Overview of networked supervisory control with imperfect communication channels

Yuting Zhu1, Liyong Lin1, Ruochen Tai1, Rong Su1

Abstract

This paper presents an overview of the networked supervisory control frameworks for discrete event systems with imperfect communication networks, which are divided into the centralized setup, the decentralized setup and the distributed setup. The state-of-the-art works on networked supervisory control of discrete-event systems addressing the channel imperfections that are caused by channel delays, data losses or non-FIFO channels are discussed. By presenting the key concepts and main results of each representative work, we analyze the pros and cons of different approaches. Finally, we also provide a summary of the existing works, which roughly follow two different lines of thinking and result in two different verification or synthesis approaches. The first approach utilizes simple, non-networked plant models but relies on the development of sophisticated concepts of network controllability and observability to capture network imperfections, while the second approach embeds relatively complex yet verifiable channel models into the model of the networked plant and adopts the standard concepts of controllability and observability for the (verification and) synthesis of networked supervisors. Some future research topics are also presented.

Keywords Networked control system · Communication channel · Supervisor synthesis · Delays · Losses

This work was financially supported by Singapore National Research Foundation via Delta-NTU Corporate Lab Program (DELTA-NTU CORP LAB-SMA-RP2 SU RONG M4061925.043) and by Singapore Ministry of Education Tier 1 Academic Research Grant (2018-T1-001-245 (RG91/18)).
1 Introduction

Networked systems are commonly seen nowadays due to the advancement of information and communication technologies. They offer many implementation advantages, and in the meantime present more development challenges. The popular concept of cyber-physical systems (CPS) is by nature a networked one, as it requires a closed-loop integration of cyber commands and physical operations via information networks. Networked systems have found numerous applications related to smart cities development and smart manufacturing movement, also known as Industry 4.0. They have been thoroughly studied in the systems and control community for about twenty years under the umbrella of multi-agent systems, leading to numerous publications that address a broad scope of topics such as cooperative/non-cooperative control (Lewis et al. 2013; Yu et al. 2017), cyber-physical system control (Cardenas et al. 2008; Sridhar et al. 2011; Baheti and Gill 2011; Mo et al. 2011), hybrid systems (Koutsoukos et al. 2000), distributed optimization (Wang and Lemmon 2010), task planning (Erol et al. 1994; Erol 1996), social sourcing and distributed learning (Choi et al. 2009), and cybersecurity analysis and control (Hu and Yue 2012; Walsh et al. 2002; Zhang et al. 2001; Kim et al. 2003). One of the most important research aspects is how information is generated, propagated and used in the network, directly affecting the overall network performance.

The discrete-event system (DES) community, in particular, the supervisory control community, has also been actively involved in this popular research trend, as illustrated in a large number of publications on modular/distributed control and decentralized control, where a target system is comprised of a set of local agents, interacting with each other via specific synchronization mechanisms, and each agent is managed by one or several local controllers via specific information fusion mechanisms. The goal is to ensure safety, i.e., no bad behaviors will happen, liveness, i.e., good behaviors will (eventually) happen, and optimality, i.e., the attained system performance should be the best among all possible ones. It is thus desirable to survey which goals have been achieved in network control of the DES based on the different frameworks and identify what challenges need to be dealt with in the future. There are several major control frameworks in the DES community, namely centralized control, decentralized control, and distributed control which also called modular control. Therefore, it is necessary to investigate the different architecture of supervisory control with imperfect communication channels, which are extended from the ideal communication channels. The major difference between decentralized control and distributed control frameworks is arguably whether each local supervisor can ensure “local” controllability and observability properties without any assistance of neighboring supervisors. However, in any circumstance, each supervisor needs to communicate with the plant and possibly other supervisors to ensure successful closed-loop control. For this purpose, this paper intends to provide a review of the state-of-the-art networked control architecture (centralized, decentralized, distributed control) with the different properties of communication channels (FIFO/non-FIFO, communication delays or losses).

Another review paper on networked DES can be found in Lin (2020). In that paper, the author first discusses how to model networked discrete-event systems and how to obtain state estimates using a networked observer, following the seminal modeling framework introduced in Shu and Lin (2014a, 2014b, 2016a). Then, the authors introduce and solve several supervisory control problems of networked DES, including (i) (nondeterministic) supervisory control problem, (ii) deterministic supervisory control problem, (iii) non-blocking supervisory control problem, (iv) robust supervisory control problem, (v) robust
deterministic supervisory control problem, (vi) deterministic decentralized supervisory control problem, and (vii) nonblocking decentralized supervisory control problem. Necessary and sufficient conditions for solving the above problems are presented. Finally, the authors list some open problems in supervisory control of networked discrete-event systems. In contrast to Lin (2020), we also provide a detailed review on different modeling frameworks, in particular, including those that are based on modelling channels as automata. We believe this work provides a complementary viewpoint on the important subject of networked control and hope to arouse interests from both DES theorists and industrial practitioners.

The paper is organized as follows. Section 2 provides an overview of supervisory control architectures under ideal communication channels and then proposes several fundamental questions that need to be addressed. Furthermore, works on networked supervisory control are categorized in Section 3, based on the approaches to model the communication channels with delays and losses, the methodology to generate resilient supervisors and the complexity of the synthesis. Finally, we raise some challenging issues in Section 4.

2 Overview of supervisory control architectures under ideal communication channels

Before presenting the main control frameworks in the DES community, some preliminary knowledge is explained in details below. Let Σ denote a finite alphabet. Let \( \mathcal{G} \) denote a finite state automaton over alphabet Σ, which is a four tuple \( \mathcal{G} = (Q, \Sigma, \delta, Q_0) \), where \( Q \) is the finite set of states, \( \delta \subseteq Q \times \Sigma \times Q \) the transition relation and \( Q_0 \subseteq Q \) the initial state set. A (finite state) supervisor over control constraint \( (\Sigma_c, \Sigma_o) \) is a deterministic finite state automaton \( S = (X, \Sigma, \xi, x_0) \) that satisfies the controllability and observability constraints:

i) (controllability) for any state \( x \in X \) and any uncontrollable event \( \sigma \in \Sigma_{uc} \), \( \xi(x, \sigma)! \),

ii) (observability) for any state \( x \in X \) and any unobservable event \( \sigma \in \Sigma_{uo} \), \( \xi(x, \sigma)! \) implies \( \xi(x, \sigma) = x \),

After the plant \( G \) generates a string \( s \), the supervisor will observe \( P_o(s) \) via natural projection \( P_o : \Sigma^* \rightarrow \Sigma_{o}^* \), and then a control command \( V(P_o(s)) \) is issued. When an (enabled) observable event occurs, the control action is instantaneously updated (Cassandras and Lafortune 2009). The supervisor \( V \) sends control action by enabling (or disabling) event based on the observations of the strings generated by plant \( G \), so as to make the closed behavior equal to a given language \( K \subseteq L(G) \). A supervisor on \( G \) w.r.t. control constraint \( (\Sigma_c, \Sigma_o) \) is a map \( V : P_o(L(G)) \rightarrow \Gamma \), where \( P_o : \Sigma^* \rightarrow \Sigma_{o}^* \) is the natural projection and \( \Gamma := \{ \gamma \subseteq \Sigma \mid \Sigma_{uc} \subseteq \gamma \} \), \( V(w) \) is the control command issued by the supervisor after observing \( w \in P_o(L(G)) \). \( V/G \) denotes the closed-loop system of \( G \) under the supervision of \( V \). The closed-behavior \( L(V/G) \) of \( V/G \) is inductively defined as follows:

1. \( e \in L(V/G) \),
2. if \( s \in L(V/G) \), \( \sigma \in V(P_o(s)) \) and \( s\sigma \in L(G) \), then \( s\sigma \in L(V/G) \),
3. no other strings belong to \( L(V/G) \).

The control map \( V \) is finitely representable if there exists a finite state supervisor \( S = (X, \Sigma, \xi, x_0) \) over \( (\Sigma_c, \Sigma_o) \) such that \( L(S\|G) = L(V/G) \), where \( \| \) is the synchronous
product operation on automata. It has been shown that, as long as a closed-loop language $K \subseteq L(G)$ is controllable (Ramadge and Murray Wonham 1987) and observable (Lin and Murray Wonham 1988), there always exists a finitely-representable supervisor control map $V$ such that $L(V/G) = K$. If the marking information is important, then we also can consider the marked behavior $L_m(G)$ and $L_m(V/G)$ (Cassandras and Lafortune 2009), where $G = (Q, \Sigma, \delta, Q_0, Q_m)$ is now a 5-tuple and $Q_m \subseteq Q$ is the set of marked states.

In a general decentralized supervisory control paradigm (Yoo and Lafortune 2002), depicted in Fig. 1, where the alphabet $\Sigma$ is covered by a set of local alphabets $\{\Sigma_i \subseteq \Sigma | i \in I = \{1, \cdots, n\}\}$, $\Sigma_{c,i} := \Sigma_c \cap \Sigma_i = \Sigma_{c,e,i} \cup \Sigma_{c,d,i}$, where $\Sigma_{c,e,i}$ is the set of controllable events which are enabled by default and $\Sigma_{c,d,i}$ is the set of controllable events which are disabled by default. The local decisions over $\Sigma_{c,e}$ are processed by the conjunctive fusion rule while the local decisions over $\Sigma_{c,d}$ are processed by the disjunctive fusion rule. The plant $G$ is controlled by a set of local supervisors $V_i : P_{o,i}(L(G)) \rightarrow \Gamma (i \in I)$ with $P_{o,i} : \Sigma^* \rightarrow (\Sigma_i \cap \Sigma_o)^*$ being the natural projection, via either the conjunctive fusion rule (denoted by “⊕”) or the disjunctive fusion rule (denoted by “⊗”). In the conjunctive fusion rule (CFR), for each $s \in L(G)$, we have

$$V_{c,i}(P_{o,i}(s)) := \{\sigma \in \Sigma_{c,e,i} \subseteq \Sigma_c \cap \Sigma_i | P_{o,i}(s)^{-1} \sigma \cap K \neq \emptyset \} \cup (\Sigma_c - \Sigma_{c,e,i}) \cup \Sigma_{uc}.$$  \hspace{1cm} (1)

The conjunctive permissive control map is $V_c : L(G) \rightarrow \Gamma$, where

$$(\forall s \in L(G))V_c(s) := \bigcap_{i \in I} V_{c,i}(P_{o,i}(s)).$$

That is, in conjunctive fusion rule, a controllable event is enabled by the decentralized supervisor if and only if it is enabled by all the local supervisors. In the disjunctive fusion rule (DFR), for each $s \in L(G)$, we have

$$V_{d,i}(P_{o,i}(s)) := \{\sigma \in \Sigma_{c,d,i} \subseteq \Sigma_c \cap \Sigma_i | (P_{o,i}(s)^{-1} \sigma \cap \overline{K}) \cap L(G) \subseteq \overline{K} \} \cup \Sigma_{uc}.$$  \hspace{1cm} (2)

The disjunctive anti-permissive control map is $V_d : L(G) \rightarrow \Gamma$, where
That is, in the disjunctive fusion rule, a controllable event is enabled by the decentralized supervisor if and only if it is enabled by at least one of the local supervisors.

The general supervisory control map is $V_g : L(G) \rightarrow \Gamma$ where

$$(\forall s \in L(G))V_g(s) := \bigcup_{i \in I} V_{d,i}(P_{o,i}(s)).$$

To make this control architecture work, i.e., a given sublanguage $K \subseteq L_m(G)$ is equal to $L_m(V_g/G)$, the sublanguage $K$ must be controllable, $L_m(G)$-closed, and co-observable, which is defined as follows:

- The language $K_c := L(V_c/G)$ is (C&P) co-observable w.r.t. $G$ and $(\Sigma_{c,e} := \bigcup_{i \in I} \Sigma_{c,e,i}, \{P_{o,i} | i \in I\})$, if for all $s \in \overline{K}$ and $\sigma \in \Sigma_{c,e}$,

  $$s\sigma \in L(G) - \overline{K_c} \Rightarrow (\exists i \in I)P_{o,i}^{-1}(s)\sigma \cap \overline{K_c} = \emptyset \land \sigma \in \Sigma_{c,e,i}.$$  

- The language $K_d := L(V_d/G)$ is (D&A) co-observable w.r.t. $G$ and $(\Sigma_{c,d} := \bigcup_{i \in I} \Sigma_{c,d,i}, \{P_{o,i} | i \in I\})$, if for all $s \in \overline{K_d}$ and $\sigma \in \Sigma_{c,d}$,

  $$s\sigma \in \overline{K_d} \Rightarrow (\exists i \in I)(P_{o,i}^{-1}(s)\sigma \cap \overline{K_d})\sigma \cap L(G) \subseteq \overline{K_d} \land \sigma \in \Sigma_{c,d,i}.$$  

Intuitively, (C&P) co-observability ensures that each bad string can be identified by at least one local supervisor, and (D&A) co-observability ensures that each good string can be confirmed by at least one local supervisor. These two co-observability concepts are not compatible when more than one local supervisor exists, but they are both reduced to the same observability concept in the centralized framework.

Each local observation in the decentralized control strategy may be enhanced by allowing event communication among local supervisors. There are a lot of works on this topic, focusing on the synchronous communication for the control of decentralized DES, where the communication is assumed to involve zero delay. In Barrett and LaFortune (2000), a novel information structure formalism is presented. It includes which actions observable by each controller, which controllers communicate to other controllers, what symbols are communicated, when controllers initiate communication, and what information may be inferred by each of the controllers following any sequence of actions. Based on this structure, both anticipating controllers and myopic controllers are studied and characterized. Laurie Ricker and Rudie (1999), Ricker (2008), and Wang et al. (2008) investigate the minimal communication policies for decentralized control where a communication policy is said to be minimal if removing one or more communications of event occurrences in the dynamic evolution of the system renders a correct solution incorrect. Specifically, the minimal communication problem is translated into one that can be solved on a Markovian mode in Ricker (2008), based on which the minimal cost communication protocol can be found by solving an optimization problem over a set of Markov chains. Under an assumption on the absence of cycles (other than self-loops) in the system model, Wang et al. (2008) proposes a polynomial-time algorithm in the size of the state space of the plant for the synthesis of communication policies, which is an improvement compared with previous works. Two approaches are summarized in Ricker (2013) for the synthesis of communication policies.
protocols: state-based communication and event-occurrence communication. However, since these works consider channels that involve zero delays, they might not be applicable for some realistic scenarios where delays happen unavoidably in the shared communication network.

