2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

A Robust Model Predictive Control Method for Networked Control Systems

Severin Beger Sandra Hirche

***Abstract*— Robustly compensating network constraints such as delays and packet dropouts in networked control systems is crucial for remotely controlling dynamical systems. This work proposes a novel prediction consistent method to cope with delays and packet losses as encountered in UDP-type communication systems. The augmented control system preserves all properties of the original model predictive control method under the network constraints. Furthermore, we propose to use linear tube MPC with the novel method and show that the system converges robustly to the origin under mild conditions. We illustrate this with simulation examples of a cart pole and a continuous stirred tank reactor.**

I. Introduction

The advancement in data driven control requires solutions for robust networked control systems (NCS) in order to outsource heavy computation. This enables the usage of complex, data hungry methods for small embedded systems, while providing high flexibility for system design and scalability as well as easy maintenance. However, the use of communication networks introduces additional challenges for closed loop control, such as time delays and packet losses. Extensive research has been conducted to develop stability analysis and controller design tools to cope with these flaws (cf. [1]), mostly in the setting of delays smaller than a sampling step.

For applications with high sample rates, such as robotic systems, networked communication results in much longer delays of several sample time steps. Model Predictive Control (MPC) is suited well for handling these network-related challenges [2], as its predictive nature can be explicitly used to compensate for delays and dropouts. One of the first to suggest using MPC for delay compensation was [3] for a teleoperation scenario. In [4] the authors introduced *prediction consistency*, a core property for deterministic predictive networked control methods under delays and packet losses. When extending a nominal predictive control method to be prediction consistent under delays and dropouts, properties from the nominal closed loop system, such as stability, are conserved. Several works have proposed prediction consistent methods to apply MPC in NCS, such as [5] and [6]. Some extensions concern using packet-based communication networks without acknowledgments (also known as User Datagram Protocol(UDP)-like) [7], event-based methods [8], and input to state stability [9], [10]. In [11] the authors address the difficulty of time synchronization between components in a network. More recent works aim to simplify the compensation schemes such as [12] and focus on realistic scenarios, e.g. using a WiFi network [13]. So far, little attention has been given to using robust MPC methods in the networked setting, although delays limit networked methods to react immediately to disturbances. However, many remotely controlled systems have some limited computing power and access to the most recent state measurements, which can be exploited efficiently. We address this idea in the paper at hand.

Our contribution is twofold. Firstly, we propose a novel method for ensuring prediction consistency when using a predictive control method over a communication network subject to bounded time-varying delays and packet dropouts. It relies on UDP-like communication and does not require time synchronization, thus making it suitable for general-purpose communication networks, which were not originally intended to be used for fast real-time control such as WiFi or 5G. Secondly, we investigate how tube MPC can be used in this setting to robustify a process subject to bounded additive noise. The work is structured as follows: at first, we state the considered dynamics under constraints and our assumptions on the communication network. Subsequently, we present our method for ensuring prediction consistency. Then, we derive our main results on the prediction consistency of our method and bounds on the tube due to network influences. Two simulation studies are presented to substantiate the efficacy of the proposed method. At last, we discuss our results and conclude our work.

II. Problem Statement

Consider the linear discrete-time system subject to additive noise w

$$x[k+1] = Ax[k] + Bu[k] + w[k], \quad x[0] = x_0 \tag{1}$$

where the states, inputs, and disturbances are in the sets

$$x[k] \in \mathbb{X},\ u[k] \in \mathbb{U},\ w[k] \in \mathbb{W} \qquad \forall k \in \mathbb{N}_0. \tag{2}$$

The sets $\mathbb{U} \subset \mathbb{R}^m$ and $\mathbb{W} \subseteq \mathbb{R}^n$ are compact and convex polytopes, while $\mathbb{X} \subset \mathbb{R}^n$ is a convex, bounded polyhedron.

We assume sensors and actuators to be collocated. The control loop is closed with a remote controller over a non-acknowledged UDP-like packet-based communication network, which imposes communication constraints in the form of time-varying delays and packet losses.

Additionally, we assume that the logic unit at the plant side,

S. Beger and S. Hirche are with the Chair of Information-oriented Control, TUM School of Computation, Information, and Technology, Technical University of Munich, 80333, Munich, Germany. Email: {severin.beger, hirche}@tum.de

This work was supported by the Federal Ministry of Education and Research of Germany (BMBF) as part of the project 6G-life (project identification number 16KISK002).

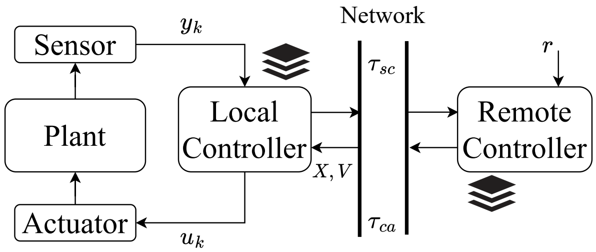

Fig. 1. Considered setup. The local controller forwards measurements to the remote side and receives actuation signals from the remote controller over a lossy network.

in the following denoted as a local controller, has enough computational power to handle our proposed algorithm and to execute a linear state feedback law.

We consider a remote controller in the network that possesses close to unlimited computational resources, runs the predictive control algorithm, and takes delays and packet losses into account. Both sides have some memory capabilities in the form of buffers. The system is depicted in fig. 1.

