assumed that both delays are known by the CAS without the usage of time stamps. Furthermore, it's also assumed that in the case where no packets reach either the controller or the actuator side, the last received packet is used instead. This means that a buffer is used on both sides.

The distribution of the network-induced delays in both channels and the distribution of the RTD is shown in figure 27. The probability of the delay in the backward channel being large is higher than the probability of the delay in the forward channel being large. The backward channel delay is typically severer than the forward channel delay, which is due to the fact that in a NCS, especially when it is combined with wireless sensor networks or Internet of Things systems, excessive sensors will be deployed in the network, which can lead to a higher time-delay [22].

It can be seen that the probability of the RTD being small is very low and that the probability of τ_k being large is high. This can be explained like the follow: first of all, the sensor-to-controller delay is generated assuming that it is uniformly distributed with the given boundaries

$$\tau_{sc,min} \leq \tau_{sc,k} \leq \tau_{sc,max} \quad \Rightarrow \quad 1 \leq \tau_{sc,k} \leq N_c - 2. \tag{74}$$

Since the RTD is not allowed to exceed the upper bound, which is assumed to be $N_c - 1$, the controller-to-actuator delay is generated assuming that it is uniformly distributed with the resulting boundaries

$$\tau_{ca,min} \leq \tau_{ca,k} \leq \tau_{k,max} - \tau_{sc,k}, \tag{75}$$

where $\tau_{k,max}$ is the upper bound of the RTD. It's clear that by using this method the probability of the forward channel delay being small is high.

Another possibility to model the network-induced delays would be to assume, that the RTD is uniformly distributed. This assumption would result into smaller RTDs compared to the RTD of the first method. The most used approach to model the networked-induced delays in NCSs is the Markov chain approach. It is one of the most attractive methods due to the reason that it is more natural to involve the temporal evolution of time-delays in the network, since the current time-delay may influence the future time-delays in practice. However, in this work the first presented method is used instead.

In figure 28 and 29 the network-induced delays and the RTD are illustrated respectively.

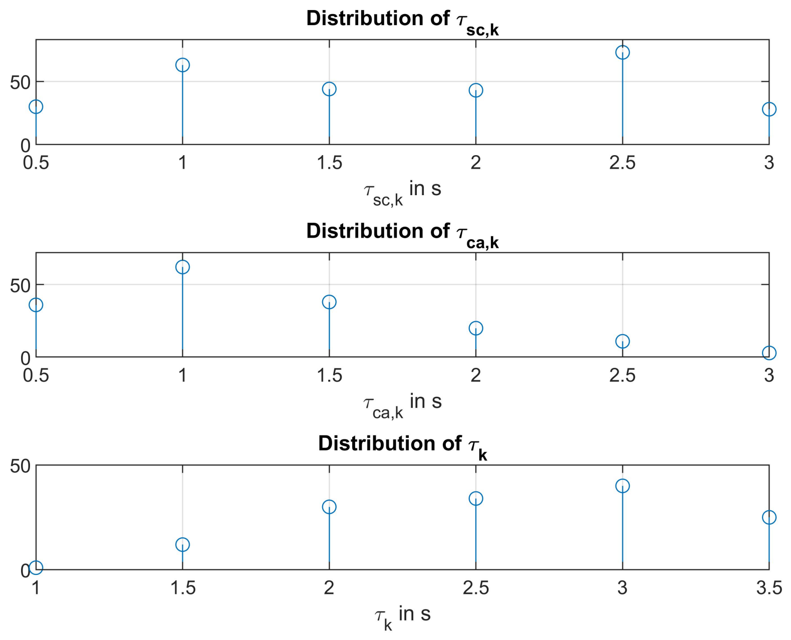

Figure 27: Distribution of the network-induced delays and the RTD.

It can be observed that the delay in the backward channel is never equal to zero, since the data from the sensors is sent at each time instant. Both delays aren't allowed to exceed the upper bound of $N_c - 2$, which is in this case $3s$. Whenever a packet reaches the controller side, a packet containing the predicted control sequence is sent to the actuator. This means that the controller isn't constantly sending information (see figure 28). Figure 29 shows at which time instants the packets reach the actuator side. The RTD isn't allowed to exceed the upper bound of $N_c - 1$, which is in this case $3.5s$.

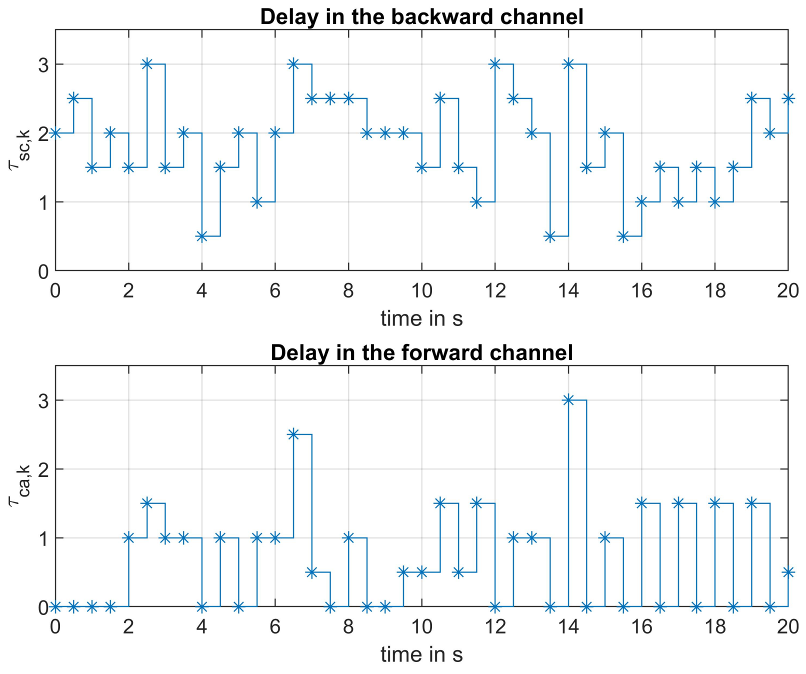

Figure 28: Backward channel delay $\tau_{sc,k}$ and forward channel delay $\tau_{ca,k}$ over the time (for 20s).

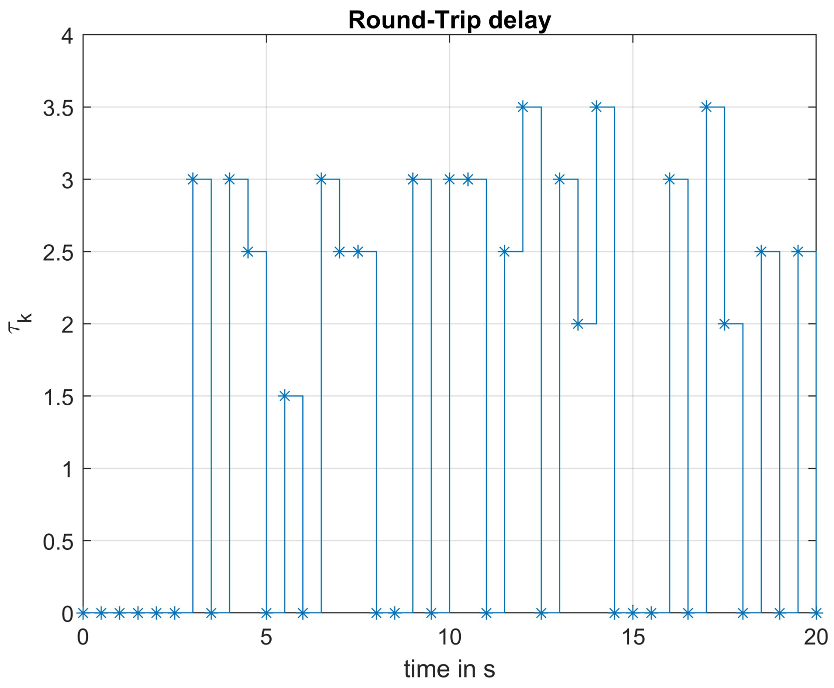

Figure 29: RTD τ_k over the time (for 20s).

Note that in this thesis the upper bounds of the network-induced delays depend on the control horizon N_c. This means that a larger control horizon will lead to larger network-induced delays. For the two tank system with a sampling period of $h = 0.5s$, a control horizon of $N_c = 20$ would result into RTDs up to $9.5s$. Such large delays are obviously difficult to handle and they result into a bad control performance and into instability. A possibility to avoid those problems and to guarantee a better control performance would be to choose the upper bounds of the network-induced delays to be independent of the control horizon N_c. This method would increase the control performance but it would also increase the computation time of the controller, since more predicted control values are computed.

The schematic of the feedback loop with the time-delay compensation without time stamps is shown in figure 30. As mentioned before, both time-delays are assumed to be directly known by the CAS.

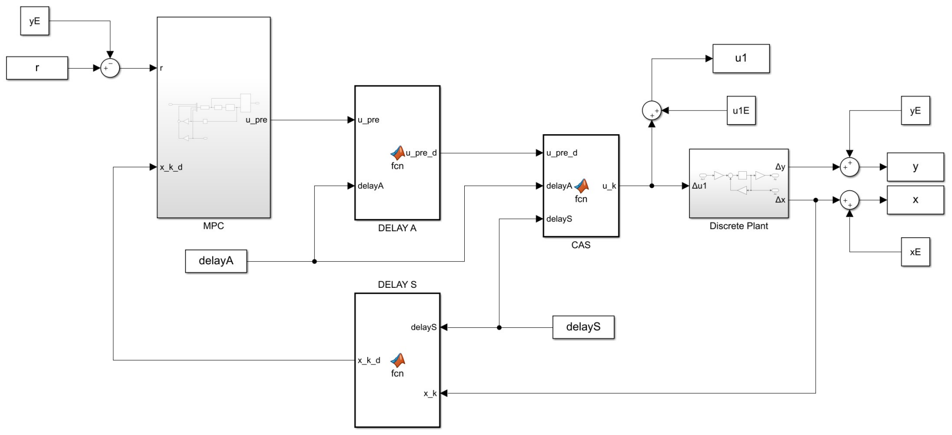

Figure 30: Schematic of the feedback loop with the linearized model in the case where no time stamps are used.

Figure 31 shows the water level in the bottom tank for the simplified time-delay compensation without time stamps. The water level in the bottom tank follows the reference signal, but it oscillates quiet strongly. This phenomenon can be explained as the follow: in this simplified implementation, the last used control signal is reused in the case where no packet reaches the CAS and this leads to the oscillation behavior of the output signal. In order to avoid this oscillation, the technique presented in section 4 is used. This means that whenever a packet doesn't reach the CAS, the next predicted control input of the last received packet, which is stored in a buffer, is applied as the new plant input.

Note that in this simplified implementation the MPC runs at each time instant, because it's assumed, that whenever a packet reaches either the controller or the actuator side, the packet is stored in a buffer. This means that the MPC always uses the packet stored in the buffer on the controller side for the computations. With this method a lot of unnecessary computation time is needed, which may degrade the control performance. It makes more sense to just run the MPC whenever a packet reaches the controller side. By doing that, only a buffer on the actuator side is needed.

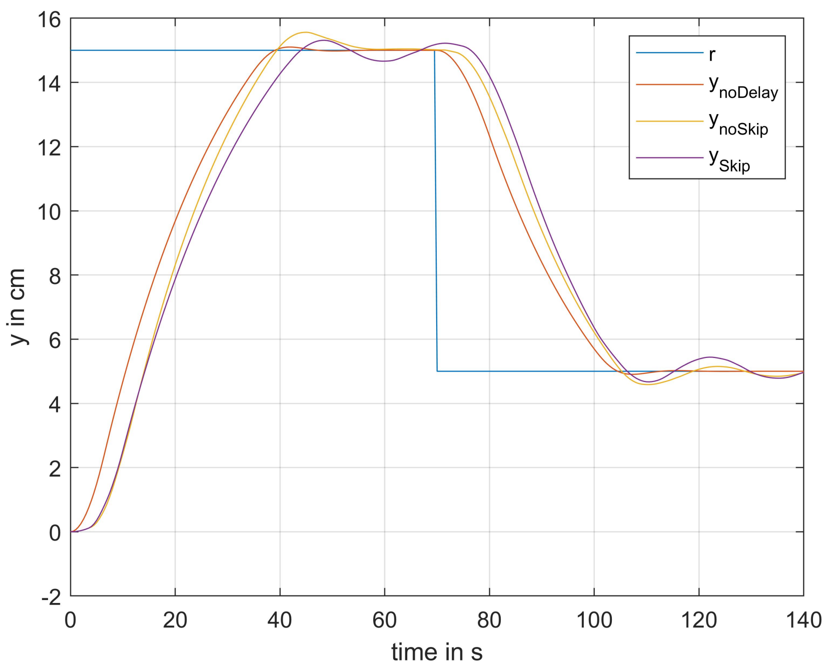

Figure 31: Water level in the bottom tank for the feedback loop with the linearized model in the case where no time stamps are used.

5.2.4 Packet-based Control With Time Synchronization

In this section, the packet-based control approach with time synchronization is applied to the system, which means that time stamps are used to identify both network-induced delays separately. In figure 32 the schematic of the feedback loop is shown and the behavior of the water level in the bottom tank can be seen in figure 33. There is almost no oscillation at the output of the system. It can be observed that it's convenient to skip older packets, because the system reacts faster.

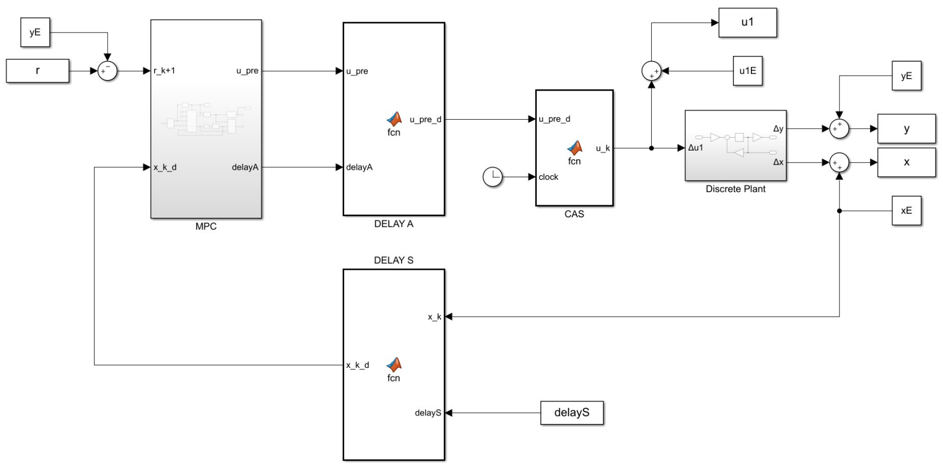

Figure 32: Schematic of the feedback loop with the linearized model in the case where time stamps are used.

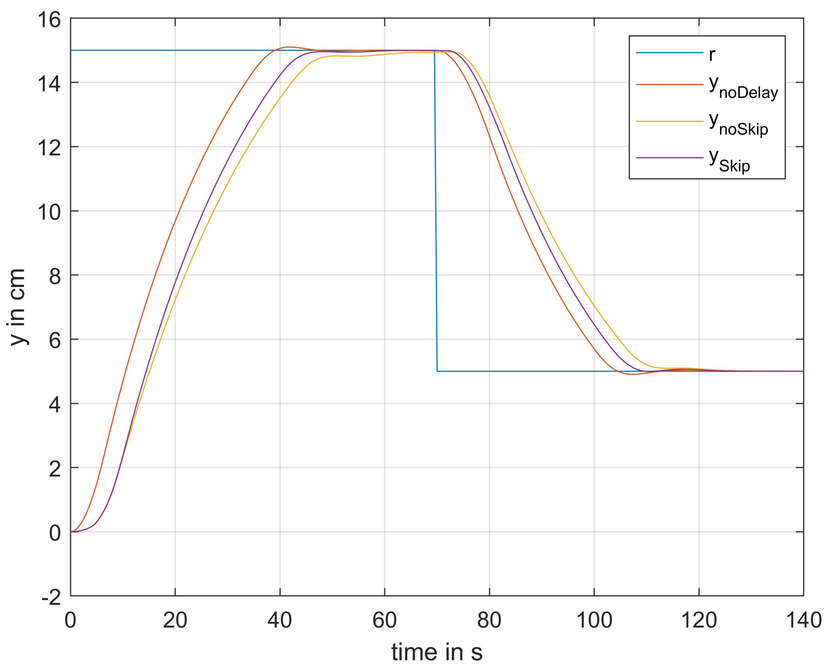

Figure 33: Water level in the bottom tank for the feedback loop with the linearized model when using the packet-based control approach with time synchronization.