Notice that in the decentralized control framework, each local control law \( V_i \ (i \in I) \) cannot ensure the local closed-loop behavior \( L(V_i/G_i) \) to be controllable and observable. Instead, it requires a genuine co-design of all local control laws \( \{V_i|i \in I\} \) to ensure global controllability and observability with proper fusion rules accompanied by suitable concepts of (C&P, D&A) co-observability, i.e., for every single undesired string in the system, it requires a specific joint effort of all local supervisors to prevent the string from happening. Unfortunately, it is often undecidable (Tripakis 2004) whether there exists a decentralized supervisor to achieve the goal unless some restrictions are imposed (Rudie and Wonham 1991). To avoid the undecidability issue faced by decentralized control, significant efforts have been made in developing a modular control (or distributed control) framework, where the set of all undesired behaviors is decomposed into a finite set of languages, each of which will be handled by one specific local supervisor. The general architecture of coordinated control is illustrated in Fig. 2, where there are multiple local components \( \{G_i|i \in I\} \), where the alphabet of each \( G_i \) is \( \Sigma_i \). There are a finite number of local requirements \( \{E_i \subseteq L_m(G_i)|i \in I\} \). The goal is to design a set of local control laws \( \{V_i:L(G_i) \rightarrow \Gamma|i \in I\} \) together with a coordinator \( C : P_C(L(\bigcup_{i \in I} G_i)) \rightarrow \Gamma \), where \( P_C : (\bigcup_{i \in I} \Sigma_i)^* \rightarrow \Sigma_C^* \) is the natural projection with \( \bigcup_{i,j \in I, i \neq j} \Sigma_i \cap \Sigma_j \subseteq \Sigma_C \subseteq \bigcup_{i \in I} \Sigma_i \) being the alphabet of the coordinator, such that the following property holds: Let \( G = \bigcap_{i \in I} G_i \).

- For each \( i \in I \),
  - \( L_m(V_i/G_i) \subseteq E_i \);  
  - \( L_m(V_i/G_i) \) is controllable and observable w.r.t. \( G_i \) and \((\Sigma_{c,i},\Sigma_{o,i})\);
Vi/Gi is nonblocking.

- Supremality or maximality can be imposed on \( L_m(V/G_i) \), depending on the choice of observability.

- The closed-loop system \( V/G \) is nonblocking, where \( V = \land_{i \in I} V_i \land C \).

Efficient synthesis methods have been developed in the literature based on either bottom-up abstraction (Feng and Wonham 2008; Rong et al. 2010a, 2010b, 2011) or top-down decomposition (Komenda et al. 2015b) to solve this problem.

In contrast to the existing works on multi-agent systems, where the quality of networked communication among agents, e.g., signal noises and transmission errors, message delays and dropouts, plays one critical role in system analysis and control, DES control theories rarely consider such imperfect communication. In the standard Ramadge-Wonham supervisory control theory, it is assumed that event executions are instantaneous and asynchronous, which was later relaxed by introducing max-plus automata (Gaubert 1995), time-weighted automata (Su et al. 2011) and timed Petri nets (Wang 2012), where events have durations and asynchrony of event executions refers to the starting times of relevant events, instead of their ending times. The key assumption of synchronous event generation, sending and receiving must hold in each component and supervisor, lead to the FIFO control and observation channels. Due to this assumption, details of network communication processes are not critically important, even when observable outputs of the plant may be delayed (but unanimously) or lost due to transmission failures. So one big question is: what will happen if the FIFO and losslessness assumption does not hold in the observation and control channels? To facilitate a simple discussion, we first narrow ourselves down to a simple network setup, depicted in Fig. 3, where there is one plant \( G \), one supervisor \( S \), one directed observation channel \( OC \) from \( G \) to \( S \), and one directed control channel \( CC \) from \( S \) to \( G \).
Each observable output generated by the plant \(G\) is transmitted to \(S\) via \(OC\), and each control command generated by \(S\) is transmitted to \(G\) via \(CC\). We will address the impact of channel imperfections on the overall performance of the (networked) closed-loop system \((G,S)\). More specifically, we will only focus on message delays and dropouts in both \(OC\) and \(CC\). The first challenge is how to model channel delays imposed on observation messages and control commands, especially, when delays can void the FIFO assumption. Due to computability and computational complexity concerns, current works mainly focus on regular languages or equivalent models. So it is a common practice to assume a known upper bound of delays for each message\(^1\). If the message has not been received before the upper bound of delays is passed, it is assumed to be lost or dropped out - so the upper bound also serves as a “time-out” mechanism. Delays can be measured in the number of elapsed ticks or in the number of elapsed plant events, depending on a user’s needs. Once a delay model is specified, the next challenge is how to model interactions between delayed channels and the plant \(G\) and supervisor \(S\). This model is crucial, as it directly affects the subsequent concepts of network controllability and network observability, thus, determines whether a proposed networked control framework is practically feasible. In this work, only discrete time models, but not dense time models, will be taken into the consideration. We remark that there are also research works on networked supervisory control that consider dense time models (Zhivoglyadov and Middleton 2003).

After having proper channel models and channel-(\(G,S)\) interaction models, the following fundamental questions need to be answered:

1. How to model the networked closed-loop system?
2. What conditions may ensure the existence of a networked supervisor resilient to network imperfections?
3. How to synthesize resilient supervisors against network imperfections such as delays and/or losses?
4. Is it possible to carry out the synthesis efficiently?

In the next section, we shall review the state-of-the-art works for each setup, aiming to answer the above questions. We will categorize existing frameworks based on specific models and synthesis strategies.

3 Review of state-of-the-art networked control frameworks

3.1 state-of-the-art networked control with \(OC\) delays

We organize the existing works depending on whether they consider i) only \(OC\) delays, or ii) both \(OC\) and \(CC\) delays. i) is a special case of ii), so we will only briefly describe those works that address i), while those works that address ii) will be discussed in more details in Section 3.2.5.

\(^1\) There are notable exceptions. For example, in Tripakis (2004), it is shown that the existence of decentralized controllers in the cases of unbounded-delay is an undecidable problem.
3.1.1 OC delays in centralized control

In Alves et al. (2017), the authors consider a non-FIFO observation channel with delays and losses, while the control channel is assumed to be lossless and has no delay (thus effectively FIFO). The model of the timed networked discrete event systems (TNDES) is proposed, where an ordinary finite automaton model \( G = (X, \Sigma, \delta, x_0) \) of the plant is augmented with a timing structure \( t : X \times \Sigma \rightarrow \mathbb{R}^+ \) to specify the minimal transition activation time. Then, an untimed one that models TNDES is recursively constructed. New properties of networked controllability and networked observability are defined and used to characterize the existence of a networked supervisor. All the relevant languages considered in Alves et al. (2017) are prefix-closed, and thus the property of non-blockingness is not studied. Compared with the timed discrete event systems (TDES) model, the number of states used for the representation of a system may be significantly reduced by using TNDES. However, a model transformation to an untimed model still needs to be carried out. This may reduce the benefit of using TNDES. It is noteworthy that the number of transitions of the untimed model is exponential in the size of the alphabet. We assume that the number of observable events is \( |\Sigma_c| \), number of controllable command is \( |\Sigma_c| \), with delay in \( N \) steps, the state size of observation channel of TDES is \( 2 \cdot \Sigma_c^* \cdot N \) while state size of untimed observation channel model is \( 2 \cdot \Sigma_c^* \). The same condition for the state space explosion exists in the control command channel.

In Alves et al. (2019), the authors consider the problem of the design of robust supervisors that are able to cope with intermittent loss of observations and also make the controlled system achieve the specification language under nominal operation. Necessary and sufficient conditions, i.e., robust controllability and \( K \)-observability, for the existence of a robust supervisor that is able to cope with intermittent loss of observations is presented. The property of relative observability is also extended to robust relative observability. Furthermore, the verification conditions of robust controllability and \( K \)-observability extends classical Ramadge-Wonham controllability and observability properties. However, there is no consideration of delays in the communication channels, which may add the work some limitations. The problem of synthesis of a robust supervisor when the characterizing conditions fail is also not studied.

3.1.2 OC delays in decentralized control

In Park and Cho (2007b), the authors introduce a particular OC delay model, where after each control command is sent, there may be several uncontrollable event firings in the system before a new control command is generated, i.e., message transmission in OC is not instantaneous, and its maximum duration is determined by the maximum number of consecutive uncontrollable event firings. Three assumptions are made, which are listed below:

1. Each locally controllable event is locally observable, i.e., \( \Sigma_{c,j} \subseteq \Sigma_{e,j} \);
2. The number of consecutive occurrences of uncontrollable events in the plant \( G \) is upper bounded with a finite bound;
3. Messages in both OC and CC (after fusion) are FIFO.

Following the framework of Ramadge and Murray Wonham (1987), a language \( K \subseteq L_m(G) \) is controllable with respect to \( G \) if \( \overline{K \Sigma_{uc}^* \cap L(G)} \subseteq \overline{K} \) and it is
L_m(G)-closed if \( \overline{K} \cap L_m(G) = K \). To achieve a given language specification \( K \subseteq L_m(G) \) of a plant \( G \), the authors introduce a new (C&P) control law, extended from the standard architecture shown in Section 1, which is stated below: for each \( s \in L(G) \), \( V_i(P_{oi}(s)) : = \{ \sigma \in \Sigma_{c,i} | P^{-1}_{oi}(s) \Sigma_{uc} \sigma \cap \overline{K} \neq \emptyset \} \cup (\Sigma - \Sigma_{c,i}) \), the basic idea is if there exists some string \( s' \) that is observably identical to \( s \) and \( s' \sigma \epsilon \in \overline{K} \) for some \( u \in \Sigma^* \), then \( \sigma \) is allowed by \( V_i \) after observing \( P_{oi}(s) \).

After the occurrence of a string \( s \in L(G) \), the set of enabled events of a decentralized supervisor is denoted as \( V_{dec}(s) \), which is also called a decentralized supervisory control action following the conjunctive fusion rule under communication delays. The control should be determined based on what the decentralized supervisor sees. \( S_{dec} \) is a decentralized supervisor, where \( S_{dec}(t) \) denotes the enabled event set when the decentralized supervisor executes \( t \).

The final (C&P) control law is shown as follows: for all \( s \in L(G) \),

\[
V_{dec}(s) := \begin{cases} 
\emptyset & (\exists t \in \Sigma^*)(\exists u \in \Sigma_{uc}^s) s = tu \land (\forall u' \in \{u\} - \{\epsilon, u\}) S_{dec}(t \land S_{dec}(tu') = \emptyset, \\
\cap_i V_i(P_{oi}(s)) & \text{otherwise.} 
\end{cases}
\]

That is, the control law \( V_{dec} \) will generate a new command, only when the current string \( s \) is not a suffix string \(^2\) of \( t \) via an uncontrollable sequence \( u \), such that there is a control command at \( t \), but there is no new command afterwards till now, due to observation delay in \( OC \). The closed-loop behaviour is defined as follow:

- \( \epsilon \in L(V_{dec}/G) \),
- for all \( s \in L(V_{dec}/G) \) and \( \sigma \in \Sigma \) with \( s \sigma \in L(G) \), \( \sigma \in L(V_{dec}/G) \) if \( (\exists t \in \Sigma^*) (\exists u \in \Sigma_{uc}^s) s = tu \land \sigma \in V_{dec}(t) \land (\forall v \in \{u\} - \{\epsilon\}) V_{dec}(tv) = \emptyset. \)

Similar to Yoo and Lafortune (2002), to make the proposed (C&P) control law works, the authors extend the concept of (C&P) co-observability to properly handle \( OC \) delays.

**Definition 1** A sub-language \( K \subseteq L_m(G) \) is delay-coobservable w.r.t. \( G \) and \( \{ \Sigma_{oi} | i \in I \} \), if for all \( s \in K, u \in \Sigma_{uc}^s \) with \( su \in K \) and for all \( \sigma \in \Sigma_{uc} \), \( s \sigma \in L(G) - K \Rightarrow (\exists i \in I) P^{-1}_{oi}(s) \Sigma_{uc}^s \sigma \cap \overline{K} = \emptyset \land \sigma \in \Sigma_{oi}. \)

**Theorem 1** Given a language specification \( K \subseteq L_m(G) \), for a plant \( G \) with communication delays, there exists a nonblocking decentralized supervisor \( S_{dec} \) such that \( L_m(S_{dec}/G) = K \) if and only if

1. \( K \) is controllable w.r.t. \( G \),
2. \( K \) is delay-coobservable w.r.t. \( (\Sigma_{oi}, \Sigma_{oi}) | i \in \{1, 2, \ldots, n\} \),
3. \( K \) is \( L_m(G) \)-closed.

---

\(^2\) a suffix of a string \( s \) is a substring that occurs at the end of \( s \)
The computational complexity of verifying the delay-coobservability of a language $K$ is $O(|Q^K|\cdot |X|^2)$, where $Q^K$ is the state space of a deterministic automaton that recognizes $K$, and $X$ is the state set of $G$.

Although the existence problem of a non-blocking supervisor has been solved in this paper, there are also some restrictions: 1) the assumption that all locally controllable events are locally observable is slightly restrictive; 2) the assumption that the number of possible consecutive occurrences of uncontrollable events in the plant is limited within a finite bound is a bit restrictive, as it excludes the possibility of a loop that only contains uncontrollable events; 3) this work only deals with the verification problem and cannot be used when the delay-coobservability property fails. Thus, it is of great importance to consider the synthesis problem when this property fails.