Consider the following assumptions on the computing units and network behavior:

- **A1)** Every message sent from sensor to controller is time-stamped with the time t_p when the measurement was taken at the plant side.
- **A2)** The sensor-to-controller delay τ_{sc} as well as the controller-to-actuator delay τ_{ca} are upper bounded[1]: $0 \leq \tau_{sc}[k] \leq \bar{\tau}_{sc}$, $0 \leq \tau_{ca}[k] \leq \bar{\tau}_{ca}$.
- **A3)** The upper bound of the round trip time $\bar{\tau}_{RTT} = \bar{\tau}_{sc} + \bar{\tau}_{ca}$ for a successful transmission without dropouts from sensor-to-controller-to-sensor is known, where $\bar{\tau}_{ca}$ includes computing time.
- **A4)** The number of consecutive packet losses over the complete loop (from sensor to actuator) is upper bounded by $\bar{n}_l \in \mathbb{N}_0$.
- **A5)** The local controller at the plant has enough computational power to execute a linear feedback policy.

Note that we do not require time synchronization. Our scheme is designed to rely exclusively on timestamps from the plant side.

In this work, we consider a robust MPC strategy for linear systems, known as tube MPC, as described, e.g., in [14]. Consider the nominal system without disturbances as

$$z[k+1] = Az[k] + Bv[k]. \tag{3}$$

The following optimal control problem (OCP) $\mathbb{P}(z)$ is solved at the remote controller:

[1]We treat all delays longer than their respective upper bound as packet losses.

$$\begin{aligned} \min_{\boldsymbol{V}} \; & \left(z[N]^T P z[N] + \sum_{j=0}^{N-1} z[j]^T Q z[j] + v[j|k]^T R v[j|k]\right) \\ \text{s.t.} \; & z[j+1] = Az[j] + Bv[j|k] \\ & z[j] \in \overline{\mathbb{X}}, v[j|k] \in \overline{\mathbb{U}}, z[N] \in \overline{\mathbb{X}}_f, \end{aligned} \tag{4}$$

where $\boldsymbol{V}[k] = \{v[0|k], \cdots, v[N-1|k]\}$ is the resulting input sequence with the notation $v[j|k]$ denoting the input at time $j+k$ computed at time k based on a measurement $x[k]$. $i \in [0, N-1]$, N is the considered horizon.

Additionally, $P = P^T \succeq 0$, $Q = Q^T \succeq 0$, and $R = R^T \succ 0$. The matrices and the terminal set $\overline{\mathbb{X}}_f$, must be chosen carefully to provide recursive feasibility and stability (c.f. [15]).

The sets $\overline{\mathbb{X}} = \mathbb{X} \ominus \mathbb{S}$, $\overline{\mathbb{U}} = \mathbb{U} \ominus K\mathbb{S}$ and $\overline{\mathbb{X}}_f = \mathbb{X}_f \ominus \mathbb{S}$ are the state and input constraints tightened with a robust positive invariant tube $\mathbb{S}$. A precalculated static feedback gain K ensures that the system under disturbance converges to the nominal trajectory. The considered feedback law at the real system is

$$u[k] = v^*[0|k] + K(x[k] - z[k]), \tag{5}$$

where $v^*[0|k]$ is the first optimal input, computed from $\mathbb{P}(z)$ with $z_0 = x[k]$, whereas the second term provides a converging behavior towards the predicted state sequence.

Because we know the round trip time $\bar{\tau}_{RTT}$ as well as the time of a measurement t_p, we can use our system model to predict the state for the next predicted time of arrival $t_{p,d}$ and solve the OCP (4) to calculate inputs for that time. Through this, we counteract communication delays. By sending the complete computed input sequence, the local controller may use the later values of the sequence as a backup in case no new control values are delivered due to a packet dropout.

The main problem arises from predicting the future state consistently without considering past predicted inputs that did not arrive at the plant side. The following definition formalizes this thought. t_a denotes the time of application at the actuator of a control input sequence $\boldsymbol{V}[t_a]$, while t_s is the time of sensing of a measurement.

Definition 1. *[4]*

(i) We call a feedback control sequence $\boldsymbol{V}[t_a]$ consistently predicted if the control sequence $[\hat{u}[t_s], \ldots, \hat{u}[t_a - 1]]$ used for the prediction of $\hat{x}[t_a]$ equals the control sequence $[u[t_s], \ldots, u[t_a - 1]]$ applied at the actuator.

(ii) We call a networked control scheme prediction consistent if at each time k the computation of $u[k]$ according to (5) *in the actuator is well defined, i.e., $k - t_a \leq N - 1$ and $\boldsymbol{V}[t_a]$ is consistently predicted.*

III. A SIMPLE CONSISTENT PREDICTION METHOD

In this section, we define the behaviors of the local controller at the plant side and of the remote controller in the network, such that they result in a prediction consistent control scheme. We use unique identification numbers for this purpose as well as a nominal and an error mode at the remote controller side.

In the following, we will denote the time at the plant and at the remote controller as t_p and t_c, respectively. All considered time stamps, delays, and dropout are a multiple of the sampling time T and thus $t_j, \tau_j, n_j \in \mathbb{N}_0 \, \forall j$. Furthermore, all input trajectories are tagged with an ID $i \in \mathbb{N}$. We denote the buffer at the plant side as B_p and at the remote controller as B_c. The packets sent over the network are labeled similarly as P_p and P_c. The buffers are used to temporarily save multiple of the corresponding packets.
Initially, the system needs to be in a safe state, e.g., an equilibrium. Our method is initiated with the first input sequence that reaches the plant side, based on the measurement x_0 at time $t_p = 0$.

