A prediction based control scheme for networked systems with delays and packet dropouts

Lars Grüne, Jürgen Pannek, Karl Worthmann

***Abstract*— We present a networked control scheme which uses a model based prediction and time-stamps in order to compensate for delays and packet dropouts in the transmission between sensor and controller and between controller and actuator, respectively. In order to analyze the properties of our scheme, we introduce the notion of prediction consistency which enables us to precisely state the network properties needed in order to ensure stability of the closed loop.**

I. INTRODUCTION

Motivated by numerous emerging applications, networked control systems have received considerable attention during the last years. In this paper we consider a setting where the controller, the actuator and the sensor of a closed loop system communicate over a network in which the transmission is subject to (not necessarily small) delays and packet dropouts.

In order to compensate for delays, we add a model based predictor to our controller, as in, e.g., [1], [5], [6], [7] and the references therein. Based on the most recent measurement available at the controller, the predictor computes an estimate of the future state from which the controller determines the control signal. This signal is then sent to the actuator and the prediction horizon is chosen large enough (based on an estimated bound of the transmission delay) such that the control signal can be expected to reach the actuator in time.

In order to compensate for packet dropouts (where delays which exceed the estimated bound for the transmission delay are treated as dropouts, too), the controller does not only compute a feedback value for the next sampling instant. Instead, it computes and transmits a whole feedback control sequence which is used by the actuator until the next sequence arrives.

The main difficulty in designing such a scheme lies in the fact that the control sequence used by the actuator needs to be known to the predictor before it is applied by the actuator. In fact, it needs to be known even before we can be sure whether it is successfully transmitted to the actuator. Thus, in order to ensure a faithful prediction, the main problem to be solved is to guarantee that the control sequences used for the prediction and at the actuator coincide, a property we call *prediction consistency*, cf. Definition 2.1. Introducing this property allows to separate the robustness analysis of the controller with respect to (inevitable but usually small) prediction errors from the analysis of the basic correctness of the prediction scheme. For reasons of space limitations we can only sketch the application of this separation principle in this paper, cf. Theorem 2.2. More general stability proofs based on this principle will be addressed in future papers.

In order to ensure prediction consistency, our proposed scheme contains a correction mechanism which uses appropriate time stamps. These enable the actuator to identify and discard non-consistent control sequences. If this happens, the controller is notified via an error message and can adjust the prediction control sequence. This idea is not entirely new, as similar schemes have been presented in, e.g., [1], [6]. Compared to these schemes, the advantages of our scheme are twofold: on the one hand, the scheme is simpler in the sense that no special "recovery mode" is necessary. On the other hand, the special buffer structure allows us to guarantee both fast performance if the network is working without errors and fast recovery after a network error has occured, see the discussion at the beginning of Section III.

This research was supported by DFG Grant GR1569/12 within the priority research program 1305

Lars Grüne, Jürgen Pannek and Karl Worthmann are with Mathematisches Institut, Universität Bayreuth, 95440 Bayreuth, Germany, [lars.gruene,juergen.pannek,karl.worthmann]@uni-bayreuth.de

II. SETUP AND PRELIMINARIES

We consider a discrete time nonlinear control system

$$x(n+1) = f(x(n), u(n))$$

with $x(n) \in X$, $u(n) \in U$, whose solution with $x(n_0) = x_0$ is denoted by $x(n; n_0, x_0, u)$. Typically, the model under consideration will be a discrete time model for a sampled data system. Thus, we often refer to the time instances n as sampling instances.

We assume that a controller is given, which generates a control sequence

$$\mu(x, k) \in U, \quad k = 0, \ldots, m^\star - 1,$$

for each $x \in X$, where $m^\star \geq 1$ is some fixed number. For instance, model predictive control algorithms naturally fit this setting.

An *admissible control horizon sequence* is a sequence of numbers $(m_i)_{i\in\mathbb{N}_0}$ with $m_i \leq m^\star$ for all $i \in \mathbb{N}_0$. Denoting the values

$$\sigma_k \quad := \quad \sum_{j=0}^{k-1} m_j \quad \text{(using the convention } \textstyle\sum_{j=0}^{-1} = 0\text{)}$$

we define the solutions $x(\cdot)$ of the *closed loop system* by

$$x(n+1) = f(x(n), u(n)), \tag{1}$$

where

$$\begin{aligned} &u(n) = \mu(x(\sigma_{k_n}), n - \sigma_{k_n}) \\ &\text{with} \quad k_n := \max\{k \in \mathbb{N}_0 \,|\, \sigma_k \leq n\}. \end{aligned} \tag{2}$$

It follows that in (1), (2) the feedback values $\mu(x(\sigma_{k_n}),0),\dots,\mu(x(\sigma_{k_n}),m_{k_n}-1)$ are used. Conditions under which a model predictive control scheme yields a stable closed loop system for all admissible control horizon sequences $(m_i)_{i\in\mathbb{N}_0}$ can, e.g., be found in [3], [8].

We consider the following delays in the control loop:

- $\tau_{sc}(n)$: communication delay between sensor and controller
- $\tau_c(n)$: computational delay, i.e., the time the controller needs to compute $\mu(x_0,\cdot)$ from x_0
- $\tau_{ca}(n)$: communication delay between controller and actuator

Here the index n stands for the n-th transmission or computation, respectively, in or between the corresponding devices. For ease of notation we assume that all these delays are integer values. In addition to these delays, packet dropouts can occur in each transmission.

For simplicity of exposition, let us assume that sensor, actuator and controller have synchronized clocks (this could be relaxed similar to [8, Section III.C]). Hence, at the time the measurement arrives at the controller, the delay $\tau_{sc}(n)$ is known but $\tau_c(n)$ and $\tau_{ca}(n)$ are unknown. Since these values are unknown, we will use upper bounds $\tau_c^{\max}$ and $\tau_{ca}^{\max}$ of the delays which we intend to compensate. Note, however, that in our scheme we will not need to assume

$$\tau_c(n)+\tau_{ca}(n)\le\tau_c^{\max}+\tau_{ca}^{\max} \tag{3}$$

because each violation of (3) can be treated as a packet dropout. Thus, it is enough to assume that the transmission is successful "sufficiently frequently", which will be made precise in Theorem 4.5 and Remark 4.6.