In Tripakis (2004), Hiraishi (2009), and Sadid et al. (2015), the authors take supervisor communication delays into consideration under the decentralized control architecture with communication among local supervisors. As a key issue in the supervisory control of networked DES, the decidability of the verification and synthesis problems is investigated in Tripakis (2004) and Hiraishi (2009). In Tripakis (2004), the problem of decentralized control with communication is studied, where delays are either bounded by a given constant or unbounded. Communication channels are assumed to be FIFO and lossless. It is shown to be undecidable to check the existence of two local controllers such that a set of responsiveness properties is satisfied, in both the case of unbounded-delay communication and the case of no communication. The decidability of joint observability with bounded-delay communication is also shown. By enforcing bisimilarity between the controlled system and the specification, the decentralized control problem is shown to be undecidable in Hiraishi (2009). This work also presents two sufficient conditions to make the decentralized control problem decidable for finite-state controllers. The first condition is when the communication involves $k$-bounded-delay, and the other is when any cycle in the state transition diagram of the system contains an event observable by all controllers.

### 3.1.3 OC delays in modular/distributed control

In Zhang et al. (2016a) and Zhang et al. (2016b), the authors start from the DES distributed control scheme called “supervisor localization”, which describes a systematic top-down approach to design distributed controllers which collectively achieve global optimal and nonblocking supervision. Assuming that inter-agent communication of selected “communication events” may be subject to unknown time delays but with no data loss, a property of ‘delay-robustness’ is proposed and shown to be polynomial-time verifiable, and that such tests serve to distinguish between events that are delay-critical and those that are not. In addition, timed DES is adopted as the system model in Zhang et al. (2016b) so that delays can be explicitly measured by the number of elapsed ticks. Then, the property of timed delay-robustness with respect to the timed channel is defined, which extends the untimed counterpart.

In Kalyon et al. (2011), the authors consider the control of distributed systems composed of subsystems communicating asynchronously; the aim is to build local controllers that restrict the behavior of a distributed system in order to satisfy a global state avoidance property. By modeling the modular systems as communicating finite state machines with reliable unbounded FIFO queues between subsystem, Kalyon et al. (2011) and Kalyon et al. (2013) adopt the technique of abstract interpretation for over-approximating reachability and co-reachability to ensure finite termination in distributed controller synthesis.
As an extension of Kalyon et al. (2011) and Kalyon et al. (2013) provides the full process allowing to derive controllers from a state-based specification and a plant by means of state-based estimates and abstract interpretation techniques, whereas (Kalyon et al. 2011) only presents the control point of view with an overview of the state-based estimates computation point of view.

Based on distributed Petri net, Darondeau and Ricker (2012) proposes to synthesize distributed controller starting from a monolithic supervisor, where the communication channel is non-FIFO and lossless. By encoding the information to be exchanged, which is a crucial contribution of Darondeau and Ricker (2012), the distributed Petri net synthesis technique is illustrated on the 3 Dining Philosopher problem.

3.1.4 State estimation and detectability related to OC delays

By modelling the system as a timed DES where an explicit tick event is used to measure the passage of one unit of time, Sadid et al. (2015) verifies the robustness of all synchronous communication protocols under conditions of fixed or finitely-bounded delay, not just optimal communication protocols. However, this work only addresses the problem of verification of robustness against delay; if the communication protocol is not robust against delays, then the results of this work cannot be used.

In Zhou et al. (2019), the authors consider the problem of observation nondeterminism. In order to find a supervisor to ensure the safety of the controlled system for the nondeterministic control problem, it requires the upper-bound language of the controlled system equals to a given specification language. The observations of event is deterministic in the conventional supervisory control framework. But in networked system with delays or losses, the observation of events become non-deterministic. Under this condition, we define the “O-observability” property to represent a supervisor is safe, which means that the upper-bound language of the controlled system equals to a given specification language. We define the key condition for the existence of such a supervisor as O-observability. A method is proposed to check the O-observability by constructing an augmented automaton. In the augmented automaton, each state is a doubleton of which the first element is the current state of the original system (which is used to track all the strings generated by the original system) and the second element is a set of state estimates of all the possible observations when a string occurs. A subset is defined that includes all the bad states in the augmented automaton. It is shown that O-observability holds if and only if there are bad states in the augmented automaton. A state-estimate-based supervisor is synthesized to control the given system to be safe when the nondeterministic control problem is solvable.

In Sasi and Lin (2018), the authors study the detectability for networked discrete event systems impacted by network delays and losses, which is concerned with the ability to determine the current and subsequent states. This work considers both network detectability and network D-detectability. Network detectability allows the determination of the state of a networked discrete event system. In contrast, networked D-detectability allows one to distinguish between some pairs of states of the systems. The characterization and verification of these two detectability properties are also provided in Sasi and Lin (2018).

The problem of state estimation under communication delays, for the non-FIFO channel, has been considered in Lin et al. (2019). It considers multiple channels, each of which is a FIFO channel with a different delay. Thus, the resulting channel is non-FIFO. The first, conservative, method for computing state estimate is directly extended from an existing approach, which computes an over-approximation in the sense that this state estimate may contain states
that the system cannot be in. The second, exact, method distinguishes between the occurrence of an event and the reception of an event, as in the general discussions in the beginning of this subsection. Each communication channel is modeled with these two types of events. State estimates can be computed based on the synchronous product of the plant model and the channel models. Both online computation and offline computation methods are proposed. This work extends the state estimation method when no communication delay is involved.

In Alves and Basilio (2019), both communication losses and delays in the observation channel are considered. The observation channel is assumed to be non-FIFO since there are multiple FIFO observation channels. The plant is in general a non-deterministic finite automaton, and delay is measured by the number of occurrences of events. By a transformation to untimed nondeterministic automaton, networked $D$-detectability definition is proposed, which is equivalent to the $D$-detectability in the untimed nondeterministic model. The $D$-detectability properties studied include strong $D$-detectability, weak $D$-detectability, strong periodic $D$-detectability, weak periodic $D$-detectability.

In this work, we shall not provide an in-depth cover of state-estimation of networked discrete event systems. For the reader that is interested in this topic, please refer to Lin et al. (2019).

### 3.2 state-of-the-art networked control with OC and CC delays

In this subsection, we organize the existing papers based on specific models of channel delays and the relevant control architectures.

#### 3.2.1 An input-output control architecture with non-FIFO channels

In Balemi and Brunner (1992), the authors present an input-output interpretation of the supervision of discrete-event systems. In their networked setup, the schematic diagram is shown in Fig. 4, where the supervisor sends the commands in $\Sigma_c$ through the control...
channel and the plant sends the responses in $\Sigma_{uc}$ to the supervisor. More specifically, the plant produces responses in reaction to commands from the supervisor and, symmetrically, the supervisor accepts the responses of the plant and produces commands for the plant. In this input-output perspective, each command from the supervisor is a controllable event, while each response from the plant is an uncontrollable event. Balemi and Brunner (1992) assumes that the observation channel can hold multiple responses in $\Sigma_{uc}$ and the control channel can only hold one command in $\Sigma_c$. The plant is given by $G = (\Sigma, L_G, M_G)$, where $L_G$ denotes the closed-behavior of $G$ and $M_G \subseteq L_G$ denotes the marked behavior of $G$. Similarly, the supervisor is given by $S = (\Sigma, L_S, M_S)$. The composition of $G$ and $S$ is denoted by $G \| S = (\Sigma, L_G^c, M_G^c)$. The composition $G \| S$ of $G$ and $S$ is said to be well-posed if $G \| S = G \| (\Sigma_{uc} \cdot \Sigma_c) S$, where $||_{(\Sigma_{uc}, \Sigma_c)}$ denotes the prioritized synchronous composition operator w.r.t. $(\Sigma_{uc}, \Sigma_c)$. In their networked supervisor synthesis problem formulation, Balemi and Brunner (1992) requires a) $\emptyset \in P_{\Sigma_{uc}} (M_G^c) \subseteq L'_{\text{spec}}$, where $L'_{\text{spec}} \subseteq \Sigma_{uc}^*$ denotes the specification and $P_{\Sigma_{uc}} : \Sigma^* \rightarrow \Sigma_{uc}^*$ denotes the natural projection, b) $G \| S$ is well-posed, c) $S$ is non-blocking in the closed-loop system and a marking state in $S$ eventually corresponds to a marking state in $G$, and d) the behavior of $S$ is unaffected by permutation of order of commands and responses in $G$. To solve the synthesis problem, the notion of a delay-insensitive language is proposed in Balemi and Brunner (1992). The main theorem that characterizes the existence of a delay-insensitive supervisor $S$ is given in the following.

**Theorem 2** There exists a delay-insensitive supervisor $S$ for $G$ such that $M_G^c = K$ iff $K$ is controllable, delay-insensitive and $M_G$-closed.

It turns out that, although the class $D(L)$ of delay-insensitive sublanguages of a language $L$ is not closed under union, the class $CD(L)$ of delay-insensitive and controllable sublanguages of $L$ is closed under union, and the supremal element $\text{supCD}(L)$ exists in $CD(L)$. Thus, the following result holds, which allows one to solve the synthesis problem even if the above characterizing conditions fail.

**Theorem 3** The supervisory control problem with delays is solvable if and only if $\emptyset \in P_{\Sigma_{uc}} (\text{supCD}(M_G \cap P_{\Sigma_{uc}}^{-1}(L'_{\text{spec}})))$. If a solution exists, then the supervisor $S = (\Sigma, K_D, K_D)$ with $K_D = \text{supCD}(M_G \cap P_{\Sigma_{uc}}^{-1}(L'_{\text{spec}}))$ is the maximally permissive solution.

Another significant result is the following, which allows a networked supervisor to be synthesized as delay-free communications.

**Theorem 4** If $M_G$ and $L'_{\text{spec}}$ are delay-insensitive, then $K = \supC(M_G \cap P_{\Sigma_{uc}}^{-1}(L'_{\text{spec}}))$ is also delay-insensitive.

In the extended work (Balemi 1994), the effect of communication delays on the connection of a plant and a supervisor is defined via a delay operator. Formally, the delay of a language $L \subseteq \Sigma^*$ with respect to $\Sigma' \subseteq \Sigma$ is denoted by $\text{delay}([\Sigma^*](L))$, which is defined to be the smallest superlanguage of $L$ such that for any $s, t \in \Sigma^*, \sigma' \in \Sigma'$, $\sigma \in \Sigma - \Sigma'$

1. $L \subseteq \text{delay}([\Sigma^*](L))$
2. $s\sigma't \in \text{delay}([\Sigma^*](L)) \implies s\sigma't \in \text{delay}([\Sigma^*](L))$

 Springer
Intuitively, $delay[\Sigma'](L)$ is the closure of $L$ under the permutation of the substrings $\sigma' \sigma \in \Sigma' (\Sigma - \Sigma')$ to $\sigma \sigma' \in (\Sigma - \Sigma')\Sigma'$. A notion of well-posedness is defined, which informally requires that any command coming from the supervisor to the plant must be accepted by the plant; the continuation of the current string in the plant with a command from the supervisor must form a new string contained again in the closed-behavior $L_G$ of the plant. The class $\omega[L_G]$ of sublanguages of $L$ for supervisors enforcing well-posedness of the connection with a plant having language $L_G$ is shown to be closed under union; the supremal element is denoted by $supo[L_G](L)$. The unmarked supervisor synthesis problem with delays is formulated in Balemi (1994) as follows. Given a plant $G = (\Sigma, L_G, G)$ and a prefix-closed specification language $L'_{spec} \subseteq \Sigma^\omega$, find a supervisor $S = (\Sigma, L_S, S)$ such that (a) $L_S \subseteq L_G$, (b) $\emptyset \subset P_{\Sigma_{uc}}(G)$, (c) the connection of $S$ and $G$ is well-posed. It is shown that the above unmarked supervisor synthesis problem with delays, where all relevant languages are prefix-closed, has a solution if and only if for the language $L_S = sup\{K \mid K \in \omega[L_G](L_G) \land L_G \cap delay[\Sigma'](K) \subset P_{\Sigma_{uc}}(L_{spec})\}$, it holds that $\emptyset \subset P_{\Sigma_{uc}}(L_S)$. If a solution exists, then the supervisor with language $L_S$ is a solution. To ensure computability, the language $L_G$ is required to be self-well-imposed, that is, $L_G \supseteq L_G \Sigma^\omega \cap delay[\Sigma_{uc}](L_G \Sigma^\omega_{uc})$ and $L_G \supseteq L_G \Sigma^\omega_{uc} \cap delay[\Sigma_{uc}](L_G \Sigma^\omega)$. If $L_G$ is self-well-imposed, then the above $L_S$ can be replaced with $supC(L_G \cap P_{\Sigma_{uc}}^{-1}(L_{spec}'))$ (cf. Theorem 4). The restriction to plants with so called memoryless languages allows one to compute in polynomial time a supervisor solving the networked supervisor synthesis problem with communication delays. A prefix-closed language $L \subseteq \Sigma^\omega$ is said to be memoryless if for any $s, s' \in L$ such that $s \in delay[\Sigma_{uc}](\{s'\})$, then for any $t \in \Sigma^\omega$, $st \in L \Leftrightarrow s't \in L$. Moreover, all the characterizing results have also been extended to dealing with non-prefix-closed cases in Balemi (1994) to address the non-blockingness property. It is argued that most systems can be properly modeled to satisfy the memoryless property and another technical condition to allow polynomial time synthesis of networked supervisors.

