Online State Estimation for Supervisor Synthesis in Discrete-Event Systems With Communication Delays and Losses

Yunfeng Hou, Yunfeng Ji, Member, IEEE, Gang Wang, Member, IEEE, Ching-Yen Weng, and Qingdu Li

Abstract—In the context of networked discrete-event systems (DESs), communication delays and losses exist between the plant and the supervisor for observation and between the supervisor and the actuator for control. In this article, we first introduce a new framework for supervisory control of networked DESs. Under the introduced framework, we address the state estimation problem for supervisor synthesis of networked DESs with both communication delays and losses. The estimation algorithm considers the effect of the controls imposed on the system. In addition, the estimation algorithm is based on the control decisions available up to the moment, and all future control decisions are assumed to be unknowable. Two notions, called “observation channel configuration” for tracking observation delays and losses and “control channel configuration” for tracking control delays and losses, are defined. Then, we introduce an online approach for state estimation of the controlled system. Compared with the existing approach, the proposed approach under the introduced framework can estimate the states of the controlled system more accurately. As an application of the proposed approach, we finally show that the existing methods can be applied to synthesize maximally permissible and safe networked supervisors.

Index Terms—Communication delays and losses, networked discrete-event systems (DESs), state estimation, supervisor synthesis.

I. INTRODUCTION

The dynamics of discrete-event systems (DESs) is driven by asynchronous event sequences. A state estimate of DESs is defined as the set of (discrete) states such that the controlled system or the closed-loop system may be in after observing a sequence of observable events. Supervisor synthesis is an important problem in the supervisory control of DESs and has drawn much attention in the DES community over the past few decades [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. In supervisor synthesis, a supervisor or controller is desired to dynamically enable or disable event occurrences so that state estimates of the controlled system always satisfy the property of interest [3], [7], [8], [9], [10]. Thus, state estimation is crucial in supervisor synthesis for determining a valid control action after each new event observation.

Recent advances in wireless local-area network (WLAN)-based and cellular-based communication systems have enabled us to connect a supervisor and a plant via a shared communication network (networked DESs). Such a networked structure provides more flexible and agile ways to control a DES. For example, it allows the supervisor to be an edge computing node, which means it can share the computing resource with other edge computing nodes. However, communication delays and losses existing in networks pose significant challenges to attaining accurate state estimations when trying to solve the supervisor synthesis problem. Most of the current works implement state estimations of networked DESs based on the open-loop system without using the information of the control actions executed by the actuator [13], [14], [15], [16], [17], [18], [19], [20], [21]. When the state estimates calculated in [13], [14], [15], [16], [17], [18], [19], [20], and [21] are used for supervisor synthesis, they may contain some states that have been prevented from being reached by the control. One exception is the work of [22], where a novel online state estimation algorithm was proposed by taking the information of the control decision’s history into consideration. Nevertheless, as will be shown in Section V, the state estimate calculated in [22] is an overapproximation of the actual state estimate of the controlled system and may contain states that the controlled system never reaches. In addition, this approach considers only control delays.

In this article, we generalize the assumption of control delays in [22] into a more general case of communication delays and losses and study how to obtain an accurate state estimate of the controlled system under the observation delays and losses and the control delays and losses. Specifically, this article assumes the following: 1) both the control channel and the observation channel satisfy first-in-first-out (FIFO); 2) the observation delays and the control delays are upper bounded by \( N_o \) and...
$N_e$ event occurrences (observable or not), respectively; and 3) the numbers of consecutive observation losses and consecutive control losses are no larger than $N_{l,o}$ and $N_{l,c}$, respectively. It is worth noting that when there exist only control delays and losses, the observation of the supervisor to a string is deterministic, and we can immediately determine which control decision has been made after the occurrence of this string. However, when observation delays and losses exist, the observation of the supervisor to a string is nondeterministic and varies with the different observation delays and losses. For different observations, the supervisor may make different control decisions. An event after this string may be allowed to occur for some of these control decisions, but not be allowed to occur for the other control decisions. We must consider all the possibilities, which complicate the state estimation problem.

To obtain an accurate state estimate of the controlled system, we must first specify an accurate “dynamics” of the controlled system. To this end, we introduce a new framework for supervisory control of networked DESs. In this framework, a communication automaton is constructed to model the interaction process between the supervisor and the plant over the control channel and the observation channel under communication delays and losses. Each state of the communication automaton records the following:

1) the state that the plant is in;
2) the state that the networked supervisor is in;
3) the sequence of observable events that have occurred but still need to be delivered to the supervisor;
4) the number of consecutive observation losses;
5) the control action in use;
6) the sequence of control actions that have been issued but are still delayed at the control channel;
7) the number of consecutive control losses.

States of the communication automaton are updated when one of the following five behaviors occurs (represented as a special event occurrence):

1) a new event occurs;
2) a new observable event is communicated;
3) a new control action is executed;
4) an observation loss occurs;
5) a control loss occurs.

The dynamics of the controlled system can be simply “decoded” from sequences that can be generated by the communication automaton.

Next, we discuss how to produce online state estimates of the controlled system under communication delays and losses. Specifically, for tracking states of the observation channel, we introduce the “observation channel configuration,” which consists of the following two parts: 1) a sequence of event-integer pairs that is used to track the delayed observable event occurrences and the number of observation delays and 2) an integer that is used to track the number of consecutive observation losses. On the other hand, for tracking the states of the control channel, we introduce the “control channel configuration,” which consists of the following three parts: 1) an admissible control action that is taking effect; 2) a sequence of command-integer pairs that is used to track the delayed control actions and the number of control delays; and 3) an integer that is used to track the number of consecutive control losses. By incorporating the control channel configurations and the observation channel configurations into the states of the plant, we can obtain a triplet. We call such a triplet an augmented state (the plant state is augmented with the observation channel configuration and the control channel configuration). An online approach is proposed for updating the augmented state estimates upon each new observation, which can be used to estimate the states of the controlled system. Compared with [22], we show that the proposed approach can estimate states of the controlled system more accurately, even if there are only control delays.

Based on a game structure called bipartite transition system (BTS) [7], a general approach for solving a set of important supervisor synthesis problems was proposed by Yin and Lafortune in [7], [8]. Benefiting from the state estimation algorithm developed in this article, it becomes a reality that a BTS can be extended to its networked counterpart networked BTS (NBTS) when communication delays and losses exist. Similar to BTS, an NBTS also consists of the following two types of states: 1) the $Y$-state and 2) the $Z$-state. The $Y$-state records all the augmented states that the controlled system may reach immediately after a new observation. The $Z$-state collects all the augmented states that are reachable from its predecessor $Y$-state in an unobservable way. Using the NBTS, techniques developed in [7] and [8] can be easily extended to solve the corresponding problems with the communication delays and losses, which can effectively simplify the research. As an example, we finally show how to construct an NBTS using the proposed state estimation approach and extend the techniques developed in [7] and [8] to synthesize a maximally permissible networked supervisor while ensuring that the safety of the controlled system is satisfied.

Due to space limitations, some proofs are omitted in this article and can be found in the Appendix of [23].

II. PRELIMINARIES

A. Preliminaries

A DES is modeled by a deterministic finite-state automaton $G = (Q, \Sigma, \delta, q_0)$, where $Q$ is the finite set of states; $\Sigma$ is the finite set of events; $\delta : Q \times \Sigma \rightarrow Q$ is the transition function; and $q_0$ is the initial state. $\delta$ is extended to $Q \times \Sigma^*$ in the usual way. “$!$” means “is defined,” and “$\not!$” means “is not defined.” $\mathcal{L}(G)$ is the language generated by $G$. $\Sigma$ is partitioned into the set of controllable events $\Sigma_c$, the set of uncontrollable events $\Sigma_{uc}$. $\Sigma$ is also partitioned into the set of observable events $\Sigma_o$ and the set of unobservable events $\Sigma_{uo}$. The natural projection $P : \mathcal{L}(G) \rightarrow \Sigma_o$ is defined as $P(\varepsilon) = \varepsilon$ and, for all $s, s' \in \mathcal{L}(G)$, $P(s \sigma) = P(s) \sigma$ if $\sigma \in \Sigma_o$, and $P(s \sigma) = P(s)$, otherwise. Given automata $G_1$ and $G_2$, we say that $G_1$ is a subautomaton of $G_2$, denoted by $G_1 \subseteq G_2$, if $G_1$ and $G_2$ have the same initial state and $G_1$ is a subgraph of $G_2$.

Given an $s$, let $[s] = \{s' : (\exists s') s = s' \varepsilon\}$ be the set of all prefixes of $s$. The length of a string $s$ is denoted by $|s|$. The prefix closure of a language $L \subseteq \Sigma^*$ is denoted as $\overline{L}$. $L$ is prefix-closed if $L = \overline{L}$. We only consider prefix-closed languages in this article. $\varepsilon$ denotes the empty string. Given a $s = \sigma_1 \sigma_2 \cdots \sigma_k$,
we write $s^i = \sigma_1 \cdots \sigma_i$ for $i = 1, \ldots, k$, and $s^0 = \varepsilon$. $\mathbb{N}$ is the set of natural numbers. Given $a, b \in \mathbb{N}$, let $[a, b]$ be the set of natural numbers between $a$ and $b$. The cardinality of a set $Z$ is denoted by $|Z|$. Given a $n \in \mathbb{N}$, let $Z^{\leq n}$ be the set of sequences (consisting of elements in $Z$) with a length no larger than $n$.

We consider a networked DES in this article. Due to the network characteristics, communication delays and losses exist for both control and observation. We make the following assumptions on the networked DESs.

1) Both the control channel and the observation channel satisfy the FIFO property, i.e., the observable event occurrences are delivered to the supervisor in the same order as they were generated, and the control actions are executed by the actuator of the plant in the same order as they were issued.

2) The communication delays in the observation (regarded as observation delays) are nondeterministic but are upper bounded by $N_o$ events, i.e., when an event occurs, the system can generate no more than $N_o$ event occurrences before this event is communicated to the supervisor. The communication delays in the control (regarded as control delays) are also nondeterministic but are upper bounded by $N_c$ events, i.e., before an issued control action is executed, the system can generate no more than $N_c$ event occurrences.

3) The consecutive losses of the observable event occurrences are assumed to be no larger than $N_{o,v}$, i.e., before a new observable event is communicated (observed), there are at most $N_{o,v}$ consecutive observable losses, and the consecutive losses of the control actions are assumed to be no larger than $N_{l,c}$, i.e., before a new control action is executed, there are at most $N_{l,c}$ consecutive control losses.

4) The actuators always implement the most recently received action, and the initial control action can be executed without any delays and losses.