In order to compensate for delays, we add a model based predictor to the controller.[1] We assume that given a state $x(n)$ at time n, a time $\sigma>n$ and a control sequence $u_n,\dots,u_{\sigma-1}$, the predictor is able to generate a prediction $\tilde{x}(\sigma;n,x(n),u)\approx x(\sigma;n,x(n),u)$.

In the networked control scheme, the predictor will use a buffered control sequence $\tilde{u}$ in order to generate the prediction $\tilde{x}(\sigma;n_s,x(n_s),\tilde{u})$. Here n_s denotes the most recent sensor time stamp, i.e., the prediction is based on the most recent measurement $x(n_s)$ available to the predictor and σ is chosen such that delays $\le\tau_c^{\max}+\tau_{ca}^{\max}$ are compensated, details will be provided below. Abbreviating $\tilde{x}(\sigma)=\tilde{x}(\sigma;n_s,x(n_s),\tilde{u})$, the feedback value sequence $\mu(x(\sigma_{k_n}),\cdot)$ in (2) will then be replaced by $\mu(\tilde{x}(\sigma_{k_n}),\cdot)$. In order to ensure a faithful prediction we introduce the following definition.

Definition 2.1: (i) We call a feedback control sequence $\mu(\tilde{x}(\sigma_{k_n}),\cdot)$ *consistently predicted* if the control sequence $\tilde{u}(n_s),\dots,\tilde{u}(\sigma_{k_n}-1)$ used for the prediction of $\tilde{x}(\sigma_{k_n})$ equals the control sequence $u(n_s),\dots,u(\sigma_{k_n}-1)$ applied at the actuator.

[1]In the sequel we think of predictor and controller as one device. In particular, $\tau_c(n)$ will denote the computational delay of this combined device.

(ii) We call a networked control scheme *prediction consistent* if at each time n the computation of $u(n)$ according to (2) in the actuator is well defined, i.e.,

$$n-\sigma_{k_n}\le m^\star-1 \tag{4}$$

and $\mu(\tilde{x}(\sigma_{k_n}),\cdot)$ is consistently predicted.

Under this condition it is easy to state various stability results. Here we only sketch a possible result which is similar to [6, Theorem 1] (see [6] for more precise assumptions and [2] or [4] for the definition and sufficient conditions for practical asymptotic stability).

Theorem 2.2: Assume that the closed loop system (1) is practically asymptotically stable if $\mu(x(\sigma_{k_n}),\cdot)$ is replaced by $\mu(\tilde{x}(\sigma_{k_n};n_s,x(n_s),u),\cdot)$ in (2), where u is the control sequence applied by the actuator. Then, if the networked control scheme is prediction consistent, the networked closed loop system is practically asymptotically stable.

Sketch of the proof: Prediction consistency implies that $u(n)$ from (2) is well defined and that the identity $\mu(\tilde{x}(\sigma_{k_n};n_s,x(n_s),u),\cdot)=\mu(\tilde{x}(\sigma_{k_n};n_s,x(n_s),\tilde{u}),\cdot)$ holds. Hence, the networked closed loop system coincides with (1) where $\mu(x(\sigma_{k_n}),\cdot)$ is replaced by $\mu(\tilde{x}(\sigma_{k_n};n_s,x(n_s),u),\cdot)$. Since this system is assumed to be practically asymptotically stable, the assertion follows. ■

A more detailed formulation of Theorem 2.2 as well as extensions to other stability notions like, e.g., input-to-state stability, will be addressed in future papers.

Essentially, the notion of prediction consistency leads to a separation principle which allows to analyze the basic *correctness* of the prediction independently from the *robustness* of the closed loop system with respect to prediction errors and from the *accuracy* of the prediction model (which in turn depends on the delay τ_s of the sensor transmissions, cf. also [6, Assumption 2]). Hence, we can leave robustness and accuracy issues aside when analyzing the basic mechanisms of a scheme. As a consequence, the prediction consistency framework allows to thoroughly and rigorously analyze the correctness and performance of more sophisticated networked control schemes, which is our focus in the remainder of this paper.

III. DESCRIPTION OF THE SCHEME

As already pointed out in the introduction, the necessity to know the control sequence for the prediction before it is applied by the actuator poses the crucial difficulty in designing a prediction consistent networked control scheme. Even worse, due to the possible packet dropouts the control sequence is needed in the predictor before we know whether it will arrive at the actuator. There is, however, an important detail in Definition 2.1 which we exploit: the control sequences only need to coincide for those feedback sequences $\mu(\tilde{x}(\sigma),\cdot)$ which are *applied* by the actuator. Hence, by adding suitable time stamps to the transmissions we enable the actuator to determine whether the received control sequence was computed from a consistent prediction. Then, the actuator can discard erroneous control sequences and notify the controller that the prediction control sequence needs to be corrected.

In the literature, two related approaches can be found: in [6] an acknowledgment based scheme has been presented, in which the actuator confirms the receipt of each control sequence. As a consequence, the controller has to wait for the acknowledgment before the next control sequence can be computed, i.e., the transmission delay limits the time between two instances at which the control loop is closed. Hence, we propose an error message based scheme similar to [1] in which the network is assumed to work without errors until the actuator sends an error message. The main difference of our scheme compared to [1] is that our different buffer structure allows for a faster "recovery" of the scheme if an error has occurred: while in [1] after a network failure the scheme is in "recovery mode" for $m^\star$ steps, our scheme will resolve a network error in at most $\tau_c^{\max} + \tau_{ca}^{\max}$ time units plus the time needed for the transmission of the error message, cf. Remark 4.6. Furthermore, we do not need any internal "recovery mode" of the actuator.

In order to describe the scheme we specify the necessary buffer structure and the algorithms used in each component of the control loop. Although the clocks are assumed to be synchronized, it will be convenient to use different symbols n_s, n_c and n_a for the time in the sensor, controller and actuator, respectively.

The main idea of the scheme is to use time stamped transmissions in order to compensate for delays. The simplest device in our scheme is the sensor. Recall that the sensor delay affects the prediction accuracy but not the prediction consistency, cf. the discussion after Theorem 2.2.