A. Local Controller

First, we develop the behavior of the local control unit at the plant side. It is sample-based and executes its algorithm with a sample rate T_s. At the beginning of each cycle, the system needs to check for a new control packet from the remote controller. Let us denote this operation with the Boolean variable $m \in \{0,1\}$. Each control packet P_c includes an ID i, the corresponding desired time of application at the local controller $t_{p,d}(i)$, the current predicted trajectory of states $\boldsymbol{X}(i) = \{x[0|t_{p,d}(i)], x[1|t_{p,d}(i)], \cdots, x[N|t_{p,d}(i)]\}$ and the associated trajectory of inputs $\boldsymbol{V}(i) = \{v[0|t_{p,d}(i)], v[1|t_{p,d}(i)], \cdots, v[N-1|t_{p,d}(i)]\}$. Additionally, it contains the ID $i_{c,last}(i)$ of the last input trajectory, which was used to predict $\hat{x}[\bar{\tau}_{RTT}|t_p(i)]$. The local controller needs the latter to determine a consistently predicted input. A control packet can be denoted as $P_c = [i, t_{p,d}(i), \boldsymbol{X}(i), \boldsymbol{V}(i), i_{c,last}(i)]$. The local controller checks, whether the time of execution is smaller or equal to the current time at the plant side: $t_{p,d} \leq t_p$. If so, the contents of the new control packet are put into the buffer B_p. If not, the new packet is discarded.
Then, the controller checks the buffer for an input expected for the current time step: $\exists i \in B_p$ for which $t_{p,d}(i) = t_p$. If such an input exists, we need to check if it is prediction consistent by comparing

$$i_{c,last}(i) = i_{p,last}[t_p - 1]. \tag{6}$$

The trajectory is used, if this holds true. We update the predicted input $v^*[t_p] = v[0|t_{p,d}(i)]$, the predicted state $\hat{x}[t_p] = x[0|t_{p,d}(i)]$ and the ID of the last applied input $i_{p,last}[t_p] = i$. The internal counter of successively used inputs of a single input trajectory is (re)set to $c_p = 1$. Older control packets with $t_{p,d}(i) < t_p$ can be pruned from the buffer.
In case the input was not prediction consistent, we delete it and all trajectories for later times

$$t_{p,d}(i) > t_p \;\; \forall i \in B_p \tag{7}$$

from the plant buffer. Now, we reuse the last consistent trajectory with ID $i_{p,last}[t_p] = i_{p,last}[t_p - 1]$ and choose $v^*[t_p] = v[c_p|t_{p,d}(i_{p,last}[t_p])]$ as well as the corresponding predicted state $\hat{x}[t_p] = x[c_p|t_{p,d}(i_{p,last}[t_p])]$. Then we increment the counter of used inputs c_p by one.
If no new control packet arrives at the plant, the procedure is the same as for the situation without prediction consistency, only that the buffer is not pruned.
In any case, we measure our system state $x[t_p]$, use it to compute the input for the plant

$$u[t_p] = v^*[t_p] + K(x[t_p] - \hat{x}[t_p]) \tag{8}$$

and apply it. Next, the plant side sends a packet P_p to the controller, which contains the current measurement $x[t_p]$, the associated time of measurement t_p and the ID of the last considered input trajectory $i_{p,last}[t_p]$ resulting in $P_p = [x[t_p], t_p, i_{p,last}[t_p]]$. A state flow diagram of the plant side algorithm is given in Fig. 2.

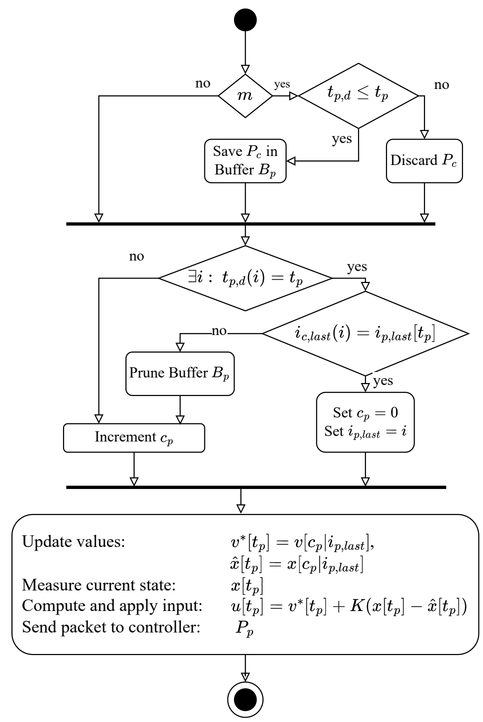

Fig. 2. Plant side algorithm. The top part handles incoming control packets, the middle part checks on prediction consistency, the bottom task computes and executes the control input.

B. Remote Controller

Next, we turn our attention towards the remote controller. It runs in two different states, which are differentiated with the Boolean variable $s \in \{0,1\}$. The first is the nominal operation for which $s = 0$. The other is the recovery mode, which is for situations where prediction consistency does not hold. Similar to the plant side, we use a Boolean variable $m \in \{0,1\}$ to determine the situations when a new message from the plant has arrived. This leaves us with four distinct situations at the remote controller.
We start with the cases, when a new measurement arrives: $m = 1$. A new ID i is computed by increasing a counter. We use it to mark the next computed control trajectory, which is supposed to be executed at the plant at $t_{p,d} = t_p + \bar{\tau}_{RTT}$. Thus, the newly arrived measurement packet is denoted as $P_p(i)$, its time stamp as $t_p(i)$ and the newest measurement as $x(i) = x[t_p(i)]$. Furthermore, the measurement packet contains the ID of the last applied input at the plant $i_{p,last}(i)$. Regardless

of the controller state s, we first need to check whether the applied input at the time of the measurement t_p matches the expected input for that point in time:

$$i[t_p] = i_{p,last}[t_p], \tag{9}$$

where $i[t_p]$ describes the ID of the input trajectory in B_c with the smallest difference of the desired application time to the measurement time as $i[t_p] = \min_{i \in B_c}(t_p - t_{p,d}(i)) \quad \text{s.t. } t_{p,d}(i) \leq t_p$.