In figure 34 and 35 the states and the input of the plant are illustrated respectively. Again, it can be observed that it's more convenient to skip older packets, since the controller reacts faster. Furthermore, it's also shown that the input voltage u_1 oscillates in the case where the data among the control components is exchanges in the form of data packets over a network. A possibility to reduce this oscillation behavior would be to increase the weighting matrix R.

In order to avoid the MPC running at each time instant, the so-called *Enabled Subsystem* block in Simulink is used. With the usage of this block it's possible to run the MPC only whenever a packet reaches the controller side. Because of this feature, a better control performance is guaranteed, since there is less computation time needed. Figure 36 shows how often the MPC is running during the simulation. When the MPC is active, the value increases by 1 and when the MPC is inactive, the value decreases by 1. It can clearly be seen that in the case where older packets don't get skipped, the MPC is more often active than if older packets get skipped. It's also shown that when older packets get skipped (in this specific case), the MPC is more often inactive than active during the simulation time.

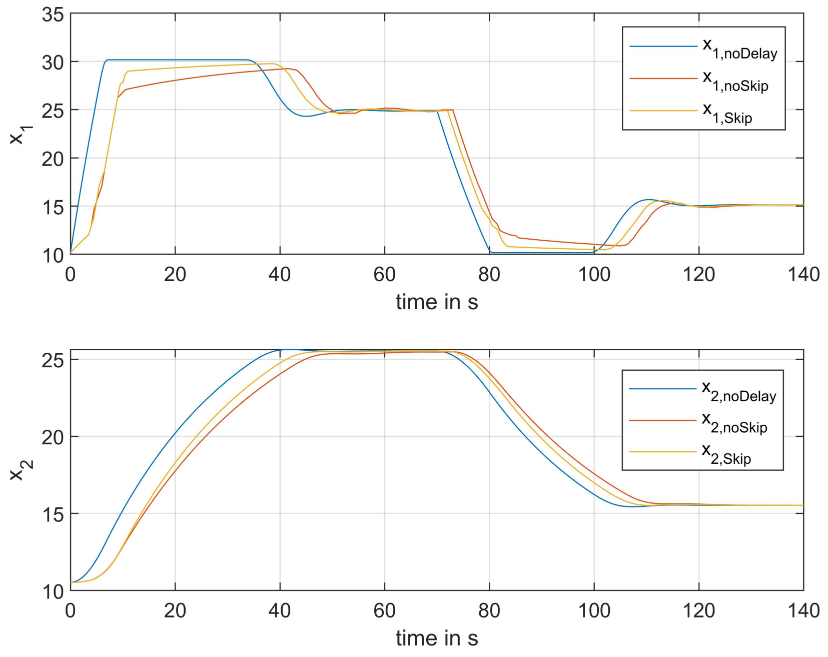

Figure 34: States of the system for the feedback loop with the linearized model when using the packet-based control approach with time synchronization.

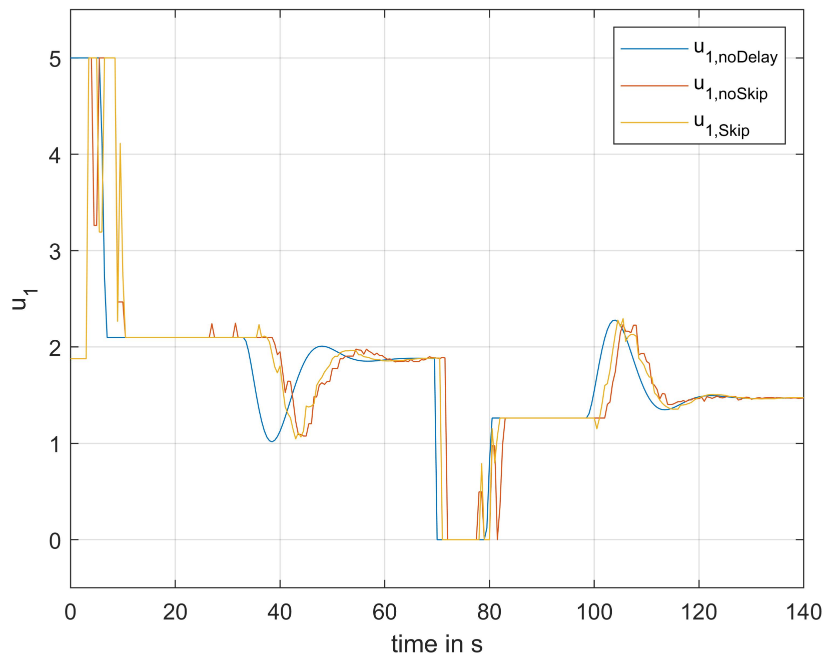

Figure 35: Input of the system for the feedback loop with the linearized model when using the packet-based control approach with time synchronization.

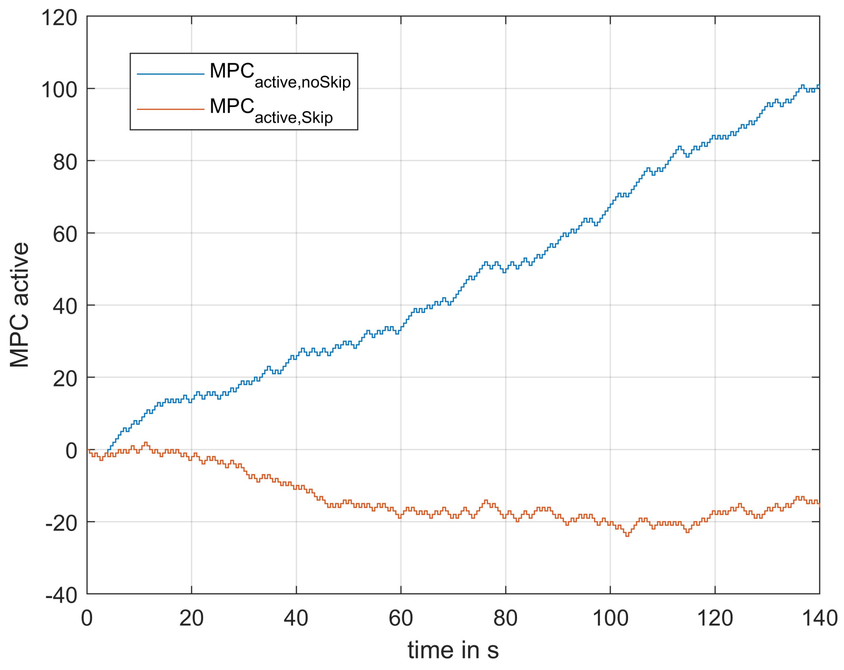

Figure 36: Active/Inactive state of the MPC during the simulation when using the packet-based control approach with time synchronization.

Until now, the MPC has always predicted the control sequence over the entire control horizon N_c. As mentioned in section 4.1, the first part of the predicted control sequence has simply be discarded. Another possibility would be to choose the boundaries of the cost function J, such that the MPC only predicts those control inputs that can effectively be used, since the controller already knows the sensor-to-controller delay $\tau_{sc,k}$ before it starts the computations. The quadratic cost function results to

$$J = \sum_{i=1}^{N_p} (\hat{y}_{k+i} - r_{k+i})^T Q_i (\hat{y}_{k+i} - r_{k+i}) + \sum_{i=\tau_{sc,k}+2}^{N_c} \Delta u_{k+i-1}^T R_i \Delta u_{k+i-1}. \tag{76}$$

Figure 37 shows how the water level in the bottom tank behaves in the case where the boundaries of the cost function are reduced. The computation time is lower, since the boundaries of the cost function are smaller. However, it can clearly be seen that the quality of the results isn't that good anymore. The water level is not able to reach the equilibrium point and there is always a certain deviation from the reference signal. A possibility to get a better control performance would be to choose the control horizon bigger than the maximum possible RTD τ_k.

Since the results of the packet-based control approach without time synchronization are the same as the results presented in this section (in the case where the MPC predicts over the entire control horizon N_c), no simulations using the approach presented in section 4.2 are performed. As already mentioned, only the size of the packet sent by the controller to the actuator side changes.

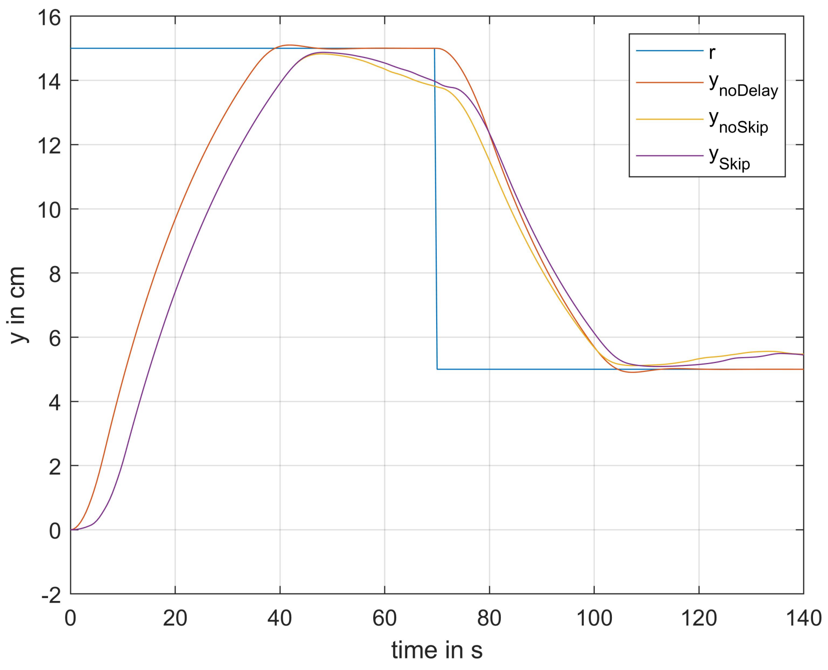

Figure 37: Water level in the bottom tank for the packet-based control approach with time synchronization with the reduced boundaries of the cost function.

5.2.5 Compensation of Data Packet Dropouts

In this section, data packet dropouts are taken into account and their impact on the control performance is investigated. For each communication channel, a uniformly distributed integer number with the given boundaries a and b is generated such as

$$\begin{aligned} a_{sc} \leq X_{sc} \leq b_{sc} \qquad &\Rightarrow \qquad 1 \leq X_{sc} \leq 5 \\ a_{ca} \leq X_{ca} \leq b_{ca} \qquad &\Rightarrow \qquad 1 \leq X_{ca} \leq 5 \end{aligned} \tag{77}$$

It has been assumed that whenever the random number X_{sc} is equal to $z_{sc} \in [a_{sc}, b_{sc}]$, the packet containing the feedback information is dismissed. In the case where X_{ca} is equal to $z_{ca} \in [a_{ca}, b_{ca}]$, the packet containing the future control actions is discarded. It is clear that by decreasing the upper bounds b_{sc} and b_{ca}, the number of data packet dropouts in the network increases. Furthermore, it's also assumed that only five consecutive data packet dropouts are tolerated. Figure 38 shows how often the MPC is running during the simulation and the time instances, where data dropouts in the forward and in the backward channel (in the case where older packets get skipped).

The three methods presented in section 4.4 are applied to the system. In figure 39 the results of the three methods are illustrated. It can be seen how the random packet losses affect the control performance.

The first method is very simple to implement, but the control input set to zero causes an unsmooth switching, which cannot always be tolerated and therefore it's difficult to get a good control performance. The difference between the results of the second and third method isn't that big in this specific case, but it's more convenient to apply the control prediction approach presented in section 4 (Method 3), since it provides the best control performance. However, it's more complex and complicated to implement.

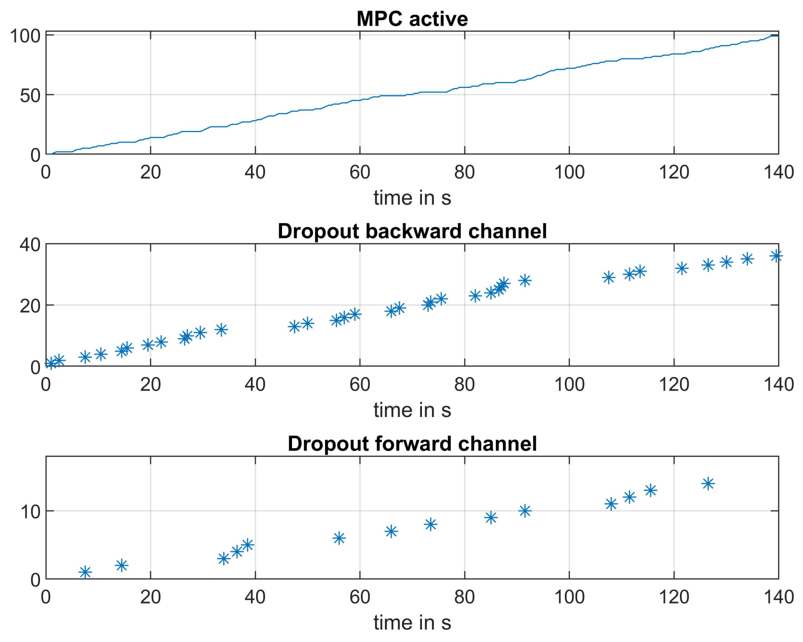

Figure 38: Status of the MPC (top). Packet dropouts in the backward channel (middle). Packet dropouts in the forward channel (bottom).

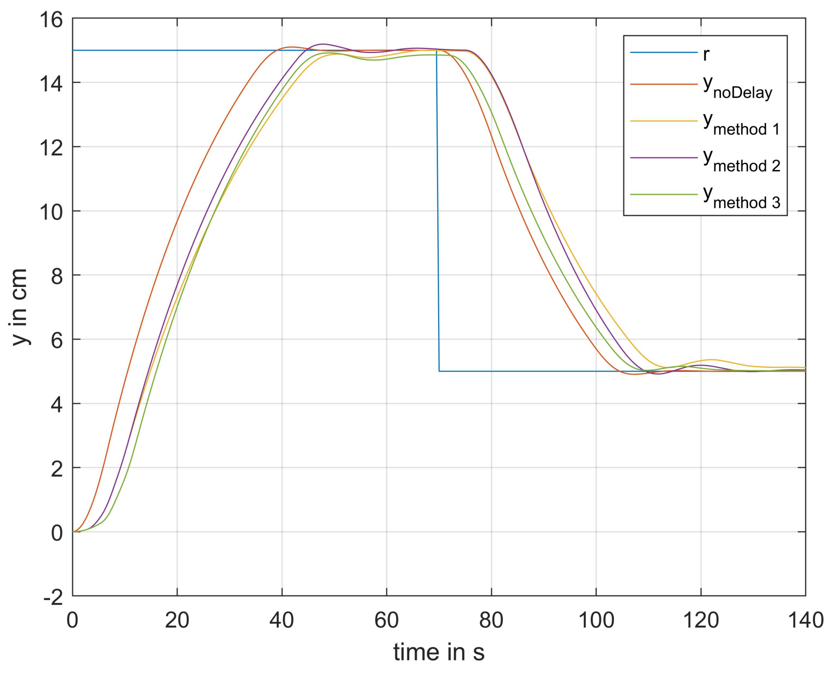

Figure 39: Water level in the bottom tank for the feedback loop with the linearized model with packet dropouts.