### 3.2.2 An implicit model of OC and CC delays in centralized control

In Park and Cho (2006), the authors investigate the existence conditions of a delay-robust non-blocking supervisor that can achieve a given language specification, and the assumption is under FIFO observation channel and control channel with bounded communication delays. The schematic diagram is shown in Fig. 5, where the delay bound is assumed to be $D$. Following the general discussion, it is assumed that a control pattern only contains controllable events, and every controllable event is disabled by default and is permitted to occur only if it is enabled by a supervisor. Thus, a supervisor is a map $S : L(G) \rightarrow 2^\Sigma$. According to the schematic diagram, uncontrollable events may subsequently occur within a delay bound $D$ (from the moment when the supervisor sends a control pattern), and thus the supervisory control action $S(t)$ for a string $t$ can be actually applied to the system either after the string $t$ or further after any subsequent occurrence of uncontrollable events $a_i \ldots a_l$ where $i \in \{1, \ldots, D\}$. The closed-loop behavior $L(S/G)$ is defined as follows: 1) $e \in L(S/G)$, and 2) for any $s \in L(S/G)$ and $\sigma \in \Sigma$ with $s\sigma \in L(S/G)$, $s\sigma \in L(S/G) \Leftrightarrow i) \sigma \in \Sigma_{uc}$ or, ii) $\exists r \in \Sigma, |s| - |t| \leq D$ and $\sigma \in S(t)$ whereas $tu = s$ for some $u \in \Sigma_{uc}$ and $S(v) = \emptyset$ for any $v \in \overline{u} - \{e\}$. For a specification $K \subseteq L_m(G)$ for $G$ subject to a delay bound $D$, the problem is to find necessary and sufficient conditions for the existence of a nonblocking supervisor $S$ such that $L_m(S/G) = K$, where as usual $L_m(S/G) = L(S/G) \cap L_m(G)$ and $S$ is nonblocking iff $L_m(S/G) = L(S/G)$. To solve the problem, it is assumed that every possible subsequent occurrence of uncontrollable events is limited within the delay bound $D$, i.e., $|ul| \leq D$ for any
\( u \in \Sigma^+ \) and \( s \in \Sigma^+ \) satisfying \( su \in L(G) \). Based on this assumption, the property of delay-nonconflictingness is formulated and the following characterization result is proved.

**Theorem 5** Given a specification \( K \subseteq L_m(G) \) for \( G \) subject to a delay bound \( D \), there exists a delay-robust nonblocking supervisor \( S \) such that \( L_m(S/G) = K \) iff

1. \( K \) is controllable w.r.t. \( G \) and \( \Sigma_{uc} \)
2. \( K \) is delay-nonconflicting w.r.t. \( G \) and \( D \)
3. \( K \) is \( L_m(G) \)-closed.

Suppose \( G \) is modelled by a finite-state automaton \( G = (Q, \Sigma, q_0, \delta, Q_m) \) and \( K \) is modelled by \( H = (\Sigma, X, x_0, X_m) \). Then, an algorithm is provided to perform the verification of the delay-nonconflictingness of \( K \), which has a computational complexity of \( O(||X||Q||\Sigma||D^D) \).

There are two limitations of this work. Firstly, there is no synthesis algorithm provided if the delay-nonconflictingness property fails. Secondly, it is assumed that every plant event is observable, which may not be realistic in practice. In Park and Cho (2007a), the authors extend the characterization results of their previous work (Park and Cho 2006) to the case of partial observation by proposing the notion of delay observability to replace the notion of delay-nonconflictingness defined for the full observation case. The delay observability property also assures no confliction in making a decision for controllable legal events under partial observation and communication delays. The setup assumes controllable events are observable, and it only addresses the verification problem, and thus the results cannot be used if the verification conditions fail.

In Lin (2014), the author considers the problem of networked supervisor synthesis, dealing with both communication delays and losses. In particular, delays and losses
occur in both the observation channel and the control channel, which are assumed to be FIFO. For the case when the control channel has no delays or losses, two observation maps are used for modelling observation delays and losses, respectively.

Let $G = (\mathcal{Q}, \Sigma, \delta, q_0)$. Now, let $\delta_o = \{(q, \sigma, q') \mid \delta(q, \sigma) = q' \land \sigma \in \Sigma_o\}$ denote the set of observable transitions and let $\delta_{uo} = \{(q, \sigma, q') \mid \delta(q, \sigma) = q' \land \sigma \in \Sigma_{uo}\}$ denote the set of unobservable transitions. Let $\delta_L \subseteq \delta_o$ denote the subset of observable transitions that may get lost in the communication. Let $\Theta_L : L(G) \rightarrow 2^\Sigma$ denote the observation mapping under losses, which is defined as follows: let $s = \sigma_1...\sigma_j...\sigma_k \in L(G)$, then $\Theta_L(s)$ is obtained by replacing each $\sigma_i$ in $s$ with $a) \{e\}$, if the corresponding transition $(q_i, \sigma_i, \delta(q_i, \sigma_i)) \in \delta_{uo}$, $b) \{\sigma_i\}$, if $(q_i, \sigma_i, \delta(q_i, \sigma_i)) \in \delta_o - \delta_L$ and $c) \{e, \sigma_i\}$, if $(q_i, \sigma_i, \delta(q_i, \sigma_i)) \in \delta_L$. That is, network losses lead to nondeterminism in observation and thus a string $s$ may lead to a set $\Theta_L(s)$ of observed strings. $\Theta_L(G)$ is recognized by $G_L = (\mathcal{Q}, \Sigma_o, \delta_{loss}, q_0)$, where $\delta_{loss} = \delta_o \cup \{(q, e, q') \mid (q, \sigma, q') \in \Sigma_{uo} \cup \delta_L\}$. To model $N$-bounded delays, $\Theta^D_N : \Sigma^* \rightarrow 2^\Sigma$ is used for delayed observation. Formally, for any $s \in L(G)$, $\Theta^D_N(s) := \{s_{i-} \mid i \in [0, N]\}$, where $s_{i-}$ is the prefix of $s$ with the last $i$ events removed. Intuitively, $s_{i-}$ may be observed after $s$ is executed, since the last $i$ events in $s$ may not be observed yet. With both observation delays and losses, the map $\Theta_{DL} = \Theta_L \circ \Theta^D_N$ is defined. The network observability is defined as follows, which is reduced to then observability when $\Theta_{DL}(s) = P_o(s)$ in the setup with no delays and no losses.

**Definition 2** Given a prefix-closed language $K \subseteq L(G)$ and the observation mapping under communication delays and losses described by $\Theta_{DL}$ with a delay upper bound $N$, $K$ is network observable with respect to $L(G)$ and $\Theta_{DL}$ if $\forall \sigma \in L(G)$, $s\sigma \in K \implies \exists t \in \Theta_{DL}(s), \forall s' \in \Theta^{-1}_{DL}(t), s' \in K \land s'\sigma \in L(G) \implies s'\sigma \in K$.

The author considers a state-estimate based networked supervisor $\gamma : 2^O \rightarrow 2^\Sigma$ that determines for each state estimate the set of events to be disabled. The state estimate of the supervisor after observing $t \in \Sigma^*$ is $E(t) = \{q \in Q | \exists s \in L(G) : t \in \Theta_{DL}(s) \land \delta(q_o, s) = q\}$. Then, the closed behavior $L(G, \gamma)$ of the supervised system is defined recursively as follows: a) $e \in L(G, \gamma)$, and b) $\forall s \in L(G, \gamma)$, $\forall \sigma \in \Sigma$, $s\sigma \in L(G, \gamma)$ iff $s\sigma \in L(G) \land (\sigma \in \Sigma_o \lor \exists t \in \Theta_{DL}(s) \land s\sigma \notin \gamma(E(t)))$. The following characterization result is obtained.

**Theorem 6** Assume a networked discrete event system $G$ with communication delays and losses in observation, described by $\Theta_{DL}$ with an upper bound $N$. Assume that there are no communication delays or losses in control. For a nonempty prefix-closed regular language $K \subseteq L(G)$, there exists a state estimate-based network supervisor $\gamma : 2^O \rightarrow 2^\Sigma$ such that $L(G, \gamma) = K$ if and only if (1) $K$ is controllable with respect to $L(G)$ and $\Sigma_{uc}$ and (2) $K$ is network observable with respect to $L(G)$ and $\Theta_{DL}$.

The author then considers the control channel with communication delays upper bounded by $M$ and $M$-bounded consecutive losses of control patterns when the observation channel has no losses or delays and under full observation. Let $K/s := \{s' \in \Sigma^* \mid ss' \in K\}$. Then, network controllability is defined in the following, which is reduced to controllability when $M = 0$.

**Definition 3** Given a prefix-closed language $K \subseteq L(G)$ and an upper bound $M$ on control delays and losses, $K$ is said to be network controllable with respect to $L(G)$ and $\Sigma_{uc}$ if $\forall s \in K, \forall \sigma \in \Sigma, s\sigma \in L(G) \land (\sigma \in \Sigma_{uc} \lor \sigma \notin K/s_{-1} \land \cdots \land \sigma \notin K/s_{-M} \lor s\sigma \in K$.
Unlike controllability, even if all events are controllable, \( K \) may not be network controllable, due to communication delays and losses in control. It is assumed that before a control action arrives, the system will use the previously received control action. Then, the closed behavior \( L(G,\gamma) \) of the supervised system is defined recursively as follows: a) \( \epsilon \in L(G,\gamma) \), and b) \( \forall s \in L(G,\gamma), \forall \sigma \in \Sigma, s\sigma \in L(G,\gamma) \) iff \( s\sigma \in L(G) \land (\sigma \in \Sigma_u \lor \sigma \notin \gamma(\delta(q_0,s-\epsilon)) \cap \gamma(\delta(q_0,s-M))) \). The following characterization result is obtained.

**Theorem 7** Assume a networked discrete event system \( G \) with communication delays and losses in control, bounded by \( M \). Assume full observation and that there are no communication delays or losses in observation. For a nonempty prefix-closed regular language \( K \subseteq L(G) \), there exists a state-based networked supervisor \( \gamma : Q \rightarrow 2^\Sigma \) such that \( L(G,\gamma) = K \) if and only if \( K \) is network controllable.

In the general case, when the control channel has both delays and losses bounded by \( M \) and the observation channel has both delays and losses, with upper bound \( N \), a separation principle holds. Thus, the following central result holds.

**Theorem 8** Assume a networked discrete event system \( G \) with communication delays and losses in observation, described by \( \Theta_{DL} \) (upper bound \( N \)), and with communication delays and losses in control bounded by \( M \). For a nonempty prefix-closed regular language \( K \subseteq L(G) \), there exists a networked supervisor \( \gamma : Q \rightarrow 2^\Sigma \) such that \( L(G,\gamma) = K \) if and only if (1) \( K \) is network controllable with respect to \( L(G) \) and \( \Sigma_{uc} \) and (2) \( K \) is network observable with respect to \( L(G) \) and \( \Theta_{DL} \).

While the non-networked supervisory control framework has been naturally extended to the networked setup in Lin (2014), this work only studies the verification conditions for characterizing the existence of a networked supervisor that achieves a given specification language. If network observability or network controllability fails, then a supervisor cannot be synthesized according to the results of this work. In Shu and Lin (2014a), the authors consider a slightly different problem setup where the behavior of the supervised system needs to be both adequate and safe. The observation channel is assumed to have a delay bound \( N_o \), and the control channel has a delay bound \( N_s \). Any control policy \( \pi \) has to satisfy both the observation feasibility and control feasibility. A supervisor may disable different events for the same sequence of event occurrences in the plant due to the random nature of the delays. Thus, to describe the behavior of the controlled system, two languages are defined in Shu and Lin (2014a). The first language, which is smaller, contains strings that can be generated under all observation and control delays, and is denoted by \( L_o(\pi/\hat{G}) \), where \( \hat{G} \) denotes the networked discrete-event system consisting of the plant \( G \) and the channel delays. The second language, which is larger, contains strings that can be generated under some possible observation and control delays, and is denoted by \( L_a(\pi/\hat{G}) \). The synthesis problem, i.e., the Supervisor Synthesis Problem for Networked Discrete Event Systems, is then formulated as follows: given a networked discrete event system \( \hat{G} \), a minimal required language \( K_r \), and a maximal admissible language \( K_a \), synthesize a supervisor with control policy \( \pi \) such that: 1) \( \pi \) is control feasible; 2) \( \pi \) is observation feasible; 3) \( L_o(\pi/\hat{G}) \supseteq K_r \); and 4) \( L_a(\pi/\hat{G}) \subseteq K_a \). It is shown in Shu and Lin (2014a) that the class of control feasible and observation feasible control policies is closed under conjunction. Furthermore, \( L_r(\pi_1 \land \pi_2/\hat{G}) = L_r(\pi_1/\hat{G}) \cap L_r(\pi_2/\hat{G}) \). Thus, there exists the minimal policy \( \pi_{min} \) such that \( L_r(\pi_{min}/\hat{G}) \supseteq K_r \). The following characterization result is then obtained.
Theorem 9 The Supervisor Synthesis Problem for Networked Discrete Event Systems is solvable if and only if \( L_a(\pi_{\text{min}}/\hat{G}) \subseteq K_a \). Furthermore, if it is solvable, then \( \pi_{\text{min}} \) is a solution.

Shu and Lin (2014a) also constructs \( \pi_{\text{min}} \) and proposes an implementation of \( \pi_{\text{min}} \) based on a state-estimated based control policy and a new observer. The off-line implementation of the minimal supervisor is of exponential complexity, while an on-line implementation can reduce the computational complexity to be polynomial in each step. An algorithm to check the condition \( L_a(\pi_{\text{min}}/\hat{G}) \subseteq K_a \) for the existence of a networked supervisor is proposed by constructing an augmented automaton \( G_{\text{aug}} \). A maximally-permissive control policy \( \pi_{\text{max}} \) can be obtained from \( \pi_{\text{min}} \) by an iterative construction. Compared with Lin (2014), the results developed in Shu and Lin (2014a) do not need to assume that a uniform delay is applied to all the events delayed in a string generated by the plant. However, due to the random delays, the language generated by the controlled system is non-deterministic, which makes it more difficult to analyze properties such as nonblockingness/deadlock-freeness. Even a string has a continuation, and the controlled system can still be blocked/deadlocked after that string, as the continuation may be disabled in some (but not all) trajectories the controlled system may take due to non-determinism.