Sensor: The sensor simply sends a time stamped measurement at each sampling instance n_s:

(S) at the sampling instance n_s the measured state and time stamp $(x(n_s); n_s)$ are sent from the sensor to the controller

While the sensor data carries only one time stamp n_s indicating the time of the measurement, the control sequences $\mu(\tilde{x}(\sigma_k), \cdot)$ sent to the actuator carry two time stamps: the first one, σ_k, indicates the time from which on the sequence is supposed to be used in the actuator and the second one, $\sigma_{pre,k}$, contains the largest time stamp of those control sequences which have been used for the prediction of $\tilde{x}(\sigma_k)$. This information will be stored in the actuator and used to detect inconsistent control sequences. Thus, prediction consistency by removing inconsistent data from the buffer is maintained.

If a missing or inconsistent control sequence is detected, an error message is sent. This way the controller is able to correct the control sequence used for prediction, if necessary. For the synchronization of the respective control sequences, the controller uses an internal variable σ_{pre} whose meaning will be described below.

Controller (including the predictor): The controller and predictor device has two buffers:

- a measurement buffer B_m for storing the most recent time stamped measurement $(x(n_s); n_s)$ received from the sensor
- a control buffer B_c for storing the time stamped feedback laws $(\mu(\tilde{x}(\sigma_k), \cdot), \sigma_k)$, which are needed in order to construct the input sequence for the prediction. We define $S_c := \{\sigma_0, \ldots, \sigma_k\}$ to be the (ordered) set of time stamps of the entries in B_c.

Note that in practice old values will, of course, be deleted from the buffer B_c. For simplicity of exposition, we will not address this issue here.

The algorithm in the controller now works as follows: At each sampling instance[2] n_c, in the Steps (C1)–(C4) the predictor estimates the future state $\tilde{x}(n_c + \tau_c^{\max} + \tau_{ca}^{\max})$ from the most recent measurement available in the measurement buffer B_m at time n_c. The control sequence $\tilde{u}$ for the prediction is constructed according to (2) from the feedback control sequences stored in B_c and the largest time stamp σ_{k_n} used in (2) is stored in σ_{pre}. The predicted state is used by the controller to compute a feedback control sequence which is sent to the actuator and stored in the buffer. Before the computation starts, an error check is performed in Step (C0): whenever the actuator detects an either missing or inconsistently predicted control sequence, an error message is sent. The error message contains the time stamps σ_{err} and σ_{cor} of the missing or inconsistent sequence and of the last consistent sequence received, respectively. If an error message is received, it is first checked whether a control sequence with time stamp σ_{err} is contained in the prediction buffer B_c, i.e., if $\sigma_{err} \in S_c$. This is the case if and only if an inconsistency occurred which has not been known before. In this case, all inconsistent control sequences are removed from the buffer B_c. At the beginning, the internal variable is initialized to $\sigma_{pre} =$ *undefined.*

At each sampling instance n_c:

(C0) if an error message $(\sigma_{err}, \sigma_{cor})$ has been received from the actuator, check whether $\sigma_{err} \in S_c$. If yes, delete all entries $(\mu(\tilde{x}(\sigma_k), \cdot), \sigma_k)$ with $\sigma_k > \sigma_{cor}$ from the control buffer B_c and set $\sigma_{pre} := \sigma_{cor}$

(C1) from the measurement $(x(n_s), n_s) \in B_m$, predict $\tilde{x}(\sigma)$ for $\sigma := n_c + \tau_c^{\max} + \tau_{ca}^{\max}$ using the prediction control sequence generated from B_c via (2)

(C2) from the predicted value, compute $\mu(\tilde{x}(\sigma), \cdot)$ (finished at time $n_c + \tau_c(n_c)$)

(C3) at time $n_c + \tau_c(n_c)$, send this control sequence, its time stamp and the time stamp of the preceding control sequence $(\mu(\tilde{x}(\sigma), \cdot); \sigma; \sigma_{pre})$ to the actuator

(C4) set $\sigma_{pre} := \sigma$ and add $(\mu(\tilde{x}(\sigma), \cdot), \sigma)$ to the control buffer B_c

Actuator: The actuator has the following buffer:

- a control buffer B_a for storing the time stamped feedback laws $(\mu(\tilde{x}(\sigma_k), \cdot), \sigma_k, \sigma_{pre,k})$ received by the controller. We define $S_a := \{\sigma_0, \ldots, \sigma_k\}$ to be the (ordered) set of time stamps which are contained in B_a.

[2]The scheme is easily extended to the case when the controller only computes a feedback control sequence at a subset of sampling instances which could also be chosen dynamically.

This buffer is similar to the control buffer B_c in the controller but also stores the σ_{pre} time stamps. Like in the controller, old values will be deleted from B_a in practice. Again, for simplicity of exposition, we will not address this issue here.

We assume that the actuator is able to insert a transmitted feedback value sequence $(\mu(\tilde{x}(\sigma), \cdot), \sigma, \sigma_{pre})$ at the correct position σ into the buffer, i.e., the set S_a remains ordered after inserting σ. This enables us to use feedback value sequences which arrive in the wrong order (with respect to the time stamp σ) due to the transmission delay.

In the actuator, we need to insert the arriving sequences into the buffer, detect missing and remove inconsistent sequences and apply the control value to the plant. This is done by the three steps of the following algorithm.