5.3 Simulation using the Nonlinear Model

As shown in section 5.2, the controller works quiet good when using the linearized model. Very often, such a model is defective in terms of reliability and authenticity, because of the neglected nonlinear terms. That's why it's of interest to investigate how the controller works on the nonlinear model as well. The parameters of the MPC are all the same as in the previous section. The constraints for the pump voltage u_1 are given as

$$\begin{aligned} u_{min} \leq u_i \leq u_{max} \\ 0 \leq u_1 \leq 5V \end{aligned} \tag{78}$$

with

$$u_{min} = \begin{bmatrix} u_{k,min} \\ u_{k+1,min} \\ \vdots \\ u_{k+N_c-1,min} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad u_{max} = \begin{bmatrix} u_{k,max} \\ u_{k+1,max} \\ \vdots \\ u_{k+N_c-1,max} \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ \vdots \\ 5 \end{bmatrix}. \tag{79}$$

The constraints for the states x_1 and x_2 are given as

$$\begin{aligned} x_{min} \leq x_i \leq x_{max} \\ h_{0i} \leq x_i \leq 20cm + h_{0i} \qquad \text{for} \quad i = 1, 2 \end{aligned} \tag{80}$$

with

$$x_{min} = \begin{bmatrix} x_{1,min} \\ x_{2,min} \\ \vdots \\ x_{N_p-1,min} \\ x_{N_p,min} \end{bmatrix} = \begin{bmatrix} 10.17 \\ 10.53 \\ \vdots \\ 10.17 \\ 10.53 \end{bmatrix} \qquad x_{max} = \begin{bmatrix} x_{1,max} \\ x_{2,max} \\ \vdots \\ x_{N_p-1,max} \\ x_{N_p,max} \end{bmatrix} = \begin{bmatrix} 30.17 \\ 30.53 \\ \vdots \\ 30.17 \\ 30.53 \end{bmatrix}. \tag{81}$$

5.3.1 Without Network

Let's first assume that the control components are all directly connected to each other, which means that no network is used. Again, the classical MPC approach presented in section 3 is used.

The behavior of the output of the plant is shown in figure 40. The water level in the bottom tank does not overshoot in the equilibrium point $y_E = 15cm$. The same result has already been shown for the linearized model. However, by comparing figure 24 with figure 40 it can be seen, that the output of the nonlinear model reaches the reference signal faster than the output of the linearized model.

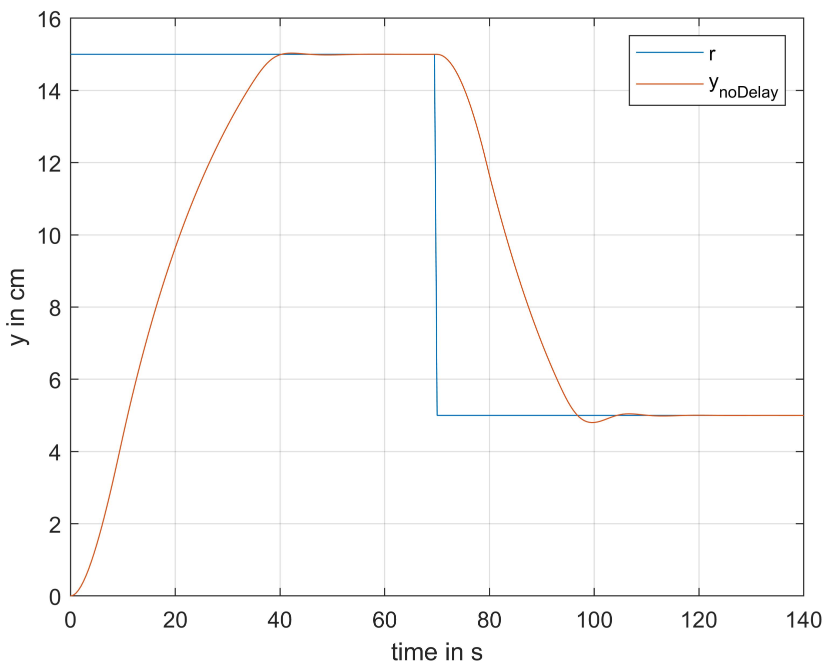

Figure 40: Water level in the bottom tank for the feedback loop with the nonlinear model without network.

5.3.2 Without Time-delay Compensation

A network is used for the communication among the control components. Random time-delays will affect the transmission between the controller and the actuator side. In order to simplify the simulation, both network-induced delays are assumed to be equal to the sampling period such as

$$\tau_{sc,k} = \tau_{ca,k} = 1. \tag{82}$$

Figure 41 shows the behavior of the water level in the bottom tank. Also in this case the simulation stops running at time instant $t = 11s$, because the given boundaries have been exceeded. The solver isn't able to find a feasible solution within the allowed regions. In figure 42 the states and input of the system are illustrated respectively.

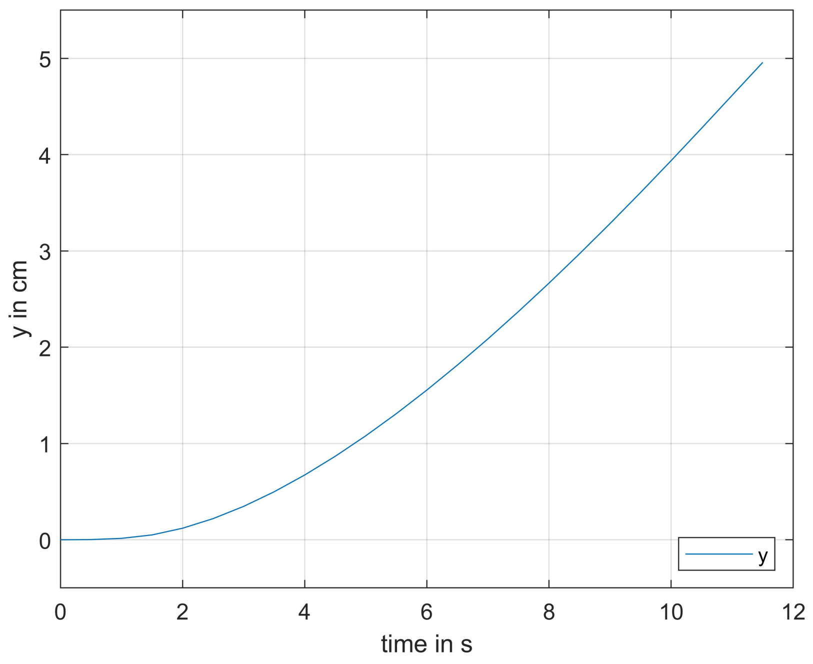

Figure 41: Water level in the bottom tank for the feedback loop with the nonlinear model without the time-delay compensation.

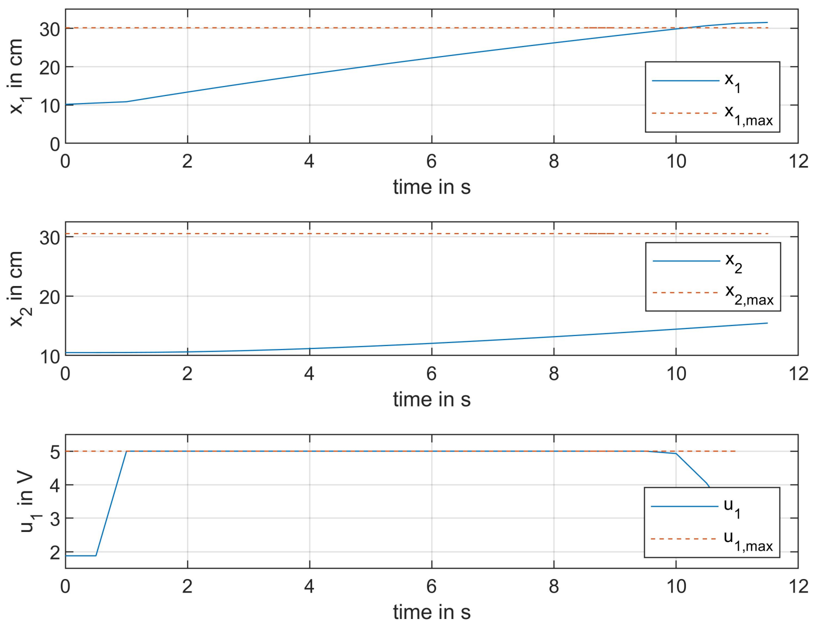

Figure 42: States and input for the feedback loop with the nonlinear model without the time-delay compensation.

The state x_1 exceeds the given upper bound and that's why the simulation stops running.

5.3.3 Time-delay Compensation Without Time Stamps

The simplified compensation approach presented in section 5.2.3 is applied on the nonlinear model as well. Both network-induced delays are assumed to be directly known by the CAS without performing any computations. Figure 43 shows the behavior of the output of the plant. The water level in the bottom tank more or less reaches the wanted level, but it oscillates quiet strongly. By comparing figure 30 with figure 43, it can be seen that the oscillation behavior of the nonlinear model is stronger than the oscillation behavior of the linearized model. In other words, the output signal of the nonlinear model is more inaccurate than the output signal in the linear case.

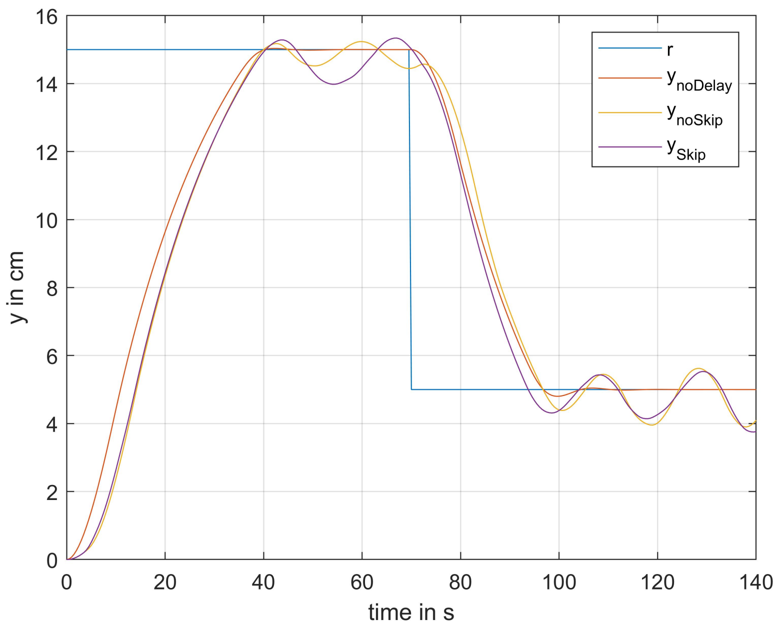

Figure 43: Water level in the bottom tank for the feedback loop with the nonlinear model in the case where no time stamps are used.

5.3.4 Packet-based Control With Time Synchronization

The packet-based control approach with time synchronization is also applied to the nonlinear model. Both assumptions 1 and 2, presented in section 4.1, have to be fulfilled and the working principle is the same as the one already illustrated in Algorithm 3.

Again, the backward channel delay $\tau_{sc,k}$ can be computed right after the packet reaches the controller side. The first values of the predicted control sequence are simply discarded and the other ones are sent to the plant side together with a time stamp. As it's shown in figure 44, the output signal of the plant follows the reference signal without oscillating. By comparing the results of the linearized model, shown in figure 33, with the results of the nonlinear model it can clearly be seen, that the network-induced delays have a bigger impact on the control performance when using the nonlinear model. I can also be observed that the output signal of the nonlinear model reaches the reference signal faster than the output signal of the linearized model.

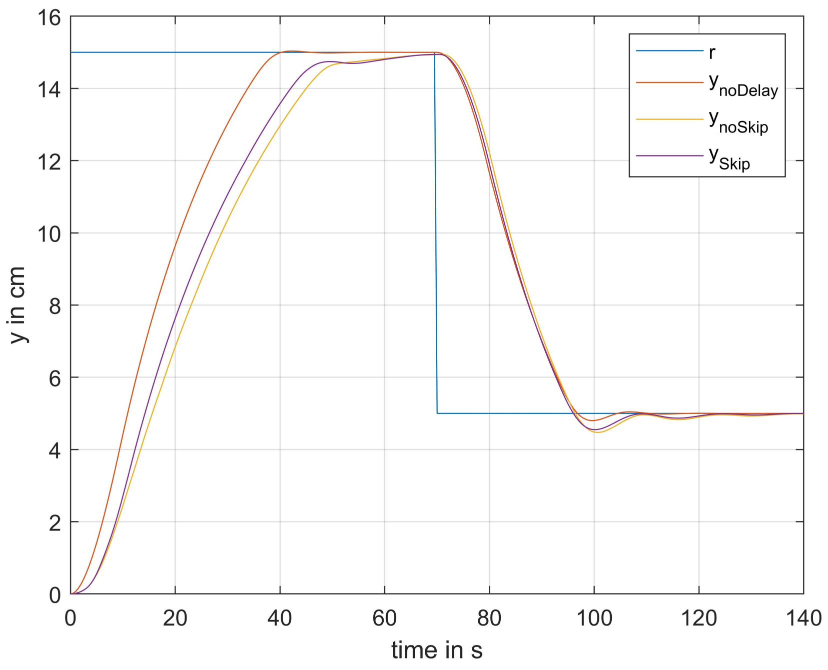

Figure 44: Water level in the bottom tank for the feedback loop with the nonlinear model when using the packet-based control approach with time synchronization.

In figure 45 and 46 the states of the system and the input signal are shown respectively.

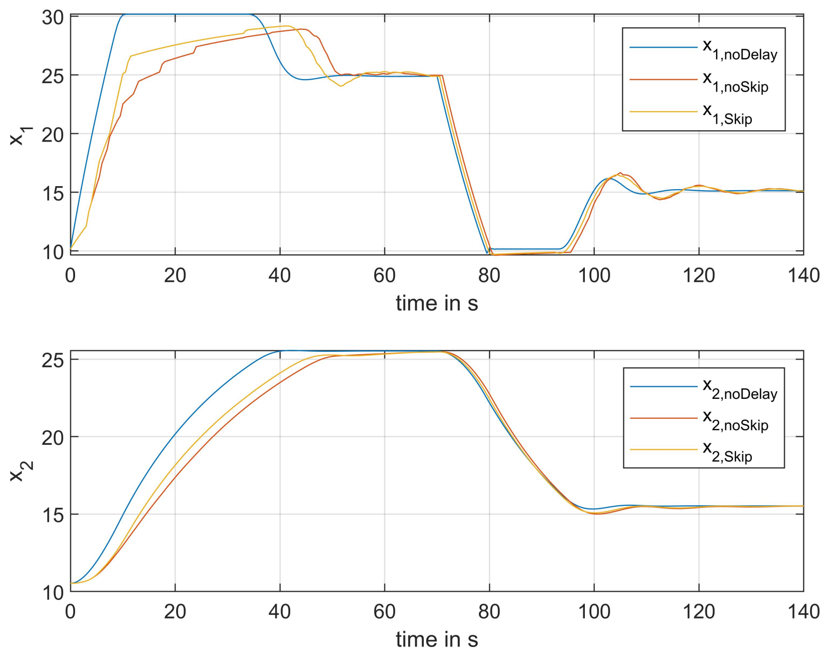

Figure 45: States of the system for the feedback loop with the nonlinear model when using the packet-based control approach with time synchronization.

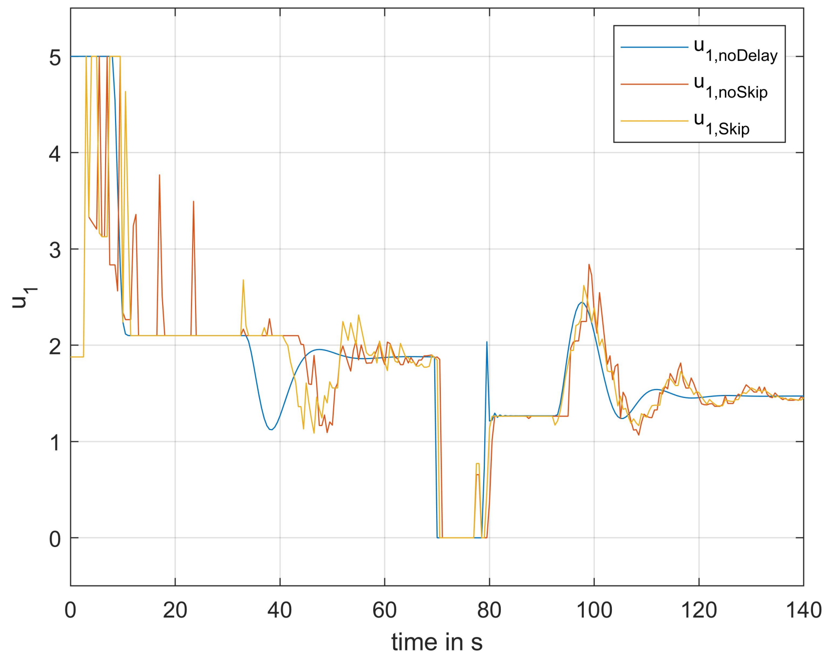

Figure 46: Input of the system for the feedback loop with the nonlinear model when using the packet-based control approach with time synchronization.