In Shu and Lin (2016a), the authors explicitly address this non-determinism. In order to capture the non-determinism caused by the communication delays in the observation channel, delay observability is defined, which says that if two event sequences have different control requirements, then all the possible observations of them must be totally different. If there are no communication delays, then delay observability is reduced to observability. In order to capture the non-determinism caused by the communication delays in the control channel, delay controllability is defined. Delay controllability says that if one needs to disable an event, then that event must be controllable, and all the possible controls must disable it. If there are no communication delays, then delay controllability is reduced to controllability. Deterministic Networked Control Problem for Discrete Event Systems is formulated as follows: given a plant \( G \) subject to observation delays and control delays, and a specification language \( K \), find an observation feasible and control feasible control policy \( \pi \) such that the controlled system \( \pi/G \) satisfies \( L_r(\pi/G) = L_a(\pi/G) = K \). It turns out that the Deterministic Networked Control Problem for Discrete Event Systems is solvable iff \( K \) is delay controllable and the augmented language \( K_{\text{aug}} \) is delay observable. Algorithms are also proposed in Shu and Lin (2016a) to verify these two properties. If the language to be synthesized is not delay observable or delay controllable, its infimal delay controllable and delay observable superlanguage and maximal delay controllable and delay observable sublanguages are also constructed in Shu and Lin (2016a).

Example 1 A networked discrete event system \( \hat{G} \) is shown in Fig. 1(a) with \( N_o = N_c = 1 \). All the events are controllable and observable. The goal is to prevent the transition \((4,\alpha,5)\) from occurring. Let us consider the following control policy \( \pi \).

1. After observing event \( \beta \), the supervisor disables event \( \alpha \).
2. For all other cases, the supervisor enables all the events.

Intuitively, there are three possible cases or outcomes after the occurrence of \( \beta \). Case 1) No delay in observation and control. In this case, is observed and is disabled in time. This is illustrated by the middle path in Fig. 6. Case 2) No delay in observation but delay in control. In this case, \( \beta \) is observed and \( \alpha \) is not disabled in time. This is illustrated by the left path in
Fig. 6 Case 3) Delay in observation. In this case, whether there is delay in control or not is inconsequential. $\alpha$ is not disabled in time. This is illustrated by the right path in Fig. 6.

Since there are different possibilities, to describe the behavior of the controlled system, we define two languages for the controlled system. The first language, which is smaller, contains strings that can be generated under all observation and control delays and is denoted by $L_r(\pi/\hat{G})$. The second language, which is larger, contains strings that can be generated under some possible observation and control delays and is denoted by $L_a(\pi/\hat{G})$. For this example, $L_r(\pi/\hat{G})$ is shown in Fig. 7 and $L_a(\pi/\hat{G})$ is shown in Fig. 7.

Here is another example illustrated in Rashidinejad et al. (2018), the proposed approach achieves a networked supervisor which is controllable and observable and provides non-blockingness and safety for the plant. For the plant $P$ depicted in Fig. 8, all events are observable and controllable. The events executed in $P$ are supposed to be transferred through observation channels introducing a delay $N_o = 1$. Then the observed plant $OP$ as given in Fig. 9 is obtained.
In Wang et al. (2016), the authors study the robust control of networked discrete-event systems, where a supervisor is used to control several possible plants under communication delays and losses. The solution methodology is by translating the robust control problem into the conventional networked control problem by constructing an augmented automaton for all possible plants and an augmented specification automaton for the corresponding specification automata. This work considers the robust networked synthesis problem with both single objective and multiple objectives. The single objective case corresponds to when all the specifications are the same. A necessary and sufficient condition for the existence of a robust networked supervisor is derived. The multiple objectives case corresponds to when the specifications are different. Only a sufficient condition for the existence of a networked supervisor is obtained. In Shu and Lin (2016b), the authors extend the work of Lin (2014) to consider predictive networked supervisor, which predicts the impacts of communication delays and losses in the control channel in determining the control actions. The existence condition of a predictive networked supervisor is derived based on controllability and network observability. It is shown that predictive networked supervisors are better than non-predictive counterparts, and predictive networked supervisor is optimal in the sense that it is always a solution if the networked supervisory control problem is solvable (Figs. 10, 11, 12).

The process to compute the predictive supervisor is described in Lin (2020). Considering both observation delays and losses, the composition of two mappings \( \Theta^N_\text{No} \) representing observation delays and losses \( \Theta^\text{L} \) are completely captured by the mapping \( \Theta^N_\text{No} = \Theta^\text{L} \circ \Theta^N_\text{D} \).

After observing a string \( w \in \Theta^N_\text{No}(L(G)) \), the nonpredictive supervisor determines the current state estimate as \( E^\text{No}(w) \). Based on the observed string \( E^\text{No}(w) \), the nonpredictive supervisor \( S_{\text{np}} \) will issue the control command \( S_{\text{np}}(w) \) without considering the delay in control channels. However, predictive networked supervisor \( S_p \) considers control delays in its computation of control commands to determine the predicted states. Since control commands may be delayed by (at most) \( N_c \) steps, the predictive supervisor will add states that...
can be reached from states in $E_{N_e}^{N_e}(w)$ within $N_e$ steps to obtain the predicted state estimate as $R_{N_e}^{N_e}(E_{N_e}^{N_e}(w))$.

Compared with the research work in Shu and Lin (2014a) and Shu and Lin (2016b), it shows that predictive networked control is better than non-predictive networked control. The first point is that the existence conditions for predictive supervisors are weaker
than those for non-predictive supervisors, and any specification that can be achieved by any supervisor can also be achieved by a predictive supervisor. One more novelty is network controllability no longer needed, and the detailed discussion will be explained in Lin (2020). The following characterization result is obtained.

**Theorem 10** Consider a networked discrete-event system $G$ with observation delays and losses described by $\Theta_o$ and control delays and losses bounded by $N_c$. For a nonempty closed language $K \subseteq L(G)$, there exists a predictive networked supervisor $S_p$ such that $L_a(S_p/G) = K$ if and only if (i) $K$ is controllable with respect to $L(G)$ and $\Sigma_c$ and (ii) $K$ is network observable with respect to $L(G)$ and $\Theta_o^{N_o} + N_c$.

In Rashidinejad et al. (2019), a state estimate-based network supervisor has been replaced by a non-predictive supervisor, on the basis of this observed plant, a non-predictive supervisor is synthesized that provides safety and non-blockingness for the observed plant, by slightly adapting the Bertil-Wonham framework of supervisor synthesis for timed discrete-event systems. To deal with control delays, the nonpredictive supervisor achieved for the observed plant is transformed to a networked supervisor that enables the events beforehand.

In Zhao et al. (2015), the authors present an application of control of networked timed discrete event systems to power distribution networks. Under the assumption that delays and losses are bounded, a necessary and sufficient condition based on networked-timed discrete-event systems defined as T-controllability and network T-observability, and they are used to characterize the existence of a networked supervisor. The results are applied to 33-node (bus) test system, where the objective is to ensure that the total sub-station transformer power stays within prespecified safety limits. In Hou et al. (2019), the authors introduce and reduce relative network observability, under communication delays and losses in the FIFO observation channel and control channel, to network observability, which allows existing solutions for network observability verification to be directly applied; the application to the calculation of the supremal controllable and relatively network observable sublanguage is also shown.

### 3.2.3 An implicit model of OC and CC delays in decentralized control

Following the channel delay model proposed in Lin (2014), the authors in Shu and Lin (2014b) discuss decentralized control and investigate how to use the local supervisors to control the system in order to satisfy given specifications under the influence of both OC and CC delays. The specifications are described in two languages: a minimal language specifying the minimal required performance to achieve, and a maximal admissible language specifying the maximal set of legal behaviors. This work is an extension of the centralized framework described in Shu and Lin (2014a) to the decentralized networked control setting, assuming each local supervisor has its own communication channel with the plant and different communication channels may have different communication delays. It is assumed that in the OC, communication delays do not change the order of the events, i.e., the observation channel is FIFO. In the control channel, the initial control message is not delayed.

Due to observation delays, local supervisors may have different observations for the same string $s \in L(G)$. By adopting mapping functions to capture the relationship between the string observed by local supervisors and the string generated by the plant, the set of possible observations for local supervisor $S_i$ ($i \in I$) is denoted by $\Theta_i(s) = \{P_o, i(t) | \exists m \in \{0, \ldots, N_o, i\} t = s_m\}$, where $N_{o, i}$ is the upper bound of delays in the OC and $s_m$ is the prefix of $s$ obtained by
removing the last \(m\) events. \(\theta_i(s) \in \Theta_i(s)\) is used to denote a particular (delayed) observation. The control policy \(\pi_i\) implemented by local supervisor \(S_i\) should be calculated based on the current observation, that is, \(\pi_i : \Sigma^* \times \Sigma_{a,i}^* \rightarrow \Gamma\). This paper adopts the conjunctive fusion rule to combine control actions of local supervisors. The decentralized control map is \(\hat{\pi} : \Sigma^* \rightarrow \Gamma\), where

\[
(\forall s \in L(G)) \hat{\pi}(s) := \bigcap_{i \in I} \pi_i(s, \theta_i(s)).
\]

The closed-loop system is defined as \(\hat{\pi}/G\) in a usual way. The decentralized control problem is stated as follows.

**Problem 1** (DCPNDES): Given a plant \(G\), a minimal required prefix language \(K_r\) recognized by \(G_r = (Q_r, \Sigma, \delta_r, q_0, Q_r)\), and a maximal admissible prefix language \(K_a\) recognized by \(G_a = (Q_a, \Sigma, \delta_a, q_0, Q_a)\), we want to find a decentralized control policy \(\hat{\pi}\) such that

1. \(\hat{\pi}\) is co-control feasible, that is,

\[
(\forall i \in I)(\forall s \in L(G))\Sigma_{ic} \subseteq \pi_i(s, \theta_i(s));
\]

2. \(\hat{\pi}\) is co-observation feasible, that is, \((\forall i \in I)(\forall s, s' \in L(G))\theta_i(s) = \theta_i(s') \Rightarrow \pi_i(s, \theta_i(s)) = \pi_i(s', \theta_i(s'))\);

3. \(K_i \subseteq L_i(\hat{\pi}/G)\), where

- \(e \in L_i(\hat{\pi}/G)\),
- \(s\sigma \in L_i(\hat{\pi}/G) \iff s \in L_i(\hat{\pi}/G) \land s\sigma \in L(G) \land (\forall i \in I)(\forall m_i \in \{0, \ldots, N_{c,i}\}) (\forall \theta_i(s_{-m_i}) \in \Theta_i(s_{-m_i})) \sigma \in \pi_i(s_{-m_i}, \theta_i(s_{-m_i})).\)

4. \(L_a(\hat{\pi}/G) \subseteq K_a\), where

- \(e \in L_a(\hat{\pi}/G)\),
- \(s\sigma \in L_a(\hat{\pi}/G) \iff s \in L_a(\hat{\pi}/G) \land s\sigma \in L(G) \land (\forall i \in I)(\exists m_i \in \{0, \ldots, N_{c,i}\}) (\exists \theta_i(s_{-m_i}) \in \Theta_i(s_{-m_i})) \sigma \in \pi_i(s_{-m_i}, \theta_i(s_{-m_i})).\)

To find a decentralized control policy \(\hat{\pi}\) for DCPNDES, a minimal control policy is constructed for each local supervisor \(S_i\) \((i \in I)\). Assuming that, after a string \(s \in L(G)\) occurs, the supervisor \(S_i\) sees the string \(\theta_i(s)\), the current state estimate of this supervisor is then given by \(E_i(\theta_i(s)) = \{q \in G_i | (\exists t \in L(G_i)) \theta_i(s) \in \Theta_i(t) \land \delta_i(q_0,t) = q\}\). For state \(q\), the enabled event set under control delays is

\[
\Gamma_{r,i}^{N_{c,i}}(q) = \bigcup_{q' \in R_{r,i}^{N_{c,i}}(q)} \Gamma_r(q')
\]

where \(\Gamma_r(q')\) is the set of events defined at state \(q'\) in \(G_r\), \(N_{c,i}\) is the upper bound of delays in the control channel between \(S_i\) and \(G\), and

\[
R_{r,i}^{N_{c,i}}(q) = \left\{ \begin{array}{ll}
\{ \delta_r(q,t) | t \in \Sigma^* \land |t| \leq N_{c,i} \} & q \in Q_r, \\
\emptyset & \text{otherwise}.
\end{array} \right.
\]

Then the minimal control policy \(\pi_{i,\text{min}}(s, \theta_i(s))\) is given by

\[
\pi_{i,\text{min}}(s, \theta_i(s)) = \bigcup_{q \in E_i(\theta_i(s))} \Gamma_{r,i}^{N_{c,i}}(q) \cup \Sigma_{a,i}
\]
where \( \Sigma_{uc,i} = \Sigma_{uc} \cap \Sigma_i \) is the set of uncontrollable events for \( S_i \). Based on the minimal control policy for each local supervisor, the decentralized conjunctive control policy \( \hat{\pi}_{min} \) can be generated. Finally, an augmented automaton similar to that in Shu and Lin (2014a) is constructed to verify the existence of a solution.

### 3.2.4 An implicit model of OC and CC delays in modular control

With the same channel delay model introduced in Lin (2014), the authors in Komenda and Lin (2016) consider the following modular control problem with OC and CC delays, which are finitely bounded respectively by \( N_{o,i} \) and \( N_{c,i} \).