At each sampling instance n_a:

```
(A1) insert all time stamped feedback sequences
     $(\mu(\tilde{x}(\sigma), \cdot), \sigma, \sigma_{pre})$ which arrived at the actuator
     since the previous sampling instance $n_a - 1$ and satisfy
     $\sigma \geq n_a$ (sequence arrived in time) into the buffer $B_a$
     at the correct position.
(A2) if $n_a \neq 0$ check whether there is $\sigma_i \in S_a$ with $n_a = \sigma_i$
         if yes, check whether $\sigma_{pre,i} = \sigma_{i-1}$
             if no, remove $(\mu(\tilde{x}(\sigma_i), \cdot), \sigma_i, \sigma_{pre,i})$ from $B_a$
             and send an error message $(\sigma_{err}, \sigma_{cor}) =$
             $(n_a, \max\{\sigma_k \in S_a \,|\, \sigma_k < n_a\})$ to the controller
         if no, send an error message $(\sigma_{err}, \sigma_{cor}) =$
             $(n_a, \max\{\sigma_k \in S_a \,|\, \sigma_k < n_a\})$ to the controller
(A3) compute the control value $u(n_a)$ from $\mu \in B_a$ via (2)
```

Note that formula (2) used in Step (A3) requires $n_a \leq \max(S_a \cap \{0, \ldots, n_a\}) + m^\star - 1$ for successful computation of $u(n_a)$. We will later derive conditions which guarantee this inequality.

In words, step (A2) of this algorithm does the following: whenever a transition from one feedback sequence $\mu(\tilde{x}(\sigma_{i-1}), \cdot)$ to the next sequence $\mu(\tilde{x}(\sigma_i), \cdot)$ occurs in the control sequence, it is checked whether $\sigma_{pre,i} = \sigma_{i-1}$ holds. If this is the case, then the actuator will use the new sequence. If $n_a = 0$, no check is performed, because no previous sequence is available in the buffer.

If, however, $\sigma_{pre,i} \neq \sigma_{i-1}$ holds, then the algorithm detects an inconsistency, deletes this sequence from the buffer and thus continues to use $\mu(\tilde{x}(\sigma_{i-1}), \cdot)$. In addition an error message containing the time stamps σ_{err} of the inconsistent sequence and σ_{cor} of the last correct sequence is sent to the controller.

If no feedback sequence $\mu(\tilde{x}(\sigma_i), \cdot)$ with $n_a = \sigma_i$ is present in the buffer, then the actuator assumes that this sequence has been lost and thus an inconsistency will occur at some later time. Hence, an error message containing the time stamp of the missing sequence and of the last correct sequence is sent to the controller. In step (C0), the check $\sigma_{err} \in S_c$ enables the controller to decide whether the sequence was indeed lost, in which case an inconsistency will occur at some later time and consequently the usual error handling is performed in the controller.

Observe that if an error message $(\sigma_{err}, \sigma_{cor})$ is sent at some time $n_a = \sigma_{err}$, then error messages with identical σ_{cor} are sent at each time $\tilde{n}_a \in \{\sigma_{cor} + 1, \ldots, n_{cons}\}$, where n_{cons} is the smallest time at which a sequence with $\sigma_{pre} = \sigma_{cor}$ is found in the buffer. In particular, since the time stamps σ_{pre} of the sequences sent in (C4) are strictly increasing unless an error message arrives, the occurrence of an error triggers an error message in each sampling instance until one of the messages reaches the controller, the prediction sequence is corrected and the corrected control sequence is received and processed in the actuator.

For simplicity of exposition, we assume that the scheme starts at $n_a = 0$ and that $0 \in S_a$ and $0 \in S_c$ for all times $n_a \geq 0$ and $n_c \geq 0$, respectively. This means that the control sequence $\mu(\tilde{x}(0), \cdot)$ has been computed, successfully transmitted, and is used in the actuator at time $n_a = 0$. In particular, this implies that S_c and S_a are never empty. This is always physically possible if the controlled process can be stopped until the first feedback control sequence has been computed and successfully transmitted. If the process to be controlled is already running when the controller is turned on, this can be obtained by applying a default control value at the plant until the first successful transmission from the controller to the actuator and resetting all times $n_s = n_c = n_a$ to 0 at this time instant.

The following figures illustrate the scheme graphically. Figure 1 shows the transmissions without errors. The controller (C) starts computing at time n_c based on the last received measurement from the sensor (S). The computation and transmission time $\tau_c + \tau_{ca}$ are within the maximal allowed interval $\tau_c^{\max} + \tau_{ca}^{\max}$ (indicated by the dashed line), hence the control sequence arrives in time.

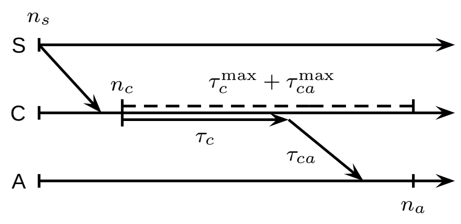
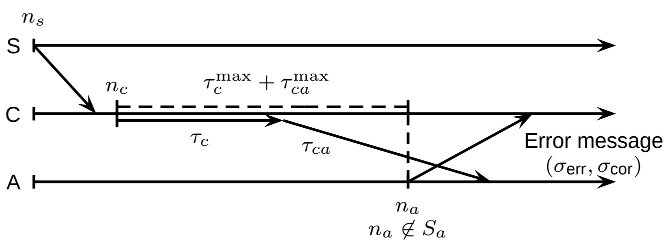


Fig. 1. Operation of scheme when no error occurs

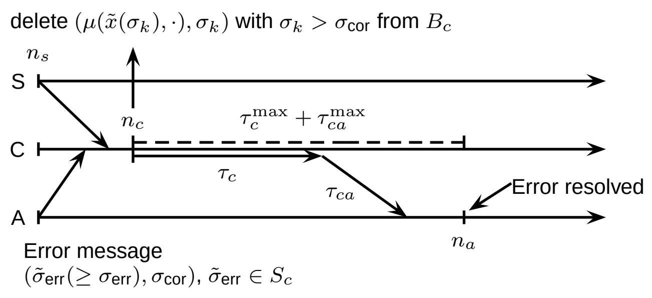
Figure 2 shows a situation in which an error occures. Here the sequence arrives too late at the controller, i.e., at a time later than $n_a = n_c + \tau_c^{\max} + \tau_{ca}^{\max}$ and is thus not inserted into the buffer B_a. The actuator detects this missing sequence at time n_a, sends an error message with the values $\sigma_{err} = n_a$ and $\sigma_{cor} = \max\{\sigma_k \in S_a \,|\, \sigma_k < n_a\}$, and continues using the feedback sequence with time stamp σ_{cor} until the error is resolved.