If this condition holds, the last used trajectory was predicted consistently. Assume that we are in nominal mode. Then, the controller uses the new measurement $x[t_p]$ to predict the next state at $t_{p,d}$ by a simple forward rollout of the dynamics with the known model and the expected last inputs for the time steps between t_p and $t_{p,d}$. In nominal mode, the controller is optimistic and assumes that all sent packets since t_p have arrived. From the state prediction $\hat{x}[t_{p,d}]$ it solves the OCP (4). The ID i is packaged alongside the calculated inputs, states, time of application at the plant side, and the ID of the last used input trajectory $i_{c,last}(i) = i-1$ as control packet P_c and simultaneously saved in the buffer B_c as well as sent to the plant.

If (9) is not true, an error occurs, and the remote controller switches to or stays in recovery mode $s = 1$. We know now that all input predictions for times later than the last applied input at t_p are inconsistent. Therefore, the plant will reject them. Thus, the controller can prune its buffer B_c and delete all predictions i with $t_{p,d}(i) > t_p \ \forall i \in B_c$. Now, the buffers at the controller and plant side are equal. Next, a new input sequence, which we call a correction trajectory, needs to be computed. When in recovery mode, the controller is pessimistic. It assumes no inputs have arrived since the last confirmed input with ID $i_{p,last}[t_p]$. It uses the input trajectory of that old prediction together with the known nominal system model to calculate the new state at the time $t_{p,d}$. As before, all necessary values are saved in the local buffer and put into a packet to be sent to the plant. What is important is that the ID of the last used input is set correctly as $i_{c,last}(i) = i_{p,last}[t_p]$. This ensures that if some recovery packets from the controller to the plant are lost due to the network, the next packet that arrives will certainly be prediction consistent. Additionally, we mark this trajectory by assigning $e(i) = t_p(i)$ and save the error time as $t_e = t_p(i)$. These values are saved locally at the controller buffer B_c.

Finally, we have the situation, where a new measurement arrives and (9) is true, but the system is in recovery mode $s = 1$. Now it is necessary to check, whether the newly arrived trajectory was a correction trajectory for the existing error by comparing $e(i_{p,last}) = t_e$. If the error correction check holds, the error is now assumed to be resolved, and hence we set $s = 0$ to return to the nominal mode. In this case, we need to prune our buffer B_c, as there might be some correction trajectories saved, which were not used at the plant. Thus, we prune all packets in the controller buffer, for whose IDs the condition

$$\left(e(i) = e(i_{p,last}[t_p])\right) \wedge \left(i \neq i_{p,last}[t_p]\right) \ \forall i \in B_c \tag{10}$$

holds. After this, B_c and B_p are consistent again and we can continue with the state prediction as in the nominal case. If the error check does not hold, we stay in the recovery mode.

Now, let's consider cases where no new measurement arrives. If the controller is in recovery mode, i.e., $(m,s) = (0,1)$, it computes a new input value regardless. To do so, it takes the last available measurement $P_p(i) = P_p(i-1)$ and uses the pessimistic recovery mode strategy to predict a new state at $t_{p,d} = t_{p,d}(i-1)+1$. Then, the same OCP as usual is solved, and corresponding control values are sent, as explained above.

In case of no new measurement in nominal mode, the controller simply stays idle. The state flow diagram for the logic used at the remote controller side is given in Fig. 3.

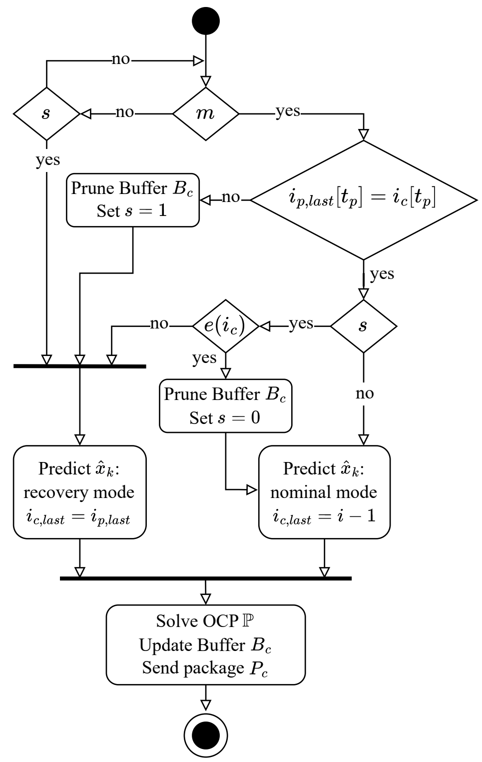

Fig. 3. Remote controller algorithm. The top part describes checks for measurement and prediction consistency, the middle part determines the state prediction, and the bottom part solves the OCP.

IV. Properties of the Method

In [4, Theorem 2.2] it is stated that a system remains practically asymptotic stable for a predictive controller, given that the nominal closed loop system is practically asymptotic stable and the used prediction method is prediction consistent. Along these lines it is noted that the prediction behavior can be separated from the robustness of the scheme.

Therefore, the analysis of our method is twofold: first, we examine all cases of occurring delays and packet losses to show the prediction consistency of our method. Secondly, we analyze the error propagation in the worst-case scenario and derive a bound for the robustly invariant tube $\mathbb{S}$. Then, when applying the findings jointly, we can extend the result to a robustly asymptotic stable system.

Lemma 4.1. *If assumptions (A1-A4) hold and the chosen time horizon for the MPC fulfills*

$$N \geq \bar{n}_l + 2\bar{\tau}_{RTT}, \tag{11}$$

then the proposed method for delay and packet loss compensation using model predictive control methodologies in networked control systems under the influence of delays and packet dropouts is prediction consistent in the sense of Definition 1.

Proof. The proof can be found in the appendix. ∎

In the following we show that the system is robustly asymptotically stable with the proposed method as well as the resulting lower bound for the disturbance invariant set.