**Problem 2** Given generators \( G_1 \) and \( G_2 \), whose alphabets are \( \Sigma_1 \) and \( \Sigma_2 \), respectively. Let \( K \subseteq L(G_1 || G_2) \) be a prefix-closed specification. Let \( G_k = P_k(G_1) \parallel P_k(G_2) \) be a properly designed coordinator and \( K \) is conditionally decomposable with respect to \( \Sigma_1, \Sigma_2 \) and \( \Sigma_k \) (Komenda et al. 2015b), i.e., \( K = P_{1+k}(K) \parallel P_{2+k}(K) \), where \( P_{i+k} : (\Sigma_1 \cup \Sigma_2)^* \rightarrow (\Sigma_1 \cup \Sigma_k)^* \) (\( i = 1, 2 \)). Find two networked supervisors \( S_1 \) and \( S_2 \) with partial observation and OC and CC delays bounded by \( N_{o,i} \) and \( N_{c,i} \), such that

\[
\begin{align*}
L(S_i/(G_i || G_k)) & \subseteq P_{1+k}(K); \\
L(S_i/(G_1 \parallel G_k)) \cup L(S_j/(G_2 \parallel G_k)) & = K.
\end{align*}
\]

The definition of conditional decomposability is used in the approach based on the coordination control framework of Komenda et al. (2015a), where the property of conditional decomposability is used to specify that the global specification can be decomposed into local specifications. Conditional decomposability of a language \( L \subseteq (\Sigma_1 \cup \Sigma_2)^* \) with respect to sub-alphabets \( \Sigma_1 \) and \( \Sigma_2 \) requires to find another sub-alphabet \( \Sigma_k \subseteq \Sigma_1 \cup \Sigma_2 \) such that the language \( L \) is decomposable with respect to \( \Sigma_1 \cup \Sigma_k \) and \( \Sigma_2 \cup \Sigma_k \), i.e., \( L = P_{1+k}(L) \parallel P_{2+k}(L) \); \( L \) is then said to be conditionally decomposable w.r.t. \( \Sigma_1, \Sigma_2 \) and \( \Sigma_k \). The definition can be naturally extended to multiple sub-alphabets. The definition of conditional decomposability for two local sub-alphabets \( \Sigma_1, \Sigma_2 \) is given in the formulation of Problem 2. By decomposing \( K \) into \( P_{1+k}(K) \) and \( P_{2+k}(K) \), the authors show the possibility to treat coordination and predictive control together to handle possible observation and command delays, which brings the advantage of lower computational complexity. By replacing \( P_{o,i} \) with \( \Theta_i \) associated with \( N_{o,i} \) and \( N_{c,i} \), the authors extend the concept of conditional observability\(^3\) from Komenda et al. (2015b) to conditional network observability, and derive the following main result: Problem 2 is solvable if and only if

\[
\begin{align*}
K & \text{ is relaxed conditional controllable (Komenda et al. 2015a);} \\
K & \text{ is conditionally network observable with respect to } N_{o,1} + N_{c,1} \text{ and } N_{o,2} + N_{c,2}.
\end{align*}
\]

The synthesis complexity is \( O(2^{|P_{1+k}(K)|} + 2^{|P_{2+k}(K)|}) \)\(^4\), which in the worst case is double exponential-time, unless both \( P_{1+k} \) and \( P_{2+k} \) are natural observers (Wong and Wonham 2004).

---

\(^3\) Conditional controllability is one of the central notions to characterize the solvability of the coordination control problem.

\(^4\) \( ||L|| \) represents the minimum state size of the automata that recognizes \( L \).
3.2.5 An explicit automaton model of OC and CC delays

There are some existing works that explicitly model the communication delays and non-FIFO observations. A new framework for centralized networked supervisory control of timed discrete event systems is proposed in Rashidinejad et al. (2018) and Rashidinejad et al. (2019). To capture the bounded delay and non-FIFO property of the observation channel, an automaton is constructed in a flexible way to model the behavior of the observed plant (i.e., the plant together with the observation channel). The predicted supervisor is established on the observed plant. In Rashidinejad et al. (2018), control channel is assumed to be FIFO, thus the enabled events by the supervisor are supposed to be stored in a sequenced list. Under the existence of control channel delays, controllable events that are enabled by a non-predictive supervisor will only be received by the plant after the passage of $N_c$ ticks. This undesirable effect is addressed by looking ahead of time and predicting the effects of control delays. The networked supervisor is then obtained from the non-predictive supervisor to ensure safety and nonblockingness for the observed plant.

A TDES $P$ is formally represented as a quintuple $P = (A, \Sigma_p, \delta_p, a_0, A_m)$ where $A, \Sigma_p, \delta_p : A \times \Sigma_p \to A$, $a_0 \in A$, and $A_m \subseteq A$ stand for the set of states, set of events, transition function, initial state, and set of the marked states, respectively. It is assumed that the set of states of any TDES contains the event $\text{tick} \in \Sigma_p$. The controllable event set is denoted as $\Sigma_c = \Sigma_p \setminus \Sigma_{nc}$, all the plant events are assumed to be observable, $\sigma_o$ denotes the relabelling that means receiving $\sigma \in \Sigma$, and the enabled event set $\Sigma_e = \{ \sigma | \sigma \in \Sigma_c \setminus \{ \text{tick} \} \}$. The events executed in the plant are transmitted through the observation channel and observed by the supervisor after the passage of $N_o$ ticks; the control commands issued by the supervisor stay in the control channel for $N_c$ ticks before being received by the plant.

Given a plant with an active event set $(\Sigma = \Sigma_p \setminus \{ \text{tick} \})$ is called the set of active events), the objective is to find a supervisor in the networked control platform, called networked supervisor, $NS = (V, \Sigma_{NS}, \delta_{NS}, v_0, V_m)$, where $V$ denotes the set of states, $\Sigma_{NS}$ denotes the set of events, $v_0 \in V$ denotes the initial state, $\delta_{NS} : V \times \Sigma_{NS} \to V$ denotes the transition function, and $V_m$ denotes the set of marked states. Finally, it is necessary to adapt the definitions of controllability, safety, and observability to fit the proposed delay settings.

In Lin (2014), a method is proposed to obtain state estimates using networked observers. When the communication delays and losses are taken into consideration, state estimates become much more complicated. Two assumptions are made (Lin 2014), the first assumption is that the communication delays are bounded by $N$ in the following sense. After the occurrence of an event, the supervisor will know that occurrence of the event before no more than additional $N$ occurrences of events. The second assumption is that the supervisor will observe the occurrences of events in the order that they occurred. In other words, the communication channel satisfies the FIFO property. The second assumption is valid only if there is one communication channel from the plant to the supervisor. The new definition of networked supervisory control. In Lin et al. (2019), the author extends the state estimation methods into a complex system where multiple communications channels exist and each
channel needs to interact with a centralized supervisor. Besides, the FIFO property holds only for each channel, and communication delays may be different for different channels.

In Rashidinejad et al. (2018), since sometimes observation channels are considered not to have the FIFO characteristic, events may not be necessarily observed in the same order as they occur; it presents a method to synthesize a networked supervisor handling delays in both observation channel and control channel.

It is worth mentioning that communication is supposed to be FIFO in [65] and [66]. However, in reality, many communication networks allow events to overtake each other. In Zhu et al. (2019) the authors present a new framework that transforms a networked control problem with $OC$ and $CC$ delays into a standard Ramadge-Wonham supervisory control problem. A schematic diagram for the centralized setup is shown in Fig. 14, which can be extended to a system of an arbitrary number of components and local supervisors.

Such a model transformation method makes it possible to apply existing supervisory control methods such as decentralized control, modular control and hierarchical control to networked control problems. It is possible to apply the approach for model transformation based on existing supervisory control framework such as decentralized control, modular control and hierarchical control to networked control problems. Besides the four network components shown in Fig. 3, i.e., the plant $G$, the supervisor $S$, the observation channel $OC$ and the control channel $CC$, there is one extra component, the Command Execution Module (CE), which is used to transduce each control pattern $\gamma \in \Gamma$ into individual events so that the channel $CC$ model can interact with the plant $G$. We will explain the details shortly.

The networked closed-loop system operates in the following way. Whenever the plant $G$ executes an observable event $\sigma \in \Sigma_o$, it sends a message $m_\sigma$, indicating the occurrence of $\sigma$ in the plant, over the observation channel; the event of sending the
message $m_\sigma$ is denoted by $\sigma^{\text{in}} \in \Sigma^{\text{in}}_o$, where $\Sigma^{\text{in}}_o$ is a copy of $\Sigma_o$ with superscript “in”, i.e., $\sigma^{\text{in}} \in \Sigma^{\text{in}}_o \iff \sigma \in \Sigma_o$. It is required that $\Sigma^{\text{in}}_o \cap \Sigma_o = \emptyset$. The event of receiving the message $m_\sigma$ by the supervisor is denoted by $\sigma^{\text{out}} \in \Sigma^{\text{out}}_o$, where $\Sigma^{\text{out}}_o$ is a copy of $\Sigma_o$ with superscript “out”, and it is required that $\Sigma^{\text{out}}_o \cap \Sigma_o = \emptyset$. $\sigma^{\text{out}}$ can occur only if $\sigma^{\text{in}}$ has already occurred. When $\sigma^{\text{in}}$ occurs, but $\sigma^{\text{out}}$ never occurs, it is assumed that the message $m_\sigma$ is lost in transmission. In this case we use $\sigma^{\text{loss}} \in \Sigma^{\text{loss}}_o$ to denote message dropout. It is required that $\Sigma^{\text{loss}}_o \cap \Sigma_o = \emptyset$. In addition, to avoid the situation where each observable event may get lost, leading to no solution, we assume that only events in $\Sigma^{\text{ol}}_o \subseteq \Sigma_o$ may get lost. Each message $m_\sigma$ is characterised by the tuple $(\sigma^{\text{in}}, \sigma^{\text{out}}, \sigma^{\text{loss}})$. The observation channel may have either a finite or an infinite capacity. After the supervisor $S$ receives an observation $\sigma^{\text{out}}$ from the observation channel, it sends a control command message $m_\gamma$ over the control channel, denoted by $\gamma^{\text{in}} \in \Gamma^{\text{in}}$, where $\Gamma^{\text{in}}$ is a copy of $\Gamma$ with superscript “in” such that $\Gamma^{\text{in}} \cap \Gamma = \emptyset$. Considering that $\Sigma_{uc}$ is always allowed by the supervisor $S$, and execution of an uncontrollable event will be done autonomously by the plant $G$, thus, never be delayed in the CC, we assume that $\Gamma \subseteq 2^{\Sigma_{uc}}$, that is, a control command only decides whether a controllable event should be disabled, denoted as the event not being included in the control pattern. The event of receiving the message $m_\gamma$ by the plant $G$ is denoted by $\gamma^{\text{out}} \in \Gamma^{\text{out}}$, where $\Gamma^{\text{out}}$ is a copy of $\Gamma$ with superscript “out” such that $\Gamma^{\text{out}} \cap \Gamma = \emptyset$. The control channel can have a finite or an infinite capacity and may also experience loss of messages. In the case that the message $m_\gamma$ gets lost in the CC, the event $\gamma^{\text{loss}} \in \Gamma^{\text{loss}}$ will be used to denote the message dropout. We assume that $\Gamma^{\text{loss}} \cap \Gamma = \emptyset$. In addition, to avoid the situation where each control message may get lost, leading to no solution, we assume that only control patterns in $\Gamma_l \subseteq \Gamma$ may get lost. The $OC$ and $CC$ channel delays are respectively upper bounded by $num^o \in \mathbb{N}$ and $num^c \in \mathbb{N}$, which are interpreted as the number of event firings in the system.

For each $\sigma \in \Sigma_o$, each $0 \leq i \leq num^o$ and each $k > 0$, let $mls = ((m_\sigma i), k)$ denote the multi-set which consists of $k$ copies of $(m_\sigma i)$, where $i$ is the time-to-leave tuple in the communication channel. Then, the content of the observation channel can be represented by
mlss ⊆ \{(m_σ, i, k) | σ ∈ \Sigma_o, 0 ≤ i ≤ num^o, k > 0\} with each multi-set in mlss having distinct \((m_σ, i)\). Let \(S_{obs}\) denote the set of all possible contents in the OC, where each message \(m_σ\) is associated with a timer value \(i\), which is upper bounded by \(num^o\). When the timer value \(i\) reaches 0, then the message \(m_σ\) must be popped out of the OC. Nevertheless, the message \(m_σ\) may be popped out before \(i\) ticks down to 0, representing that the channel delay for \(m_σ\) can be any value between 0 and \(num^o\). Two assumptions are made below:

1. The closed-loop system is asynchronous, i.e., no more than one event can fire at each time.
2. The firing of each event in \(\Sigma^{out} \cup \Sigma^{loss}\) does not trigger relevant timers to count down.
3. The sending, receiving and loss of control commands do not trigger relevant timers to count down

A non-FIFO OC is modelled as a nondeterministic finite automaton 
\[ G_{OC} = (Q^{obs} := \{q^{mlss} \mid mlss \in S_{obs}\}, S_{obs} := \Sigma_i \cup S^{out} \cup \Sigma^{loss} \cup \Sigma_{uo}, q_0 := q(1), \delta^{obs} \), where the transition relation \(\delta^{obs} \subseteq Q^{obs} \times S_{obs} \times Q^{obs}\) is defined as follows, where \(m(mlss)\) denotes the minimum of the time-to-leaves for the multi-sets in mlss:

1. For each \(mlss \in S_{obs}\) and each \(σ \in \Sigma_o\), we define

   a) \((q^{mlss}, σ^{in}, q^{mlss'}) \in δ^{obs}\), where \(mlss' \in S_{obs}\) is obtained from \(mlss \in S_{obs}\) by: i) replacing each \((m_σ, i, k)\) in \(mlss\) with \((m_σ, i - 1, k)\), where \(σ' \in \Sigma_o\), and ii) adding \((m_σ, num^o, 1)\), if \(m(mlss) \geq 1\) or \(mlss = \{\}\).

   b) \((q^{mlss}, σ^{out}, q^{mlss'}) \in δ^{obs}\), where \(mlss' \in S_{obs}\) is obtained from \(mlss \in S_{obs}\) by replacing some \((m_σ, i, k)\) with \((m_σ, i, k - 1)\), if \(k > 1\), or by removing some \((m_σ, i, 1)\).

   c) \((q^{mlss}, σ^{loss}, q^{mlss'}) \in δ^{obs}\), where \(σ \in \Sigma_{uo}\) and \(mlss'\) is obtained from \(mlss\) by replacing some \((m_σ, i, k)\) in \(mlss\) with \((m_σ, i, k - 1)\), if \(k > 1\), or by removing some \((m_σ, i, 1)\) in \(mlss\). We require \(i \geq 1\) for the time-to-leave item.

2. For each \(mlss \in S_{obs}\) and each \(σ \in \Sigma_{uo}\), we define \((q^{mlss}, σ, q^{mlss'}) \in δ^{obs}\), where \(mlss' \in S_{obs}\) is obtained from \(mlss\) by replacing each \((m_σ', i, k)\) in \(mlss\) with \((m_σ', i - 1, k)\), where \(σ' \in \Sigma_o\), if \(m(mlss) \geq 1\) or \(mlss = \{\}\).