Finally, Figure 3 shows how this error is resolved. Upon arrival of an error message (possibly a later one than the one shown in Figure 2 but with identical σ_{cor}) the prediction sequence is updated at time n_c and the next feedback value sequence is computed based on the corrected prediction. If this sequence arrives at the actuator in time (which is the situation in the figure), the error is resolved at time $\sigma = n_c + \tau_c^{\max} + \tau_{ca}^{\max}$ and the sequence $\mu(\tilde{x}(\sigma), \cdot)$ is used.



n_s
S
C
A
n_c
$\tau_c^{\max} + \tau_{ca}^{\max}$
τ_c
τ_{ca}
Error message
$(\sigma_{\text{err}}, \sigma_{\text{cor}})$
n_a
$n_a \notin S_a$

Fig. 2. Delay which leads to an error



delete $(\mu(\tilde{x}(\sigma_k), \cdot), \sigma_k)$ with $\sigma_k > \sigma_{\text{cor}}$ from B_c
n_s
S
C
A
n_c
$\tau_c^{\max} + \tau_{ca}^{\max}$
τ_c
τ_{ca}
Error resolved
n_a
Error message
$(\tilde{\sigma}_{\text{err}}(\geq \sigma_{\text{err}}), \sigma_{\text{cor}})$, $\tilde{\sigma}_{\text{err}} \in S_c$

Fig. 3. Resolving the error of Figure 2

IV. ANALYSIS OF THE SCHEME

In this section we analyze the prediction consistency of our proposed scheme. We proceed in three steps: first, in Lemma 4.2 we show that only consistently predicted feedback control sequences are applied by the actuator. Second, in Lemma 4.3 we derive a bound on the time needed for the error recovery of our scheme. Third, in Lemma 4.4 we give conditions under which (4) is satisfied on a finite interval. These three steps are then put together in Theorem 4.5 providing conditions on the network ensuring prediction consistency of our scheme.

Since the time stamp sets S_c and S_a vary with time, it will be convenient to define $S_c(n_c) := \{\sigma_0, \ldots, \sigma_k\}$ and $S_a(n_a) := \{\sigma_0, \ldots, \sigma_{\tilde{k}}\}$ to be the ordered set of time stamps which are contained in S_c and S_a after step (C4) and (A3) of the respective algorithms have been executed. Since in both algorithms no time stamp once removed from these sets can be inserted again, the inclusions

$$\begin{array}{rcl} S_c(\tilde{n}_c) \cap \{0, \ldots, n_c\} & \subseteq & S_c(n_c) \cap \{0, \ldots, n_c\} \\ S_a(\tilde{n}_a) \cap \{0, \ldots, n_a\} & \subseteq & S_a(n_a) \cap \{0, \ldots, n_a\} \end{array} \qquad (5)$$

follow for all $\tilde{n}_c \geq n_c \geq 0$ and all $\tilde{n}_a \geq n_a \geq 0$.

Since both the control sequence applied by the actuator and the control sequence used for prediction are generated via (2), we first clarify which feedback sequences (which are uniquely determined via the entries in the respective time stamp sets S_c and S_a) are actually needed in the respective computations of $u(n)$ in (2).

Lemma 4.1: (i) The control values used for the prediction in step (C1) at time $n_c \in \{n_s, \ldots, \sigma\}$ are uniquely determined by the feedback sequences corresponding to time stamps in the set

$$S_c(n_c) \cap \{\sigma_c, \ldots, \sigma - 1\},$$

where $\sigma_c := \max\{s \in S_c(n_c) \,|\, s \leq n_s\}$.

(ii) Let $n_1, n_2 \in \mathbb{N}_0$ with $n_1 \leq n_2$. Then, the control values applied at the plant in Step (A2) at time $n_a \in \{n_1, \ldots, n_2\}$ are uniquely determined by the feedback sequences corresponding to time stamps in the set

$$S_a(n_2) \cap \{\sigma_a, \ldots, n_2\},$$

where $\sigma_a := \max\{s \in S_a(n_2) \,|\, s \leq n_1\}$.

Proof: (i) This assertion follows immediately from (2).

(ii) From (2) it follows that the control value used at time n_a is uniquely determined by $\max\{s \in S_a(n_a) \,|\, s \leq n_a\}$. From (5) it follows that this value coincides with $\max\{s \in S_a(n_2) \,|\, s \leq n_a\}$ which implies the assertion. ■

From this lemma it follows that prediction consistency is guaranteed if (4) and

$$S_c(n_c) \cap \{\sigma_c, \ldots, \sigma - 1\} = S_a(\sigma) \cap \{\sigma_a, \ldots, \sigma - 1\} \quad (6)$$

hold[3] for all $n_c \geq 0$, using the notation from Lemma 4.1 with $n_1 = n_s$, $n_2 = \sigma - 1$, and $\sigma = n_c + \tau_c^{\max} + \tau_{ca}^{\max}$. The following lemma shows that this condition is satisfied for those feedbacks control sequences which are actually used in the actuator.

Lemma 4.2: Consider times n_c and n_a such that $\sigma = n_c + \tau_c^{\max} + \tau_{ca}^{\max} = n_a$ holds and let n_s be the time stamp of the measurement used in the controller in step (C1) at time n_c. Assume that $\sigma \in S_a(\sigma)$ holds in step (A3), i.e., that the computed feedback $\mu(\tilde{x}(\sigma), \cdot)$ is applied at the plant. Then (6) is satisfied, i.e., only consistently predicted feedback values $\mu(\tilde{x}(\sigma), \cdot)$ are accepted in the actuator.

Proof: Let $S_a(\sigma) = \{\sigma_0, \ldots, \sigma_k\}$ and $S_c(n_c) = \{\tilde{\sigma}_0, \ldots, \tilde{\sigma}_{\tilde{k}}\}$, both numbered in increasing order. We show the equality

$$S_a(\sigma) \cap \{0, \ldots, \sigma - 1\} = S_c(n_c) \cap \{0, \ldots, \sigma - 1\} \quad (7)$$

which implies (6).

