Supervisory Control of Networked Timed Discrete Event Systems with Bandwidth Constraints

Zhaoyu Xiang, Yufeng Chen, *Senior Member, IEEE*, Naiqi Wu, *Fellow, IEEE*, and Zhiwu Li, *Fellow, IEEE*

***Abstract*— In this paper, we investigate the supervisory control of a networked timed discrete event system modeled with a timed automaton. In this system, a plant communicates with its supervisor through multiple observation channels and a single control channel. Any observation channel is imposed a bandwidth constraint such that the number of events transmitted through the channel per time unit should not exceed a given non-negative integer. The control channel is subject to bounded control delays and losses. Our goal is to synthesize a supervisor against control delays and losses such that the bandwidth constraint of any observation channel is satisfied. To this end, the system is first transformed to a region automaton that is a finite representation of the system's semantics. Then, by considering the impact of control delays and losses on system state estimation, the conventional bipartite transition systems are extended to networked timed ones, from which a necessary and sufficient condition for the existence of a desired supervisor is derived. An algorithm is proposed for building such a supervisor if existing.**

***Index Terms*— Networked discrete event system, supervisory control, bandwidth constraint, timed automaton.**

I. Introduction

Event-driven discrete event systems (DESs) have an important role in modeling and optimizing processes in logistics and manufacturing [1]. The supervisory control theory initiated by Ramadge and Wonham [2] is used to synthesize a supervisor ensuring that a DES satisfies a control specification. In this paper, we address an open problem in the supervisory control community, namely the supervisory control problem under bandwidth constraints as reported in [5]. The control specification considered in this problem is bandwidth safety, which limits the number of events transmitted through an observation channel per time unit to not exceed the channel's assigned bandwidth, represented by a non-negative integer. Then a solution to this problem aims to build a supervisor for a networked discrete event system (NDES) to ensure the enforcement of this control specification. Exploring this problem can not only enrich the supervisory control theory of DESs but also expand its application.

To solve the supervisory control problem under bandwidth constraints in NDESs, a key issue is to measure control delays and losses in the control channel. The works in [10]–[12] adopt a timed discrete event system (TDES) introduced by Brandin and Wonham [7] (the B-W framework), where the event $tick$ is used to represent the elapse of one time unit. Then, control delays and losses are gauged by counting the occurrences of $tick$. The timing constraints of the B-W framework are designed by restricting the occurrence of any non-$tick$ event within a specified lower and an upper time bound. However, as stressed in [14], in terms of real-life applications, the B-W framework is less versatile than timed automata [19]–[21], as the latter can model multiple and different timing constraints by introducing invariants (constraining the amount of time that a system is allowed to sojourn in locations) and guards (stipulating the time when a system's location transition can be enabled).

Timed automata are classified into discrete-timed and dense-timed automata [17], where the former consists of integer-valued clock variables whose values can be any non-negative integer in $\mathbb{N} = \{0, 1, 2, \dots\}$ [16], while the latter is composed of real-valued clock variables whose values can be any non-negative real number in $\mathbb{R}_{\geq 0}$ [15]. Since the semantics of a timed automaton is represented by an infinite state transition semantic graph [14], [19], [21], it is sometimes necessary to convert the timed automaton into a region automaton, which is a finite abstraction of the semantics, to solve the problems such as supervisor synthesis [1], verification of detectability [15] and opacity [16], and state estimation [18]. In a region automaton, a special timed event τ is introduced to model the time elapse [15]. For discrete-timed automata, the occurrence of τ represents the elapse of one time unit [16], [17], whereas for dense-timed automata, it indicates a nondeterministic time passage within $\{d \in \mathbb{R}_{\geq 0} | 0 < d < 1\}$ [1], [18]. A TDES can be modeled by a discrete-timed automaton, thereby offering greater versatility than the TDESs represented within the B-W framework. Furthermore, its region automaton is capable of measuring control delays and losses using the timed event τ, akin to the $tick$ event!

The discrete-timed automata introduced in [16], [17] do not consider invariants, restricting their application. In this paper, a TDES is modeled by a discrete-timed automaton taking both guards and invariants into account. The corresponding region

This work was supported in part by the National Key R&D Project of China under Grant 2018YFB1700104, and the Science and Technology Development Fund, MSAR, under Grant 0064/2021/A2. *(Corresponding author: Zhiwu Li.)*

Zhaoyu Xiang, Yufeng Chen, Naiqi Wu, and Zhiwu Li are with the Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China, and Zhiwu Li is also with the School of Electro-Mechanical Engineering, Xidian University, Xian 710071, China (e-mails: 1909853gmi30002@student.must.edu.mo; chyf01@163.com; nqwu@must.edu.mo; zwli@must.edu.mo).