Theorem 4.2. *Given a linear system with additive bounded disturbance as in (1) with constraints (2) and a robustly asymptotically stabilizing tube MPC law as in (5) based on the OCP (4) and a precomputed, stabilizing Feedback gain K. Assume that assumptions (A1-A5) hold and the initial state x_0 lies in the feasible set. Assume that the chosen time horizon fulfills (11). Then the system is robustly asymptotically stable and its robust invariant set is lower bounded by*

$$\bigoplus_{j=0}^{L} (A+BK)^{j-1}\mathbb{W} \subseteq \mathbb{S} \tag{12}$$

where $L = \max[N, 2\bar{n}_l + 3\bar{\tau}_{RTT} - 1]$.

Proof. From Lemma 4.1 the prediction consistency of our scheme follows. By making use of [4, Theorem 2.2] we can assume that an already nominally asymptotically stable MPC method preserves this property if a prediction consistent method is used.
Robust asymptotically stable behavior for the nominal closed loop system is provided by linear tube MPC as long as our system operates within disturbance invariant sets. To derive a bound on the tube in our scenario, we need to consider the worst possible error from the considered disturbances at the maximum prediction step N for a given input sequence.
Given a state measurement $x[0]$ at time $k = 0$, and the current error due to disturbance $e[0] = 0$, the error develops as follows

$$e[k] = \sum_{j=1}^{k} (A+BK)^{j-1} w(k-j). \tag{13}$$

Let's consider a drop from the controller to the actuator. As a result, no new input trajectory arrives at the plant, and it has to reuse the previous one. Let's assume the remote controller is notified about the prediction inconsistency with the measurement $t_p = k$. It starts to compute and send correction trajectories based on this measurement in every time step, until the error is resolved. In the worst case for disturbance, all measurements are dropped for the maximum

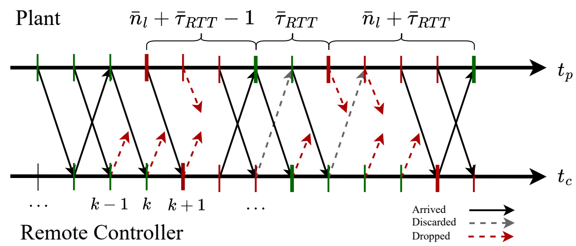

Fig. 4. Worst case delay and dropout scenario of the presented method. Arrows represent packets. $\bar{\tau}_{RTT} = \bar{n}_l = 2$.

amount of steps $\bar{n}_l$, but right before a new measurement is received, the first successful transmission of a correction trajectory based on the measurement at k takes place after $\bar{n}_l - 1$ steps. It is applied at $t_{p,cor} = k + \bar{n}_l + \bar{\tau}_{RTT} - 1$. At this point in time, the predicted state and the actual state differ by $e[t_{p,cor}] = x[t_{p,corr}] - \hat{x}[\bar{n}_l + \bar{\tau}_{RTT} - 1|k]$. Now, in the worst-case situation for the switch from recovery mode to nominal mode, the input trajectory, which is computed from the first measurement after the arrival of the correction trajectory, drops out. Again, recovery mode needs to be triggered, and a correction trajectory needs to be sent. Assuming a complete blackout for $\bar{n}_l$ steps followed by the maximum round trip time to deliver the correction trajectory, the time between the two correction trajectories arriving at the plant is $2\bar{\tau}_{RTT} + \bar{n}_l$. At this point in time, the error originating from the first error at time k has progressed to its maximum value $e[3\bar{\tau}_{RTT} + 2\bar{n}_l - 1]$. Thus, the maximum of either $3\bar{\tau}_{RTT} + 2\bar{n}_l - 1$ or the freely chosen prediction horizon N gives us the number of steps to consider for error development and, therefore, a bound on the minimum disturbance invariant set. ∎

Fig. 4 illustrates the described worst-case situation with $\bar{n}_l = \bar{\tau}_{RTT} = 2$.

V. Simulation Examples

In the following, we present two simulation examples. For both, we use a simulated network composed of two Markov chains, one for a delay state and the other for dropouts. While the latter uses two states (dropout or successful transmission), the former has three states, representing the load of the network. Each state operates on a different Weibull distribution to represent situations of low, medium, and high network traffic.
In both simulations we compare three approaches. We call the first *naive MPC*, as it does not apply our prediction consistent scheme for delay and dropout compensation. Instead, it uses any received measurement as the initial state to compute a new control trajectory using OCP (4). On the plant side, it applies the most recent trajectory it gets and iterates it, should there not be a new input. Additionally, it also considers a linear feedback law locally to counter disturbances. The other two approaches both consider our method, one without the tube and the other with it.

A. Cart Pole

The first use case is a cart pole system. We use the linearized version of a real system introduced in [16] and used in [13] amongst others. The state vector $x = [s, \alpha, \dot{x}, \dot{\alpha}]^T$

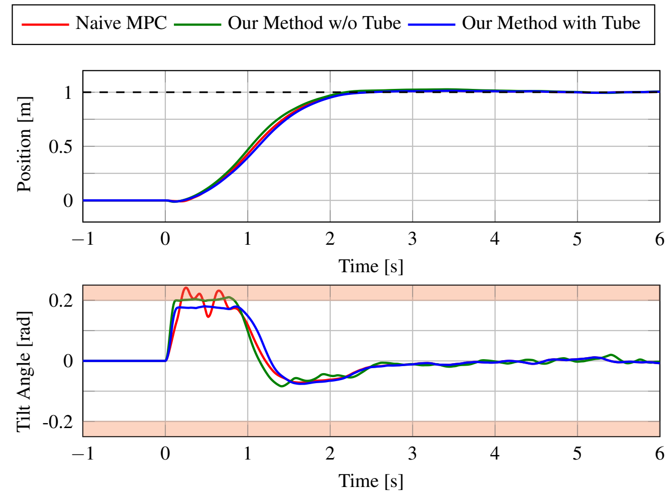

Fig. 5. Position and tilt angle of the balance robot. We compare a nominal MPC with the tube-based MPC method.