In order to show (7), let $S_t(n_c) = \{\hat{\sigma}_0, \ldots, \hat{\sigma}_{\hat{k}}\}$ be the set of all σ-values for which steps (C0)–(C4) have been performed up to time n_c, i.e., which have been computed and transmitted to the actuator. Clearly, both $S_a(\sigma)$ and $S_c(n_c)$ are subsets of $S_t(n_c)$.

More precisely, the values

$$\hat{\sigma}_i \in (S_t(n_c) \setminus S_a(\sigma)) \cap \{0, \ldots, \sigma - 1\}$$

are exactly the values transmitted to the actuator which were either removed or not received. The values

$$\hat{\sigma}_i \in (S_t(n_c) \setminus S_c(n_c)) \cap \{0, \ldots, \sigma - 1\}$$

are exactly those values which were inserted in step (C4) but removed in step (C0) at some later time.

In order to prove (7), we show

$$\hat{\sigma}_j \notin S_a(\sigma) \;\;\Leftrightarrow\;\; \hat{\sigma}_j \notin S_c(n_c)$$

for all $\hat{\sigma}_j \in S_t(n_c) \cap \{0, \ldots, \sigma - 1\}$.

[3]Under the somewhat artificial condition "$\mu(\tilde{x}(s), m) \neq \mu(\tilde{x}(s+m), 0)$ for all $s = 0, \ldots, \sigma - 1$ and all $m = 0, \ldots, m^\star$" Equation (6) is also necessary for prediction consistency.

We first show that every $\hat{\sigma}_j \in S_t(n_c) \cap \{0, \dots, \sigma - 1\}$ with $\hat{\sigma}_j \notin S_a(\sigma)$ satisfies $\hat{\sigma}_j \notin S_c(n_c)$. Indeed, if $\hat{\sigma}_j \notin S_a(\sigma)$ then error messages with $\sigma_{cor} < \hat{\sigma}_j$ are sent at each time instant in a discrete interval $I \ni \hat{\sigma}_j$. Since by assumption $\sigma \in S_a(\sigma)$, one of these error messages must have reached the controller at some time instant $\bar{n}_c \le n_c$, because otherwise the prediction control sequence would not have been corrected and $\mu(\tilde{x}(\sigma), \cdot)$ would have been inconsistent, thus $\sigma \notin S_a(\sigma)$. While processing this error message, $\mu(\tilde{x}(\hat{\sigma}_j), \cdot)$ was removed from B_c, hence $\hat{\sigma}_j \notin S_c(\bar{n}_c)$ implying $\hat{\sigma}_j \notin S_c(n_c)$ by (5).

Conversely, we show that every $\hat{\sigma}_j \in S_t(n_c) \cap \{0, \dots, \sigma - 1\}$ with $\hat{\sigma}_j \notin S_c(n_c)$ satisfies $\hat{\sigma}_j \notin S_a(\sigma)$. We assume that $\mu(\tilde{x}(\hat{\sigma}_j), \cdot)$ arrived at the actuator in time, because otherwise $\hat{\sigma}_j \notin S_a(\sigma)$ follows immediately.

Let $n_{c,i}$ be the time at which $\mu(\tilde{x}(\hat{\sigma}_j), \cdot)$ has been computed in step (C2). Since $\hat{\sigma}_j \notin S_c(n_c)$, at some time $n_{c,j} > n_{c,i}$ an error message with $\sigma_{cor} < \hat{\sigma}_j$ must have been processed in (C0) which caused the deletion of $\hat{\sigma}_j$ from $S_c(n_c)$. Let $n_{c,j} > n_{c,i}$ be minimal with this property. Let $n_{a,j} \le n_{c,j}$ denote the time this error was detected in step (A2). Then from the error condition in the actuator it follows that error messages were sent at each time instant $n_a = \sigma_{cor} + 1, \dots, n_{a,j}$.

Now we distinguish two cases:

Case 1: $n_{a,j} \ge \hat{\sigma}_j$. In this case, since $\sigma_{cor} < \hat{\sigma}_j \le n_{a,j}$, the actuator sent an error message at time $n_a = \hat{\sigma}_j$. This implies that $\mu(\tilde{x}(\hat{\sigma}_j), \cdot)$ was deleted from B_a at time instant $\hat{\sigma}_j$ and hence $\hat{\sigma}_j \notin S_a(\sigma)$ by (5).

Case 2: $n_{a,j} < \hat{\sigma}_j$. In this case, the transmission with time stamp $\hat{\sigma}_j$ had not yet been processed in (A2) when the error was sent. However, since the error message arrived at the controller at time $n_{c,j} > n_{c,i}$, at time $n_{c,i}$ when $\mu(\tilde{x}(\hat{\sigma}_j), \cdot)$ was computed and transmitted, the error was not yet known and thus not yet corrected. Hence, $\mu(\tilde{x}(\hat{\sigma}_j), \cdot)$ was inconsistent at time $n_a = \hat{\sigma}_j$ and hence $\hat{\sigma}_j$ was removed from $S_a(\hat{\sigma}_j)$. Again it follows that $\hat{\sigma}_j \notin S_a(\sigma)$ by (5). ■

This lemma shows that in our scheme the actuator only uses consistently predicted feedback values. Thus, in order to obtain prediction consistency in the sense of Definition 2.1(ii) it remains to prove that (2) is well defined, i.e., that (4) holds for all times $n = n_a$ in the actuator.

In order to derive a condition guaranteeing (4), we first derive an upper bound for the time needed to resolve an error. We say that the network *works without errors* on a discrete time interval $\{n_1, \dots, n_2\}$ if all feedback sequences computed within this interval reach the actuator with a transmission delay less or equal the maximal delay defined in the scheme. Furthermore, we say that a feedback value sequence $\mu(\tilde{x}(\sigma), \cdot)$ is *successfully transmitted* if it is accepted (and thus used) by the actuator.

Lemma 4.3: Assume that the network works without errors on a discrete time interval $\{n_1, \dots, n_2\}$ with

$$n_2 \ge n^\star := n_1 + \tau_{ca}^{\max} + \tau_c^{\max} + \tau_{err}, \tag{8}$$

where $\tau_{err} := n_{err} - n_1$ and n_{err} denotes the time the first error message sent in the time interval $\{n_1, \dots, n_2\}$ is processed in the controller. Then all feedback value sequences $\mu(\tilde{x}(\sigma), \cdot)$ with time stamp $\sigma = \sigma^\star, \dots, n_2 + \tau_c^{\max} + \tau_{ca}^{\max}$, where

$$\sigma^\star := n^\star + \tau_{ca}^{\max} + \tau_c^{\max} \tag{9}$$

are successfully transmitted.