automaton is then constructed. By integrating the TDES and a network consisting of multiple communication channels, a networked timed discrete event system (NTDES) is obtained. The supervisory control problem under bandwidth constraints is formulated for the NTDES. To solve the problem, a state estimation method is introduced for an NTDES, which computes the set of states in which the system may currently be, utilizing the over-approximation technique [6]. Specifically, a state in an NTDES is deemed reachable if and only if it is reached through a string permitted to occur for some control decisions delayed in the control channel. The proposed state estimation method gives more precise estimation results than the open-loop state estimation approach in [4]–[5], [11], where control decisions are not considered, and than the one in [6], where time elapses are modeled inappropriately [10]. By tracking state estimates of an NTDES, the traditional bipartite transition systems (BTSs) [3], which cannot handle control delays and losses, are extended to networked timed bipartite transition systems (NTBTSs), from which a necessary and sufficient condition for the existence of a solution supervisor for the considered problem is derived. An algorithm is presented to decode such a supervisor from the NTBTS, if existing. In summary, this paper makes the following contributions:

1) A supervisor synthesis problem for NTDESs with bandwidth constraints is formulated.
2) An online state estimation approach is proposed for NTDESs.
3) A necessary and sufficient condition for the existence of a supervisor for the formulated problem is derived and an algorithm is proposed to compute such a supervisor, if existing.

The rest of the paper is organized as follows. The notions of discrete-timed automata and their region automata are introduced in Section II. Section III defines a networked timed discrete event system and formulates a corresponding supervisory control problem under bandwidth constraints. An online over-approximation state estimation approach for networked timed discrete event systems is designed in Section IV. A networked timed bipartite transition system is constructed in Section V for the formulated problem. Section VI concludes the paper.

II. Preliminary

A. System Model

Let Σ be a finite set of events. A finite string is a sequence of events defined over Σ. The length of a string s, denoted by $|s|$, is the number of event labels forming s. A string with length zero is called the empty string and is denoted by ε. The notation Σ^*, called the Kleene-closure of Σ, is the set of all finite strings defined over Σ. A subset of Σ^* is said to be a language. For a j-tuple $y = (y_1, y_2, \ldots, y_j)$, the notation $y[j']$ for $j' \in \{1, 2, \ldots, j\}$ is used to denote the j'-th entry of y, i.e., $y[j'] = y_{j'}$.

An integer-valued clock c, called clock for short, is a variable whose value can be any non-negative integer in $\mathbb{N}$. The finite set of clocks is denoted by C. A clock valuation is defied as a function $v : C \to \mathbb{N}$, assigning to a clock $c \in C$ its current value $v(c)$. We write V_C the set of all clock valuations over C. Given a clock valuation $v \in V_C$ and a subset $\kappa \subseteq C$ of clocks, the notation $v_{[\kappa]}$ is a new clock valuation that assigns 0 to all the clocks in κ, while keeping the other clocks unchanged. That is, $v_{[\kappa]}(c) = 0$ if $c \in \kappa$; $v_{[\kappa]}(c) = v(c)$ otherwise. Let us denote by 0_C the initial clock valuation such that $0_C(c) = 0$ for all $c \in C$. Given a natural number $d \in \mathbb{N}$ and a clock valuation $v \in V_C$, let $v + d$ be a clock valuation such that $(v + d)(c) = v(c) + d$ for all $c \in C$ [15], measuring the time elapse d.

An atomic clock constraint is defined in the form of $c \sim d$, for $c \in C$, $d \in \mathbb{N}$, and $\sim \in \{\leq, <, \geq, >, =\}$ [15]. A clock valuation $v \in V_C$ satisfies an atomic clock constraint $c \sim d$, denoted by $v \models c \sim d$, if $v(c) \sim d$. A clock constraint is defined as a conjunction of a finite number of atomic clock constraints. Let $A(C)$ be the set of all clock constraints over C. Given $g \in A(C)$ and $v \in V_C$, v satisfies g, denoted by $v \models g$, if v satisfies all the atomic clock constraints in g [19].

A timed discrete event system (TDES) is modeled by a deterministic discrete-timed automaton (DTA)

$$G = (L, l_0, \Sigma, C, Inv, E),$$

where

- L is the finite set of locations;
- $l_0 \in L$ is the initial location;
- Σ is the finite set of events;
- C is the finite set of clocks;
- $Inv : L \to A(C)$ is the invariant function which assigns a location $l \in L$ an invariant $Inv(l) \in A(C)$ [1], [15];
- $E \subseteq L \times A(C) \times \Sigma \times 2^C \times L$ is the finite set of edges. An edge is of the form $e = (l, g, \sigma, \kappa, l')$, where $l, l' \in L$ are respectively the source and final locations, $g \in A(C)$ is a guard, $\sigma \in \Sigma$ is an event, and $\kappa \in 2^C$ is the set of clocks that will be reset to zero after σ occurs. Given any two different edges $e = (l, g, \sigma, \kappa, l')$ and $e' = (l, g', \sigma, \kappa', l'')$ with the same source location l and event σ, it requires that for any $v \in V_C$, $[v \models g \Rightarrow v \not\models g'] \wedge [v \models g' \Rightarrow v \not\models g]$ [1], [14].

The determinism of G results in a deterministic region automaton, that is, the occurrence of an event in a state in the region automaton leads to a unique state [1]. Next, the synchronous product operator proposed in [14] is introduced to model the joint behavior of timed automata.