Figure 47 shows how often the MPC is active during the simulation time.

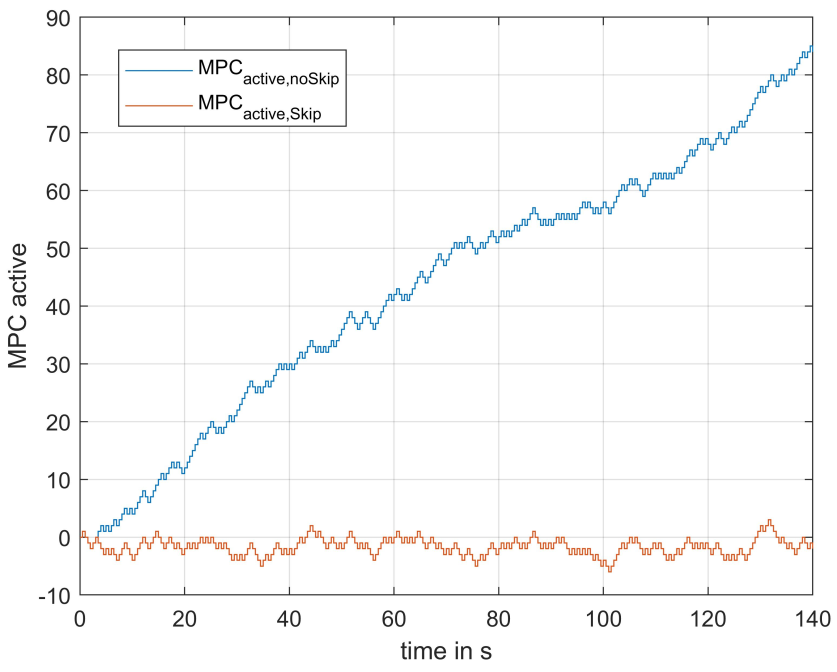

Figure 47: Active/Inactive state of the MPC during the simulation when using the packet-based control approach with time synchronization.

The controller is more active in the case where older packets don't get skipped. It can also be observed that in the case where older packets get skipped (in this specific case), the MPC is equally often active than inactive during the simulation.

The boundaries of the cost functional are reduced as it has been shown in equation (76), since the backward channel delay is already known by the controller before it starts with the computations. The MPC only predicts those control inputs that can also be applied to the plant. By doing that, the size of the packet sent by the controller is reduced and it varies at each time instant.

Figure 48 shows the behavior of the water level in the bottom tank. As it has already been shown for the linearized model, the output of the plant isn't able to follow the desired target, because the MPC predicts over less time steps. Whenever this approach is applied, it would make more sense to choose the upper bound of the RTD independently from the control horizon N_c. Figure 49 and 50 show the states of the nonlinear model and the plant input respectively. It can be seen that the state x_2 doesn't oscillate, however, the state x_1 and especially the plant input u_1 oscillate significantly during the simulation time.

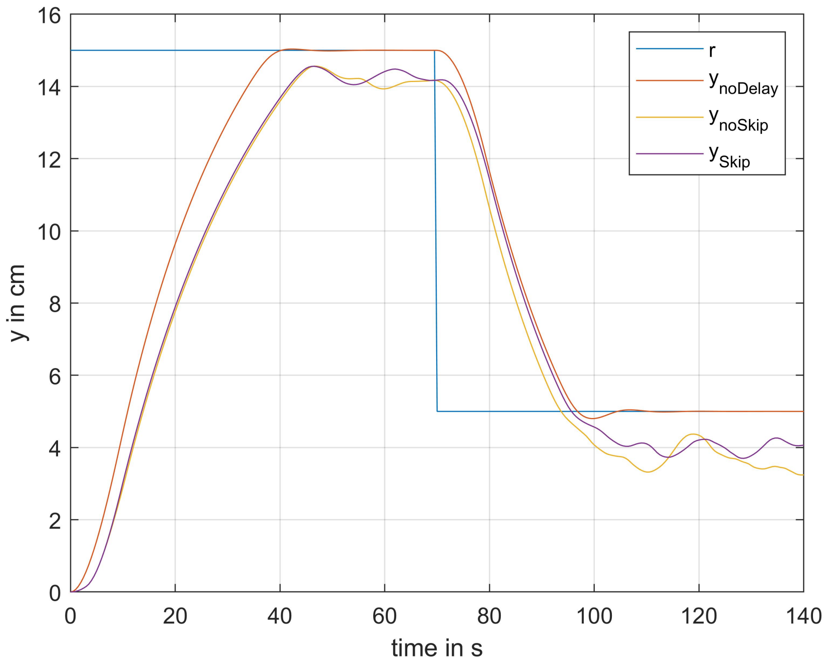

Figure 48: Water level in the bottom tank for the packet-based control approach with time synchronization with the shorted cost function.

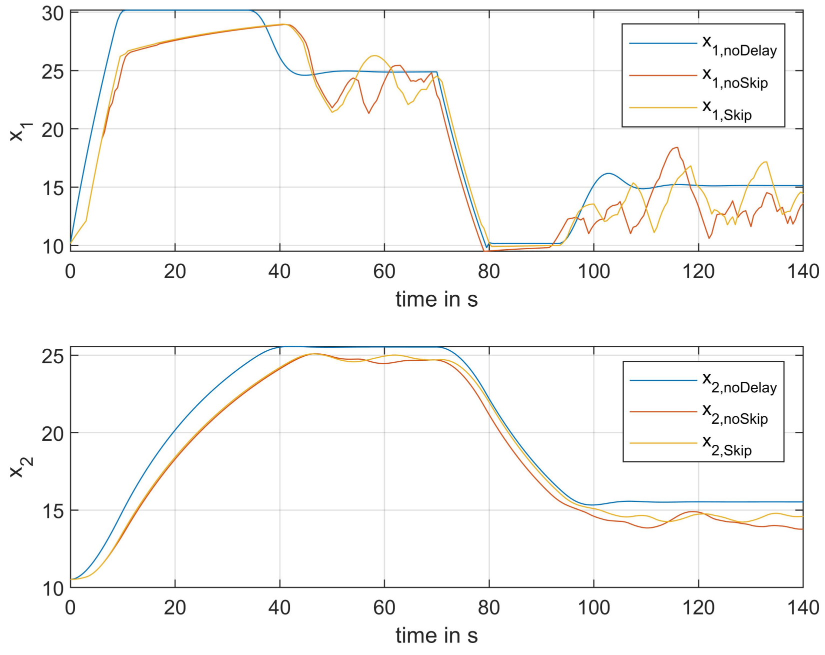

Figure 49: States of the nonlinear model for the packet-based control approach with time synchronization with the shorted cost function.

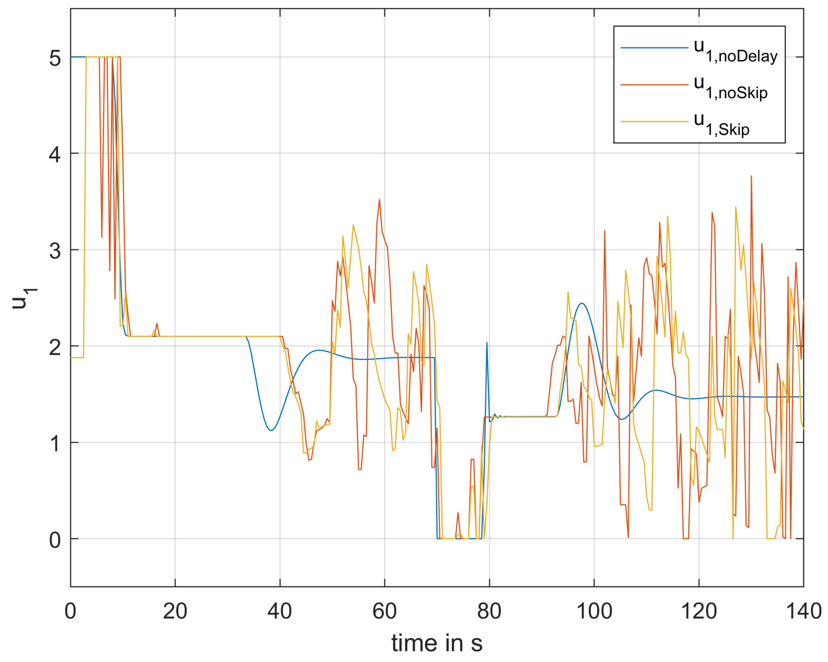

Figure 50: Input of the nonlinear model for the packet-based control approach with time synchronization with the shorted cost function.

5.3.5 Compensation of Data Packet Dropouts

In this section, data packet dropouts are taken into consideration. The three methods presented in section 4.4 can be applied to the nonlinear model, if assumption 5 holds. Figure 51 shows how often the MPC is active during the simulation and the exact time instances, when packets get lost in the forward and in the backward channel (in the case where older packets get skipped).

In figure 52 the results of the three methods are illustrated. It can be seen that it's more convenient to apply the control prediction approach presented in section 4 (Method 3), since the other two methods do not guarantee an optimal control performance.

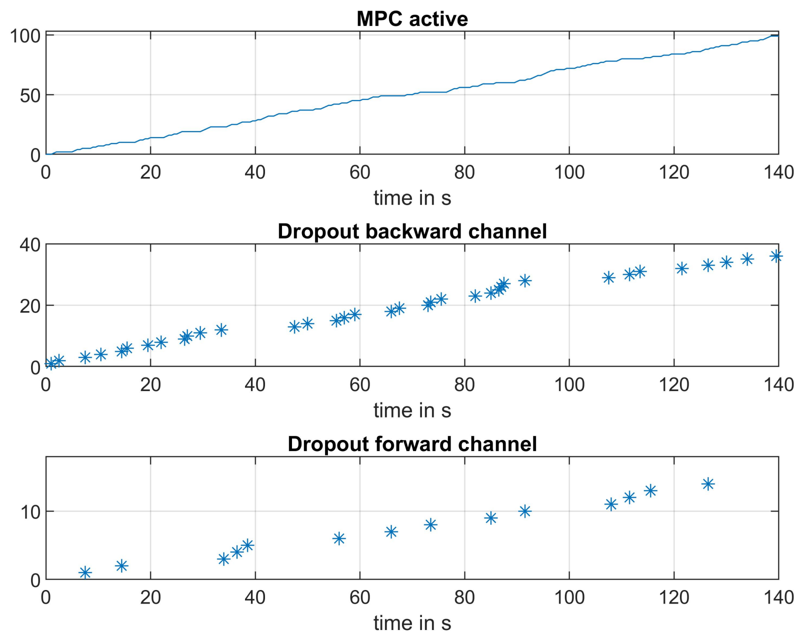

Figure 51: Status of the MPC (top). Packet dropouts in the backward channel (middle). Packet dropouts in the forward channel (bottom).

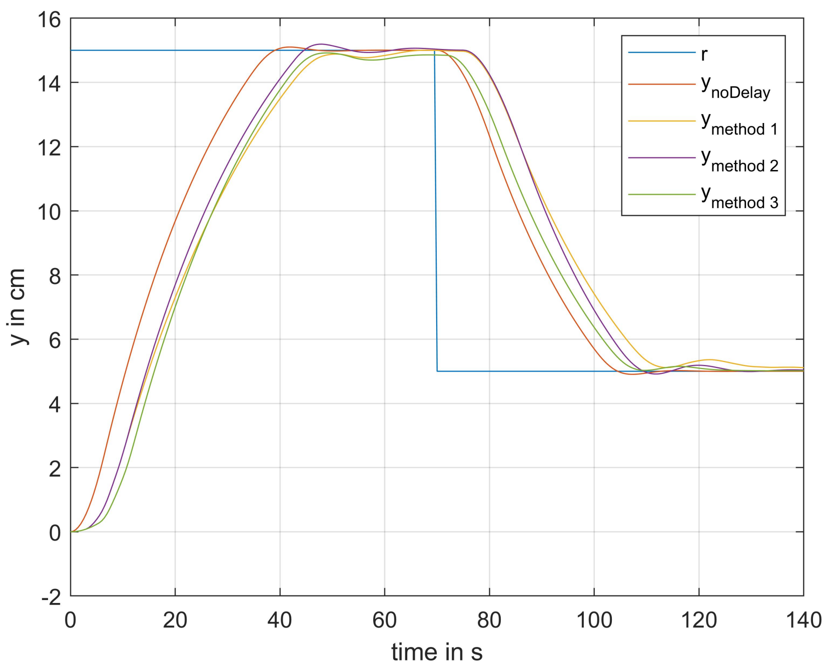

Figure 52: Water level in the bottom tank for the feedback loop with the nonlinear model with packet dropouts.

6 Experiments

As it has been shown in the previous sections, there is an important difference between the usage of the linearized model and the nonlinear model. The simulation results in the linear case are better than the simulation results in the nonlinear case, because of the nonlinear effects which have been neglected when calculating the linearized model.

Very often, the mathematical model does not fully describe the system in real life. That's why it's interesting to observe how the simulations performed in sections 5.2 and 5.3 look like for the real model. First of all, the unknown parameters of the model have to be identified.

6.1 Parameter Identification

It's very important to perform the parameter identification at the beginning of this experiment. If older values for the parameters are used, the calculated mathematical model will lead to a bad control performance and even to instability, since the values of the parameters change over time.

Capacitive sensors of the type *Liquicap T FMI21* are used to measure the water level in the top and in the bottom tank [15]. First of all, in order to convert the measured signals into a value in cm, an offset d and a gain V have to be computed. The parameters h_{0i} and k_i for $i = 1, 2$ are identified by performing an outflow test. Water is pumped into the tank, until a steady state of the water level is reached. Next, the water inflow is switched off and the water level in the tank is measured. The simplified mathematical model results to

$$\frac{dx}{dt} = -k\sqrt{x}. \tag{83}$$

This nonlinear differential equation can be computed analytically with the so-called *separation of variables* approach.

$$\begin{aligned} \frac{dx}{dt} &= -k\sqrt{x} \\ \frac{dx}{\sqrt{x}} &= -kdt \\ 2\sqrt{x} &= -kt + C \end{aligned} \tag{84}$$

In order to compute the constant of integration C, the time instant t_e, where the tank is fully emptied, is introduced such that

$$x(t_e) = 0 \qquad \Rightarrow \quad C = kt_e. \tag{85}$$

Regarding (84) yields

$$x(t) = \frac{1}{4}\big(-kt + C\big)^2. \tag{86}$$

The analytical solution for the fill height $h(t) = x(t) - h_0$ is given as

$$h(t) = \begin{cases} \frac{k^2}{4}\big(t - t_e\big)^2 - h_0 & \text{for} \quad t_0 \leq t \leq t_e \\ -h_0 & \text{for} \quad t \geq t_e \end{cases}. \tag{87}$$

With the usage of the least squares method, it's possible to identify the unknown parameters h_{0i} and k_i for $i = 1, 2$ with

$$E = \sum_{i=1}^{n} \left[s_i - \left(\frac{k_{approx}^2}{4} (t_i - t_{e,approx})^2 - h_{0,approx} \right) \right]^2, \tag{88}$$

where s_i is the i-th measured value of the fill height, t_i is the corresponding time instance and n is the amount of measured values. Figure 53 shows the measured and the approximate water level in the top tank and figure 54 shows the measured and the approximate water level in the bottom tank (after performing the outflow test).