Proof: Since the network works without errors for $n_c \in \{n_1, \dots, n_2\}$, each feedback sequence $\mu(\tilde{x}(\sigma), \cdot)$ with $\sigma = n_c + \tau_c^{\max} + \tau_{ca}^{\max}$ arrives at the actuator in time. Hence, it is successfully transmitted if and only if it is accepted at the actuator which in turn happens if and only if one of the following conditions hold:

(i) The feedback value sequence $\mu(\tilde{x}(\sigma - 1), \cdot)$ was successfully transmitted.

(ii) An error message with $\sigma_{err} \in S_c$ is received at time $n_c = \sigma - \tau_c^{\max} - \tau_{ca}^{\max}$.

In particular, condition (i) implies by induction that if a sequence $\mu(\tilde{x}(\sigma_{ok}), \cdot)$ for some $\sigma_{ok} \in \{n_1 + \tau_c^{\max} + \tau_{ca}^{\max}, \sigma^\star\}$ was transmitted successfully, then all subsequent sequences are transmitted successfully and the assertion of the lemma follows.

Thus, in order to prove the lemma we need to show that there exists a time stamp $\sigma_{ok} \in \{\sigma_1, \sigma^\star\}$ for which one of the conditions holds, where we define $\sigma_1 = n_1 + \tau_c^{\max} + \tau_{ca}^{\max}$.

If condition (i) holds for some $\sigma \in \{\sigma_1, \dots, \sigma^\star\}$, then there is nothing to show. Hence, assume that this is not the case, i.e., that none of the feedback sequences $\mu(\tilde{x}(\sigma_1 - 1), \cdot), \dots, \mu(\tilde{x}(\sigma^\star - 1), \cdot)$ is successfully transmitted.

Since by (8) the network works without errors at the time instances $n_1, \dots, n^\star$, this implies that neither condition (i) nor condition (ii) is satisfied for $\sigma \in \{\sigma_1, \dots, \sigma^\star - 1\}$. In particular, no error message was processed at $n_c = n_1, \dots, n^\star - 1$ (because otherwise condition (ii) would have been satisfied). Hence, no entries have been removed from B_c during these times, implying in particular $\sigma_1 \in S_c(n^\star - 1)$ and thus $\sigma_1 \in S_c$ at the beginning of step (C0) at time $n_c = n^\star$.

On the other hand, since the network works without errors at time n_1, the sequence $\mu(\tilde{x}(\sigma_1), \cdot)$ arrives at the actuator in time but is rejected there. Thus, an error message with $\sigma_{err} = \sigma_1$ is sent which arrives at the controller at time $\sigma_1 + \tau_{err} = n^\star$. Since $\sigma_1 \in S_c$ in step (C0) at time $n_c = n^\star$, condition (ii) is satisfied for

$$\sigma_{ok} = n^\star + \tau_c^{\max} + \tau_{ca}^{\max} = \sigma^\star$$

which shows the assertion. ■

In other words, Lemma 4.3 shows that if an error occurred, then the networked scheme "recovers" from this error after at most $\tau_{ca}^{\max} + \tau_c^{\max} + \tau_{err}$ time steps. We now use this information in order to derive a condition under which inequality (4) holds.

Lemma 4.4: Consider times $n_0 \le n_1 < n_2 \in \mathbb{N}_0$ with n_1, n_2 satisfying (8) and assume that the feedback control sequence $\mu(\tilde{x}(\sigma_0), \cdot)$ with time stamp $\sigma_0 = n_0 + \tau_{ca}^{\max} + \tau_c^{\max}$ is successfully transmitted to the actuator. Assume that the network works without errors on the interval $\{n_1, \dots, n_2\}$

(i.e., there may occur errors on the interval $\{n_0, \ldots, n_1 - 1\}$) and let the inequality

$$m^\star \geq n_1 - n_0 + \tau_{ca}^{\max} + \tau_c^{\max} + \tau_{err} \tag{10}$$

hold, where τ_{err} is defined as in Lemma 4.3. Then inequality (4) is satisfied for all $n \in \{\sigma_0, \ldots, n_2 + \tau_c^{\max} + \tau_{ca}^{\max}\}$. Furthermore, all feedback control sequences $\mu(\tilde{x}(\sigma), \cdot)$ for $\sigma = \sigma^\star, \ldots, n_2 + \tau_c^{\max} + \tau_{ca}^{\max}$ with $\sigma^\star$ from (9) are successfully transmitted.

Proof: From Lemma 4.3 we obtain that the feedback sequences $\mu(\tilde{x}(\sigma), \cdot)$ for the time stamps $\sigma = \sigma^\star, \ldots, n_2 + \tau_c^{\max} + \tau_{ca}^{\max}$ with $\sigma^\star$ from (9) are successfully transmitted.

Since both σ_0 and $n_2 + \tau_c^{\max} + \tau_{ca}^{\max}$ are transmission times, in order to prove (4) it is sufficient to show

$$\sigma_{k+1} - \sigma_k \leq m^\star \tag{11}$$

holds for all successful transmission times in $\{\sigma_0, \ldots, n_2 + \tau_c^{\max} + \tau_{ca}^{\max}\}$.

Since at least the times σ_0 and $\sigma^\star, \ldots, n_2 + \tau_c^{\max} + \tau_{ca}^{\max}$ are successful transmission times, in the worst case the two consecutive transmission times with the largest difference are σ_0 and $\sigma^\star$. By (10) we get

$$\begin{aligned} & \sigma^\star - \sigma_0 \\ = \;& n_1 + 2\tau_{ca}^{\max} + 2\tau_c^{\max} + \tau_{err} - n_0 - \tau_{ca}^{\max} - \tau_c^{\max} \\ = \;& n_1 - n_0 + \tau_{ca}^{\max} + \tau_c^{\max} + \tau_{err} \;\leq\; m^\star \end{aligned}$$

which proves (11). ■

Combining Lemma 4.2 and Lemma 4.4 we prove the following main theorem on the prediction consistency of our proposed scheme.