Definition 1: (Synchronous product) The synchronous product of two deterministic discrete-timed automata $G_1 = (L_1, l_{1_0}, \Sigma_1, C_1, Inv_1, E_1)$ and $G_2 = (L_2, l_{2_0}, \Sigma_2, C_2, Inv_2, E_2)$ is the deterministic discrete-timed automaton $G_1||G_2 = (L_1 \times L_2, (l_{1_0}, l_{2_0}), \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, Inv_{12}, E_{12})$, where for any $(l, l') \in L_1 \times L_2$, it holds $Inv_{12}(l, l') = Inv_1(l) \wedge Inv_2(l')$, and E_{12} is defined as follows:

1) For any $\sigma \in \Sigma_1 \setminus \Sigma_2$, $(l_1, g, \sigma, \kappa, l_1') \in E_1$ with $\kappa \cap C_2 = \emptyset$, and $l_2 \in L_2$, $((l_1, l_2), g, \sigma, \kappa, (l_1', l_2)) \in E_{12}$;
2) For any $\sigma \in \Sigma_2 \setminus \Sigma_1$, $(l_2, g, \sigma, \kappa, l_2') \in E_2$ with $\kappa \cap C_1 = \emptyset$, and $l_1 \in L_1$, $((l_1, l_2), g, \sigma, \kappa, (l_1, l_2')) \in E_{12}$;
3) For any $\sigma \in \Sigma_1 \cap \Sigma_2$, $(l_1, g_1, \sigma, \kappa_1, l_1') \in E_1$, and $(l_2, g_2, \sigma, \kappa_2, l_2') \in E_2$ with $\kappa_1 \cap C_2 = \kappa_2 \cap C_1$, $((l_1, l_2), g_1 \wedge g_2, \sigma, \kappa_1 \cup \kappa_2, (l_1', l_2')) \in E_{12}$. $\diamond$

Example 1: Consider the TDESs G_1 and G_2 as shown in Fig. 1(a) and (b), respectively. The label $(c_s \geq 1, p_1, \{c_s\})$ in G_1 denotes guard $c_s \geq 1$, event p_1, and the set of clocks $\{c_s\}$ to be reset to zero after p_1 occurs. This label between locations l_0 and l_1 signifies that G_1 can transition from l_0 to l_1, when the guard $c_s \geq 1$ holds. After this transition, event p_1 occurs and the clock c_s is reset to zero. Both invariants of l_0 and l_1 in G_1 equal true and are omitted for simplicity. The label $[c'_s \leq 1]$ next to a location in G_2 denotes that the invariant of that location is $c'_s \leq 1$. The synchronous product of G_1 and G_2 is the TDES $G = G_1||G_2 = (L, l_0, \Sigma, C, Inv, E)$ as shown in Fig. 1(c) with the set of clocks $C = \{c_s, c'_s\}$. □

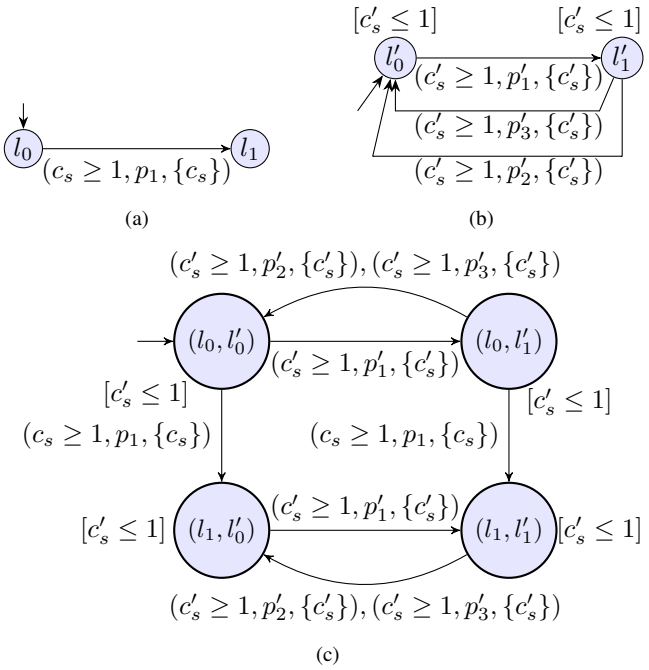

Fig. 1: (a) TDES G_1, (b) TDES G_2, and (c) TDES $G = G_1||G_2$ with the set of clocks $C = \{c_s, c'_s\}$.

B. Region Automata

By transforming the construction method of a region automaton for a dense-timed automaton in [19] to one for a deterministic discrete-timed automaton, this subsection develops a region automaton for a TDES G. We start by converting the notion of clock equivalence for real-valued clocks [21] to integer-valued clocks. Given a clock $c \in C$, let $M(c)$ be the maximal constant that appears in the conditions involving c in guards or invariants in G [1], [19].

Definition 2: Let $G = (L, l_0, \Sigma, C, Inv, E)$ be a TDES. Two clock valuations $v, v' \in V_C$ are said to be clock-equivalent, denoted by $v \cong v'$, if for all $c \in C$, either $v(c) > M(c)$ and $v'(c) > M(c)$, or $v(c) = v'(c)$. ◇

A clock region of G is an equivalence class of clock valuations under $\cong$ [15], [19]. The clock region where any clock valuation assigns to each clock $c \in C$ a value that is larger than $M(c)$ is called the unbounded clock region, denoted by $r_\infty = \{v \in V_C | (\forall c \in C) v(c) > M(c)\}$ [19]. Let us write $V_C/\cong$ as the set of all clock regions of G [19]. Any clock valuation corresponds to a unique clock region [15].