Intuitively, Rule 1. a) says that the event \(σ^{in}\) constitutes a time step and will add a message \(m_σ\) coupled with the time-to-leave \(num^o\). Rule 1. b) says that the event \(σ^{out}\) removes some message \(m_σ\) from the observation channel. Rule 1. c) says that any message \(m_σ\) with time-to-leave at least 1 can get lost\(^7\), if \(σ \in \Sigma_{uo}\). Rule 2) states that any unobservable event \(σ \in \Sigma_{uo}\) also constitutes a time step. We here emphasize that each plant event constitutes one time step and any other event does not.

A non-FIFO CC can be treated in a similar way.

---

\(^5\) In the definition for subset inclusion between collections of multi-sets, we shall treat each multi-set as an element. Thus, \((((m_σ, 1), 3)) \not\subseteq (((m_σ, 1), 3), ((m_σ, 3), 2)))\), while in our convention we write \((((m_σ, 1), 1)) \not\subseteq (((m_σ, 1), 3), ((m_σ, 3), 2)))\).

\(^6\) If \(mlss = \{\}\), then clearly \(mlss' = \{\}\).

\(^7\) When the time-to-leave is zero, the corresponding message must leave the queue within the current time step and cannot get lost.
1. For each $mlss \in S_{com}$ and each $r \in \Gamma$, we define
   
   (a) $(q^{\text{in}}, \gamma^\text{in}, q'^{\text{in}}_r) \in \delta^{\text{com}}$, where $mlss' \in S_{com}$ is obtained from $mlss \in S_{com}$ by replacing $((m_r, \text{num}_r), k)$ in $mlss$ with $((m_r, \text{num}_r), k + 1)$.
   
   (b) $(q^{\text{loss}}, \gamma^\text{loss}, q'^{\text{loss}}_r) \in \delta^{\text{com}}$, where $mlss' \in S_{com}$ is obtained from $mlss \in S_{com}$ by replacing some $((m_r, i), k)$ with $((m_r, i), k - 1)$.
   
   (c) $(q^{\text{out}}, \gamma^\text{out}, q'^{\text{out}}_r) \in \delta^{\text{com}}$, where $mlss' \in S_{com}$ is obtained from $mlss$ by replacing each $((m_r, i), k)$ in $mlss$ with $((m_r, i), k - 1)$. We require $i \geq 1$ for the time-to-leave item.

2. For each $mlss \in S_{com}$ and each $\sigma \in \Sigma_{\text{in}} \cup \Sigma_{\text{to}}$, we define $(q^{\text{mlss}}, \sigma, q'^{\text{mlss}}_r) \in \delta^{\text{com}}$, where $mlss' \in S_{com}$ is obtained from $mlss$ by replacing each $((m_r, i), k)$ in $mlss$ with $((m_r, i - 1), k)$, if $m(mlss) \geq 1$ or

Intuitively, Rule 1. a) says that the sending of a control pattern $\gamma^\text{in}$ adds a message $m_r$ coupled with the time-to-leave $\text{num}_r$. Rule 1. b) says that the event $\gamma^\text{inout}$ removes some message $m_r$ from the control channel. Rule 1.c) says that any message $m_r$ with time-to-leave at least 1 can get lost. Rule 2) states that any unobservable event $\sigma \in \Sigma_{\text{uo}}$ and any event $\sigma^{\text{in}} \in \Sigma^{\text{in}}$ constitutes one time step, which updates the time-to-leave item of each message in $CC$.

The output of the $CC$ model is a control pattern $\gamma^\text{out} \in \Gamma^{\text{out}}$, which cannot be recognized by the plant $G$, whose alphabet is $\Sigma^{\text{in}} \cup \Sigma_{\text{uo}}$. To link up these two models, we need to create an interface called the command execution automaton $G_{CE}$, which maps each control pattern $\gamma^\text{out}$ to a set of events in $\Sigma^{\text{in}} \cup \Sigma_{\text{uo}}$. Let $G_{CE} = (Q_{CE}, \Sigma_{CE}, \delta_{CE}, q^{CE}_0)$, where $Q^{CE} = \{ q^{\gamma}_r | \gamma \in \Gamma \} \cup \{ q^{\text{wait}} \}$, $\Sigma_{CE} = \Gamma^{\text{out}} \cup \Sigma^{\text{in}} \cup \Sigma_{\text{uo}}$, $q^{CE}_0 = q^{\text{wait}}$, $\delta_{CE} : Q^{CE} \times \Sigma_{CE} \rightarrow Q^{CE}$ is defined as follows.

1. for any $\sigma \in \Sigma_{\text{uc}} \cap \Sigma_{\text{uo}}$, $\delta_{CE}(q^{\text{wait}}, \sigma) = q^{\text{wait}}$.
2. for any $\sigma \in \Sigma_{\text{uc}} \cap \Sigma_{\text{uo}}$, $\delta_{CE}(q^{\text{wait}}, \gamma^\text{in}) = q^{\text{wait}}$.
3. for any $\gamma \in \Gamma$, $\delta_{CE}(q^{\text{wait}}, \gamma^\text{out}) = q^\gamma$.
4. for any $\gamma, \gamma' \in \Gamma$, $\delta_{CE}(q^\gamma, \gamma^\text{out}) = q^{\gamma'}$.
5. for any $q'$, if $\sigma \in \Sigma_{\text{uo}} \cap (\gamma \cup \Sigma_{\text{uc}})$, $\delta_{CE}(q^\gamma, \sigma^\text{in}) = q^{\text{wait}}$.
6. for any $q'$, if $\sigma \in \Sigma_{\text{uo}} \cap (\gamma \cup \Sigma_{\text{uc}})$, $\delta_{CE}(q^\gamma, \sigma) = q'$.
7. and no other transitions are defined.

In the case that the original plant is $G = (Q, \Sigma, \delta, q_0, Q_{\text{in}})$, we replace each $\sigma \in \Sigma_{\text{uc}}$ with $\sigma^\text{in} \in \Sigma_{\text{in}}$ and create a new plant model $G^{\text{mod}} = (Q, \Sigma_{\text{mod}} = \Sigma_{\text{uc}} \cup \Sigma_{\text{uo}}, \delta_{\text{mod}}, q_0, Q_{\text{in}})$, where for any $\sigma \in \Sigma_{\text{uo}}$, $\delta_{\text{mod}}(q, \sigma) = q'$ iff $\delta(q, \sigma) = q'$, and for any $\sigma \in \Sigma_{\text{uc}}$, $\delta_{\text{mod}}(q, \sigma^\text{in}) = q'$ iff $\delta(q, \sigma) = q'$. We now treat $G^{\text{mod}}$ as the plant, and rename it as $G$ in accordance with Fig. 14. Thus, $G$ is over $\Sigma_{\text{uo}} \cup \Sigma_{\text{in}}$ (after relabelling). Let $\mathcal{P} := G || G_{OC} || G_{CC} || G_{CE}$ be the new networked system plant, where the alphabet is $\Sigma^{\mathcal{P}} := \Sigma_{\text{uo}} \cup \Sigma_{\text{in}} \cup \Sigma_{\text{out}} \cup \Sigma_{\text{loss}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \cup \Gamma_{\text{loss}}$, the controllable alphabet is $\Sigma^{\mathcal{P}}_c := \Gamma_{\text{in}}$, and the observable alphabet to the supervisor $S$ is $\Sigma^{\mathcal{P}}_o := \Sigma_{\text{out}} \cup \Gamma_{\text{in}}$. We have the following networked control problem:

**Problem 3** Given the networked plant $\mathcal{P}$ and a specification $E \subseteq L_m(G)$, design a supervisor $S$ over $\Sigma^{\mathcal{P}}$ such that

---

8 If such a k does not exist, we treat $k = 0$.
9 If $mlss = \{ \}$, then clearly $mlss' = \{ \}$.
\[
R(P_{\Sigma_{\mathcal{P}}} \cdot (\Sigma_{\mathcal{P}} \cup \Sigma_{\mathcal{S}i})) \subseteq E,
\]
where \( R \) is a relabelling operator that relabels each \( \sigma_{in} \in \Sigma_{\mathcal{P}} \) with \( \sigma_{o} \in \Sigma_{\mathcal{S}} \);
- \( P\parallel \mathcal{S} \) is nonblocking;
- \( P\parallel \mathcal{S} \) is state-controllable w.r.t. \( P \) (Su et al. 2010a; Su et al. 2010b);
- \( P\parallel \mathcal{S} \) is state-observable w.r.t. \( P \) and \( P_{o} \) (Su et al. 2010a; Su et al. 2010b), where \( P_{o} : (\Sigma_{\mathcal{P}})^{*} \rightarrow (\Sigma_{\mathcal{P}}_{o})^{*} \) is the natural projection.

The problem can be solved by using automaton-based synthesis methods, e.g., Su et al. (2010a), Su et al. (2010b), and Su et al. (2011), which allows the plant \( P \) to be a nondeterministic finite-state automaton, and the final synthesized supervisor \( S \) to be deterministic. The synthesis tool SuSyNA\(^{10}\) can solve supervisor synthesis problem for nondeterministic plants. If FIFO channels are considered (Zhu et al. 2019), the non-determinism caused by bounded channel delays can be subsumed by partial-observation. In this case, the tool of TCT (Feng and Wonham 2006) can be used.

In Rashidinejad et al. (2018), the authors consider the setup where the (lossless) observation channel is non-FIFO while the (lossless) control channel is FIFO; both the channels are allowed to be of infinite capacities. The paper studies the networked supervisor synthesis problem for timed discrete-event systems; thus, the elapse of time is measured by the number of occurrences of ticks. Both the observation channel and the control channel are assumed to have a fixed communication delay, which may not be practical. Since activity loops are prohibited, the two channels are effectively reduced to be of bounded capacities. To model the asynchronous interaction between the plant and the supervisor, the asynchronous product of the plant and the supervisor is used, which is equivalent to the standard synchronous product of the plant, the supervisor, and two channel models. To deal with observation delays and disorderings, an automaton is proposed which models the behavior of the observed plant, i.e., the plant together with the observation channels. On the basis of this observed plant, a nonpredictive supervisor is synthesized that provides safety and nonblockingness for the observed plant, by slightly adapting the Bertil-Wonham framework of supervisor synthesis for timed discrete-event systems. To deal with control delays, the nonpredictive supervisor achieved for the observed plant is transformed into a networked supervisor that enables the events beforehand. It is not known or discussed how such a two-step approach can deal with setups with lossy channels, non-FIFO control channel, channels with bounded delays in a flexible manner. The assumption that all the plant events are observable may also be difficult to relax. The asynchronous product operation defined in Rashidinejad et al. (2018) is also presented in Rashidinejad et al. (2019), which considers the fact that enablement, execution, and observation of an event do not occur simultaneously but with some delay. We remark that an observation message in Rashidinejad et al. (2019) encodes an observable event.

4 Discussions of existing challenges

Currently, there are two main approaches to modelling channel delays:

\(^{10}\) Synthesis tool can be found at https://www.ntu.edu.sg/home/rsu/Downloads.htm.
F1) An implicit channel delay model: in this framework, with Lin (2014) being the representative work, the impact of channel delays on observability and controllability of a closed-loop system is explicitly envisioned, as captured by properly defined concepts of network observability and network controllability, without providing a detailed delay process model. Liu et al. (2019) proposed an online approach for solving the safety supervisory control problem of networked DES with control delays.

F2) An explicit channel delay model: in this framework, with Zhu et al. (2019) being the representative work, a detailed delay process model is explicitly given, upon which its impact on observability and controllability becomes part of the system analysis and control task, and is not explicitly embedded in those definitions. In Rashidinejad et al. (2018) and Rashidinejad et al. (2019), a new framework for centralized networked supervisory control of TDES is proposed. By defining an automaton that models the behaviour of the plant plus the observation channels, a supervisor is synthesized by adapting conventional synthesis methods. The obtained supervisor provides safety, controllability, observability and nonblocking-ness for the observed plant.

To illustrate the difference between these two approaches, let $D$ denotes the ($OC$ and/or $CC$) delay process, $\mathcal{C}(G, S, D)$ be the system controllability of $(G, S)$ under the influence of $D$, and $\mathcal{O}(G, S, D)$ be the system observability of $(G, S)$ under the influence of $D$. The key research focus of networked control is to understand and precisely describe the following implications:

$$D \Rightarrow \mathcal{C}(G, S, D) \quad (5)$$

$$D \Rightarrow \mathcal{O}(G, S, D) \quad (6)$$

In F1, because $D$ is not precisely modelled, the networked supervisor becomes too complicated to specify the controllability and observability $\mathcal{C}(G, S, D)$ and $\mathcal{O}(G, S, D)$. The catch is that it is unclear whether there is a specific physically realizable delay process $D$ that make $\mathcal{C}(G, S, D)$ and $\mathcal{O}(G, S, D)$ physically feasible. In addition, the network controllability and observability concepts are typically very complicated and hard to follow. In F2, by precisely modeling $D$, the impact of $D$ on the system can be precisely modelled as $G||D$, which is then treated as a new plant. The concepts of network controllability and observability become $\mathcal{C}(G||D, S)$ and $\mathcal{O}(G||D, S)$, which are simply the standard concepts of controllability and observability in the classical supervisory control theory without explicitly mentioning delays. In other words, the actual impact of $D$ becomes part of the plant behaviours. Thus, in principle, all existing synthesis methods such as centralized control, modular control, decentralized control, hierarchical control, and state-based control may be applied. The challenges for the transformation based framework is how to combine control pattern and what information shall supervisor send and how to use the received information from other supervisors.

Although channels may be either FIFO or non-FIFO in the literature, all existing networked control frameworks assume a target system $(G, S, D)$ to be asynchronous, which may not be applicable in reality, as communication channels typically operate in a concurrent manner, i.e., the message input and output of each single channel typically take place concurrently. It is unclear how concurrency can be handled in F1. But it could be handled in F2. For example, by considering an elaborated channel delay model based on the one proposed in Zhu et al. (2019), where events in $\Sigma^{in}_o$ and $\Sigma^{out}_o$ may take place either synchronously or asynchronously in the $OC$,
and so do control messages in $\Gamma^{in}$ and $\Gamma^{out}$ in the CC. This essentially calls for a concurrent networked supervisory control framework, which shall match reality better.