contains the position s of the device on a straight line as well as its tilt angle α. We use a sampling time of $T_s = 0.01s$. The corresponding discretized dynamic matrices are

$$A = \begin{bmatrix} 1 & 0.002 & 0.010 & 0 \\ 0 & 1.003 & 0 & 0.010 \\ 0 & 0.437 & 0.963 & 0.0353 \\ 0 & 0.551 & 0.019 & 0.981 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0.048 \\ 0.032 \end{bmatrix}. \quad (14)$$

For the sake of simplicity we assume full state measurements. We set $N = 50$, $R = 1$, and $Q = diag(10, 1000, 1, 1)$. Furthermore, we obtain K_{LQR} as well as $Q_N = P$ from solving the discrete algebraic Riccati equation with the proposed matrices. Finally, for our network, we set $\bar{\tau}_{RTT} = 7$ and $\bar{n}_l = 3$. The considered delays are $w[k] = \begin{bmatrix} 0 & 0 & 10 & 1 \end{bmatrix}^T d[k]$ with $d[k] \in [-1, 1]$. Two constraints are imposed on the system. Firstly, the tilting angle needs to be in $[-0.2, 0.2]$rad to ensure that the linearization holds. Additionally, the input needs to be between $[-20, 20]$V, which is a physical limitation of the real system.

Fig. 5 shows position x and tilt angle α over time. As our horizon is much larger than the assumed delays and dropouts, the MPC is able to stabilize the system in all three scenarios despite the imposed communication constraints. Notably, the naive MPC does not stay within the angle constraints and shows oscillatory behavior, although it has a stabilizing linear feedback locally. Furthermore, as any reaction to disturbance may be delayed up to 70ms, our approach without the tube also fails to stay within the tilt angle constraints. The combined approach, on the other hand, fulfills all constraints, although the considered tightening of the bounds seems to be overly conservative. In Fig 6 the round trip time is depicted. Despite the frequent packet losses the method manages to safely guide the system to its desired location, even under substantial disturbances.

B. Continuous Stirred Tank Reactor

Our second use case is from the process industry. It is interesting, as it is nonlinear as well as operates on a much different time scale than the cart pole example. We consider

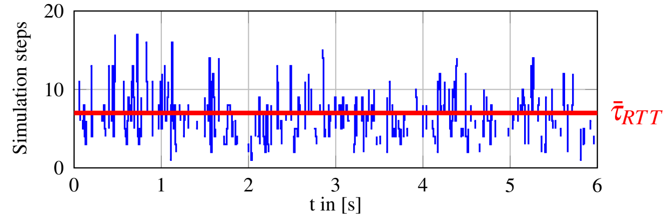

Fig. 6. Round trip time of the balance robot example. Everything above $\bar{\tau}_{RTT}$ as well as all steps where the line is broken is considered as a dropout.

an adiabatic continuous stirred tank reactor (CSTR) with the following dynamics

$$\frac{dT}{dt} = \frac{F}{V}(T_{A0} - T) + \sum_{i=1}^{3} \frac{-\Delta H_i}{\rho c_p} k_{i0} e^{\left(\frac{-E_i}{RT}\right)} C_A + \frac{Q}{\sigma c_p V_r} \quad (15)$$

$$\frac{dC_A}{dt} = \frac{F}{V_r}(C_{A0} + \Delta C_{A0} - C_A) - \sum_{i=1}^{3} k_{i0} e^{\left(\frac{-E_i}{RT}\right)} C_A \quad (16)$$

with states $x = [T, C_A]^T$, which denote the substance temperature as well as the molar concentration of the considered reactant. As input we apply or remove heat through $u = Q$, which is bounded by $|Q| \leq 10^5$ kJ/h. Furthermore, we consider an unknown, bounded time-varying uncertainty $\Delta C_{A0} \in [-0.25, 0.25]$ mol/l. All parameters of the system can be found in [17]. For our simulation we consider $T_s = 0.025$ h, $T_{end} = 0.6$ h, $N = 10$, $Q = diag(1, 1000)$, $R = 10^{-6}$. The cost function is quadratic, as shown in the OCP (4). However, we do not consider terminal costs but imply the final state x_N to lie within an ellipse specified by $\mathbb{X}_f = \{x \in \mathbb{R}^2 | (x_1 - x_{1,d})^2 + \frac{(x_2 - x_{2,d})^2}{0.2^2} \leq 1\}$, meaning that the final state shall not deviate more than 1K and 0.2mol/l from the desired state, which is $x_d = [388\text{K}, 3.59\text{mol/l}]$. For the network we choose $\bar{\tau}_{RTT} = 4$ and $\bar{n}_l = 2$ with a prediction horizon $N = 10 = 2\bar{\tau}_{RTT} + \bar{n}_l$. As our system is nonlinear, we compute a linear feedback gain for each newly computed MPC input trajectory using a linearization around the final predicted state. This gain is then forwarded alongside the inputs and state trajectories to the local controller and iterated like the predicted input. The states over time are plotted in fig. 7, whereas fig. 9 shows the phase portrait of the system. As can be seen, the naive approach shows a heavy overshoot of the desired states. Our approaches are similar in their transient response. Differences can be seen during the settling of the temperature, where the version without the linear gain leaves the terminal set $\mathbb{X}_f$, while the one with the tube, on the other hand, manages to keep the system in the desired terminal set, even though the disturbance on the molar concentration influences the system heavily. This example shows that combining delay and dropout compensation with the tube approach for disturbance rejection yields performance advantages when compared to schemes only applying one of the methods.

It must be noted that we don't use the proposed constraint tightening in this scenario and thus don't guarantee save constraint fulfillment anymore. Nonetheless, this approach seems to be an effective strategy, especially for systems like the CSTR, where time and input constraints are not of too much concern.