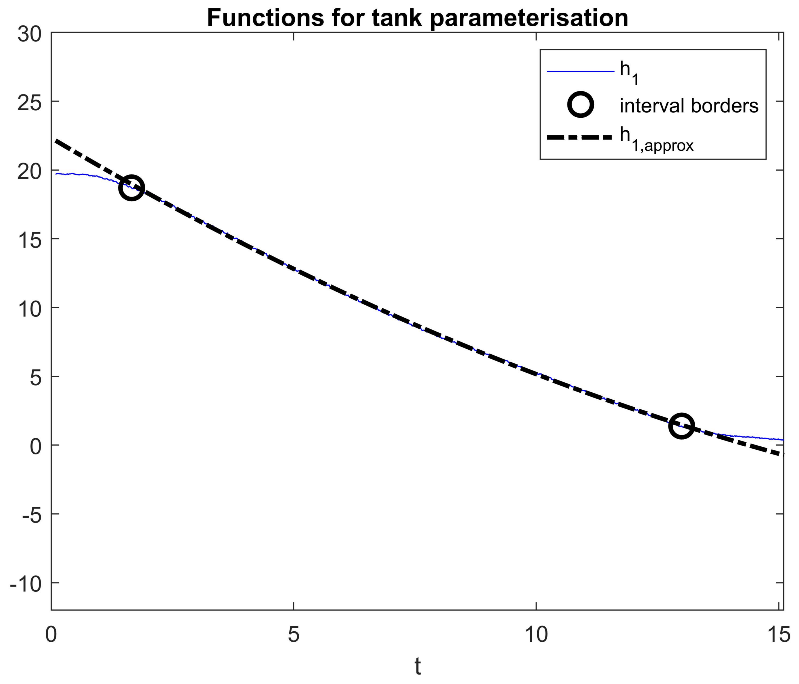

Figure 53: Measured and approximated water level in the top tank, after performing the outflow test.

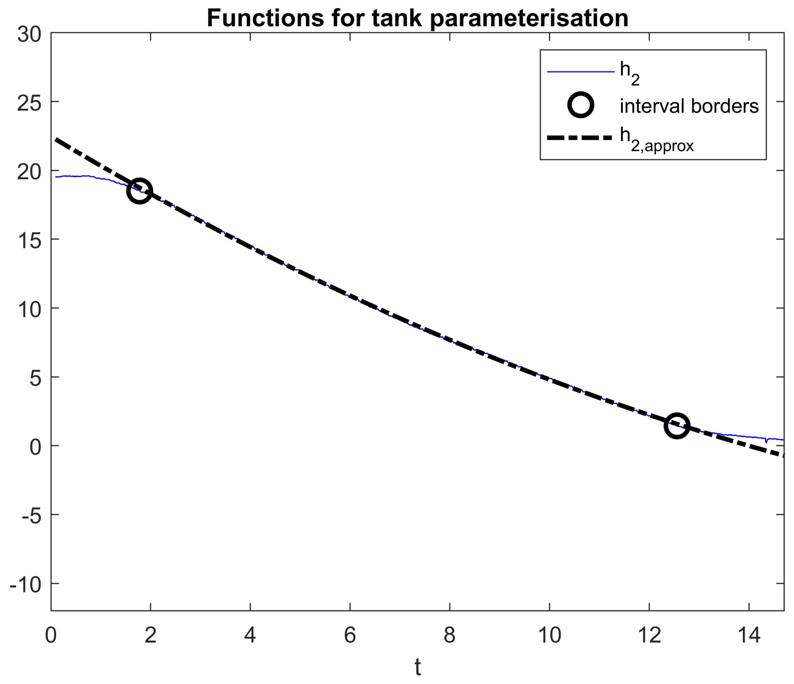

Figure 54: Measured and approximated water level in the bottom tank, after performing the outflow test.

For the identification of the pump parameters the so-called *steady-state method* is used. A constant pump voltage u_i is applied and after a while, a steady state of the fill height is reached.

$$\frac{dx_i}{dt} = 0 = -k\sqrt{x_i} + z_{p,i} \tag{89}$$

The inflow $z_{p,i}$ can be computed as

$$z_{p,i} = k_i\sqrt{x_i}, \tag{90}$$

since the parameters k_1 and k_2 are already known. For the first pump, this method is performed 11 times, with voltages between $3.6V$ and $6.6V$ in $0.3V$ steps. For the second pump, this method is performed 14 times, with voltages between $3V$ and $5.6V$ in $0.2V$ steps. At this point, the pump parameters α_i, β_i and γ_i can be computed by interpolation. Figure 55 shows the u_1/z_p-curve of the first pump and figure 56 shows the u_1/z_p-curve of the second pump.

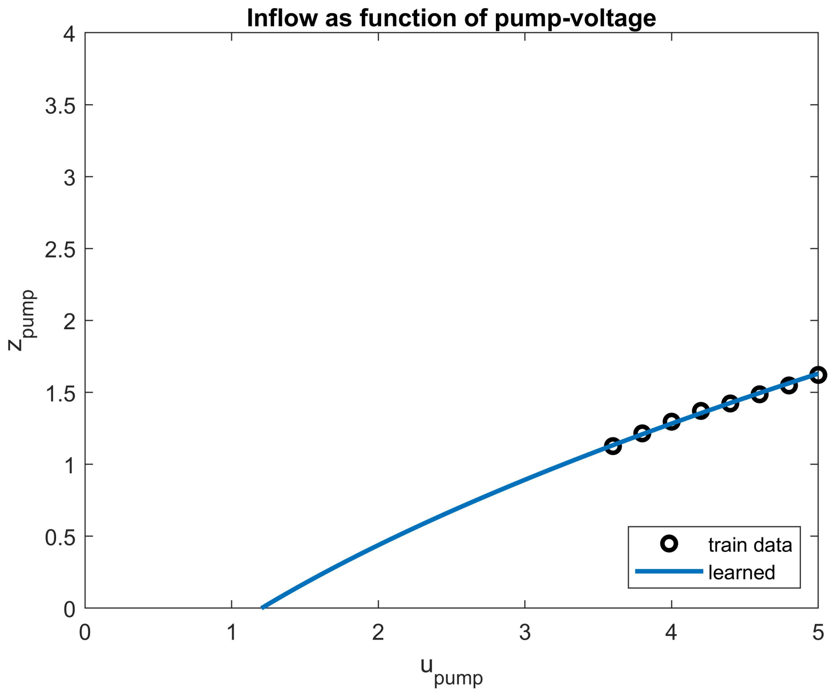

Figure 55: $u_{1,pump}/z_p$-curve of the first pump.

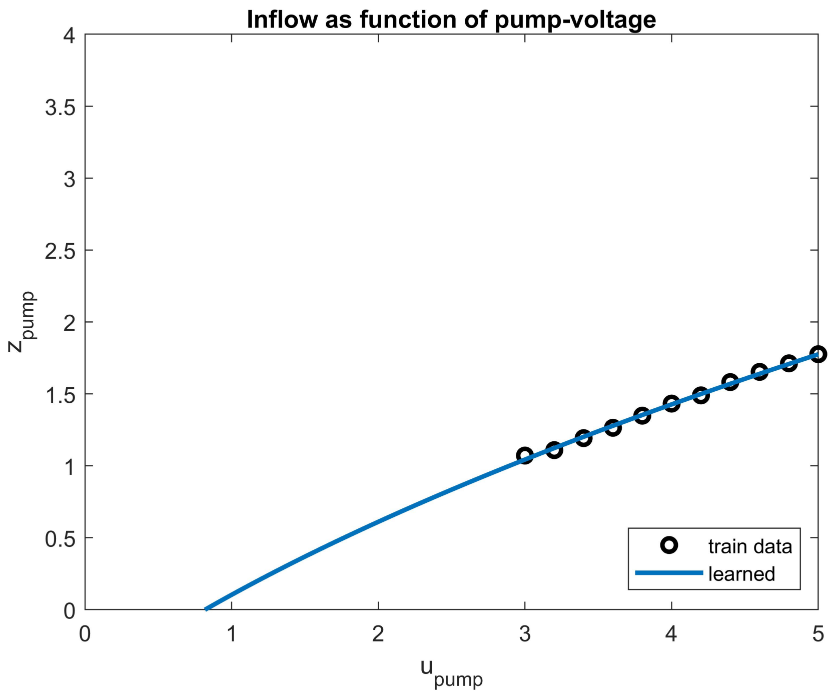

Figure 56: $u_{1,pump}/z_p$-curve of the second pump.

In table 3 the identified parameters of the two tank system are illustrated.

Parameter	**Tank 1**	**Tank 2**
$offset_{hi}$	-16.1964	-16.9361
$gain_{hi}$	-76.6903	-79.1449
$k_i[cm^{\frac{1}{2}}s^{-1}]$	0.3767	0.4029
$h_{0i}[cm]$	7.6654	6.6288
$\alpha_i[cms^{-1}]$	-1.6024	-2.3402
$\beta_i[cm^2s^{-2}]$	-0.7026	3.0621
$\gamma_i[cm^2s^{-2}V^{-1}]$	2.2686	2.7854

Table 3: Parameters of the two tank system.

6.2 Real-time Control Software - QUARC

QUARC is the most efficient way to implement real-time applications on hardware using Simulink. With the so-called *HIL Simulation Block* it's possible to communicate with the real world. In this thesis, two analog inputs are needed to collect the data from the sensors and one analog output is needed to activate the first pump. Since YALMIP isn't suited for real-time applications, another way to compute the optimization problem has to be found. In this work, the so-called *qpOASES* is used [38]. It's an open-source C++ parametric active-set algorithm for quadratic programming [23]. qpOASES solves quadratic problems of the following form:

$$\begin{aligned} \min_{\xi} \quad & \frac{1}{2}\xi^T \hat{W} \xi + \hat{c}^T \xi \\ \text{s.t.} \quad & A_{iq}\xi \le b_{iq} \end{aligned} \tag{91}$$

This means that the optimization problem

$$\begin{aligned} \min_{\substack{\Delta\bar{u}_k \in \mathbb{R}^{mN_c} \\ \bar{\varepsilon}\in\mathbb{R}}} \quad & \Delta\bar{u}_k^T(H^TQH+R)\Delta\bar{u}_k + 2\Delta\bar{u}_k^T H^T Q\bar{e}_k + \rho\bar{\varepsilon} \\ \text{subject to} \quad & W\Delta\bar{u}_k \le \bar{w} + V\bar{\varepsilon} \\ & \bar{\varepsilon} \ge 0, \end{aligned} \tag{92}$$

has to be specified in the form given in equation (91). In the case where soft constraints are not used, the optimization problem is given by

$$\begin{aligned} \min_{\xi} \quad & \frac{1}{2}\xi^T 2(H^TQHR)\xi + 2H^TQ\bar{e}_k\xi \\ \text{s.t.} \quad & W\xi \le \bar{w} \end{aligned} \tag{93}$$

where

$$\xi = \Delta\bar{u}_k. \tag{94}$$

In the case where soft constraints are used, the optimization problem is

$$\begin{aligned} \min_{\xi} \quad & \frac{1}{2}\xi^T \begin{bmatrix} 2(H^TQHR) & 0 \\ 0 & 0 \end{bmatrix} \xi + \begin{bmatrix} 2H^TQ\bar{e}_k \\ \rho \end{bmatrix} \xi \\ \text{s.t.} \quad & \begin{bmatrix} W & -V \\ 0 & -1 \end{bmatrix} \xi \le \begin{bmatrix} \bar{w} \\ 0 \end{bmatrix}, \end{aligned} \tag{95}$$

where

$$\xi = \begin{bmatrix} \Delta\bar{u}_k \\ \bar{\varepsilon} \end{bmatrix}. \tag{96}$$

6.3 Without Network

First of all, it's assumed that the control components are all directly connected to each other, in other words no network is used. The prediction and control horizons are again chosen as

$$N_p = 20 \qquad \text{and} \qquad N_c = 8. \tag{97}$$

The weighting matrices Q and R are chosen as

$$Q = \begin{bmatrix} 0.1 & 0 & \cdots & 0 \\ 0 & 0.1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0.1 \end{bmatrix} \qquad \text{and} \qquad R = \begin{bmatrix} 0.2 & 0 & \cdots & 0 \\ 0 & 0.2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0.2 \end{bmatrix}, \tag{98}$$

with their respective sizes $Q \in \mathbb{R}^{pN_p \times pN_p}$ and $R \in \mathbb{R}^{mN_c \times mN_c}$. The constraints for the pump voltage u_1 are given as

$$\begin{gathered} u_{min} \leq u_i \leq u_{max} \\ 0 \leq u_1 \leq 10V \end{gathered} \tag{99}$$

with

$$u_{min} = \begin{bmatrix} u_{k,min} \\ u_{k+1,min} \\ \vdots \\ u_{k+N_c-1,min} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad u_{max} = \begin{bmatrix} u_{k,max} \\ u_{k+1,max} \\ \vdots \\ u_{k+N_c-1,max} \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ \vdots \\ 10 \end{bmatrix}. \tag{100}$$

The constraints for the states x_1 and x_2 are given as

$$\begin{aligned} x_{min} &\leq x_i \leq x_{max} \\ h_{0i} &\leq x_i \leq 20cm + h_{0i} \qquad \text{for} \quad i = 1, 2 \end{aligned} \tag{101}$$

with

$$x_{min} = \begin{bmatrix} x_{1,min} \\ x_{2,min} \\ \vdots \\ x_{N_p-1,min} \\ x_{N_p,min} \end{bmatrix} = \begin{bmatrix} 10.17 \\ 10.53 \\ \vdots \\ 10.17 \\ 10.53 \end{bmatrix} \qquad x_{max} = \begin{bmatrix} x_{1,max} \\ x_{2,max} \\ \vdots \\ x_{N_p-1,max} \\ x_{N_p,max} \end{bmatrix} = \begin{bmatrix} 30.17 \\ 30.53 \\ \vdots \\ 30.17 \\ 30.53 \end{bmatrix}. \tag{102}$$

For the simulations using the real model, soft constraints are used as it has been shown in equation (95). The weighting parameter ρ and the array V are chosen as

$$\rho = 1000 \qquad \text{and} \qquad V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{2N_c} \\ v_{2N_c+1} \\ \vdots \\ v_{2N_c+4N_p} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{103}$$

Note that the first $2N_c$ values of the array V are responsible for the constraints on the input signal and the rest $4N_p$ values are responsible for the constraints on the states of the plant. Since the given boundaries of the pump voltage u_1 cannot be exceeded, hard constraints are used. However, for the states x_1 and x_2 soft constraints are applied. This means that the states are allowed to exceed the given boundaries without the simulation stops running. Figure 57 shows the behavior of the water level in the bottom tank in the case where no network is used.

It can be seen that the water level in the second tank follows the reference signal, but there is always a small deviation from the wanted fill height. It's shown that the deviation between the output signal and the reference signal is smaller for lower reference signal values and larger for higher reference signal values. This deviation emerges because of the inaccurate model description. Very often, the position of the assumed equilibrium point deviates from the position of the equilibrium in real life. That's why it makes sense to investigate the control behavior of the presented methods on a practical implementation as well. A possibility to compensate this deviation would be to incorporate an integral part into the optimization problem. Since the goal of this thesis is to investigate how the communication constraints affect the control performance, no compensation is made.

It can also be observed that there is a slight oscillation behavior of the water level (especially when leaving the equilibrium point). The control horizon is reduced to $N_c = 3$. As figure 58 shows, the oscillating behavior of the water level in the second tank is reduced significantly.