Definition 3: Let $G = (L, l_0, \Sigma, C, Inv, E)$ be a TDES and $v \in V_C$ be a clock valuation. The clock region of v, denoted by $[v]$, is defined as $[v] = \{v' \in V_C | v' \cong v\}$. ◇

Example 2: Consider the TDES G as shown in Fig. 1(c). It holds $M(c) = 1$ for all $c \in C = \{c_s, c'_s\}$. The initial clock valuation 0_C satisfies $0_C(c_s) = 0_C(c'_s) = 0$. Let v_1 be a clock valuation such that $v_1(c_s) = v_1(c'_s) = 2$. The two clock valuations 0_C and v_1 are not clock-equivalent due to $v_1(c_s) = 2 > M(c_s)$ and $0_C(c_s) = 0 < M(c_s)$. Let v_2 be a clock valuation such that $v_2(c_s) = v_2(c'_s) = 3$. We have $v_1 \cong v_2$, thanks to $v_1(c) > M(c)$ and $v_2(c) > M(c)$ for all $c \in C$. The clock region of 0_C, called the initial clock region, is $[0_C]$ that is the set of all clock valuations v such that $v(c) = 0$ for $c \in C$. Note that a clock region is usually represented by a finite set of clock constraints it satisfies, see, e.g., Page 714 in [19]. For instance, $[0_C]$ can be written as $\{(c_s, c'_s) | c_s = 0 \wedge c'_s = 0\}$. The notation $\{(c_s, c'_s) | c_s = 1 \wedge c'_s > 1\}$ denotes the clock region of all clock valuations $v \in V_C$ satisfying $v(c_s) = 1$ and $v(c'_s) > 1$. For G, the unbounded clock region is $r_\infty = \{(c_s, c'_s) | c_s > 1 \wedge c'_s > 1\}$. □

To build a region automaton, we first show how to model the evolution of clock regions. The following notions are introduced. Let τ be the timed event. Given $\kappa \subseteq C$ and $r \in V_C/\cong$, let $r_{[\kappa]} = \{v_{[\kappa]} | v \in r\}$ be the clock region obtained by resetting the clocks in κ within r to zero [19]. By converting the notion of successor regions for a dense-timed automaton [19] to one for a deterministic discrete-timed automaton, we have the following definitions.

Definition 4: Let $r, r' \in V_C/\cong$ be two clock regions and $\kappa \subseteq C$ be a subset of clocks. The clock region r' is the κ-successor clock region of r if $r' = r_{[\kappa]}$. The region r' is the τ-successor clock region of r if either

1) $r = r_\infty$ is the unbounded clock region and $r = r'$, or
2) $r \neq r_\infty$, $r \neq r'$, and for all $v \in r$, there exists $d \in \{1, 2, \dots\}$ such that $v + d \in r'$ and for all $d' \in \{0, 1, \dots, d\}$, $v + d' \in r \cup r'$. ◇

In plain words, a clock region r transitions to its κ-successor clock region without time elapse but through resetting the clocks in κ to zero. The τ-successor clock region of r results from one time unit passage.

Example 3: Consider the clock valuations 0_C and v_1 as developed in Example 2. Their corresponding clock regions are $[0_C] = \{(c_s, c'_s) | c_s = 0 \wedge c'_s = 0\}$ and $[v_1] = \{(c_s, c'_s) | c_s > 1 \wedge c'_s > 1\} = r_\infty$. Let $r = [v_1]$, $r' = [0_C]$, and $\kappa = \{c_s, c'_s\} \subseteq C$. It follows $r' = r_{[\kappa]}$. Hence, r' is the κ-successor clock region of r. By $[v_1] = r_\infty$, its τ-successor clock region is itself. The τ-successor clock region of $[0_C]$ is $\{(c_s, c'_s) | c_s = 1 \wedge c'_s = 1\}$. Note that r_∞ is not the τ-successor clock region of $[0_C]$, as explained in the following. Given a clock valuation $v \in [0_C]$, $v + 1$ is a clock valuation such that $(v + 1)(c) = 1$ for all $c \in C$. Hence, one has $v + 1 \notin [0_C] \cup r_\infty$. By Definition 4, if r_∞ is the τ-successor clock region of $[0_C]$, there must exist $d \geq 2$ such that $v + d \in r_\infty$. Moreover, $v + 1 \in [0_C] \cup r_\infty$ should be satisfied, which is not possible. □

A clock region r satisfies a clock constraint $g \in A(C)$, denoted by $r \models g$, if for all $v \in r$, it holds $v \models g$. Finally, we

present the formal definition of a finite-state region automaton.