**Remark 1:** We assume that both the control channel and the observation channel satisfy FIFO, since it is often the case that there is only one communication channel from the plant to the supervisor and from the supervisor to the plant. This is slightly different from [24], [25], [26], [27], and [28], where FIFO is not required for the communication channels. In addition, since the communication losses are usually small, we assume that both the consecutive loss occurrences and the consecutive control losses have upper bounds. The same assumption can be found in [29] and [30]. Meanwhile, the delays are measured by the number of event occurrences (observable or not), which is different from [24], [25], [31], [32], [33], and [26] where time is explicitly considered. We also assume that the initial control action has been deployed in the execution module of the plant before it starts to work. Thus, when the plant is initialized, the initial control action can be executed without any delays and losses.

Similar to [25], the networked supervisor is defined as a pair $S = (A, \gamma)$, such that $A = (X, \Sigma_o, \xi, x_0)^1$ is a deterministic automaton with $L(A) = \Sigma_o^*$ and $\gamma : X \rightarrow 2^\mathbb{Lc}$ is a function that specifies the set of events to be enabled. Specifically, for any $t \in \Sigma_o^*$, we denote $\gamma(\xi(x_o, t))$ by the set of events to be enabled after observing $t$. With a slight abuse of notation, we also write $\gamma(\xi(x_o, t)) = S(t)$. Let $\Pi = \{\pi \in 2^{\Sigma} : \Sigma_{uc} \subseteq \pi\}$ be the set of all admissible control actions. Since we cannot disable an uncontrollable event, $S(t)$ should be admissible, i.e., $S(t) \in \Pi$.

**Remark 2:** Note that when communication losses exist, strings that may be observed by the networked supervisor are no longer $P(L(G))$ because some observable event occurrences may be lost during the transmission. To implement supervisory control under observation delays and losses, techniques were developed in [25] to construct an untimed automaton that models all the possible system observations in the presence of the observation delays and losses. And a networked supervisor that maps each possible system observation to an admissible control action was proposed in [25]. In contrast to [25], the networked supervisor $S$ is defined over the entire $\Sigma_o^*$ in this article. For those $t \in \Sigma_o^*$ that cannot be observed by $S$, we can define a special state $x_{spe}$ in $A$ such that $\xi(x_{0}, t) = x_{spe}$ and $\gamma(x_{spe}) = \Sigma_{uc}$.

**Example 1:** Consider system $G$ depicted in Fig. 1(a) with $\Sigma_c = \Sigma = \{\alpha, \beta, \lambda\}$. The networked supervisor $S = (A, \gamma)$ is depicted in Fig. 1(b). The function $\gamma$ is specified by the set of events associated with each state in Fig. 1(b). As shown in Fig. 1(b), when $S$ is in $p_1$, $S(\varepsilon) = \gamma(p_1) = \pi_1 = \{\alpha, \lambda\}$. Once $\alpha$ is observed, $S$ moves to state $p_2$, and $S(\alpha) = \gamma(p_2) = \pi_2 = \{\beta\}$. For all $t \in \Sigma_o^* \setminus \{\varepsilon, \alpha\}$, $S(t) = \gamma(p_2) = \pi_2 = \emptyset$.

Given two networked supervisors $S_1$ and $S_2$, we say that $S_1$ is smaller than $S_2$, denoted by $S_1 \subseteq S_2$, if $S_1(t) \subseteq S_2(t)$; for all $t \in \Sigma_o^*$, we say $S_1$ is strictly smaller than $S_2$, denoted by $S_1 \subset S_2$ if $S_1(t) \subset S_2(t)$; and there exists $t \in \Sigma_o^*$ such that $S_1(t) \subset S_2(t)$.

**III. DYNAMICS OF THE CONTROLLED SYSTEM**

To accurately estimate the states of a controlled system, we must first specify an accurate language that can be generated by the controlled system. To this end, we consider a framework of supervisory control of networked DESs in this section. In this framework, we construct a communication automaton that explicitly models the interaction process between the plant and the supervisor over the observation channel and the control channel, where communication delays and losses exist. We show that the “dynamics” of the controlled system can be inferred from the constructed communication automaton.
We first define the following four special types of events to characterize the behaviors of the communication delays and losses.

1) To describe the loss of an observable event occurrence, define bijection \( o : [1, N_o + 1] \rightarrow \Sigma^o \) such that \( \Sigma^o = \{ o(i) : i \in [1, N_o + 1] \} \), where \( o(i) \) indicates the loss of the \( i \)th observable event in the observation channel.

2) To describe the loss of a control action, define bijection \( c : [1, N + 1] \rightarrow \Sigma^c \) such that \( \Sigma^c = \{ c(i) : i \in [1, N + 1] \} \), where \( N = N_o + N_c \) and \( c(i) \) indicates the loss of the \( i \)th control action in the control channel.

3) To keep track of what observable event has been communicated, define bijection \( f : \Omega \rightarrow \Sigma^f \) such that \( \Sigma^f = \{ f(\sigma) : \sigma \in \Sigma \} \), where \( f(\sigma) \) indicates that the occurrence of \( \sigma \) has been communicated to the supervisor.

4) To model which control decision is taken, define bijection \( g : \Pi \rightarrow \Sigma^g \) such that \( \Sigma^g = \{ g(\pi) : \pi \in \Pi \} \), where \( g(\pi) \) indicates that the control action \( \pi \) has been executed.

Note that \( \Sigma , \Sigma^o , \Sigma^c , \Sigma^f , \) and \( \Sigma^g \) are mutually disjoint.

Given a system \( G \) and a networked supervisor \( S = (A, \gamma) \), such that \( A = (X, \Sigma_o , \Xi , x_o) \), we denote each state of the communication automaton by a seven-tuple \( \tilde{q} = (q, x, n, \phi, y, m, p) \in Q \times (\Sigma_o \times [0, N_o])^{\leq N_o+1} \times \Pi \times [0, N_c] \times X \) where:

1) \( q \in Q \) tracks the state that the plant is in.
2) \( x \) is a sequence of pairs \( (\sigma_1, n_1) \cdots (\sigma_k, n_k) \in (\Sigma_o \times [0, N_o])^{\leq N_o+1} \) such that \( \sigma_1 \cdots \sigma_k \in \Sigma^o \) tracks a sequence of observable events that have occurred but still need to be communicated (delivered) to the supervisor, and the integer \( n_i \) tracks the number of event occurrences while \( \sigma_i \) is waiting to be communicated.
3) \( n \in [0, N_c] \) counts the number of consecutive observation losses.
4) \( \phi \in \Pi \) is the control action in use.
5) \( y \) is a sequence of pairs \( (\pi_1, m_1) \cdots (\pi_l, m_l) \in \Pi \times [0, N_c]^{\leq N_c+1} \) such that \( \pi_1 \cdots \pi_l \in \Sigma^c \) tracks a sequence of admissible control actions that have been issued but have not been executed due to control delays, and the integer \( m_l \) tracks the number of event occurrences while the control action \( \pi_l \) is delayed at the control channel.
6) \( m \in [0, N_c] \) counts the number of consecutive control losses.
7) \( p \in X \) tracks the state that the networked supervisor \( S \) is in.

Remark 3: Note that the lengths of \( x \) and \( y \) are both finite. Since the observation delays are assumed to be upper bounded by \( N_o \), there could be \( N_o \) additional event occurrences at most before an observable event is communicated. Thus, the number of events delayed at the observation channel is \( N_o + 1 \) at most, and the length of \( x \) is no larger than \( N_o + 1 \). On the other hand, due to control delays and observation delays, the control action in use could be anyone issued in the past \( N \) steps. When a new event occurs, at least one control action issued in the past \( N + 1 \) steps is executed. Therefore, the length of \( y \) is no longer than \( N + 1 \).

Given a \( x \in (\Sigma_o \times [0, N_o])^{\leq N_o+1} \), if \( x = (\sigma_1, n_1) \cdots (\sigma_k, n_k) \neq \varepsilon \), we define \( \text{NUM}(x) = n_1 \) as the integer in the first pair of \( x \), and if \( x = \varepsilon \), we define \( \text{NUM}(x) = 0 \). Since \( \sigma_1 \) is the first event queued at the observation channel, \( \text{NUM}(x) \) records the maximum observation delays at the moment. To update the observation delays after a new event occurrence, we define \( x^+ = (\sigma_1, n_1 + 1) \cdots (\sigma_k, n_k + 1) \), if \( x = (\sigma_1, n_1) \cdots (\sigma_k, n_k) \neq \varepsilon \), and \( x^+ = \varepsilon \), if \( x = \varepsilon \). Similarly, for any \( y \in (\Pi \times [0, N_c])^{\leq N_c+1} \), if \( y = (\pi_1, m_1) \cdots (\pi_l, m_l) \neq \varepsilon \), we define \( \text{NUM}(y) = m_1 \) and \( y^+ = (\pi_1, m_1 + 1) \cdots (\pi_l, m_l + 1) \), and if \( y = \varepsilon \), we define \( \text{NUM}(y) = 0 \) and \( y^+ = \varepsilon \).