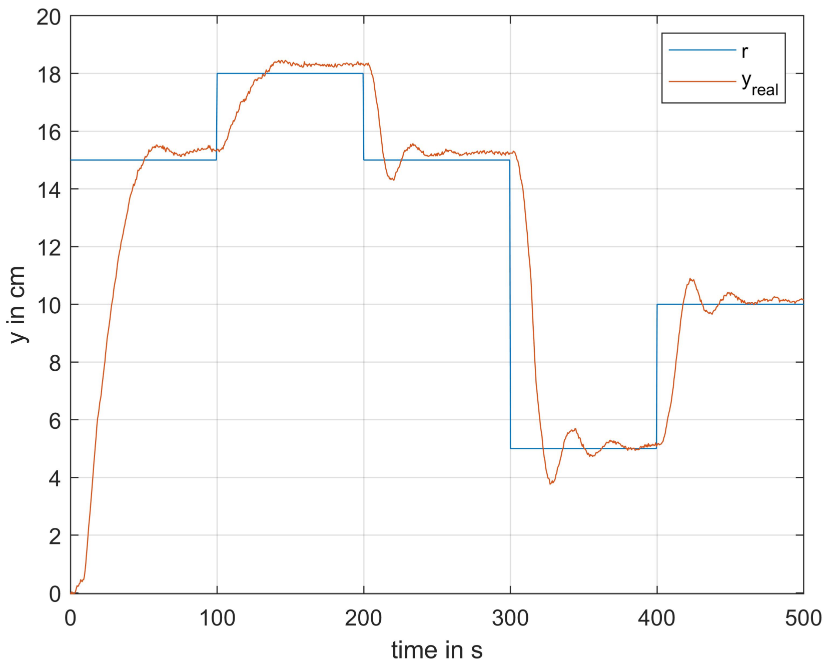

Figure 57: Water level in the bottom tank for the feedback loop with the real model without network (for $N_c = 8$).

As already mentioned in this work, in order to always have enough predicted control values it's assumed that the network-induced delays in the forward and in the backward channel are not allowed to exceed the upper bound of $N_c - 2$. This means that a control horizon of $N_c = 3$ would lead to network-induced delays which are equal to the sampling period h. A tradeoff between the oscillating behavior and the maximum value of the network-induced delays is needed. That's why the control horizon is chosen to be equal to $N_c = 6$. The boundaries of the network-induced delays result to

$$\begin{aligned} 1 \leq \tau_{sc,k} \leq N_c - 2 \quad &\Rightarrow \quad 1 \leq \tau_{sc,k} \leq 4 \\ 1 \leq \tau_{ca,k} \leq N_c - 2 \quad &\Rightarrow \quad 1 \leq \tau_{ca,k} \leq 4 \end{aligned} \tag{104}$$

and the boundaries of the RTD to

$$2 \leq \tau_k \leq N_c - 1 \quad \Rightarrow \quad 2 \leq \tau_k \leq 5. \tag{105}$$

A control horizon of $N_c = 6$ leads to network-induced delays in the forward and in the backward channel up to $2s$ a RTDs up to $2.5s$.

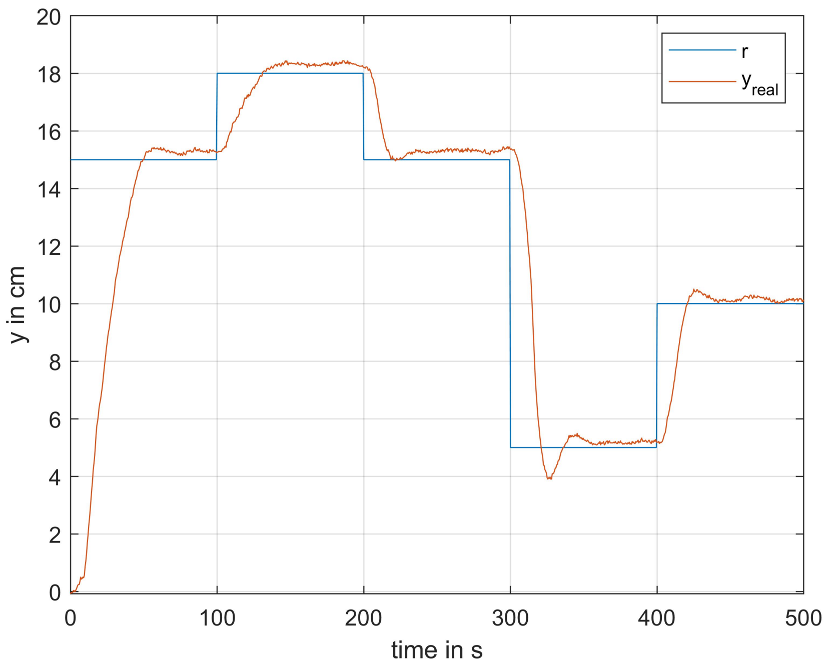

Figure 58: Water level in the bottom tank for the feedback loop with the real model without network (for $N_c = 3$).

Figure 59 shows the water level in the bottom tank for a control horizon of $N_c = 6$.

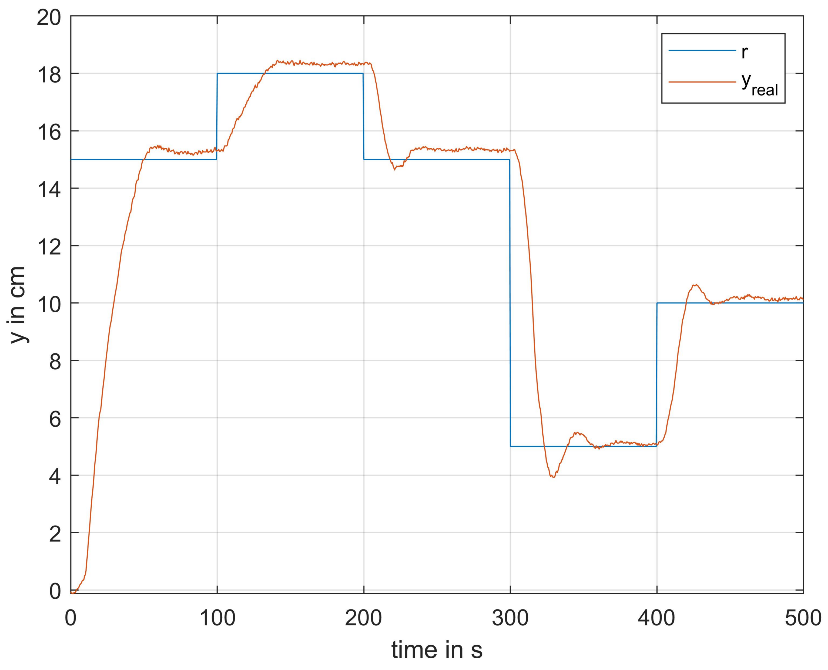

Figure 59: Water level in the bottom tank for the feedback loop with the real model without network (for $N_c = 6$).

The states and the input of the plant are illustrated in figure 60 and 61 respectively.

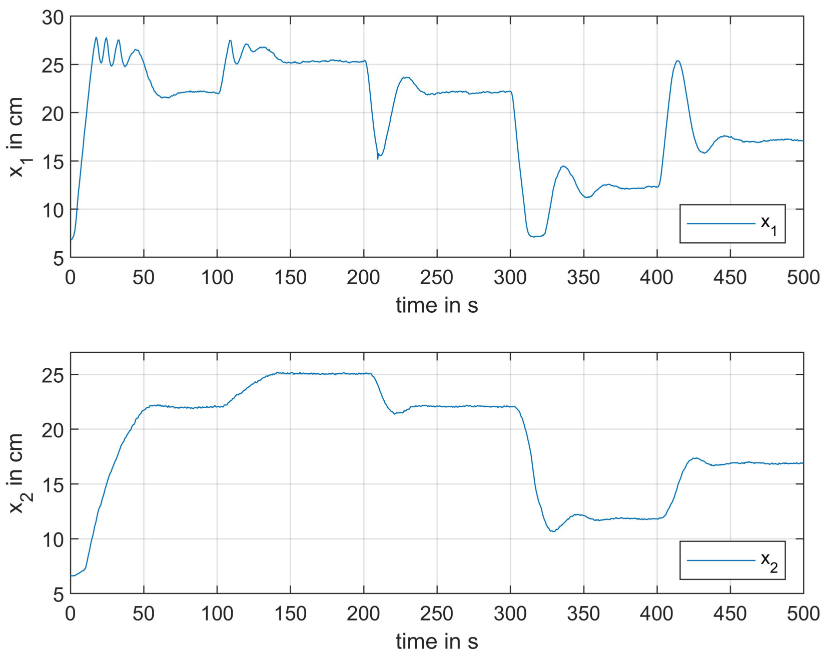

Figure 60: States of the real model in the case where no network is used (for $N_c = 6$).

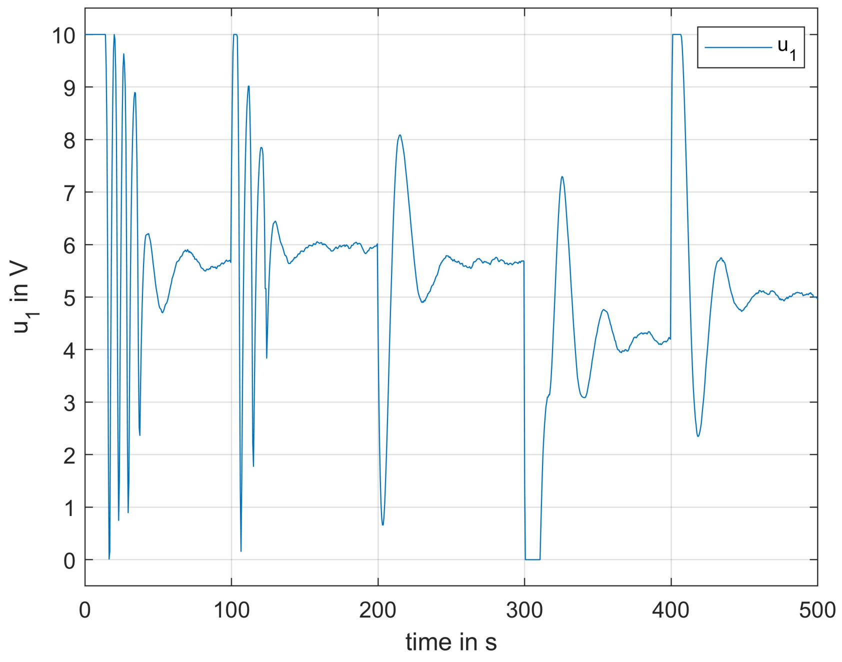

Figure 61: Input of the real model in the case where no network is used (for $N_c = 6$).

As it can be observed in figure 61, there is an unsmooth behavior of the voltage of the first pump. A possibility to improve this behavior would be to increase the values of the weighting matrix R, but it turned out that increasing the weighting matrix R leads to an increase of the deviation between the reference signal and the output of the plant.

6.4 Packet-based Control With Time Synchronization

In this section, the packet-based control approach with time synchronization is applied to the model in real life and the results are investigated. First of all, the prediction and control horizons are chosen to be equal to

$$N_p = 20 \qquad \text{and} \qquad N_c = 8. \tag{106}$$

This means that RTDs up to $3.5s$ can occur. The network-induced-delays are implemented in the same way as it has already been shown in section 5.2.3. Figure 62 shows the distribution of the network-induced delays and the distribution of the RTD. The behavior of the water level in the bottom tank is illustrated in figure 63. The output of the plant oscillates quiet strongly due to the network-induced delays. This oscillation gets worst the greater the distance from the equilibrium point. Furthermore, it can also be observed that it's more convenient to skip older packets, which has also been observed in the simulations performed with the linear and nonlinear model.

Figure 64 shows how often the MPC is running during the simulation. As expected, the controller runs more often in the case where older packets don't get skipped and in this specific case, the MPC is more inactive than active in the case where older packets get skipped.

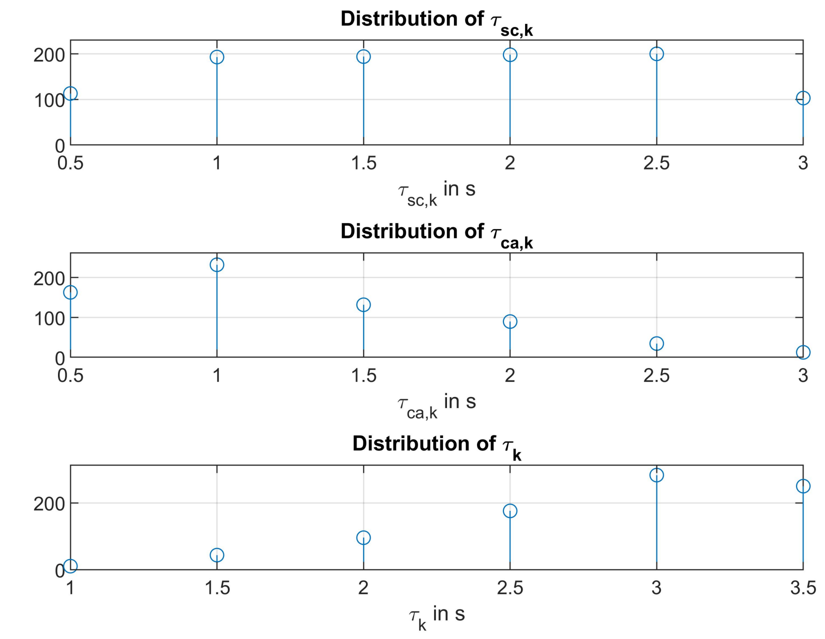

Figure 62: Distribution of the network-induced delays and the RTD ($N_c = 8$).

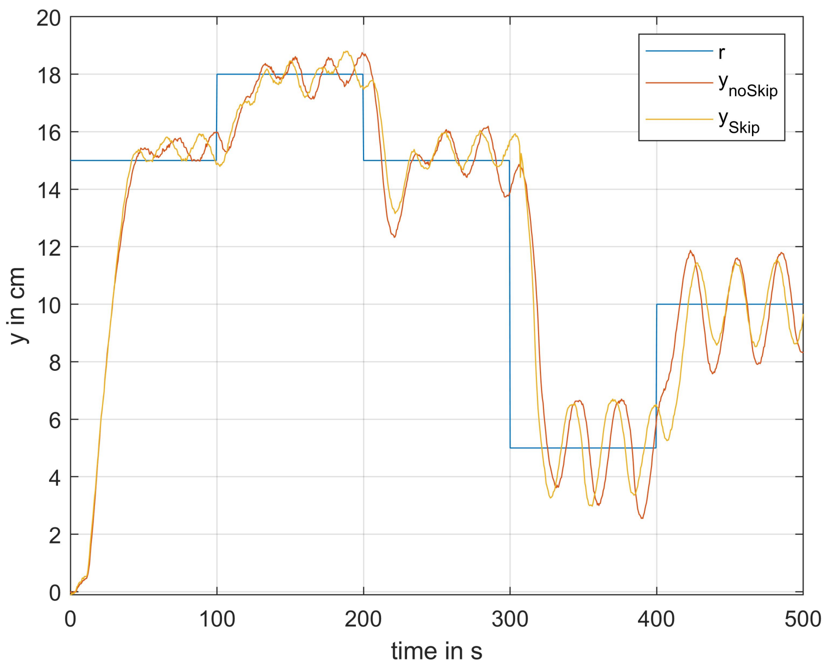

Figure 63: Water level in the bottom tank for the feedback loop with the real model using the packet-based control approach with time synchronization ($N_c = 8$).

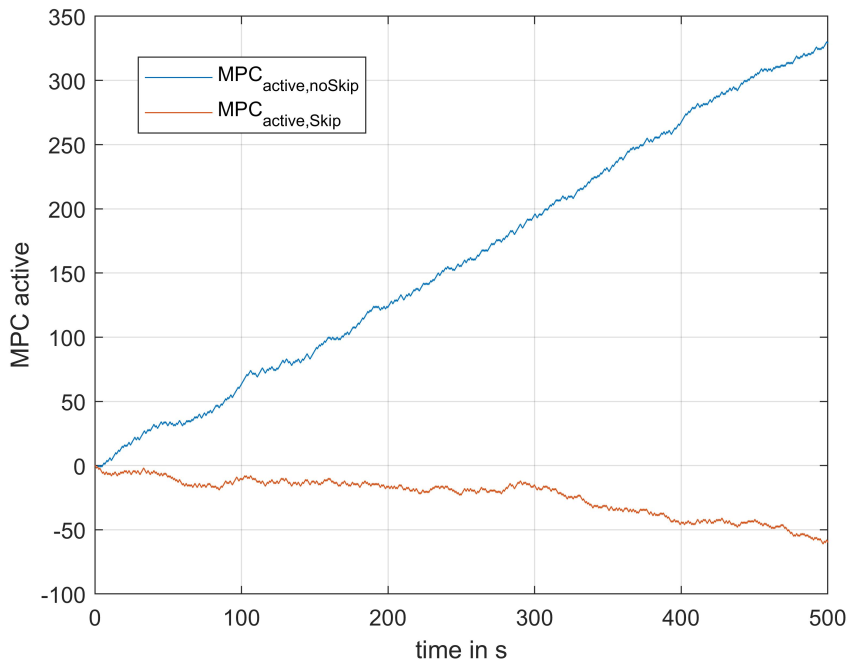

Figure 64: Active/Inactive state of the MPC during the simulation when using the packet-based control approach with time synchronization ($N_c = 8$).