Definition 5: Let $G = (L, l_0, \Sigma, C, Inv, E)$ be a TDES. The finite-state region automaton of G is a quadruple $R(G) = (Q, q_0, \Sigma_R, \delta)$, where

- $Q = L \times (V_C/\cong)$ is the finite set of states [15], [19];
- $q_0 = (l_0, [0_C])$ is the initial state;
- $\Sigma_R = \Sigma \cup \{\tau\}$ with $\tau \notin \Sigma$ is the finite set of events;
- $\delta : Q \times \Sigma_R \to Q$ is the partial transition function such that for any $q = (l, r), q' = (l', r') \in Q$, and $\sigma \in \Sigma_R$, $\delta(q, \sigma) = q'$ is defined if either
  1) $\sigma = \tau$, $l = l'$, and r' is the τ-successor clock region of r such that $r, r' \models Inv(l)$, or
  2) $\sigma \neq \tau$, there exists $e = (l, g, \sigma, \kappa, l') \in E$ such that $r \models g$, r' is the κ-successor clock region of r, and $r' \models Inv(l')$. ◇

For any $q, q' \in Q$ and $\sigma \in \Sigma_R$, write $(q, \sigma, q') \in \delta$ if $\delta(q, \sigma) = q'$. The transition function δ can be extended to $\delta : Q \times (\Sigma_R)^* \to Q$ in the usual way [1]. Intuitively, when $R(G)$ enters a state $q = (l, r) \in Q$, it means that G has transitioned to location l with the current clock valuation $v \in r$. For the timed event τ to occur at q, both r and its τ-successor clock region r' must satisfy the invariant $Inv(l)$ of l to ensure that all the clock valuations in r and r' satisfy $Inv(l)$. After τ occurs, $R(G)$ arrives at state $q' = (l, r') \in Q$. This signifies that G stays at l, while updating v to a clock valuation $v' \in r'$ that satisfies $Inv(l)$ to record the elapse of one time unit.

For a non-τ event $\sigma \in \Sigma$ to occur at $q = (l, r)$, there necessarily exists an edge $e = (l, g, \sigma, \kappa, l') \in E$ such that (1) r satisfies the guard g to guarantee that the current clock valuation v satisfies g and (2) the κ-successor clock region r'' of r should satisfy the invariant $Inv(l')$ of l' to ensure that all the clock valuations in r'' not only have reset the clocks in κ to zero, but also satisfy $Inv(l')$. Upon the occurrence of σ, $R(G)$ arrives at state $q' = (l', r'') \in Q$. This indicates that G enters location l', while updating v to a clock valuation $v'' \in r''$ that satisfies $Inv(l')$ to reset all the clocks in κ to zero.

Remark 1: For any clock region $r \in V_C/\cong$, its successor clock region is unique [19]. Hence, the occurrence of timed event τ in any state $q = (l, r)$ of a region automaton $R(G)$ leads to a unique state. Assume that a non-τ event $\sigma \in \Sigma$ occurs at (l, r). For any two different edges $e = (l, g, \sigma, \kappa, l'), e' = (l, g', \sigma, \kappa', l'') \in E$, since G is deterministic, $r \models g$ and $r \models g'$ cannot simultaneously hold. As a result, σ also induces a unique state. Consequently, $R(G)$ is deterministic. ■

Example 4: Consider the TDES G as shown in Fig. 1(c). Its region automaton $R(G)$ is depicted in Fig. 2, where the states are displayed in Table I. To save space, we use simplified notations to denote clock regions. For instance, the notation $[0, 0]$ in state q_0 refers to the initial clock region $[0_C]$ of all clock valuations $v \in V_C$ satisfying $v(c_s) = v(c'_s) = 0$. The notation $[> 1, 1]$ in q_5 represents the clock region of all clock valuations $v \in V_C$ satisfying $v(c_s) > 1$ and $v(c'_s) = 1$. After string τp_1, $R(G)$ enters state q_2, indicating that after 1 time unit, G moves to location (l_1, l'_0), triggering event p_1. □

The language generated by a region automaton $R(G)$ is defined by $\mathcal{L}(R(G)) = \{s \in (\Sigma_R)^* | \delta(q_0, s)!\}$, where ! represents "is defined". Hereafter, we focus on a TDES G whose region automaton $R(G)$ satisfies the following constraints.

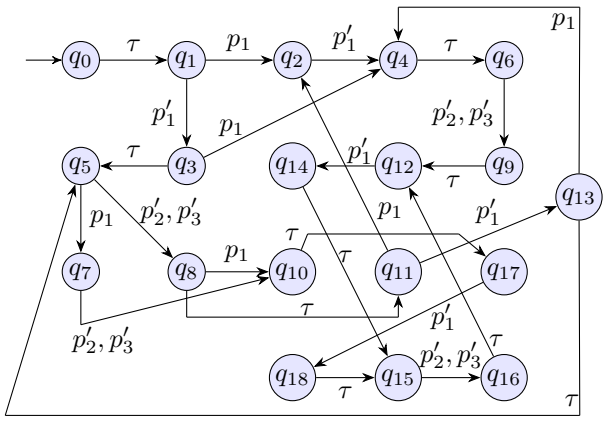

Fig. 2: Region automaton $R(G)$ with $\Sigma_o = \{p_1, p'_1, p'_2, \tau\}$, $\Sigma_{uo} = \{p'_3\}$, $\Sigma_c = \{p_1, p'_3\}$, $\Sigma_{uc} = \{p'_1, p'_2\}$, and $\Sigma_{for} = \{p_1\}$.

TABLE I: States of the region automaton $R(G)$.