With the above preparations, we formally construct the communication automaton \( G_S = (Q, \Sigma, \delta, \tilde{q}_0) \), where \( Q \subseteq Q \times (\Sigma_o \times [0, N_o])^{\leq N_o+1} \times (\Pi \times [0, N_c])^{\leq N_c+1} \times X \) is the state space; \( \Sigma \subseteq \Sigma^o \cup \Sigma^c \cup \Sigma^f \cup \Sigma^y \) is the event set; \( \tilde{q}_0 = (q_0, x_0, \varepsilon, 0, S_0) = (q_0, x_0, \varepsilon, 0, x_0) \) is the initial state; and the transition function \( \delta : Q \times X \rightarrow Q \) is defined as follows:

1) For all \( \tilde{q} = (q, x, n, \phi, y, m, p) \in \tilde{Q} \) and all \( \sigma \in \Sigma \)

\[
\tilde{\delta}(\tilde{q}, \sigma) = \begin{cases} 
\tilde{q}' & \text{if } \delta(q, \sigma) \land \sigma \in \phi \land \text{NUM}(x^+) \leq N_o \\
\tilde{q} & \text{if } \sigma \in \phi \land \text{NUM}(y^+) \leq N_c \\
\text{undefined} & \text{otherwise}
\end{cases}
\]

2) For all \( \tilde{q} = (q, x, n, \phi, y, m, p) \in \tilde{Q} \) and all \( o(i) \in \Sigma^o \), if \( x = \varepsilon \), \( \delta(\tilde{q}, o(i)) \) is not defined, and if \( x \neq \varepsilon \), we write \( x = (\sigma_1, n_1) \cdots (\sigma_k, n_k) \) for \( \sigma_j \in \Sigma_o \) and \( n_j \in [0, N_o] \), and then

\[
\tilde{\delta}(\tilde{q}, o(i)) = \begin{cases} 
\tilde{q}' & \text{if } i \in [1, k] \land n + 1 \leq N_i \land \text{NUM}(x^+) \leq N_o \\
\text{undefined} & \text{otherwise}
\end{cases}
\]

3) For all \( \tilde{q} = (q, x, n, \phi, y, m, p) \in \tilde{Q} \) and all \( f(\sigma) \in \Sigma^f \), if \( x = \varepsilon \), then \( \delta(\tilde{q}, f(\sigma)) \) is not defined, and if \( x \neq \varepsilon \), we write \( x = (\sigma_1, n_1) \cdots (\sigma_k, n_k) \) for \( \sigma_j \in \Sigma_o \) and \( n_j \leq N_o \), and then

\[
\tilde{\delta}(\tilde{q}, f(\sigma)) = \begin{cases} 
\tilde{q}' & \text{if } \sigma = \sigma_1 \land \text{NUM}(x^+) \leq N_o \\
\text{undefined} & \text{otherwise}
\end{cases}
\]
with \( q' = (q', x', n', \phi', y', m', p') \), where:

a) \( q' = q \);

b) \( x' = (\sigma_2, n_2) \cdots (\sigma_k, n_k) \);

c) \( n' = 0 \);

d) \( \phi' = \phi \);

e) \( y' = y(\gamma(\xi(p, \sigma)), 0) \);

f) \( m' = m \);

and all \( g(\pi) = (q, x, n, \phi, y, m, p) \in \hat{Q} \) if \( x = \varepsilon \), the observation channel is empty, and no control actions can be lost or executed. As communicated, if \( y = \varepsilon \), the occurrence of \( \sigma \) is allowed by the control action in use, i.e., \( \sigma \in \phi \), and the observation delays and control delays after the occurrence of \( \sigma \) are no longer than \( N_o \) and \( N_c \), respectively, i.e., \( \text{NUM}(x^+) \leq N_o \) and \( \text{NUM}(y^+) \leq N_c \). If \( \sigma \) occurs at \( \hat{q} \), since \( q \) is used to track the state that the plant is in, we have \( q' = \delta(q, \sigma) \). Furthermore, if an unobservable \( \sigma \in \Sigma_{uo} \) occurs at \( \hat{q} \), the sequence delayed at the observation channel still is \( x \) but all the numbers in \( x \) if \( x \neq \varepsilon \) should add 1 for counting the observation delays. Therefore, we set \( x = x^+ \) if \( \sigma \in \Sigma_{uo} \) in (1). However, if \( \sigma \in \Sigma_o \), by FIFO, \((\sigma, 0)\) should be added to the end of \( x \) for tracking the new observable event occurrence, which is illustrated by \( x^* = x^+(\sigma, 0) \) if \( \sigma \in \Sigma_o \) in (1). Meanwhile, after the occurrence of \( \sigma \), the numbers in \( y \) should add 1 for recording the control delays. Hence, we set \( y' = y^+ \) in (1). For the remaining components in \( \hat{q} \), the state of the supervisor can be updated only when a new event is communicated, and the control action in use can be updated only when a new control action is executed. Therefore, \( p \) and \( \phi \) have no change after the occurrence of \( \sigma \), i.e., \( p' = p \) and \( \phi' = \phi \). Since there are no observation losses and control losses, \( n = n \) and \( m = m \).

For any \( \hat{q} = (q, x, n, \phi, y, m, p) \in \hat{Q} \), if \( x = \varepsilon \), the observation channel is empty. Thus, no observable events can be lost or communicated. As we can see, if \( y = \varepsilon \), \( o(i) \in \Sigma^o \) is not defined at \( \hat{q} \) in (2), and \( f(\sigma) \in \Sigma^f \) is not defined at \( \hat{q} \) in (3). Otherwise, if \( x = (\sigma_1, n_1) \cdots (\sigma_k, n_k) \), by assumption, all the observable events queued at the observation channel may be lost if the consecutive observation losses are not larger than \( N_{lo} \) after the observation loss. Therefore, if \( n + 1 \leq N_{lo} \), \( o(i) \) is defined at \( \hat{q} \) for all \( i \in [1, k] \) in (2). When \( o(i) \) occurs at \( \hat{q} \), we remove \((\sigma_i, n_i)\) from \( x \) and update \( n \) to \( n + 1 \) to record the observation loss. On the other hand, since \( x = (\sigma_1, n_1) \cdots (\sigma_k, n_k) \), by FIFO, \( f(\sigma) \) is defined at \( \hat{q} \) iff \( \sigma = \sigma_1 \) in (3). When the occurrence of \( \sigma_1 \) is communicated, the remaining sequence to be communicated is \( \sigma_2 \cdots \sigma_k \). Therefore, we have \( x' = (\sigma_2, n_2) \cdots (\sigma_k, n_k) \) in (3). Following the communication of \( \sigma_1 \), the state of the supervisor is updated to \( \xi(p, \sigma) \), and the supervisor sends \( \gamma(\xi(p, \sigma)) \) to the actuator of the plant, which are illustrated, respectively, by \( p' = \xi(p, \sigma) \) and \( y' = y(\gamma(\xi(p, \sigma)), 0) \) in (3). Furthermore, since \( n \) is used to count the number of consecutive observation losses, we reset \( n \) to 0 after a new event communication, as \( n' = 0 \) in (3).

For any \( \hat{q} = (q, x, n, \phi, y, m, p) \in \hat{Q} \), if \( x = \varepsilon \), the control channel is empty, and no control actions can be lost or executed. Thus, if \( y = \varepsilon \), then \( c(i) \in \Sigma^c \) is not defined at \( \hat{q} \) in (4), and \( g(\pi) \in \Sigma^g \) is not defined at \( \hat{q} \) in (5). Otherwise, if \( y = (\pi_1, m_1) \cdots (\pi_h, m_h) \notin \varepsilon \), by assumption, all the control actions queued at the control channel may be lost if the consecutive control losses are not larger than \( N_{lc} \) after the control loss. Thus, if \( m + 1 \leq N_{lc} \), \( c(i) \) is defined at \( \hat{q} \) for all \( i \in [1, h] \) in (4). When \( c(i) \) occurs at \( \hat{q} \), we remove \((\pi_i, m_i)\) from \( y \) and update \( m \) to \( m + 1 \) to record the control loss. On the other hand, since \( y = (\pi_1, m_1) \cdots (\pi_h, m_h) \notin \varepsilon \), by FIFO, \( g(\pi) \) is defined at \( \hat{q} \) iff \( \pi = \pi_1 \). When \( g(\pi) \) occurs at \( \hat{q} \), the control action that is taking effect becomes \( \pi \), and the control actions queued at the control channel are \( \pi_2 \cdots \pi_h \), which are illustrated, respectively, by \( \phi' = \pi \) and \( y' = (\pi_2, m_2) \cdots (\pi_h, m_h) \) in (5). Since \( m \) is used to count the number of consecutive control losses, we reset \( m \) to 0 in (5) when a new control action is executed. Example 2 further illustrates how to construct \( G_S \).

**Example 2:** Again, we consider the \( G \) and the \( S \) depicted in Fig. 1(a) and (b), respectively. As shown in Example 1, \( \Sigma_c = \Sigma = \{ \alpha \} \). Let \( N_c = N_o = N_{lo} = N_{lc} = 1 \). The communication automaton \( G_S \) is constructed in Fig. 2.

Let us consider the initial state \( \hat{q}_0 = (q_1, \varepsilon, 0, \pi_1, \varepsilon, 0, p_1) \). Since the control channel is empty, no control actions can be executed or lost. Hence, all the events in \( \Sigma^c \) and \( \Sigma^g \) are not defined at \( \hat{q}_0 \). Similarly, all the events in \( \Sigma^c \) and \( \Sigma^f \) are also not defined at \( \hat{q}_0 \) since the observation channel is empty. By Fig. 1(a), \( \delta(q_1, \alpha) = q_2 \). Moreover, since \( \alpha \in \Sigma_o \), \( \alpha \in \pi_1 \), and...
NUM(ε+) = 0 ≤ Nc, No, by (1), we have δ̃(q0, α) = ̃q1 = (q2, (α, 0), 0, π1, ε, 0, p1).

Next, we consider state ̃q1 = (q2, (α, 0), 0, π1, ε, 0, p1). By Fig. 1 (α), δ(q2, β) = q2 and δ(q2, λ) = q3. Since β /∈ π1, by (1), β cannot occur at ̃q1. However, since λ /∈ π1, λ /∈ π1, NUM((σ, 0)+) = 1 ≤ No, and NUM(ε+) = 0 ≤ Nc, by (1), δ̃(q1, λ) = ̃q1 = (q3, (α, 1), 0, π1, ε, 0, p1). Moreover, since (α, 0) /∈ ε, by (3), (α, 0) is defined at ̃q1. When σ is communicated, S moves to state (p1, α) = p2, and a new control action γ(p2) = π2 = {β} is issued. By (3), δ̃(q1, α, 0(1)) = (q2, ε, 0, π1, ε, 0, p1). In addition, since (α, 0) /∈ ε and 0 + 1 = 1 ≤ No, the event occurrence of α may be lost from the observation channel at ̃q1. By (2), δ̃(q1, α, 0(1)) = (q2, ε, 0, π1, ε, 0, p1). In this way, we can construct GS.

Remark 4: The interaction process between the plant and the supervisor is illustrated in Fig. 3. When a new observable event σ occurs in the plant, it is immediately pushed into the observation channel. When the first event σ1 queued at the observation channel is delivered to the supervisor, a new control action π = γ(ξ(p, σ1)) can be immediately issued and inserted into the control channel. The first control action π1 queued at the control channel cannot be executed until it is delivered to the actuator of the plant. Both the control actions delayed at the control channel and the observable events delayed at the observation channel may be lost. The supervisor has no idea what observable events are now queued at the observation channel and what control actions are now queued at the control channel. As will be shown in the next section, the supervisor makes state estimation based on only the observable events that have been communicated to it.

Given a string μ ∈ ̃Σ*, let ψ(μ) and ψf(μ) be the string obtained by removing all the events in ̃Σ \ Σ and ̃Σ \ Σf from μ, respectively, without changing the order of the remaining events. Define f−1 as, for all f(σ) ∈ Σf, f−1(f(σ)) = σ. We extend ψ, ψf, and f−1 to a set of strings in the usual way. We consider μ = αf(1)c(1) ∈ ̃L(GS) in Fig. 2. By definitions, ψ(μ) = α ∈ ̃L(G), ψf(μ) = f(α), and f−1(ψf(μ)) = α.