In figure 65 the output of the real model with a control horizon of $N_c = 3$ is shown.

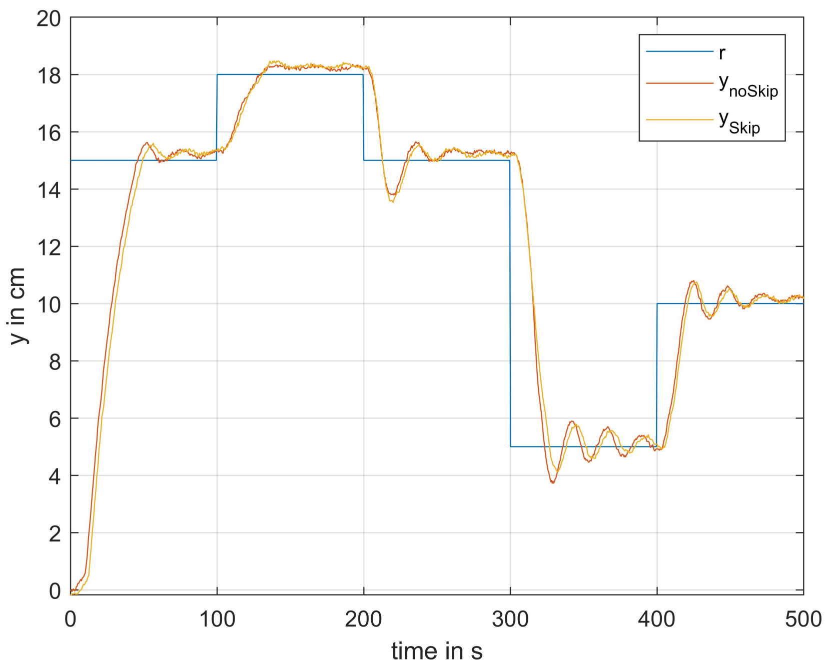

Figure 65: Water level in the bottom tank for the feedback loop with the real model using the packet-based control approach with time synchronization ($N_c = 3$).

The oscillation behavior is significantly smaller compared to the previous result, since both network-induced delays are equal to the sampling period h. As already mentioned, in order to have larger delays, the control horizon is set to $N_c = 6$. The distribution of the resulting delays is presented in figure 66.

As it can be observed, the probability of the RTD being large is very high. This means that all the simulations presented in this thesis show the worst case scenarios which can occur in a NCS, since the RTDs are always assumed to be near the upper bound. Normally, the probability of the RTD being that large is lower.

With RTDs up to $2.5s$, the water level in the second tank can still be controlled quiet good, in the case where the reference signal is near the equilibrium point. By leaving the equilibrium point, the oscillation behavior gets significantly worst. This phenomenon is perfectly shown in figure 67.

The states and the input of the plant are illustrated in figure 68 and 69 respectively. The state x_1, which is the water level in the first tank, oscillates quiet strongly during the entire simulation. Because of this, no steady state of the water level in the second tank can be reached.

Figure 70 shows how often the MPC is running during the simulation. As expected, by decreasing the upper bound of the network-induced delays, the active state of the controller is increased, since smaller delays affect the transmission of the feedback information.

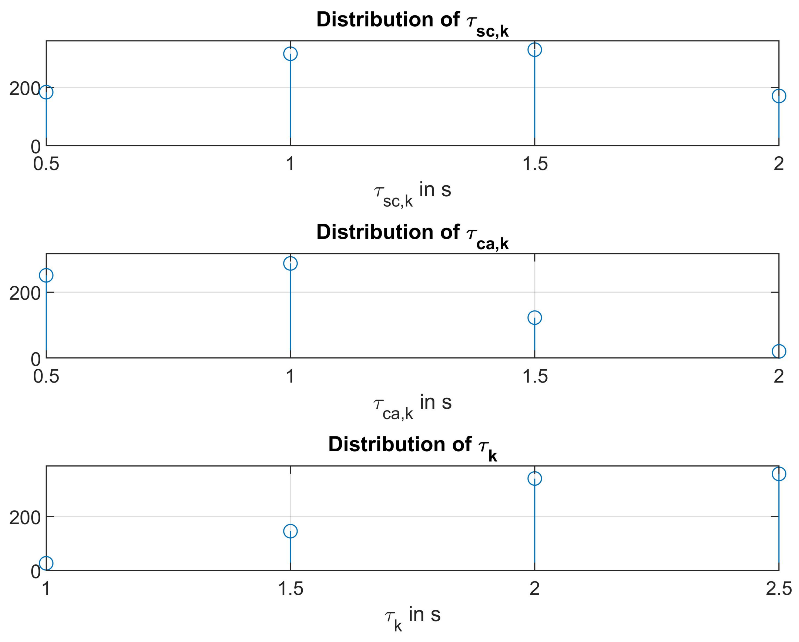

Figure 66: Distribution of the network-induced delays and the RTD.

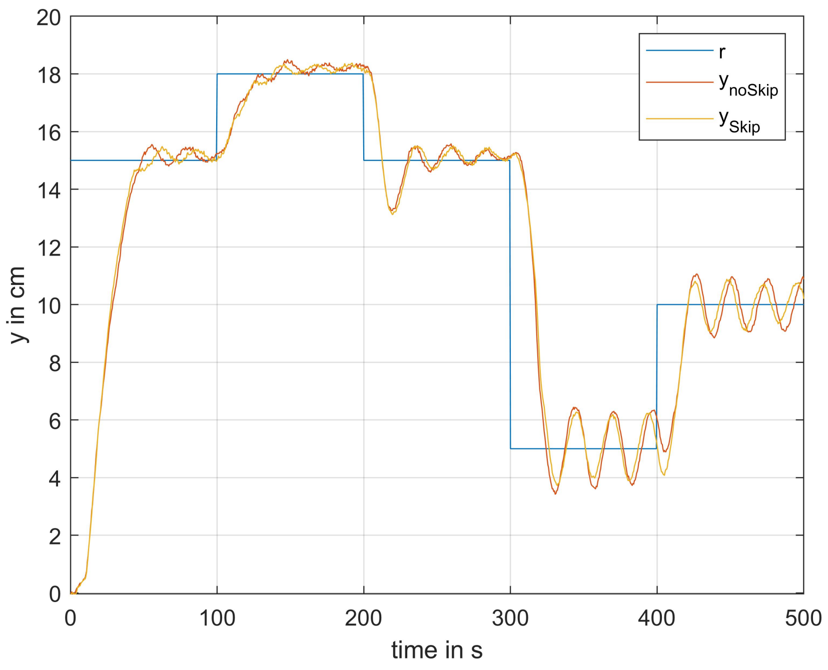

Figure 67: Water level in the bottom tank for the feedback loop with the real model using packet-based control approach with time synchronization ($N_c = 6$).

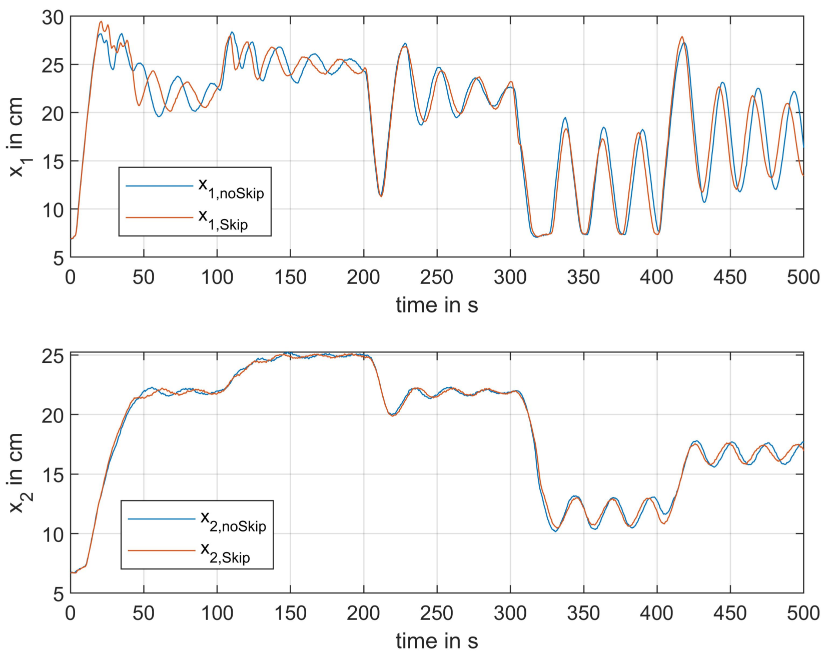

Figure 68: States of the real model for the packet-based control approach with time synchronization ($N_c = 6$).

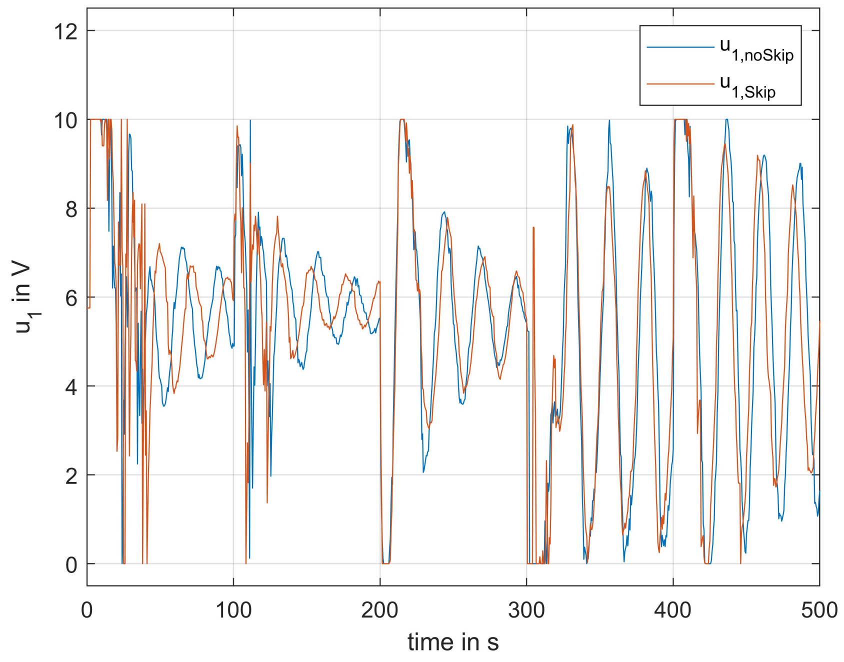

Figure 69: Input of the real model for the packet-based control approach with time synchronization ($N_c = 6$).

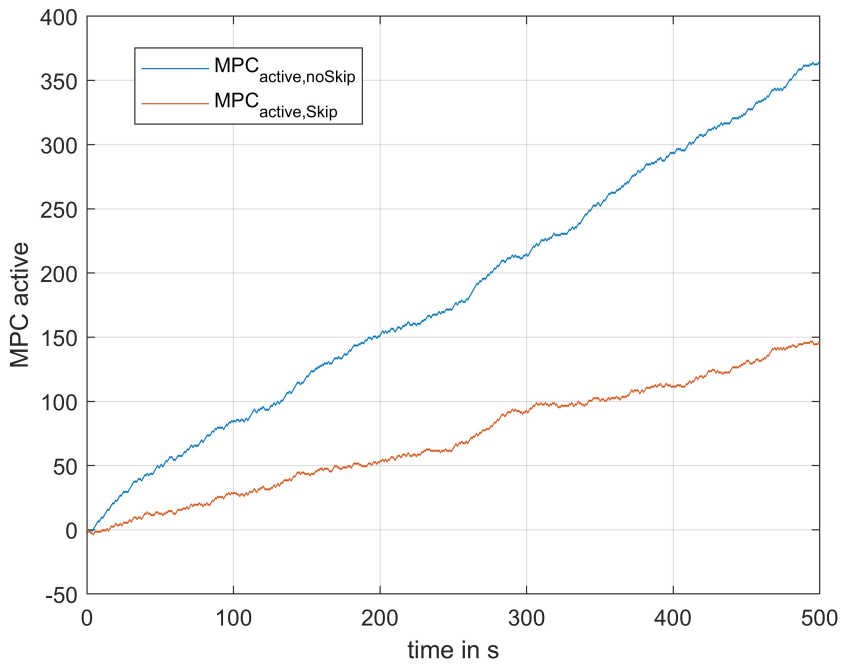

Figure 70: Active/Inactive state of the MPC during the simulation when using the packet-based control approach with time synchronization ($N_c = 6$).

6.5 Compensation of Data Packet Dropouts

In this section, data packet dropouts are taken into account and their impact on the control performance using the real model is investigated. The three methods presented in section 4.4 are applied to the system in real life as well. The implementation is the same as it has already been shown for the simulation example. Figure 71 shows how often data packet dropouts occur in the backward and in the forward channel. Whenever a packet drops, the value is increased my one. The data packets sent from the plant side to the controller side drop way more often than the packets sent by the controller. This is obviously the case, since packets containing the feedback information are sent at each time instant. The MPC only sends information whenever a packet reaches the controller side.

In figure 72, the results of the three methods are illustrated. As expected, the control prediction approach presented in section 4 leads to the best control performance, whereas the results of the first and the second method are more inaccurate. As shown, in this specific example data dropouts occur nearly 220 times in the backward channel and nearly 45 times in the forward channel. Again, the worst case scenarios are investigated, since usually the amount of data packet dropouts is way lower.

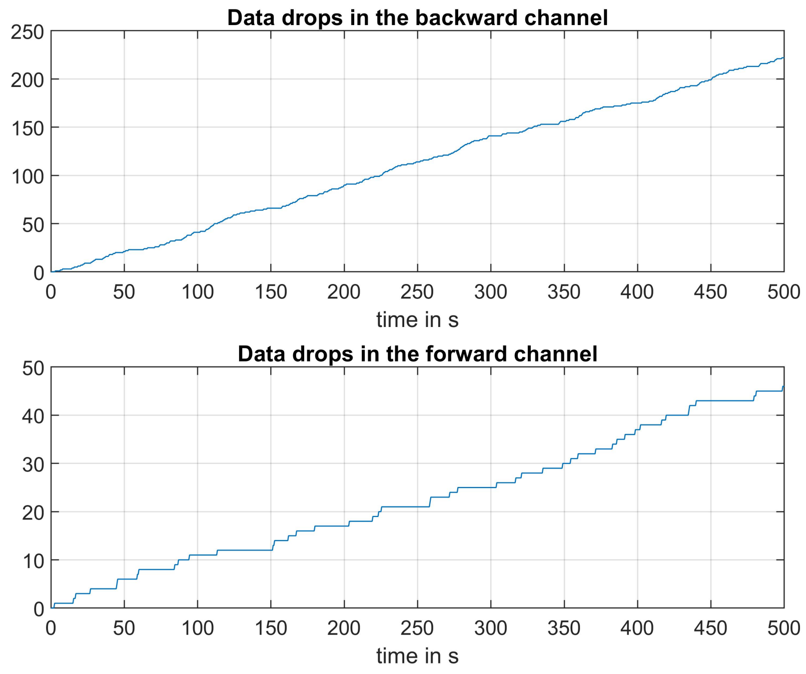

Figure 71: Data dropouts in both channels during the simulation ($N_c = 6$).