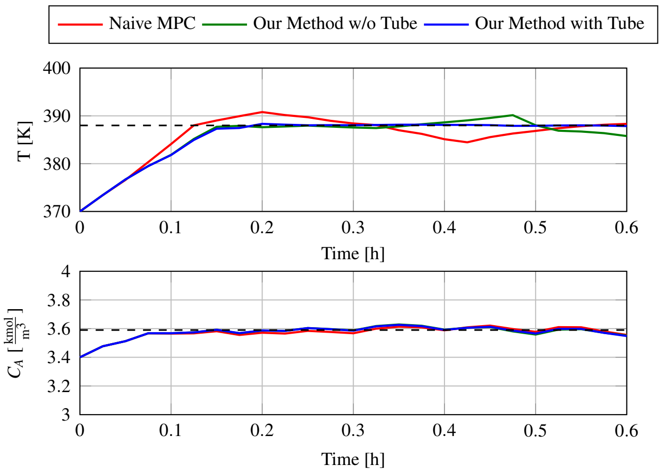

Fig. 7. State development over time for the CSTR simulation. The top depicts the temperature T, concentration C_A is shown on the bottom.

Due to its long sampling time, this simulation example is also suitable to investigate the timing of our proposed delay and dropout compensation method. In fig. 8 the network behavior under the proposed method is visualized.

VI. Discussion

Our prediction consistent method stands out in two ways. First of all, it is suitable for a broad variety of communication networks as it does not rely on any form of acknowledged messaging as in [4], [5] or time synchronization [6]. Additionally, our method performs consistently in networks, where average delays from controller to actuator are above the actual sample rate. As we do not consider a changing time horizon depending on the current delay, our method is also simpler to implement and less conservative than, e.g., [12] while showing comparable performance, as can be seen from the similar simulations on the CSTR. Secondly, it adds robustness to the networked control system using tubes, as was demonstrated in the simulations. Using a linear feedback controller on the local side allows to react to disturbances fast while staying true to the specified constraints. The main drawback of our approach is its rigidity due to buffering up to a bounded delay. This introduces artificial delays in situations, where the round trip was actually shorter, and thus it may decrease performance. Adapting online to a current best guess of the round trip delay or introducing time

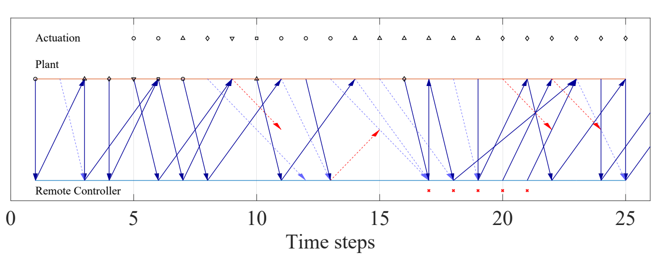

Fig. 8. Network behavior of the CSTR simulation. Dashed lines show dropped (red) and discarded (blue) packets. The geometric symbols (circles, triangles, diamonds) denote, which packet was used for the corresponding actuation. As expected, the distance from the appearance of the symbol on the plant line to its time of application on the actuation line is $\bar{\tau}_{RTT} = 4$.Red crosses on the bottom indicate the recovery mode at the remote controller.

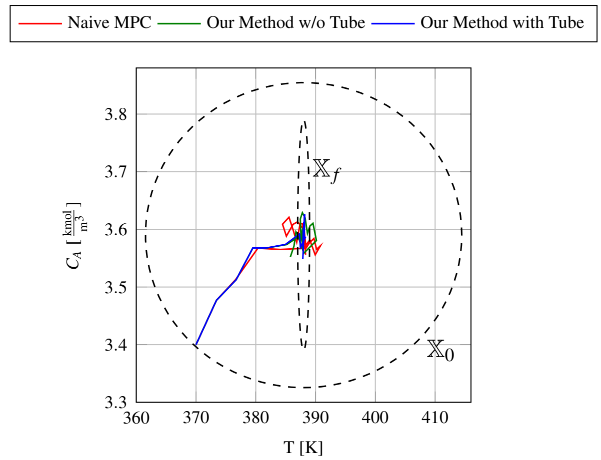

Fig. 9. Phase portrait of the CSTR simulation.

synchronization may leverage this disadvantage.

The result from theorem 4.2 on the minimum invariant disturbance set states the expected. If $N < L$ we have to tighten the constraints more than for the nominal system due to delays and packet dropouts. However, the classical approach of approximating the tube with the maximum invariant disturbance set $\mathbb{S}_\infty$ still holds. Additionally, if the horizon is long enough, it dominates the influences of delays and dropouts and thus provides the same bounds as in the nominal scenario. Therefore, the condition seems rather mild, while the method ensures robust stability.

VII. Conclusion

In the presented work we have introduced a novel predictive method for NCS that deals efficiently with communication constraints such as delays and packet losses. The technique relies on few, mild assumptions and is therefore applicable in a general setting. Thanks to the property of prediction consistency, it can be used with any type of provably stabilizing predictive control, linear or nonlinear.

Additionally, we have introduced tube MPC to the problem of Networked Predictive Control. We derived a bound on the minimum robust invariant disturbance set. This bound arises due to additional delays and longer error developments compared to the nominal setting due to the communication constraints. However, it does not restrict the method much, as in practice often a maximum positive invariant disturbance set is approximated, which contains the bounded set.

Further research may include investigations of adaptive techniques to the changing quality-of-service parameters of networks. Other areas of interest surround using online updated versions of tube MPC or learning MPC with the networked approach. Finally, further investigations on the scalability and the robustness sensitivity of the proposed method are of interest.