Intuitively: 1) ψ(̃L(GS)) specifies all the languages that can be generated under S and 2) f−1(ψf(̃L(GS))) specifies all the behaviors that can be observed by the networked supervisor. We formally prove them in the following proposition.

**Proposition 1:** Given a μ ∈ ̃L(GS), let us write δ̃(q0, μ) = (q, x, n, φ, y, m, p). Then, we have the following: 1) q = δ̃(q0, ψ(μ)) and 2) p = χ(q0, f−1(ψf(μ))).

By Proposition 1, the “dynamics” of the controlled system can be simply obtained by removing all the events in ̃Σ \ Σ from the sequences generated by GS.

**Definition 1:** Given a system G and a networked supervisor S = (A, γ) with A = (X, Σ, x0), we construct GS as described above. The language that may be generated by the controlled system under the communication delays and losses, denoted by ̃L(S/G), is defined as ̃L(S/G) = ψ(̃L(GS)).

**Proposition 2:** Given two networked supervisors S1 = (Ai) with Ai = (X, Σ, x0), for any t ∈ 1 \ (̃L(GS)), define

$$
E(S) = \{ q ∈ Q : (\exists μ ∈ ̃L(GS)) q = δ(q0, ψ(μ)) \land t = f−1(ψf(μ)) \}
$$

as the NSE of t under S, which is the set of all the possible states that the plant G may be in after observing t (subject to communication delays and losses) under S.

If S is given beforehand, we can calculate E(S) by constructing an observer of GS with the set of observable events Σf \ [34]. However, when we solve the supervisor synthesis problem, S is unknowable. The state estimate should be calculated online immediately after each new observation without using the future observations and controls [7], [8], [9], [10]. This is exactly the problem we want to solve in this article.

**IV. ONLINE NETWORKED STATE ESTIMATION**

In this section, we discuss how to produce online estimates of the states of a controlled system under communication delays and losses. To determine which state the controlled system is in, we should estimate not only the states of the plant, but also the observable event occurrences delayed at the observation channel as well as the control actions delayed at the control channel. To this end, we introduce the notions of observation channel configuration and control channel configuration as follows.

**Definition 3:** The observation channel configuration is defined as: θ0 = ((σ1, n1), ···, (σk, nk), n), where (σ1, n1) ···.
$(σ_k, n_k) \in (Σ_σ \times [0, N_σ]) \leq N_σ + 1$ is a sequence of pairs such that $σ_1 \cdots σ_k$ is a sequence of observable events currently delayed at the observation channel (in the same order as they were generated) and $n_k$ is the number of event occurrences since $σ_1$ has occurred, and $n \in [0, N_{t,o}]$ tracks the number of consecutive observation losses.

We denote by $Θ_o \subseteq (Σ_σ \times [0, N_σ]) \leq N_σ + 1 \times [0, N_{t,o}]$ the set of all the possible observation channel configurations. By Definition 3, $θ_o$ can be updated if one of the following three behaviors happens: 1) an event occurs; 2) an observable event is communicated or 3) an observable event is lost. To update $θ_o$, we define three operations as follows. Given a $θ_o = (x, n) \in Θ_o$.

1) If an event $σ \in Σ$ occurs, $x$ should be updated to $x^+$ immediately to count the observation delays. Meanwhile, if $σ \in Σ_σ$ by FIFO, we still need to add $(σ, 0)$ to the end of $x^+$ to record the new event occurrence. Formally, for any $θ_o = (x, n) \in Θ_o$ and any $σ \in Σ$, we define $IN_{obs}(θ_o, σ) = (x', n')$, where $σ \in Σ_σ, x' = x^+(σ, 0)$ and $n' = n + 1$. If a new event occurs in $Σ, x$ is communicated or 3) an observable event is lost. To update $θ_o$, we define operations as follows. Given $θ_o = (x, n) \in Θ_o$.

2) If a new $σ \in Σ_σ$ is communicated, by FIFO, $σ$ is the first event queued at the observation channel. If we write $x = (σ_1, n_1) \cdots (σ_k, n_k)$, then $σ = σ_1$ and the remaining events delayed at the observation channel are $σ_2 \cdots σ_k$. In addition, since $n$ is used to track the number of consecutive observation losses, we reset $n$ to 0 after a new event communication. Formally, for any $θ_o = (x, n) \in Θ_o$, if $x = (σ_1, n_1) \cdots (σ_k, n_k) \neq ε$, we define $OUT_{obs}(θ_o) = (x', n')$, where $x' = (σ_2, n_2) \cdots (σ_k, n_k)$ and $n' = 0$.

3) If the $ith$ event in the observation channel is lost, we should remove it from $x$. Meanwhile, since a new observation loss occurs, the number of consecutive observation losses should be updated to $n + 1$. Thus, for any $θ_o = (x, n) \in Θ_o$, if $x = (σ_1, n_1) \cdots (σ_k, n_k) \neq ε$ and any $i \in [1, k]$, we define $LOSS_{obs}(θ_o, i) = (x', n')$, where $x' = (σ_1, n_1 \cdots (σ_{i-1}, n_{i-1}) (σ_{i+1}, n_{i+1}) \cdots (σ_k, n_k)$ and $n' = n + 1$.

**Definition 4:** The control channel configuration is defined as: $θ_c = (φ, y = (π_1, m_1) \cdots (π_h, m_h), m)$, where $φ \in Π$ is the control action in use, $(π_1, m_1) \cdots (π_h, m_h) \in (Π \times [0, N_π]) \leq N_π + 1$ is a sequence of pairs such that $π_1 \cdots π_h$ are control actions currently queued at the control channel, and $m_k$ is the number of event occurrences since control action $π_k$ has been issued, and $m \in [0, N_{t, c}]$ counts the number of consecutive control losses.

We denote by $Θ_c \subseteq Π \times (Π \times [0, N_π]) \leq N_π + 1 \times [0, N_{t, c}]$ the set of all the possible control channel configurations. By Definition 4, $θ_c$ can be changed if one of the following four behaviors happens.

1) A new control action is issued.
2) A new control action is executed.
3) A control action is lost.
4) A new event occurs.

To update $θ_c$, we next define four operations as follows. Given $θ_c = (φ, y, m) \in Θ_c$.

1) If a new control action $π \in Π$ is issued, by FIFO, we need to add $(π, 0)$ to the end of $y$. Formally, for any $θ_c = (φ, y, m) \in Θ_c$ and any $π \in Π$, we define $IN_{ctr}(θ_c, π) = (φ', y', m')$, where $φ' = φ, y' = y(π, 0)$, and $m' = m$.

2) If a new control action $π \in Π$ is executed, by FIFO, $π$ is the first control action queued at the control channel. After execution, the control action that is taking effect would be $π$. Meanwhile, since $m$ is used to track the number of consecutive control losses, we need to reset $m$ to 0 after a new control action execution. Formally, for any $θ_c = (φ, y, m) \in Θ_c$, if $y = (π_1, m_1) \cdots (π_h, m_h) \neq ε$, define $OUT_{ctr}(θ_c) = (φ', y', m')$, where $φ' = π_1, y' = (π_2, m_2) \cdots (π_h, m_h)$, and $m' = 0$.

3) If the $ith$ control action in the control channel is lost, by definition, we need to remove it from $y$. Meanwhile, since a new control loss occurs, the number of consecutive control losses becomes $m + 1$. Formally, for any $θ_c = (φ, y, m) \in Θ_c$ with $y = (π_1, m_1) \cdots (π_h, m_h) \neq ε$ and any $i \in [1, h]$, define $LOSS_{ctr}(θ_c, i) = (φ', y', m')$, where $y' = (π_1, m_1) \cdots (π_{i-1}, m_{i-1})(π_{i+1}, m_{i+1}) \cdots (π_h, m_h)$, $φ' = φ$, and $m' = m + 1$.

4) If a new event occurs in $G$, all the natural numbers in $y$ (if $y \neq ε$) should increment for tracking the control delays. Hence, for any $θ_c = (φ, y, m) \in Θ_c$, define $PLUS(θ_c) = (φ', y', m')$, where $φ' = φ, y = y + 1$, and $m' = m$.

Given a $θ_o = (x, n) \in Θ_o$, let $[θ_o]_1 = x$ and $[θ_o]_2 = n$ be the first and second components of $θ_o$, respectively. Similarly, given a $θ_c = (φ, y, m) \in Θ_c$, let $[θ_c]_1 = φ$, $[θ_c]_2 = y$, and $[θ_c]_3 = m$ be the first, second, and third components of $θ_c$, respectively.

As mentioned above, in addition to $q \in Q$, we also need to estimate $θ_o \in Θ_o$ and $θ_c \in Θ_c$ since they can affect the future behaviors of the controlled system. Thus, we denote each state of the controlled system by a triplet $(q, θ_o, θ_c) \in Q \times Θ_o \times Θ_c$. We call such a state an augmented state. Next, we show how to update the augmented state estimate upon each new communication. The procedure can be briefly summarized as follows: next executing the following two steps.

**Step 1:** Let $Z \subseteq Q \times Θ_o \times Θ_c$ be a set of augmented states calculated immediately after a new observation or the initial $Z = \emptyset$. The delayed unobservable reach of $Z$ under an admissible control action $π \in Π$, denoted by $DUR(Z, π)$, is defined as follows.

1) Initially, if $Z = \emptyset$, we have

$$DUR(Z, π) = \{(q_0, (ε, 0), (π, ε, 0))\} \subseteq DUR(Z, π). \quad (7)$$

Otherwise, if $Z = \emptyset$, for all $(q, θ_o, θ_c) \in Z$

$$DUR(Z, π) = \{(q, θ_o, θ_c) \in Z \} \subseteq DUR(Z, π). \quad (8)$$

2) Then, we repeatedly apply the following operations until convergence is achieved.