$q_0 = ((l_0, l'_0), [0, 0])$	$q_1 = ((l_0, l'_0), [1, 1])$
$q_2 = ((l_1, l'_0), [0, 1])$	$q_3 = ((l_0, l'_1), [1, 0])$
$q_4 = ((l_1, l'_1), [0, 0])$	$q_5 = ((l_0, l'_1), [> 1, 1])$
$q_6 = ((l_1, l'_1), [1, 1])$	$q_7 = ((l_1, l'_1), [0, 1])$
$q_8 = ((l_0, l'_0), [> 1, 0])$	$q_9 = ((l_1, l'_0), [1, 0])$
$q_{10} = ((l_1, l'_0), [0, 0])$	$q_{11} = ((l_0, l'_0), [> 1, 1])$
$q_{12} = ((l_1, l'_0), [> 1, 1])$	$q_{13} = ((l_0, l'_1), [> 1, 0])$
$q_{14} = ((l_1, l'_1), [> 1, 0])$	$q_{15} = ((l_1, l'_1), [> 1, 1])$
$q_{16} = ((l_1, l'_0), [> 1, 0])$	$q_{17} = ((l_1, l'_0), [1, 1])$
$q_{18} = ((l_1, l'_1), [1, 0])$	

Constraint 1: $R(G)$ is Σ-loop free, that is,

$$(\forall q \in Q)(\forall s \in \Sigma^* \setminus \{\varepsilon\})\delta(q, s) \neq q. \tag{1}$$

Constraint 2: For any state $q \in Q$, there exists at least one event $\sigma \in \Sigma_R$ such that $\delta(q, \sigma)!$.

These constraints are commonly adopted in many works, see, e.g., in [7], [11], [19]. Specifically, Constraint 1 excludes an unrealistic situation where $R(G)$ continuously generates non-τ events without time elapsing. Constraint 2 ensures that $R(G)$ will not stop at a state without further event occurrences. The two constraints guarantee the continuous time progress in G, which is known as timelock freedom [15], [19].

C. Supervisory Control of TDESs

We apply the supervisory control of the B-W framework in [7], [8] to a region automaton $R(G) = (Q, q_0, \Sigma_R, \delta)$. The set of non-$\tau$ events are divided into $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$, where "$\dot\cup$" indicates the disjoint union of sets, Σ_c is the set of controllable events that can be disabled to occur by a supervisor, and Σ_{uc} is the set of uncontrollable events that cannot be disabled. The timed event τ can be preempted by forcible events [7]. The set of forcible events is denoted by $\Sigma_{for} \subseteq \Sigma$ [8].

The event set Σ_R is partitioned into $\Sigma_R = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o and Σ_{uo} are the sets of observable and unobservable events, respectively. In particular, let $\tau \in \Sigma_o$ in the sense that its occurrence is usually transparent to a plant and its supervisor [11]. Let $P : (\Sigma_R)^* \to \Sigma_o^*$ be a natural projection, recursively defined by $P(\varepsilon) = \varepsilon$ and for any $s \in (\Sigma_R)^*$ and $\sigma \in \Sigma_R$, $P(s\sigma) = P(s)P(\sigma)$ such that $P(\sigma) = \sigma$ if $\sigma \in \Sigma_o$, otherwise $P(\sigma) = \varepsilon$. Projection P can be extended to languages in the usual way [1]. The set of all control decisions issued by a supervisor is defined by $\Gamma = \{\gamma \in 2^{\Sigma} \times 2^{\Sigma_{for}} | \Sigma_{uc} \subseteq \gamma[1] \wedge \gamma[2] \subseteq \gamma[1] \cap \Sigma_{for}\}$. Specifically, $\gamma[1]$ is the set of events enabled by γ to occur, while $\gamma[2]$ is the set of forcible events enabled and forced by γ to occur to preempt the timed event τ. A supervisor is defined as a function $S : P(\mathcal{L}(R(G))) \to \Gamma$. Let us denote by S/G the closed-loop system. The language generated by S/G, denoted by $\mathcal{L}(S/G)$, is defined recursively as follows [8]:

- $\varepsilon \in \mathcal{L}(S/G)$;
- For any string $s \in \mathcal{L}(S/G)$ and non-τ event $\sigma \in \Sigma$, $s\sigma \in \mathcal{L}(S/G)$ if and only if $\sigma \in \Sigma_{R(G)}(s) \cap S(P(s))[1]$ holds, where $\Sigma_{R(G)}(s) = \{\sigma' \in \Sigma_R | s\sigma' \in \mathcal{L}(R(G))\}$ is the set of events of $R(G)$ after s;
- For any $s \in \mathcal{L}(S/G)$, $s\tau \in \mathcal{L}(S/G)$ if and only if $\tau \in \Sigma_{R(G)}(s)$ and $\Sigma_{R(G)}(s) \cap S(P(s))[2] = \emptyset$ hold.

After S/G generates a string $s \in \mathcal{L}(S/G)$, τ can occur if and only if τ is currently active (i.e., $\tau \in \Sigma_{R(G)}(s)$) and no forcible event enabled and forced by S can occur, i.e., $S(P(s))[2] \cap \Sigma_{R(G)}(s) = \emptyset$. The non-$\tau$ event $\sigma \in \Sigma$ can occur after s if and only if it is currently active and enabled, i.e., $\sigma \in \Sigma_{R(G)}(s) \cap S(P(s))[1]$. To prevent a supervisor from stopping the time progress in a TDES G, we assume that the region automaton $R(G)$ satisfies the following constraint.