Theorem 4.5: Consider the networked control scheme described in Section III and assume that the feedback sequence $\mu(\tilde{x}(0), \cdot)$ with time stamp $\sigma = 0$ (computed at time $q_0 := -\tau_{ca}^{\max} - \tau_c^{\max}$) is successfully transmitted and that there exist times $q_0 \leq q_1 < q_2 < q_3 \ldots$ in $\mathbb{Z}$ such that the network works without errors on $\{q_{2i-1}, \ldots, q_{2i}\}$ for each $i \in \mathbb{N}$. Let τ_{err} be an upper bound for the transmission delay of the error messages sent in the intervals $\{q_{2i-1}, \ldots, q_{2i}\}$, $i \in \mathbb{N}$ and assume

(i) $q_{2i} - q_{2i-1} \geq \tau_{ca}^{\max} + \tau_c^{\max} + \tau_{err}$ for all $i \in \mathbb{N}$
(ii) $q_{2i+1} - q_{2i} \leq m^\star - \tau_{ca}^{\max} - \tau_c^{\max} - \tau_{err}$ for all $i \in \mathbb{N}_0$

Then the networked control scheme is prediction consistent.

Proof: From Lemma 4.2 it follows that only consistently predicted feedback control sequences are used in the actuator. Hence it remains to be shown that (4) holds for all $n \in \{q_0, \ldots, 0\} \cup \mathbb{N}$. To this end, using Lemma 4.4 we prove the following property by induction over $i \in \mathbb{N}_0$:

$$\begin{aligned} & \text{Inequality (4) is satisfied for } n = q_{2i}, \ldots, q_{2i+2} \\ & \text{and the sequence } \mu(\tilde{x}(q_{2i+2} + \tau_{ca}^{\max} + \tau_c^{\max}), \cdot) \\ & \text{is successfully transmitted} \end{aligned} \tag{12}$$

For $i = 0$, the assumptions of Lemma 4.4 are satisfied for $n_0 = q_0$, $n_1 = q_1$ and $n_2 = q_2$ since $0 = n_0 + \tau_{ca}^{\max} + \tau_c^{\max}$ holds, $\mu(\tilde{x}(0), \cdot)$ is successfully transmitted by assumption and the conditions (i) and (ii) imply (8) and (10), observing that τ_{err} defined here is an upper bound for τ_{err} in Lemma 4.4. Thus, Lemma 4.4 implies that (4) is satisfied for $n = 0, \ldots, q_2$ and the sequence $\mu(\tilde{x}(q_2 + \tau_{ca}^{\max} + \tau_c^{\max}), \cdot)$ is successfully transmitted, i.e., (12) for $i = 0$.

For $i \to i + 1$, we use that by the induction assumption (12) for i the sequence $\mu(\tilde{x}(q_{2i+2} + \tau_{ca}^{\max} + \tau_c^{\max}), \cdot)$ is successfully transmitted. As above for $i = 0$, the conditions (i) and (ii) imply that Lemma 4.4 can be applied with $n_0 = q_{2(i+1)}$, $n_1 = q_{2(i+1)+1}$, $n_2 = q_{2(i+1)+2}$ implying that (4) is satisfied for $n = q_{2i+2}, \ldots, q_{2(i+1)+2}$ and that $\mu(\tilde{x}(q_{2(i+1)+2} + \tau_{ca}^{\max} + \tau_c^{\max}), \cdot)$ is successfully transmitted, i.e., (12) for $i + 1$. ■

Remark 4.6: In other words, in order to ensure prediction consistency for our scheme the duration of network failures must be limited by $m^\star - \tau_{ca}^{\max} - \tau_c^{\max} - \tau_{err}$ sampling periods and between two failure periods the network must work without errors for at least $\tau_{ca}^{\max} + \tau_c^{\max} + \tau_{err}$ sampling periods. Given that the latter time is precisely the network's round trip time, i.e., the time needed for sending an error message to the controller and a corrected feedback sequence back to the actuator, and that the former is exactly the maximal control horizon $m^\star$ minus this round trip time, this appears to be the optimal behavior one can expect from a networked control scheme.

From this estimate it is also easy to see that if the network fails for n_{fail} sampling instances, then the system operates in open loop for at most $n_{fail} + \tau_{ca}^{\max} + \tau_c^{\max} + \tau_{err}$ sampling periods.

References

[1] A. Bemporad, *Predictive control of teleoperated constrained systems with unbounded communication delays*, in Proceedings of the 37th IEEE Conference on Decision and Control, 1998, pp. 2133–2138.

[2] L. Grüne and D. Nešić, *Optimization based stabilization of sampled–data nonlinear systems via their approximate discrete–time models*, SIAM J. Control Optim., 42 (2003), pp. 98–122.

[3] L. Grüne, J. Pannek, and K. Worthmann, *A networked unconstrained nonlinear MPC scheme.* Preprint, 2008. Submitted to ECC09.

[4] D. Nešić and A. R. Teel, *A framework for stabilization of nonlinear sampled-data systems based on their approximate discrete-time models*, IEEE Trans. Automat. Control, 49 (2004), pp. 1103–1122.

[5] G. Pin and T. Parisini, *Stabilization of networked control systems by nonlinear model predictive control: a set invariance approach*, in Int. Workshop on Assessment and Future Directions of NMPC, L. Magni, D. M. Raimondo, and F. Allgöwer, eds., 2008. CD-ROM, Paper 160.pdf.

[6] I. G. Polushin, P. X. Liu, and C.-H. Lung, *On the model-based approach to nonlinear networked control systems*, Automatica, 44 (2008), pp. 2409–2414.

[7] P. L. Tang and C. W. de Silva, *Stability validation of a constrained model predictive networked control system with future input buffering*, Internat. J. Control, 80 (2007), pp. 1954–1970.

[8] P. Varutti and R. Findeisen, *Compensating network delays and information loss by predictive control methods*. Preprint, 2008. Submitted to ECC09.