1) For all $(q, θ_o, θ_c) \in DUR(Z, π)$, if $δ(q, σ)$ and $σ \in [θ_c]_1$ and $NUM([θ_o]_1) \leq N_o$ and $NUM([θ_c]_1) \leq N_c$

$$DUR((q, σ), θ_o, θ_c) = \{(δ(q, σ), θ_o, θ_c) \in DUR(Z, π)\} \subseteq DUR(Z, π). \quad (9)$$

2 By assumption, the plant does not work until it is initialized. Thus, before the initial control action is executed (the plant starts to work), we let $Z = \emptyset$.\[250\] IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, VOL. 11, NO. 1, MARCH 2024
2) For all \((q, \theta_a, \theta_c) \in \text{DUR}(Z, \pi), \) if \([\theta_c]_2 \neq \varepsilon, \) then

\[
(q, \theta_a, \text{OUT}^{ext}(\theta_c)) \in \text{DUR}(Z, \pi). \tag{10}
\]

3) For all \((q, \theta_a, \theta_c) \in \text{DUR}(Z, \pi), \) if \([\theta_a]_1 \neq \varepsilon \) and \([\theta_a]_2 + 1 \leq N_i, \) then for all \(i \in [1, [|\theta_a|]_1] \)

\[
(q, \text{LOSS}^{\text{obs}}(\theta_a, i), \theta_c) \in \text{DUR}(Z, \pi) \tag{11}
\]

4) For all \((q, \theta_a, \theta_c) \in \text{DUR}(Z, \pi), \) if \([\theta_a]_2 \neq \varepsilon \) and \([\theta_c]_3 + 1 \leq N_i, \) then for all \(i \in [1, [|\theta_a|]_2] \)

\[
(q, \theta_a, \text{LOSS}^{ext}(\theta_a, i)) \in \text{DUR}(Z, \pi). \tag{12}
\]

\textbf{Remark 5:} In the context of networked DESs, only when \(\) “an observable event is communicated” \(\) is observable. The behaviors of \(\) “an event (observable or not) occurs,” \(\) “a control action is executed,” \(\) “an observable event is lost,” \(\) “a control action is lost” \(\) are all unobservable. They are considered by \(\) “an observable event is communicated” \(\) is observable. The \(\) “observable” \(\) way. Specifically, if \(\) \(\) is active at \(\) \(\), \(\) \(\) control action may be lost from the control channel. Delayed at the control channel is not empty, \(\) i.e., \(\)

\[
\text{DUR}(Z, \pi) \text{ includes all the augmented states that can be reached from augmented states in } Z \text{ following a new communication of } \sigma. \text{ By FIFO, an observable event can be communicated iff it is the first event queued at the observation channel. Hence, we only consider all the } \sigma \in \Sigma_o \text{ such that there exists } (q, \theta_a, \theta_c) \in Z \text{ with } [\theta_c]_3 + 1 \leq \varepsilon \land \sigma = \sigma. \text{ When } \sigma \text{ is communicated, we remove } (\sigma_1, n_k) \text{ from } \theta_o. \text{ Thus, after communication of } \sigma, \theta_o \text{ is updated to } \text{OUT}^{\text{obs}}(\theta_o). \text{ We assume that } DOR(Z, \pi) \text{ is updated immediately after a new observable event is communicated before the next control action is issued. Therefore, we keep } \theta_o \text{ unchanged in (13).}
\]

For a communicated string \(t \in \Sigma_o^*, \) let the set of augmented states calculated by alternatively applying Step 1 and Step 2 be the augmented state estimate for \(t. \) Formally,

\[\]

\textbf{Definition 5:} Given a system \(G\) and a networked supervisor \(S\) defined over \(\Sigma_o, \) for any \(t \in f^{-1}(\{\varepsilon\} \cup L(G))\), let \(\hat{E}_S(t)\) be the augmented state estimate calculated by alternatively applying \(\) \(DUR(\cdot)\) and \(DOR(\cdot)\) as follows:

\[
1) \text{ Initially, } \hat{E}_S(\varepsilon) = \text{DUR}(\emptyset, S(\varepsilon)); \\
2) \text{ For all } t', t'\sigma_{i+1} \in [T], i = 0, 1, \ldots, |t| - 1, \\
\hat{E}_S(t'\sigma_{i+1}) = \text{DUR}(\text{DOR}(\hat{E}_S(t'), \sigma_{i+1}), S(t'\sigma_{i+1})).
\]

An example to illustrate the state estimation process will be provided in the next section. We next discuss the relationship between \(\hat{E}_S(t)\) and \(\hat{E}_S(t).\)

\[\]

\textbf{Proposition 3:} Given a system \(G\) and a networked supervisor \(S\) defined over \(\Sigma_o, \) for all \(t \in f^{-1}(\{\varepsilon\} \cup L(G))\), we have

\[
(q, \theta_a, \theta_c) \in \hat{E}_S(t) \Rightarrow (\exists \mu \in L(G)) \\
f^{-1}(\{\varepsilon\}) = t \land \delta(q_0, \mu) = (a, x, n, \phi, y, m, p) \land q = a \land \theta_a = (x, n) \land \theta_c = (\phi, y, m).
\]

\[\]

\textbf{Proposition 4:} For any \(\mu \in L(G), \) we write \(\hat{\delta}(q_0, \mu) = \hat{q} = (a, x, n, \phi, y, m, p). \) Then, \(q, \theta_a, \theta_c) \in \hat{E}_S(f^{-1}(\{\varepsilon\}) \mu), \) where \(q = a, \theta_a = (x, n), \) and \(\theta_c = (\phi, y, m).\)

Given a set of augmented states \(Z \in 2^{Q \times \Theta_a \times \Theta_c}, \) let FC(Z) \(= \{q \in Q : (\exists q, \theta_a, \theta_c) \in Z\} \) be the set of first components of augmented states in \(Z. \) The following theorem shows that FC(\(\hat{E}_S(t)\)) indeed estimates the states of the controlled system.

\[\]

\textbf{Theorem 1:} Given automaton \(G\) and a networked supervisor \(S, \) for all \(t \in f^{-1}(\{\varepsilon\} \cup L(G))\), FC(\(\hat{E}_S(t)\)) = \(\hat{E}_S(t).\)

\textbf{Proof:} We first prove FC(\(\hat{E}_S(t)\)) \(\subseteq \hat{E}_S(t).\) For any \(q, \theta_a, \theta_c) \in \hat{E}_S(t), \) by Proposition 3 and Definition 2, \(q \in \hat{E}_S(t).\) Therefore, FC(\(\hat{E}_S(t)\)) \(\subseteq \hat{E}_S(t).\) Next, we prove that \(\hat{E}_S(t) \subseteq FC(\hat{E}_S(t)).\) For any \(q \in \hat{E}_S(t), \) by Definition 2, \(\exists \mu \in L(G), \) such that \(f^{-1}(\{\varepsilon\}) = t \land q = \delta(q_0, \mu). \) We write \(\hat{\delta}(q_0, \mu) = (a, x, n, \phi, y, m, p). \) By Proposition 1, \(q = a. \) By Proposition 4, \(q, \theta_a, \theta_c) \in \hat{E}_S(f^{-1}(\{\varepsilon\}) \mu) = \hat{E}_S(t), \) where \(q = a, \theta_a = (x, n), \) and \(\theta_c = (\phi, y, m). \) Therefore, \(q = \hat{\delta}(q_0, \mu) = (a, x, n, \phi, y, m, p). \)
a = q’ ∈ FC(ΔS(t)). Since q is arbitrarily given, ΔS(t) ⊆ FC(ΔS(t)).

**Remark 6:** The online process for estimating the states of the controlled system under communication delays and losses is depicted in Fig. 4, which is briefly summarized as repeatedly executing the following: 1) an observable event occurrence σ ∈ Σo is communicated to the networked supervisor, and Z is updated to Z’ = DOR(Z, σ) and 2) a newly issued control action π ∈ Π is sent to the actuator of the plant, and the augmented estimate Z = DUR(Z’, π) is then calculated using (7)–(12). By Theorem 1, the current NSE can be obtained by taking all the first components of augmented states in Z, i.e., FC(Z).

**Remark 7:** By Fig. 4, the augmented state estimate is updated when a new control action π ∈ Π is issued (following a new observation of σ ∈ Σo). For any Z ∈ 2Q×Θo×Θε, the complexities for computing DOR(Z, σ) and DUR(DOR(Z, σ), π) are linear in the size of Θo and Q × Θo × Θε, respectively. Therefore, the computational complexity for the augmented state estimation is a stepwise order of Θ(|Q| × |Θo| × |Θε|).

Since Θε ⊆ (Σo × [0, No])Nε+1 × [0, No] and Θo ⊆ Π × (Q × [0, No])Nσ+1 × [0, No], we have

\[
\mathcal{O}(|Q| × |Θo| × |Θε|) = \mathcal{O}(|Q| × |Σ|Nσ × 2Nσ|Σ| × (No + 1)Nσ × (No + 1)Nε × (Nε + 1)Nε × No × No).
\]

The complexity of the proposed approach after each new observation is polynomial w.r.t. |Q|, No, and Nε, but is exponential w.r.t. |Σ|, Nσ, and No. The complexity of the proposed approach grows rapidly with the cardinality of the event set and the delay bounds Nσ and Nε. Thus, the proposed approach is more suitable for estimating the states of a networked DES with relatively small |Σ|, Nσ, and Nε.

**V. COMPARISON WITH THE EXISTING WORK**

In this section, we show the difference between the proposed state estimation approach with that proposed in [22]. To be consistent with [22], we assume in this section that there are only control delays with an upper bound of Nε, and there are no control losses, observation delays, and observation losses, i.e., Nε = No = Nl = 0.

We first recall the framework adopted in [22] for specifying the language of the controlled system under control delays. The framework adopted by [22] was first proposed in [21], where an event σ can occur after a string s ∈ L(G) if σ is allowed to occur by one of the control actions issued in the past Nε steps, i.e., σ ∈ S(P(s−i)) for some i ∈ [0, Nε]. Formally, the dynamics of the controlled system were specified in [21] and [22] as follows.

**Definition 6:** For a networked DES G with a supervisor S, the language \( L_a(S/G) \) that may be generated by the controlled system is defined recursively as follows:

\[
\varepsilon \in L_a(S/G)
\]

\[
\sigma s \in L_a(S/G) \iff s \in L_a(S/G)
\]

\[
\land \sigma s \in L_a(G) \land (\exists i \in [0, N_ε]) \sigma \in S(P(s−i)).
\]

For all \( t \in P(L_a(S/G)) \), techniques were developed in [22] for estimating states \( q \in Q \) such that there exists \( s \in L_a(S/G) \) with \( q = \delta(q_0, s) \) and \( t = P(s) \). Specifically, in [22], the state of the control channel is modeled as a set of control actions that have been issued in the past Nε steps, and the state of the controlled system (named as the extended state in [22]) consists of both the state of the plant and the state of the control channel. An event can occur at an extended state, and if only if, it is active at the plant state and is allowed to occur by one of the control actions issued in the past Nε steps. Then, the state estimate of the controlled system can be calculated by estimating all the possible extended states that the controlled system may be in. Next, we show that the state estimate calculated in [22] is actually an overapproximation of the exact state estimate of the controlled system and may contain states that the controlled system never reaches.