Constraint 3: For any $q \in Q$, if $\delta(q, \tau)$ is not defined, there exists an uncontrollable event $\sigma \in \Sigma_{uc}$ such that $\delta(q, \sigma)!$.

This constraint may restrict G's application; however, due to the theoretical ease of operation, Constraint 3 has been adopted by many works, see, e.g., [7], [8], [11], to ensure time progression.

Example 5: Consider the TDES G as shown in Fig. 1(c) and its region automaton $R(G)$ depicted in Fig. 2. Assume that the events p_1, p_1', p_2', and τ are observable, while p_3' is unobservable. Moreover, suppose that both p_1 and p_3' are controllable, but p_1' and p_2' are uncontrollable. Finally, let p_1 be forcible. As a result, one has $\Sigma_o = \{p_1, p_1', p_2', \tau\}$, $\Sigma_{uo} = \{p_3'\}$, $\Sigma_c = \{p_1, p_3'\}$, $\Sigma_{uc} = \{p_1', p_2'\}$, and $\Sigma_{for} = \{p_1\}$.

Let us define a supervisor S by $S(\alpha) = (\{p_1, p_1', p_2'\}, \{p_1\})$ for all $\alpha \in P(\mathcal{L}(R(G)))$. It is a static supervisor which, upon observing an event, disables p_3', enables p_1, p_1', and p_2', and forces p_1. Due to $S(P(\tau p_1'))[2] = \{p_1\}$, $\Sigma_{R(G)}(\tau p_1') = \{\tau, p_1\}$, and $\Sigma_{R(G)}(\tau p_1') \cap S(P(\tau p_1'))[2] = \{p_1\}$, at state q_3, the timed event τ will be preempted by the forced occurrence of p_1. The closed-loop system S/G is shown in Fig. 3. □

III. Networked Timed Discrete Event Systems

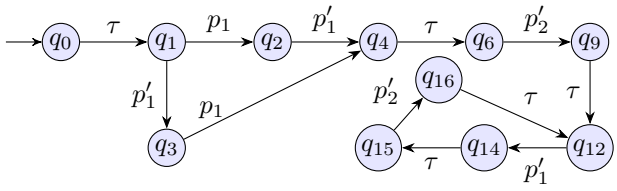

Fig. 3: Closed-loop system S/G.

A. NTDESs with Bandwidth Constraints

Definition 6: (Networked timed discrete event system) A networked timed discrete event system (NTDES) is a triple

$$\Omega = (G, oc, cc), \tag{2}$$

where

- $G = (L, l_0, \Sigma, C, Inv, E)$ is a TDES;
- oc is a k-tuple with $oc = (oc_1, oc_2, \ldots, oc_k)$ modeling k observation channels such that for any $i \in \{1, 2, \ldots, k\}$, $oc_i = (\Sigma_{o,i}, AB_i)$ is the i-th observation channel, where $\Sigma_{o,i} \subseteq \Sigma_{on} = \Sigma_o \setminus \{\tau\}$ is the set of non-τ observable events whose occurrences will be sent to a supervisor through oc_i, and $AB_i \in \mathbb{N}$ is the allocated bandwidth for oc_i, which is the maximum number of the events in $\Sigma_{o,i}$ that can be transmitted through oc_i per time unit;
- $cc \in \mathbb{N}$ is the upper time bound of control delays and losses in the control channel. ◇

In an NTDES $\Omega = (G, oc, cc)$, after the timed event τ occurs, τ is directly observed by a supervisor, without being transmitted through observation channels [10], [11]. In contrast, upon the occurrence of a non-τ event $\sigma \in \Sigma_{o,i}$ with $i \in \{1, 2, \ldots, k\}$, σ is communicated with a supervisor through observation channel oc_i. The allocated bandwidth AB_i for oc_i limits the number of events in $\Sigma_{o,i}$ that can be transmitted through oc_i per time unit to at most AB_i, which is called *observation channel bandwidth constraint* imposed on oc_i. The *global observation channel bandwidth constraint* imposed on Ω stipulates that any observation channel bandwidth constraint imposed on an observation channel must be satisfied. After a supervisor observes an event, it issues a control decision to G through the control channel. Finally, the following assumptions are made for an NTDES Ω.

A1: For any $i', i'' \in \{1, 2, \ldots, k\}$ with $i' \neq i''$, it holds $\Sigma_{o,i'} \cap \Sigma_{o,i''} = \emptyset$ and $\bigcup_{i \in \{1,2,\ldots,k\}} \Sigma_{o,i} = \Sigma_{on}$.

A2: A supervisor observes an observable event immediately after its occurrence.

A3: At any instant, Ω must use one of the control decisions issued by a supervisor in the past cc time units.

Assumption A1 means that the occurrence of any non-τ observable event σ is sent to a unique observation channel [9]. Assumption A2 states that observation delays and losses in observation channels are not considered in this paper. If they are deemed important, one can use the method in [9], [10] to transform a region automaton $R(G)$ to an observed plant that captures all possible delays and losses in the observation channels. Then, we replace $R(G)$ with this observed plant, using it as a new model for modeling the behavior of G.