APPENDIX

A. Proof of Lemma 4.1

Proof. To fulfill prediction consistency in the sense of definition 1, we need to ensure two things:

1) Buffer B_c is consistent with B_p before each state prediction at the remote controller

2) The local controller at the plant side distinguishes wrongly predicted inputs and discards them

The second condition is ensured through the prediction consistency check (6) (and subsequent pruning if applicable), which relies on the IDs of the input sequences. Alas, we need to look at the ID assignment at the remote controller to guarantee the first condition.

The proof consists of analyzing the situations of packet losses and maximum delays in the backward (sensor-to-controller) and forward (controller-to-actuator) channels, both separately and together. Additionally, we need to differentiate between nominal and recovery mode.

Delays and dropouts in the backward channel do not cause prediction inconsistencies. As long as a measurement arrives in time, the remote controller uses it to predict the next input series and sends the result to the plant. If a measurement packet P_p is lost, the remote controller does not change behavior, as it will either do nothing when the controller is in nominal mode or it will rely on the previous measurement in recovery mode. On both occasions, the buffer contents of controller B_c and plant B_p stay consistent, and thus, the future state predictions are also consistent. The critical behavior occurs in the forward channel. Input delays with a total round trip time stays below $\bar{\tau}_{RTT}$ are not problematic. Under these circumstances, the computed inputs arrive before or just at their application time, and the plant holds them in its buffer B_p until they become valid. If a drop occurs in the forward channel, the buffers on the plant and controller side differ from each other. As soon as the next measurement reaches the remote controller, the prediction inconsistency is detected as the test (9) fails, and the recovery mode is activated. Through the pruning strategy (7) buffer B_c is made consistent with B_p again. By design, all predictions in recovery mode are solely based on $i_{p,last}$ of the last arrived measurement, which is the ID of the last consistently applied input at the plant. This guarantees that the next input trajectory, which arrives at the plant, is consistently predicted. In recovery mode, a new sequence is sent at every time step, regardless of a new measurement. The maximum number of time steps from the point of detection of an error at the plant until a new consistently predicted correction trajectory arrives is

$$M = \bar{n}_l + \bar{\tau}_{RTT}. \tag{17}$$

This safety procedure is executed until a new measurement arrives at the remote controller, which carries the ID of a correction trajectory. As a result, the remote controller needs to correct its buffer B_c using condition (10). This ensures, that Buffers B_c and B_p are consistent again. Now the remote controller can resume its nominal operation. In the worst case, the system must endure the maximum amount of steps as in (17). Additionally, if the system just recovered from a previous error, but the first value in nominal mode is dropped, it takes a full round trip time until the new error is detected by the remote controller. Therefore, we need (11) to hold to cope with the worst-case situation. This poses a lower bound on the prediction horizon for MPC. ■

References

[1] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A Survey of Recent Results in Networked Control Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[2] Yang Shi and Kunwu Zhang, "Advanced model predictive control framework for autonomous intelligent mechatronic systems: A tutorial overview and perspectives," *Annual Reviews in Control*, vol. 52, pp. 170–196, 2021.

[3] A. Bemporad, "Predictive control of teleoperated constrained systems with unbounded communication delays," in *Conference on Decision and Control*, 1998, 2133–2138 vol.2.

[4] L. Grüne, J. Pannek, and K. Worthmann, "A prediction based control scheme for networked systems with delays and packet dropouts," in *Conference on Decision and Control*, 2009, pp. 537–542.

[5] G. Pin, M. Filippo, and T. Parisini, "Networked MPC for constrained linear systems: a recursive feasibility approach," in *Conference on Decision and Control*, 2009, pp. 555–560.

[6] P. Varutti and R. Findeisen, "Compensating network delays and information loss by predictive control methods," in *European Control Conference*, 2009, pp. 1722–1727.

[7] G. Pin, M. Filippo, and T. Parisini, "A Coordinated Nonlinear Model Predictive Control Scheme Over Non-acknowledged Networks," *IFAC Proceedings Volumes*, vol. 43, no. 14, pp. 563–568, 2010.

[8] P. Varutti and R. Findeisen, "Event-based NMPC for networked control systems over UDP-like communication channels," in *American Control Conference*, 2011, pp. 3166–3171.

[9] R. Findeisen, L. Grüne, J. Pannek, and P. Varutti, "Robustness of Prediction Based Delay Compensation for Nonlinear Systems," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 203–208, 2011.

[10] G. Pin and T. Parisini, "Networked Predictive Control of Uncertain Constrained Nonlinear Systems: Recursive Feasibility and Input-to-State Stability Analysis," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 72–87, 2011.

[11] P. Varutti and R. Findeisen, "On the synchronization problem for the stabilization of networked control systems over nondeterministic networks," in *American Control Conference*, 2009, pp. 2216–2221.

[12] G. Pin, G. Fenu, V. Casagrande, D. Zorzenon, and T. Parisini, "Robust Stabilization of a Class of Nonlinear Systems Controlled Over Communication Networks," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 3036–3051, 2021.

[13] M. Pezzutto, M. Farina, R. Carli, and L. Schenato, "Remote MPC for Tracking Over Lossy Networks," *IEEE Control Systems Letters*, vol. 6, pp. 1040–1045, 2022.

[14] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[15] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: Theory, computation, and design*, 2nd edition. Madison, Wisconsin: Nob Hill Publishing, 2017.

[16] F. Branz, R. Antonello, M. Pezzutto, S. Vitturi, F. Tramarin, and L. Schenato, "Drive-by-Wi-Fi: Model-Based Control Over Wireless at 1 kHz," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 3, pp. 1078–1089, 2022.

[17] J. Liu, D. M. de La Pena, P. D. Christofides, and J. F. Davis, "Lyapunov-based model predictive control of nonlinear systems subject to time-varying measurement delays," in *47th IEEE Conference on Decision and Control*, 2008, pp. 4632–4639.