**Example 3:** Again, we consider G depicted in Fig. 1(a) with \( \Sigma_c = \Sigma = \{ \alpha, \beta, \gamma \} \) and \( \Sigma_o = \{ o \} \). The networked supervisor \( S = (A, \gamma) \) is depicted in Fig. 1(b). We have \( S(\varepsilon) = \pi_1 = \{ \alpha, \lambda \} \), \( S(\sigma) = \pi_2 = \{ \beta \} \), and \( S(t) = \pi_3 = \emptyset \) for all \( t \in \Sigma \setminus \{ \varepsilon, \sigma \} \). The control delays are upper bounded by 2, i.e., \( N_ε = 2 \), and \( N_l = N_o = N_l = 0 \). We first show that \( s = \alpha \beta \in L_a(S/G) \).

1) \( \varepsilon \in L_a(S/G) \).

2) Since \( \alpha \in S(\varepsilon) \), we have \( \alpha \in L_a(S/G) \).

3) Since \( \beta \in S(P(\alpha)) = S(\alpha) \), \( \alpha \in L_a(S/G) \), and \( \alpha \beta \in L_a(S/G) \), we have \( \alpha \beta \in L_a(S/G) \).

4) Since \( \lambda \in S(P(\alpha \beta \gamma)) = S(\varepsilon) \), \( \alpha \beta \in L_a(S/G) \), and \( \alpha \beta \lambda \in L_a(S/G) \), we have \( \alpha \beta \lambda \in L_a(S/G) \).

Since \( s = \alpha \beta \in L_a(S/G), P(s) = (s) = \delta(q_0, s) \), \( q_0 \) is included in the state estimate calculated in [22] after observing \( \alpha \in P(L_a(S/G)) \). Next we show that \( \alpha \beta \) never occurs. Since \( \lambda \in S(\varepsilon) \) and \( \lambda \notin S(\alpha) \), \( \lambda \) can occur after \( \alpha \beta \) only if the control action that is taking effect after \( \alpha \beta \) is \( S(\varepsilon) \). However, since \( \beta \in S(\alpha) \) and \( \beta \notin S(\varepsilon) \), \( S(\alpha) \) must have been executed at the time \( \beta \) occurs after \( \alpha \). That is, \( S(\varepsilon) \) has been replaced by \( S(\alpha) \) after the occurrence of \( \alpha \beta \). Thus, \( \alpha \beta \) never occurs in reality, and the controlled system never reaches \( q_0 \).

By Example 3, not all the control actions issued in the past Nε steps can take effect at the moment. Thus, an event that is active at a plant state and is allowed to occur by one of the past Nε control actions may never occur in practice. As a result, the state estimate calculated in [22] may contain states that the
controlled system never is in. In contrast to [22], we introduce a new modeling framework for supervisory control of networked DESs in Section III. The state estimation approach proposed under the framework differs crucially from [22] in the sense that the proposed approach explicitly models the control action (the first component of the control channel configuration) that can really take effect. As shown in the following example, the proposed approach improves the previous approach because it excludes those states that the controlled system never reaches.

**Example 4:** We continue with Example 3. We calculate \( \delta_S (\alpha) \) and \( \epsilon_S (\alpha) \) using approaches proposed in this article.

Initially, by Definition 5 and (7), \( q_1 = (q_1, (e, 0), (\pi_1, e, 0)) \in \delta_S (e) \). Since \( \delta (q_1, \alpha), \alpha \in \pi_1 \), and \( \text{NUM} (\epsilon^+) = 0 \leq \text{N}_0 \), by \( (9) \), \( q_2 = (q_2, ((\alpha, 0), 0), (\pi_1, e, 0)) \in \delta_S (e) \). Thus, \( \delta_S (e) = \{ q_1, q_2 \} \).

Next, if event \( \alpha \) is observed, by (13), \( \text{DOR} (\delta_S (\alpha)) = \{ (q_2, (e, 0), (\pi_1, e, 0)) \} \). Upon the observation of \( \alpha \), S issues \( \pi_2 \). By Definition 5, \( \delta_S (\alpha) = \text{DUR} (\text{DOR} (\delta_S (\alpha), \pi_2)) \). By (8), \( \tilde{q}_3 = (q_2, (e, 0), (\pi_1, \pi_2, 0, 0)) \in \delta_S (\alpha) \). Since \( \delta (q_2, \lambda) = q_3 \), \( \lambda \in \pi_1 \), \( \text{NUM} (\epsilon^+) = 0 \leq \text{N}_0 \), and \( \text{NUM} (\pi_2, 0^+) = 1 \leq \text{N}_0 \), only event \( \lambda \) can occur at \( \tilde{q}_3 \) (\( \beta \) is disabled by \( \pi_1 \)). By (10), \( \tilde{q}_4 = (q_3, (e, 0), (\pi_1, (\pi_2, 1, 0))) \in \delta_S (\alpha) \). When control action \( \pi_2 \) is executed, by (10), \( \tilde{q}_5 = (q_2, (e, 0), (\pi_2, 0, 0)) \in \delta_S (\alpha) \). Since \( \delta (q_2, \beta) = q_4, \beta \in \pi_2 \), and \( \text{NUM} (\epsilon^+) = 0 \leq \text{N}_0 \), by (9), we have that \( \tilde{q}_7 = (q_4, (e, 0), (\pi_2, 0, 0)) \in \delta_S (\alpha) \). Since \( \lambda \notin \pi_2 \) and only \( \lambda \) is active at \( q_4 \) in \( \tilde{G} \), \( \lambda \) cannot occur at \( \tilde{q}_7 \). Thus, \( \alpha \beta \lambda \) will never occur under \( S \). Overall, \( \delta_S (\alpha) = \{ q_4, q_5, q_6, q_7 \} \).

By Theorem 1, \( \delta_S (\alpha) = \{ q_4, q_5, q_6, q_7 \} \). By Example 4, \( \delta_S (\alpha) \) does not contain \( q_5 \). We have shown in Example 3 that the controlled system never reaches \( q_5 \) under \( S \).

The above example justifies the difference and advantage of the proposed approach compared with that proposed in [22].

### VI. APPLICATION

In this section, we consider the application of the proposed approach. We first introduce the definition of networked safety. We then show how to apply the proposed approach to construct an NBTS. Finally, we discuss how to synthesize a maximally permissible and networked safe supervisor from an NBTS.

#### A. Networked Safety

We start by defining the networked safety of the DESs under communication delays and losses. Let \( H = (Q_H, \Sigma, \delta_H, q_0) \subseteq G \) be a subautomaton of \( G \) that characterizes the specification language (safe behaviors). That is, \( Q_H \) captures all the safe behaviors in the sense that all the strings generated by \( G \) are safe if they are ended in states in \( Q_H \) and unsafe if they are ended in states in \( Q \setminus Q_H \). Then, the networked safety property of the DESs can be defined as follows.

**Definition 7:** Given a system automaton \( G \), a specification automaton \( H \), and a networked supervisor \( S \) defined over \( \Sigma_0 \), we say that \( \mathcal{L}(S/G) \) is network safe w.r.t. \( Q_H \subseteq Q \) and \( G \), if \( (\forall s \in \mathcal{L}(S/G)) \delta(q_0, s) \in Q_H \).

We next show that networked safety can be formulated as a state-estimate-based (SE-based) property (or information-state-based property in [7] and [8]). We define the SE-based property \( \varphi \) w.r.t. \( G \) as a function \( \varphi : 2^Q \rightarrow \{ 0, 1 \} \), where for all \( Z \in 2^Q \), \( \varphi(Z) = 1 \) means that \( Z \) satisfies property \( \varphi \).

To ensure that \( \mathcal{L}(S/G) \) is network safe, by Definition 7, we must ensure all the states that the controlled system may reach are within \( Q_H \). In this regard, a state \( Z \in 2^Q \) is safe if and only if \( Z \subseteq Q_H \). Thus, the definition of the SE-based property \( \varphi_{\text{safe}} \) is defined as follows.

**Definition 8:** The SE-based property \( \varphi_{\text{safe}} : 2^Q \rightarrow \{ 0, 1 \} \) is defined as follows: For any \( Z \in 2^Q \)

\[
\varphi_{\text{safe}}(Z) = 1 \iff Z \subseteq Q_H.
\]

**Proposition 5:** Given automata \( G \) and \( H \) and a networked supervisor \( S \) defined over \( \Sigma_0 \), \( \mathcal{L}(S/G) \) is network safe w.r.t. \( Q_H \subseteq Q \) and \( G \), if and only if all the state estimates that may be generated by the controlled system satisfy \( \varphi_{\text{safe}} \), i.e., \( \forall t \in f^{-1}(\psi(\mathcal{L}(G(S))))/\varphi_{\text{safe}}(\delta(S)(t)) = 1 \).

**Proof:** \( \Rightarrow \) By contradiction. Suppose \( \exists t \in f^{-1}(\psi(\mathcal{L}(G(S)))) \) such that \( \varphi_{\text{safe}}(\delta(S)(t)) = 0 \). By (15), \( \exists q \in \delta(S)(t) \) such that \( q \in Q \setminus Q_H \). By the definition of \( \delta(S)(\cdot) \), \( \exists \mu \in \mathcal{L}(G(S)) \) such that \( f^{-1}(\psi(\mu)) = t \). Therefore, \( \exists \psi(\mu) \in \psi(\mathcal{L}(G(S))) \) such that \( \delta(q_0, \psi(\mu)) = q \in Q \setminus Q_H \). By Definition 7, \( \mathcal{L}(S)(G) \) is not network safe. \( \leftarrow \) By contradiction. Suppose that \( \mathcal{L}(S)(G) \) is not network safe. By Definitions 1 and 7, \( \exists \mu \in \mathcal{L}(G(S)) \) such that \( \delta(q_0, \psi(\mu)) \in Q \setminus Q_H \). We write \( \delta(q_0, \psi(\mu)) = (q, x, m, \phi, y, m, p) \) and \( f^{-1}(\psi(\mu)) = t \). By Proposition 1, \( q = \delta(q_0, \psi(\mu)) \in Q \setminus Q_H \). By the definition of \( \delta(S)(\cdot), q \in \delta(S)(t) \). Since \( q \in Q \setminus Q_H \), \( \varphi_{\text{safe}}(\delta(S)(t)) = 0 \), which contradicts \( \varphi_{\text{safe}}(\delta(S)(t)) = 1 \).