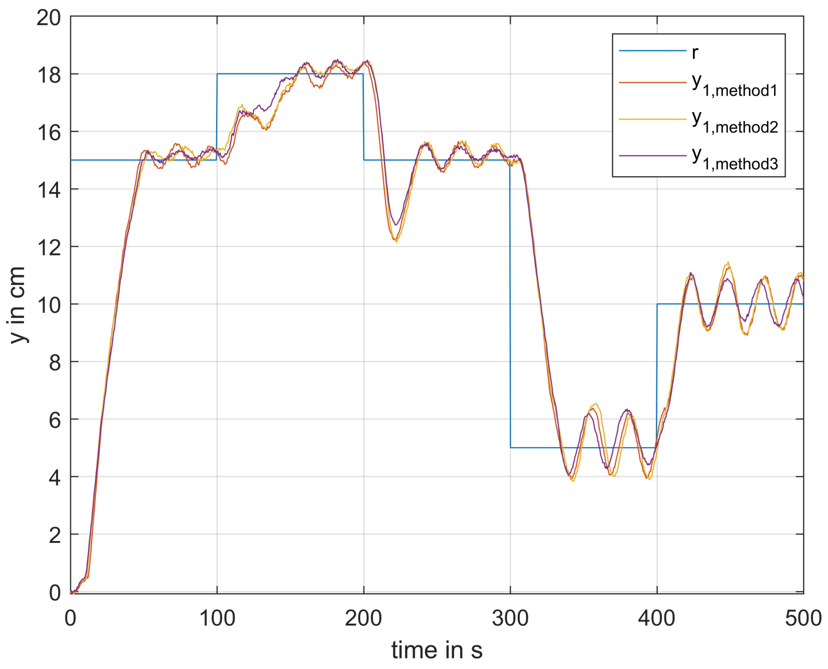

Figure 72: Water level in the bottom tank for the feedback loop with the real model with packet dropouts ($N_c = 6$).

At this point, the number of data packet dropouts is reduced, as it can be seen in figure 73. As expected, the control performance increases.

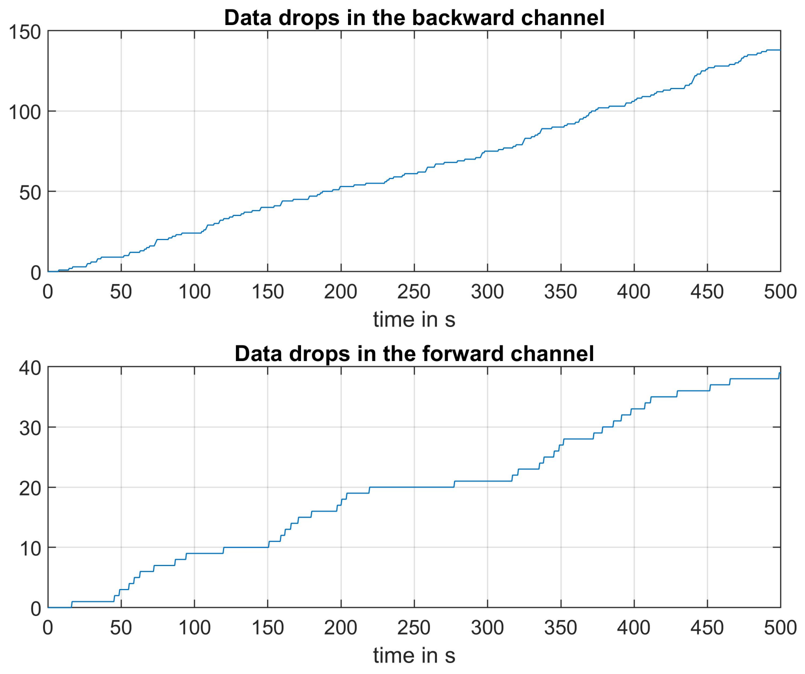

Figure 73: Data dropouts in both channels during the simulation ($N_c = 6$).

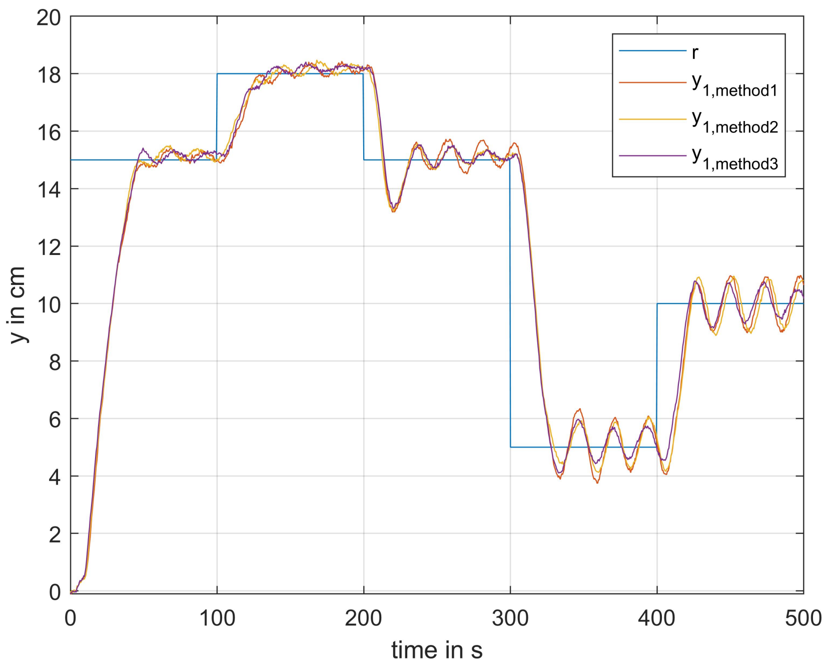

Figure 74: Water level in the bottom tank for the feedback loop with the real model with packet dropouts ($N_c = 6$).

7 Conclusions

In this thesis, several methods based on the MPC approach that can be applied for NCSs subject to random communication time-delays are presented. The UDP is used, because it is best suited for applications that require high speed and efficiency, which is normally a fundamental characteristic in control technology. With the usage of this protocol, packets may get lost during the transmission (data packet dropout) and they may be delivered out of order (data packet disorder). These communication constraints can lead to a degradation of the control performance and they have to be explicitly considered when designing the controller. This thesis is focused on the controller design and on the performance analysis within the framework of data packet dropouts and data packet disorder.

Since the MPC is able to predict the control input over a fixed number of steps, it can be utilized to compensate the random network-induced delays. To identify the unknown communication delays, the time stamp technique is used. Whenever a packet is sent either to the controller or to the actuator side, a time stamp containing the time instant of the sampled data packet is attached. It's possible to identify the network-induced delays separately (packet-based control with time synchronization) or only the RTD, which is the sum of the delays in the forward and in the backward channel, is required (packet-based control without time synchronization). With the first method, the risk of network congestion decreases, since the size of the packet sent by the controller varies over time. However, this method implies that the time synchronization between the controller and the plant is required. With the second method, the risk of network congestion increases, since the size of the packet sent by the controller is always the maximum. For these two approaches, the controller calculating delay is assumed to be zero, since it's normally very small compared to the other two delays. For fast systems, where the controller calculating delay is larger than the sampling period h, the calculating time-delay has to be compensated (as shown in this work). However, this method is not used very often, since measuring the network-induced delays using the time stamp technique takes significant time and an even higher time-delay can be caused.

A simulation example is given, in order to show the feasibility and efficiency of the proposed methods. For the communication network, an artificial network is created based on the UDP working principle. The results of the simulations performed using the linearized model are better than the results of the simulations using the nonlinear model, because of the nonlinear effects which have been neglected when calculating the linearized model. Very often, the mathematical model does not fully describe the model in real life. That's why a practical implementation is presented as well. It's shown that the proposed methods can be utilized to compensate the communication constraints that can occur in NCSs. Measuring the random time-delays with the time stamp technique is simple to implement, but it can take significantly long time and an even higher time-delay may be caused. For slow systems this restriction can simply be neglected, however, for fast systems the time stamp-induced delays can lead to a bad control performance. It's also shown that the data packet disorder problem can successfully be eliminated with a comparison rule defined at the actuator side, which is very easy to implement. In order to compensate for the random data packet dropouts, a control prediction approach should be used, since it leads to the best possible control performance. It's important to point out, that the presented methods only work if certain assumptions are fulfilled. The network-induced delays are assumed to be equal to an integer multiple of the sampling time h and they are also assumed to be upper and lower bounded. Furthermore, it is also assumed that all control components are time-synchronized, which means that they all work based on the same clock. The number of consecutive data packet dropouts must also be limited, otherwise the NCS can become an open loop. Obviously, the larger the communication delays, the worse the control performance gets.

8 References

[1] G. P. Liu , J. X. Mu , D. Rees and S. C. Chai (2006), *Design and stability analysis of networked control systems with random communication time delay using the modified MPC*, International Journal of Control, 79:4, 288-297

[2] Guo-Ping Liu, Yuanqing Xia, Jie Chen, D. Rees and Wenshan Hu, *Networked Predictive Control of Systems With Random Network Delays in Both Forward and Feedback Channels*, IEEE, VOL. 54, 2007

[3] Mohohlo S. Tsoeu, Thabo Koetje, *Unconstrained MPC Tuning for Prediction Accuracy in Networked Control Systems*, International Conference on Networking, Sensing and Control Delft, the Netherlands, April 2011

[4] Grzegorz Ewald, Mietek A. Brdys, *Model Predictive Controller for Networked Control Systems*, IFAC Proceedings Volumes, VOL. 43, Issue 8, 274-279, 2010

[5] Jing Wu, Tongwen Chen, *Model predictive control for networked control systems*, International Journal of Robust and Nonlinear Control, June 2009

[6] Yun-Bo Zhao, Guo-Ping Liu and D. Rees, *Actively Compensating for Data Packet Disorder in Networked Control Systems*, IEEE transactions on circuits and systems-II: express briefs, VOL. 57, NO. 11, November 2010

[7] Yun-Bo Zhao, Guo-Ping Liu, Yu Kang, Li Yu, *Packet-Based Control for Networked Control Systems - A Co-Design Approach*, Science Press, Beijing and Springer Nature Singapore Pte Ltd., 2018

[8] G. P. Liu, *Predictive Controller Design of Networked Systems With Communication Delays and Data Loss*, EEE transactions on circuits and systems-II: express briefs, VOL. 57, NO. 6, JUNE 2010

[9] Ke Zhang, Hai Huang and Jianming Zhang, *MPC-Based Control Methodology in Networked Control Systems*, Simulated Evolution and Learning, 814-820, 2006

[10] Brian Roffel, Ben Betlem, *Model Predictive Control*, Process Dynamics and Control, 2011

[11] Yun-Bo Zhao, Xi-Ming Sun, Jinhui Zhang and Peng Shi, *Networked Control Systems: The Communication Basics and Control Methodologies*, Mathematical Problems in Engineering, June 2015

[12] Yun-Bo Zhao, Guo-Ping Liu and D. Rees, *Design of a Packet-Based Control Framework for Networked Control Systems*, IEEE transaction on control systems technology, VOL. 17, NO. 4, July 2009

[13] Bo Yu, Yang Shi, Ji Huang, *Modified Generalized Predictive Control of Networked Systems With Application to a Hydraulic Position Control System*, Journal of Dynamic Systems Measurement and Control, May 2011

[14] Joo P. Hespanha, Payam Naghshtabrizi, Yonggang Xu, *A Survey of Recent Results in Networked Control Systems*, IEEE, VOL. 95, Issue: 1, Jan 2007

[15] Erich Kobler, *Parametrisierung Zweitankmodell*, Bachelorarbeit Telematik, Graz University of Technology, Februar 2013

[16] Martin Steinberger, Markus Tranninger, Martin Horn, Karl Henrik Johansson, *How to Simulate Networked Control Systems with Variable Time Delays?*, IFAC-PapersOnLine: Proceedings of the 21st IFAC World Congress, July 2020

[17] Martin Steinberger, *Lecture: Optimization and Control*, Graz University of Technology, 2019

[18] URL: https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html (status: 13.11.2020)

[19] URL: https://www.sciencedirect.com/topics/engineering/model-predictive-control (status: 22.09.2020)

[20] URL: https://de.mathworks.com/help/matlab/ref.html (status: 24.11.2020)

[21] Fei-Yue Wang, Derong Liu, *Networked Control Systems: Theory and Applications*, Springer, 2008

[22] Pengyi Jia, Xianbin Wang, *Nested Markov Chain - A Novel Approach to Model Network-Induced Constraints*, IEEE Annual Information Technology, Electronics and Mobile Communication Conference, November 2017

[23] URL: https://www.coin-or.org/qpOASES/doc/3.0/manual.pdf (status: 21.11.2020)

[24] Rachana Ashok Gupta, Mo-Yuen Chow, *Networked Control System: Overview and Research Trends*, IEEE, Vol. 57, Issue: 7, July 2010

[25] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla and S. S. Sastry, *Foundations of control and estimation over lossy networks*, Proceedings of the IEEE, vol. 95, no. 1, pp. 163-187, 2007

[26] M. Palmisano, M. Steinberger and M. Horn, *Optimal finite-horizon control for networked control systems in the presence of random delays and packet losses*, IEEE Control Systems Letters, vol. 5, no. 1, pp. 271-276, 2021

[27] W. P. M. H. Heemels and N. van de Wouw, *Stability and stabilization of networked control systems*, in Networked Control Systems, ser. Lecture Notes in Control and Information Sciences, Bemporad A., Heemels M., Johansson M., Ed. Springer, London, 2010, vol. 406, pp. 203-253

[28] L.Repele, R. Muradore, D. Quaglia and P. Fiorini, *Improving performance of networked control systems by using adaptive buffering*, IEEE Trans. Ind. Electron., vol. 61, no. 9, pp. 4847-4856, Sep. 2014

[29] J. Ludwiger, M. Steinberger and M. Horn, *Spatially distributed networked sliding mode control*, IEEE Control Syst. Lett., vol. 3, no. 4, pp. 972-977, Oct. 2019

[30] J. E. Normey-Rico and E. F. Camacho, *Unified approach for robust dead-time compensator design*, Journal of Process Control, vol. 19, no. 1, pp. 38-47, 2009

[31] D. Yue, Q. L. Han and J. Lam, *Network-based robust H_∞ control of systems with uncertainty*, Automatica, vol. 41, no. 6, pp. 999-1007, Jun. 2005

[32] M. Cloosterman, L. Hetel, N. van de Wouw, W. Heemels, J. Daafouz and H. Nijmeijer, *Controller synthesis for networked control systems*, Automatica, vol. 46, no. 10, pp. 1584-1594, 2010

[33] Goodwin GC, Haimovich H, Quevedo DE, Welsh JS, *A moving horizon approach to networked control system design*, IEEE Transactions on Automatic Control, vol. 49, no. 9, pp. 1427-1445, 2004)

[34] Imer OC, Yuksel S, Basar T, *Optimal control of LTI systems over unreliable communication links*, Automatica, vol. 42, no. 9, pp. 1429-1439, 2006

[35] Tang Pl, De Silva CW, *Compensation for transmission delays in an ethernet-based control network using variable-horizon predictive control*, IEEE Transactions on Control Systems Technology, vol. 14. no. 4, pp. 707-718, 2006

[36] Xia Y, Liu GP, Rees D, *Predictive control of networked systems with random delay and data dropout*, Proceedings of IEEE International Conference on Networking, Sensing and Control, Ft. Lauderdale, Florida USA, pp. 643-648, 23-25 April 2006

[37] Martin Steinberger, *Automatisierung Mechatronischer Systeme - Inverses Pendel Labormodell*, WS. 2018/19

[38] Hans Joachim Ferreau, *qpOASES User's Manual*, ABB Corporate Research, Switzerland, Dec. 2014

[39] D. Zhang, P. Shi, Q. Wang, and L. Yu, *Analysis and synthesis of networked control systems: A survey of recent advances and challenges*, ISA Trans., vol. 66, pp. 376–392, Jan. 2017

[40] X. Zhang et al., *Networked control systems: A survey of trends and techniques*, IEEE/CAA J. Automatica Sinica, vol. 7, no. 1, pp. 1–17, Jan. 2020