Computational complexity is always one major concern for supervisory control theory, which seems an even more daunting challenge for F1. How to efficiently determine the existence of a networked supervisor and, in case it exists, how to efficiently compute it are important problems to be solved. The state explosion problem due to channels and the partial observation lead to doubly exponential complexity synthesis algorithms. It is of interest to see whether we could borrow ideas about minimal communication proposed in Laurie Ricker and Rudie (1999) and Ricker (2008) to handle both observation messages and control messages. Minimal communication is also important to enhance attack-resilience of networked systems, which shall continue to be one important research direction. Finally, it could be the time to consider a new supervisory control architecture, especially the supervisory control map, which might be more robust to channel delays than the standard Ramadge-Wonham supervisory control architecture.

Acknowledgements We would like to thank Prof. Lin Feng for providing valuable suggestions in our preparation of this work.

Declarations

Conflict of interests The authors declare that they have no affiliations with or involvement in any organization or entity with any financial interest, or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

Alves MVS, Basilio JC (2019) State estimation and detectability of networked discrete event systems with multichannel communication networks. In: 2019 American control conference (ACC), IEEE, pp 5602–5607
Alves MVS, Carvalho LK, Basilio JC (2017) Supervisory control of timed networked discrete event systems. Conference on Decision and Control 56:4859–4865
Alves MVS, da Cunha Antonio EC, Carvalho LK, Moreira MV, Basilio JC (2019) Robust supervisory control of discrete event systems against intermittent loss of observations. Int J Control, pp 1–13
Baheti R, Gill H (2011) Cyber-physical systems. The Impact of Control Technology 12(1):161–166
Balemi S (1994) Input/output discrete event processes and communication delays. Discrete Event Dynamic Systems 4(1):41–85
Balemi S, Brunner UA (1992) Supervision of discrete event systems with communication delays. In: 1992 American control conference, IEEE, pp 2794–2798
Barrett G, Lafortune S (2000) Decentralized supervisory control with communicating controllers. IEEE Trans Autom Control 45(9):1620–1638
Cardenas AA, Amin S, Sastry S (2008) Secure control: Towards survivable cyber-physical systems. In: 2008 The 28th international conference on distributed computing systems workshops, IEEE, pp 495–500
Cassandras CG, Lafortune S (2009) Introduction to discrete event systems. Springer Science & Business Media
Choi J, Oh S, Horowitz R (2009) Distributed learning and cooperative control for multi-agent systems. Automatica 45(12):2802–2814
Darondeau P, Ricker L (2012) Distributed control of discrete-event systems A first step. In: Transactions on petri nets and other models of concurrency VI, Springer, pp 24–45
Erol K (1996) Hierarchical task network planning: formalization, analysis, and implementation. PhD thesis
Erol K, Hendler JA, Nau DS (1994) Umcp: A sound and complete procedure for hierarchical task-network planning. In: Aips, vol 94, pp 249–254
Feng L, Wonham WM (2006) Tct: A computation tool for supervisory control synthesis. In: 2006 8Th international workshop on discrete event systems, IEEE, pp 388–389
Feng L, Wonham WM (2008) Supervisory control architecture for discrete-event systems. IEEE Trans Autom Control 53(6):1449–1461
Gaubert S (1995) Performance evaluation of (max,+) automata. IEEE Trans Autom Control 40(12):2014–2025
Hiraishi K (2009) On solvability of a decentralized supervisory control problem with communication. IEEE Trans Autom Control 54(3):468–480
Hou Y, Wang W, Zang Y, Lin F, Yu M, Gong C (2019) Relative network observability and its relation with network observability. IEEE Transactions on Automatic Control
Hu S, Yue D (2012) Event-based h filtering for networked system with communication delay. Signal Process 92(9):2029–2039
Kalyon G, Gall TL, Marchand Hervé, Massart T (2011) Synthesis of communicating controllers for distributed systems. In: 2011 50Th IEEE conference on decision and control and european control conference, IEEE, pp 1803–1810
Kalyon G, Gall TL, Marchand H, Massart T (2013) Symbolic supervisory control of distributed systems with communications. IEEE Trans Autom Control 59(2):396–408
Kim D-S, Lee YS, Kwon WH, Park HS (2003) Maximum allowable delay bounds of networked control systems. Control Eng Pract 11(11):1301–1313
Komenda J, Lin F (2016) Modular supervisory control of networked discrete-event systems. In: 2016 13Th international workshop on discrete event systems (WODES), IEEE, pp 85–90
Komenda J, Masopust T, van Schuppen JH (2015) On a distributed computation of supervisors in modular supervisory control. In: 2015 International conference on complex systems engineering (ICCSE), IEEE, pp 1–6
Komenda J, Masopust T, van Schuppen JH (2015) Coordination control of discrete-event systems revisited. Discrete Event Dynamic Systems 25 (1-2):65–94
Koutsoukos XD, Antsaklis PJ, Stiver JA, Lemmon MD (2000) Supervisory control of hybrid systems. Proc IEEE 88(7):1026–1049
Laurie Ricker S, Rudie K (1999) Incorporating communication and knowledge into decentralized discrete-event systems. In: Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304), vol 2, IEEE, pp 1326–1332
Lewis FL, Zhang H, Hengster-Movric K, Das A (2013) Cooperative control of multi-agent systems: optimal and adaptive design approaches. Springer Science & Business Media
Lin F (2014) Control of networked discrete event systems: dealing with communication delays and losses. SIAM J Control Optim 52(2):1276–1298
Lin F (2020) Modeling and control of networked discrete-event systems. Wiley Encyclopedia of Electrical and Electronics Engineering, pp 1–27
Lin F, Murray Wonham W (1988) On observability of discrete-event systems. Inf Sci 44(3):173–198
Lin F, Wang W, Han L, Shen B (2019) State estimation of multichannel networked discrete event systems. IEEE Trans Control of Newt Syst 7 (1):53–63
Liu Z, Yin X, Shu S, Li S (2019) Online supervisory control of networked discrete-event systems with control delays. In: 2019 IEEE 58Th conference on decision and control (CDC), IEEE, pp 6706–6711
Mo Y, Kim TH-J, Brancik K, Dickinson D, Lee H, Perrig A, Sinopoli B (2011) Cyber–physical security of a smart grid infrastructure. Proc IEEE 100(1):195–209
Park S-J, Cho K-H (2006) Delay-robust supervisory control of discrete-event systems with bounded communication delays. IEEE Trans Autom Control 51(5):911–915
Park S-J, Cho K-H (2007) Supervisory control of discrete event systems with communication delays and partial observations. Systems & Control Lett 56 (2):106–112
Park S-J, Cho K-H (2007) Decentralized supervisory control of discrete event systems with communication delays based on conjunctive and permissive decision structures. Automatica 43(4):738–743
Ramadge PJ, Murray Wonham W (1987) Supervisory control of a class of discrete event processes. SIAM J Control Optim 25(1):206–230
Rashidinejad A, Reniers M, Fabian M (2019) Supervisory control of discrete-event systems in an asynchronous setting. In: 2019 IEEE 15Th international conference on automation science and engineering (CASE), IEEE, pp 494–501
Rashidinejad A, Reniers M, Feng L (2018) Supervisory control of timed discrete-event systems subject to communication delays and non-fifo observations. IFAC-PapersOnLine 51(7):456–463
Ricker SL (2008) Asymptotic minimal communication for decentralized discrete-event control. In: 2008 9Th international workshop on discrete event systems, IEEE, pp 486–491
Ricker L (2013) An overview of synchronous communication for control of decentralized discrete-event systems. In: Control of discrete-event systems, Springer, pp 127–146
Su R, van Schuppen JH, Roorda JE (2010a) Model abstraction of nondeterministic finite-state automata in supervisor synthesis. IEEE Trans Autom Control 55(11):2527–2541
Su R, Van Schuppen JH, Rooda JE (2010b) Aggregative synthesis of distributed supervisors based on automaton abstraction. IEEE Trans Autom Control 55(7):1627–1640
Su R, Van Schuppen JH, Rooda JE (2011) The synthesis of time optimal supervisors by using heaps-of-pieces. IEEE Trans Autom Control 57 (1):105–118
Rudie K, Wonham WM (1991) Act locally Think globally Decentralized supervisory control. In: 1991 American control conference, IEEE, pp 898–903
Sadid WH, Ricker L, Hashtrudi-Zad S (2015) Robustness of synchronous communication protocols with delay for decentralized discrete-event control. Discrete Event Dynamic Systems 25(1-2):159–176
Sasi Y, Lin F (2018) Detectability of networked discrete event systems. Discrete Event Dynamic Systems 28(3):449–470
Shu S, Lin F (2014a) Supervisor synthesis for networked discrete event systems with communication delays. IEEE Trans Autom Control 60(8):2183–2188
Shu S, Lin F (2014b) Decentralized control of networked discrete event systems with communication delays. Automatica 50(8):2108–2112
Shu S, Lin F (2016) Deterministic networked control of discrete event systems with nondeterministic communication delays. IEEE Trans Autom Control 62(1):190–205
Shu S, Lin F (2016) Predictive networked control of discrete event systems. IEEE Trans Autom Control 62(9):4698–4705
Sridhar S, Hahn A, Govindarasu M (2011) Cyber–physical system security for the electric power grid. Proc IEEE 100(1):210–224
Triapakis S (2004) Decentralized control of discrete-event systems with bounded or unbounded delay communication. IEEE Trans Autom Control 49 (9):1489–1501
Walsh GC, Ye H, Bushnell LG (2002) Stability analysis of networked control systems. IEEE Trans Control Syst Technol 10(3):438–446
Wang J (2012) Timed Petri nets: Theory and application, volume 9 Springer Science & Business Media
Wang W, Lafortune S, Lin F (2008) Minimization of communication of event occurrences in acyclic discrete event systems. IEEE Trans Autom Control 53(9):2197–2202
Wang X, Lemmon MD (2010) Event-triggering in distributed control systems. IEEE Trans Autom Control 56(3):586–601
Wang F, Shu S, Lin F (2016) Robust networked control of discrete event systems. IEEE Trans Autom Sci Eng 13(4):1528–1540
Wong KC, Wonham WM (2004) On the computation of observers in discrete-event systems. Discrete Event Dynamic Systems 14(1):55–107
Yoo T -S, Lafortune S (2002) A general architecture for decentralized supervisory control of discrete-event systems. Discrete Event Dynamic Systems 12 (3):335–377
Yu W, Wen G, Chen G, Cao J (2017) Distributed cooperative control of multi-agent systems. John Wiley & Sons
Zhang W, Branicky MS, Phillips SM (2001) Stability of networked control systems. IEEE Control Syst Mag 21(1):84–99
Zhang R, Cai K, Gan Y, Wonham WM (2016) Distributed supervisory control of discrete-event systems with communication delay. Discrete Event Dynamic Systems 26(2):263–293
Zhang R, Cai K, Gan Y, Wonham WM (2016) Delay-robustness in distributed control of timed discrete-event systems based on supervisor localisation. Int J Control 89(10):2055–2072
Zhao B, Lin F, Wang C, Zhang X, Polis MP et al (2015) Supervisory control of networked timed discrete event systems and its applications to power distribution networks. IEEE Transactions on Control of Network Systems 4(2):146–158
Zhivoglyadov PV, Middleton RH (2003) Networked control design for linear systems. Automatica 39(4):743–750
Zhou L, Shu S, Lin F (2019) Supervisory control of discrete event systems under nondeterministic observations. In: 2019 18Th european control conference (ECC), IEEE, pp 4192–4197
Zhu Y, Lin L, Ware S, Su R (2019) Supervisor synthesis for networked discrete event systems with communication delays and lossy channels. In: 2019 IEEE 58Th conference on decision and control (CDC), IEEE, pp 6730–6735

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.
Yuting Zhu received the B.S. degree from Southeast University, Jiangsu, China, in 2016. She is currently pursuing the Ph.D. degree with Nanyang Technological University, Singapore. Her research interests include networked control and cyber security of discrete event systems.

Liyong Lin received the B.E. degree and Ph.D. degree in electrical engineering in 2011 and 2016, respectively, both from Nanyang Technological University, where he has also worked as a project officer. From June 2016 to October 2017, he was a postdoctoral fellow at the University of Toronto. Since December 2017, he has been working as a research fellow at the Nanyang Technological University. His main research interests include supervisory control theory, formal methods and machine learning. He previously was an intern in the Data Storage Institute, Singapore, where he worked on single and dual-stage servomechanism of hard disk drives.
Ruochen Tai received the B.E. degree in electrical engineering from the Nanjing University of Science and Technology in 2016, and M.S. degree in automaton from the Shanghai Jiao Tong University in 2019. He is currently pursuing the Ph.D. degree with Nanyang Technological University, Singapore. His current research interests include security issue of cyber-physical systems, multi-robot systems, safe autonomy in cyberphysical-human systems, formal methods, and discrete-event systems.

Rong Su received the Bachelor of Engineering degree from University of Science and Technology of China in 1997, and the Master of Applied Science degree and PhD degree from University of Toronto, in 2000 and 2004, respectively. He was affiliated with University of Waterloo and Technical University of Eindhoven before he joined Nanyang Technological University in 2010. Currently, he is an associate professor in the School of Electrical and Electronic Engineering. Dr. Su’s research interests include multiagent systems, cyber security of discrete-event systems, supervisory control, model-based fault diagnosis, control and optimization in complex networked systems with applications in flexible manufacturing, intelligent transportation, human-robot interface, power management and green buildings. In the aforementioned areas he has more than 220 journal and conference publications, and 5 granted USA/Singapore patents. Dr. Su is a senior member of IEEE, and an associate editor for Automatica, Journal of Discrete Event Dynamic Systems: Theory and Applications, and Journal of Control and Decision. He was the chair of the Technical Committee on Smart Cities in the IEEE Control Systems Society in 2016-2019, and is currently the chair of IEEE Control Systems Chapter, Singapore, and a cochair of IEEE Robotic and Automation Society Technical Committee on Automation in Logistics.