Proposition 5 says that the networked safety enforcement problem can be reduced to a SE-based property \( \varphi_{\text{safe}} \) enforcement problem. To ensure that the language of the controlled system is network safe, all the state estimates that may be generated by the controlled system must satisfy \( \varphi_{\text{safe}} \). Using the proposed algorithm, a BTS can be easily extended to its network counterpart NBTS, which exhaustively searches all the admissible control actions and state estimates that may be generated under these control actions. Benefiting from such a “global view,” we can always synthesize a supervisor (if it exists) from the NBTS such that all the state estimates that may be generated under it satisfy \( \varphi_{\text{safe}} \).

### B. Networked Supervisor Synthesis

We first generalize a BTS to an NBTS using the introduced techniques. Formally, an NBTS \( T \) w.r.t. \( G \) is a seven-tuple

\[
T = (Q_T^F, Q_T^E, h_T^F, h_T^E, \Sigma_T, \alpha, \pi, y_0)
\]

where \( Q_T^F \subseteq Q \times \Theta_x \times \Theta_x \times \Pi \) is the set of \( Y \)-states; \( Q_T^E \subseteq (Q \times \Theta_x \times \Theta_x \times \Pi) \times \Pi \) is the set of \( Z \)-states, and each \( Z \)-state \( z = (I(z), \Pi(z)) \) consists of two parts such that \( I(z) \) and \( \Pi(z) \) denote the information state and the control command parts of \( z \), respectively; \( h_T^F : Q_T^F \times Y \rightarrow Q_T^F \) is a transition function from \( Y \)-states to \( Z \)-states, which is defined as follows: For any
Algorithm 1: Calculating $T'$.

**Input:** Automaton $G$ and SE-based property $\varphi_{\text{safety}}$

**Output:** An AINC $T'$

1. Construct an NBTS $T$ using $G$ as described above;
2. Remove all the Z-states $z \in Q^T_Z$ in $T$ such that $\varphi_{\text{safety}}(FC(I(z))) = 0$ from $T$, and set $T \leftarrow Ac(T)$;
3. repeat
   4. Set $Q'_Y \leftarrow Q^T_Y$ and $Q'_Z \leftarrow Q^T_Z$;
   5. for $y \in Q'_Y$, one by one do
      6. if $C_T(y) = 0$ then
         7. Set $Q'_Y \leftarrow Q'_Y \setminus \{y\}$;
      8. Set $T \leftarrow Ac(T)$;
   9. for $z \in Q'_Z$, one by one do
      10. if $(\exists (q, \theta_n, \theta_c) \in I(z)) [\theta_n] = \sigma_1, n_1 \cdots (\sigma_k, n_k) \not\in \varepsilon \land \sigma_1$ implies $h^T_Z(y, \pi)!$ holds
         11. Set $Q'_Z \leftarrow Q'_Z \setminus \{z\}$;
      12. Set $T \leftarrow Ac(T)$;
   13. until $Q'_Y = \emptyset$ and $Q'_Z = \emptyset$;
14. return $T' \leftarrow T$.

$y \in Q^T_Y$ and any $\pi \in \Pi$, $h^T_{\alpha \beta}(y, \pi) = (\text{DUR}(y, \pi), \pi)$; $h^T_{\alpha \beta} : Q^T_Y \times \Sigma_o \rightarrow Q^T_Y$ is a transition function from Z-states to Y-states, which is defined as follows: For any $z \in Q^T_Z$ and any $\sigma \in \Sigma_o$, $h^T_{\alpha \beta}(z, \sigma) = \text{DOR}(I(z), \sigma); \Sigma_o$ is the set of observable events; $\Pi$ is the set of admissible control commands; $y_0 = \emptyset$ is the initial Y-state.

**Remark 8:** The NBTS is an extension of the BTS in the case of communication delays and losses. An NBTS also consists of the following two types of states: 1) Y-states and 2) Z-states. The Y-state estimates all the augmented states that the system can reach immediately after a new observable event communication (by applying “delayed observable reach” on its predecessor). From the Y-state, all the admissible control decisions are considered. The Z-state collects all the augmented states that are reachable from its predecessor Y-state under a given control action (by applying “delayed observable reach” on its predecessor).

We say that the Z-state $z \in Q^T_Y$ satisfies the SE-based property $\varphi_{\text{safety}}$ if all the first components of its information state part satisfy $\varphi_{\text{safety}}$, i.e., $\varphi_{\text{safety}}(FC(I(z))) = 1$. An NBTS traverses the entire reachable space of the Y- and Z-states. Specifically, in each Y-state $y$, the NBTS considers all the admissible control actions and Z-states that can be reached from $y$ following the execution of these control actions. Some of these control actions may be “bad” decisions since they may cause the Z-states to violate the property of $\varphi_{\text{safety}}$ now or in the future. To exclude all these “bad” control actions, we next compute the largest subgraph of an NBTS, called all inclusive networked controller (AINC), which searches only the “good” control decisions. Before we formally construct the AINC, we first introduce several notions.

**Definition 9:** We say that an NBTS $T$ is complete, if
1. For all $y \in Q^T_Y$, $C_T(y) \neq 0$;
2. For all $z \in Q^T_Z$ and all $\sigma \in \Sigma_o$, $\exists (q, \theta_n, \theta_c) \in I(z) [\theta_n] = (\sigma_1, n_1) \cdots (\sigma_k, n_k) \not\in \varepsilon \land \sigma_1$ implies $h^T_Z(y, \pi)!$ holds.

The first property says that for any reachable Y-state, there exists at least one control action that is defined at this state, and the second property says that if an observable event is active at the Z-state, we cannot disable its occurrence. Then, we introduce the definition of the AINC as follows.

**Definition 10:** The AINC is the largest subgraph of an NBTS $T$, denoted by $T' = (Q^T_Y, Q^T_Z, h^T_{\alpha \beta}, h^T_{\gamma \delta}, \Sigma_o, \Pi, y_0)$, such that: 1) $T'$ is complete and 2) $T'$ satisfies the SE-based property $\varphi_{\text{safety}}$, i.e., $\varphi_{\text{safety}}(FC(I(z))) = 1$.

The AINC is an extension of all inclusive controller (AIC) [7] in the networked DESs. It is the largest subgraph of an NBTS $T$ that satisfies completeness and the SE-based property $\varphi_{\text{safety}}$. Given an NBTS $T$, we denote $Ac(T)$ the accessible part of $T$. $Ac(T)$ can be obtained by deleting all the Y- and Z-states in $T$ that are not reachable from the initial Y-state $y_0$, and all the transitions that are attached to these states. Algorithm 1 formally constructs the AINC for $T$.

Algorithm 1 builds the AINC in a similar way as the authors in [7] constructed the AIC. Roughly speaking, the procedure consists of the following two steps: 1) We construct the NBTS $T$ and prune all its Z-states $z \in Q^T_Z$ that violate $\varphi_{\text{safety}}$, i.e., $\varphi_{\text{safety}}(FC(I(z))) = 0$ (Lines 1 and 2) and 2) we repeatedly prune all the states violating completeness from the remaining part of $T$ until convergence is achieved (the repeat-until loop on Line 3). We next prove the correctness of Algorithm 1.

**Proposition 6:** Algorithm 1 correctly constructs the AINC $T'$.

Let $T'$ be the returned AINC of Algorithm 1. We write $y \xrightarrow{\sigma_1} z$ if $h^T_{\alpha \beta}(y, \pi) = z$ and $z \xrightarrow{\sigma_2} y$ if $h^T_{\alpha \beta}(z, \sigma) = y$. Let $S$ be a networked supervisor included in $T'$ and $t = \sigma_1 \cdots \sigma_n \in f^{-1}(\psi^T(\mathcal{Z}(G_S)))$ be an observed string. The execution of $t$ leads to an alternating sequence of Y-states and Z-states $y_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} y_n \xrightarrow{\sigma_1 \cdots \sigma_n} z_n$.

We denote $IS^Y_S(t)$ and $IS^Z_S(t)$ as the last Y- and Z-states of $y_0 z_n \cdots y_n z_n$, respectively, i.e., $IS^Y_S(t) = y_n$ and $IS^Z_S(t) = z_n$. We now show how to “decode” a networked supervisor from $T'$.

**Definition 11:** A networked supervisor $S = (A, \gamma)$ with $A = (X, \Sigma, \xi, x_0)$ is said to be included in an AINC $T'$, if for all $t \in f^{-1}(\psi^T(\mathcal{Z}(G_S)))$, $\xi(x_0, t) = I(IS^Y_S(t))$ and $\gamma(\xi(x_0, t)) \in C_T(IS^Z_S(t))$. To complete the definition of $S$, for all $t \in \Sigma^*_o \setminus f^{-1}(\psi^T(\mathcal{Z}(G_S)))$, define $\xi(x_0, t) = \text{spec}$ with $\gamma(\xi(\text{spec})) = \Sigma_{\text{uc}}$.

We denote by $S(T')$ all the networked supervisors included in $T'$. The following theorem and its corollary show that $S(T')$ collects only networked safe supervisors.

**Theorem 2:** Let $S \in S(T')$ be a networked supervisor included in $T'$. For all $t \in f^{-1}(\psi^T(\mathcal{Z}(G_S)))$, $\mathcal{E}_S(t) = FC(I(IS^Z_S(t)))$.

**Corollary 1:** Let $S \in S(T')$ be a networked supervisor included in $T'$. Then, $\mathcal{L}(S/G)$ is networked safe w.r.t. $Q_H \subseteq Q$ and $G$.}

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
**Proof:** By Theorem 2 and the fact that $T^*$ is an AINC, we have $\varphi_{\text{safe}}(S(t)) = 1$ for all $t \in f^{-1}(y) (L(G_S))$. By Proposition 5, $S(G) \cup \mathcal{L}(G_S)$ is networked safe.

By Corollary 1, we can always select a maximal networked safe supervisor from $S(T^*)$ in the following sense: Let $IS^Y_{T^*}(t) \in Q^Y_T$ be the current $Y$-state such that AINC $T^*$ is in (after observing $t$). Since $S(T^*)$ collects only networked safe supervisors, all the control actions in $CT_{T^*}(IS^Y_{T^*}(t))$ are safe control actions. We can simply pick a "greedily maximal" control action from $CT_{T^*}(IS^Y_{T^*}(t))$. Since we focus on estimating states in this article, the formal algorithm for synthesizing a maximal supervisor is beyond the scope of this article.

**VII. CONCLUSION**

In supervisory control, communication delays and losses are unavoidable when communication between the plant and the supervisor for observation, and between the supervisor and the actuator for control are carried out over some shared networks. We assume, in this article, that: 1) the delays do not change the order of the observations and controls; 2) both the observation delays and control delays have upper bounds; and 3) both the consecutive observation losses and the consecutive control losses also have upper bounds. A novel framework for supervisory control under communication delays and losses has been established. Under this framework, an algorithm for online state estimation of a controlled system has been proposed. The proposed algorithm can be used to solve the supervisor synthesis problem in networked DESs. As an application, we show how to use the existing methods to synthesize maximally permissible and safe networked supervisors.

**REFERENCES**

[1] P. J. Ramadge and W. M. Wonham, “Supervisory control of a class of discrete event processes,” SIAM J. Control Optim., vol. 25, no. 1, pp. 206–230, 1987.

[2] F. Lin and W. M. Wonham, “On observability of discrete-event systems,” Inf. Sci., vol. 44, no. 3, pp. 173–198, 1988.

[3] M. Heymann and F. Lin, “On-line control of partially observed discrete event systems,” Discrete Event Dyn. Syst.: Theory Appl., vol. 4, no. 3, pp. 221–236, 1994.

[4] N. B. Hadjilouanne, S. Lafortune, and F. Lin, “Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation,” Discrete Event Dyn. Syst.: Theory Appl., vol. 6, no. 4, pp. 379–427, 1996.

[5] S. Takai and T. Ushio, “Effective computation of an $l_{\infty}(g)$-closed, controllable, and observable sublanguage arising in supervisory control,” Syst. Control Lett., vol. 49, no. 3, pp. 191–200, 2002.

[6] K. Cai, R. Zhang, and W. M. Wonham, “Relative observability of discrete-event systems and its supremal sublanguages,” IEEE Trans. Autom. Control, vol. 60, no. 3, pp. 659–670, Mar. 2015.

[7] X. Yin and S. Lafortune, “Synthesis of maximally permissible supervisors for partially-observed discrete-event systems,” IEEE Trans. Autom. Control, vol. 61, no. 5, pp. 1239–1254, May 2016.

[8] X. Yin and S. Lafortune, “A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems,” IEEE Trans. Autom. Control, vol. 61, no. 8, pp. 2140–2154, Aug. 2016.

[9] X. Yin and S. Lafortune, “Synthesis of maximally-permissive supervisors for the range control problem,” IEEE Trans. Autom. Control, vol. 62, no. 8, pp. 3914–3929, Aug. 2017.

[10] X. Yin and S. Lafortune, “Synthesis of maximally permissive nonblocking supervisors for the lower bound containment problem,” IEEE Trans. Autom. Control, vol. 63, no. 12, pp. 4435–4441, Dec. 2018.

[11] M. V. S. Alves, L. K. Carvalho, and J. C. Basilio, “New algorithms for verification of relative observability and computation of supremal relatively observable sublanguage,” IEEE Trans. Autom. Control, vol. 62, no. 11, pp. 5902–5908, Nov. 2017.

[12] Y. Yi, X. Yin, and S. Lafortune, “Local mean payoff supervisory control for discrete event systems,” IEEE Trans. Autom. Control, vol. 67, no. 5, pp. 2282–2297, May 2022.

[13] F. Wang, S. Shu, and F. Lin, “On network observability of discrete event system,” in Proc. IEEE 54th Conf. Decis. Control, 2015, pp. 3528–3533.

[14] S. Ji, X. Yin, and S. Lafortune, “Supervisor synthesis for networked discrete event systems with communication delays,” IEEE Trans. Autom. Control, vol. 60, no. 8, pp. 2183–2188, Aug. 2015.

[15] S. Shu and F. Lin, “Deterministic networked control of discrete event systems with nondeterministic communication delays,” IEEE Trans. Autom. Control, vol. 62, no. 1, pp. 190–205, Jan. 2017.

[16] M. V. S. Alves and J. C. Basilio, “State estimation and detectability of networked discrete event systems with multi-channel communication networks,” in Proc. IEEE Amer. Control Conf., 2019, pp. 5602–5607.

[17] Y. Hou, W. Wang, Y. Zang, F. Lin, M. Yu, and C. Gong, “Relative network observability and its relation with network observability,” IEEE Trans. Autom. Control, vol. 65, no. 8, pp. 3584–3591, Aug. 2020.

[18] F. Lin, W. Wang, L. Han, and B. Shen, “State estimation of multi-channel networked discrete event systems,” IEEE Trans. Control Netw. Syst., vol. 7, no. 1, pp. 53–63, Mar. 2020.

[19] L. Zhou, S. Shu, and F. Lin, “Detectability of discrete-event systems under nondeterministic observations,” IEEE Trans. Autom. Sci. Eng., vol. 18, no. 3, pp. 1315–1327, Jul. 2021.

[20] S. Shu and F. Lin, “Predictive networked control of discrete event systems,” IEEE Trans. Autom. Control, vol. 62, no. 9, pp. 4698–4705, Sep. 2017.

[21] F. Lin, “Control of networked discrete event systems: Dealing with communication delays and losses,” SIAM J. Control Optim., vol. 52, no. 2, pp. 1276–1298, 2014.

[22] L. Liu, X. Yin, S. Shu, F. Lin, and S. Li, “Online supervisory control of networked discrete-event systems with control delays,” IEEE Trans. Autom. Control, vol. 67, no. 5, pp. 2314–2329, May 2022.

[23] Y. Hou, Y. Ji, G. Wang, C.-Y. Weng, and Q. Li, “Online state estimation for supervisor synthesis in discrete event systems with communication delays and losses,” 2022, arXiv:2201.04800.

[24] A. Rashidinejad, M. Reniers, and L. Feng, “Supervisory control of timed discrete-event systems subject to communication delays and non-FIFO observations,” in Proc. 14th IFAC Workshop Discrete Event Syst., vol. 51, no. 7, pp. 456–463, 2018.

[25] M. Alves, L. Carvalho, and J. Basilio, “Supervisory control of networked discrete event systems with timing structure,” IEEE Trans. Autom. Control, vol. 66, no. 5, pp. 2206–2218, May 2021.

[26] R. Tai, L. Lin, Y. Zhu, and R. Su, “A new modeling framework for networked discrete-event systems,” Automatica, vol. 138, pp. 1–7, 2022.

[27] L. Lin, Y. Zhu, R. Tai, S. Ware, and R. Su, “Networked supervisor synthesis against lossy channels with bounded network delays as non-networked synthesis,” Automatica, vol. 142, 2022, Art. no. 110279.

[28] Y. Zhu, L. Lin, R. Tai, and R. Su, “Distributed control of timed networked system against communication delays,” in Proc. IEEE 17th Int. Conf. Control Autom., 2022, pp. 1008–1013.

[29] Y. Zhu, L. Lin, S. Ware, and R. Su, “Supervisor synthesis for networked discrete event systems with communication delays and lossy channels,” in Proc. IEEE 58th Conf. Decis. Control, 2019, pp. 6730–6735.

[30] Z. Liu, J. Hou, X. Yin, and S. Li, “Modeling and analysis of networked supervisory control systems with multiple control channels,” in Proc. IEEE 66th Conf. Decis. Control, 2021, pp. 316–323.

[31] B. Zhao, F. Lin, C. Wang, X. Zhang, M. Polis, and Y. Wang, “Supervisory control of networked timed discrete event systems and its applications to power distribution networks,” IEEE Trans. Control Netw. Syst., vol. 4, no. 2, pp. 146–158, Jun. 2017.

[32] S.-J. Park and K.-H. Cho, “Nonblocking supervisory control of timed discrete event systems under communication delays: The existence conditions,” Automatica, vol. 44, no. 4, pp. 1011–1019, 2008.

[33] C. Miao, S. Shu, and F. Lin, “Predictive supervisory control for timed discrete event systems under communication delays,” in Proc. IEEE 55th Conf. Decis. Control, 2016, pp. 6724–6729.

[34] C. G. Cassandras and S. Lafortune, Introduction to Discrete Event Systems, 2nd ed. New York, NY, USA: Springer, 2008.
Yunfeng Hou received the B.Eng. degree in electrical engineering from the Shandong University of Technology, Zibo, China, in 2012, and the M.S. and Ph.D. degrees in control science and engineering from the University of Shanghai for Science and Technology, Shanghai, China, in 2015 and 2020, respectively. He is currently a Research Associate with the Institute of Machine Intelligence, University of Shanghai for Science and Technology. His research interests include modeling, control, and optimization in discrete-event systems, intelligent transportation systems, and cooperative control for multirobot systems.

Yunfeng Ji (Member, IEEE) received the B.S. degree in information science and technology and M.S. degree in control engineering from the University of Shanghai for Science and Technology, Shanghai, China, in 2012 and 2015, respectively, and the Ph.D. degree in sport training from the China Table Tennis College, Shanghai, China, in 2018. In 2018, he was a Senior Visiting Scholar with the Department of Informatik, University of Hamburg, Hamburg, Germany. He is currently with the Institute of Machine Intelligence, University of Shanghai for Science and Technology. His research interests include table tennis robots, image processing, machine learning, and object detection.

Gang Wang (Member, IEEE) received the B.Sc. degree in information and computing science and the Ph.D. degree in systems analysis and integration from the University of Shanghai for Science and Technology, Shanghai, China, in 2012 and 2017, respectively. From 2017 to 2019, he was a Research Associate with the Department of Electrical and Biomedical Engineering, University of Nevada, Reno, NV, USA. From 2021 to 2022, he was a Postdoctoral Fellow with the Hong Kong Centre for Logistics Robotics and the T Stone Robotics Institute, The Chinese University of Hong Kong, Hong Kong. He is currently an Associate Professor with the Institute of Machine Intelligence, University of Shanghai for Science and Technology. His research interests include distributed control of nonlinear systems, adaptive control, and robotics. Dr. Wang was a finalist for the Best Paper Award at the 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.

Qingdu Li received the B.S. degree in electronic engineering and the M.S. degree in microelectronics engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2002 and 2004, respectively, and the Ph.D. degree in robotics from Nanyang Technological University, Singapore, in 2008. In 2011, he was a Postdoctoral Researcher with the Department of Theoretical and Applied Mechanics, Cornell University, Ithaca, NY, USA. In 2017, he was a Research Fellow with the Department of Informatics, University of Hamburg, Hamburg, Germany. He is currently a Professor with the Institute of Machine Intelligence, University of Shanghai for Science and Technology, Shanghai, China. His research interests include biped robots, dynamic walking, nonlinear circuits, complex systems, and bifurcation and chaos. Dr. Li serves as an Editorial Board Member of Complexity. He was a recipient of the Chongqing Natural Science Award (Second Prize), in 2016.