Assumption A3 is widely adopted in the supervisory control community to deal with control delays and losses [4], [6], [11].

Remark 2: The *control channel bandwidth constraint* is that the number of control decisions transmitted through the control channel per time unit should not exceed an allocated bandwidth value, which will not be considered in this paper. Here, we provide a method to deal with that constraint. In the supervisory control framework, in one time unit, the number of control decisions issued by a supervisor is equal to the number of events observed [1]. If the control channel bandwidth constraint is not satisfied, one can ensure its satisfaction by reducing the number of events that can be sent to the supervisor per time unit, that is, by enforcing a new global observation channel bandwidth constraint. ■

Example 6: Consider the TDES G as shown in Fig. 1(c) and its region automaton $R(G)$ depicted in Fig. 2. Assume that observation channel oc_1 (resp. oc_2) transmits events p_1 and p_1' (resp. p_2') to a supervisor. We assign bandwidths $AB_1 = 1$ and $AB_2 = 1$ for oc_1 and oc_2, respectively. It follows $oc = (oc_1, oc_2)$ with $oc_1 = (\Sigma_{o,1} = \{p_1, p_1'\}, AB_1 = 1)$ and $oc_2 = (\Sigma_{o,2} = \{p_2'\}, AB_2 = 1)$. Note that the occurrence of unobservable event p_3' will not be sent to an observation channel. Let S/G be the closed-loop system developed in Example 5 and displayed in Fig. 3. As can be seen, the observation channel bandwidth constraint imposed on oc_2 is satisfied, since at most one event p_2' is transmitted through oc_2 in one time unit. On the other hand, the occurrence of string $\tau p_1 p_1' \tau \in \mathcal{L}(S/G)$ indicates that two events p_1 and p_1' have been conveyed through oc_1 within the same time unit, which violates the observation channel bandwidth constraint on oc_1. Hence, the global observation channel bandwidth constraint is violated.

Let us define another supervisor S' by $S'(\tau p_1' \tau p_2') = S'(\tau p_1' \tau p_2' \tau) = \gamma_1 = (\{p_1, p_1', p_2'\}, \{p_1\})$ and $S'(\alpha) = \gamma_2 = (\{p_1', p_2'\}, \emptyset)$ for all $\alpha \in P(\mathcal{L}(R(G))) \setminus \{\tau p_1' \tau p_2', \tau p_1' \tau p_2' \tau\}$. That is, S' enables p_1, p_1', and p_2', and forces p_1, upon observing strings $\tau p_1' \tau p_2'$ and $\tau p_1' \tau p_2' \tau$, while enabling p_1' and p_2' but forcing nothing for other observations. The closed-loop system S'/G is shown in Fig. 4, where the occurrences of p_1 and p_1' are separated by one time unit. Now, the global observation channel bandwidth constraint is satisfied. Supervisor S' is bandwidth-safe.

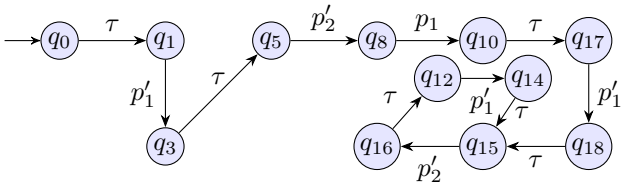

Fig. 4: Closed-loop system S'/G.

For an NDES in [4]– [6], since the occurrence of any event represents the elapse of one time unit, when the upper time bound of control delays and losses is $cc = 0$, no control delays and losses exist. In contrast, the NTDES modeled by the B-W framework in [11] uses the timed event $tick$ to denote time elapse. Hence, even when $cc = 0$, there exist control delays and losses. Specifically, the control decisions issued by a supervisor upon observing non-timed events within one time unit may still be delayed or lost [11]. So does this fact in an NTDES modeled by a timed automaton in this paper, since the timed event τ is adopted to model time passage. It seems that the NTDES developed in this paper is not as general as the one modeled by the B-W framework in the sense that the former cannot simulate the absence of control delays and losses. However, in most real-world network systems, control delays and losses are inevitable. Thus, this limitation does not have a significant impact on the application of the model considered in this paper. On the other hand, if there are no control delays and losses, an NTDES reduces to a conventional TDES and we can use the non-networked supervisory control theory in [7], [8] to handle the reduced system.

We assume $cc = 0$ for the sake of simplifying the demonstration of the proposed method. This results in an NTDES $\Omega = (G, oc, cc)$, where the observation channels oc are constructed previously. When supervisor S' observes string $\tau p_1' \tau p_2'$, the control decisions issued in the past 0 time unit are $S'(\tau p_1' \tau p_2') = \gamma_1$ and $S'(\tau p_1' \tau) = \gamma_2$. Thanks to Assumption A3, under control delays and losses, although S' issues γ_1 after $\tau p_1' \tau p_2'$, the control decision actually being used by the closed-loop system S'/Ω may be γ_2. Note that after $\tau p_1' \tau p_2'$ occurs, S'/Ω reaches state q_8. Since γ_2 does not force p_1 at q_8 to preempt τ, S'/Ω can enter q_{11} by executing τ. Subsequently, S' issues $S'(\tau p_1' \tau p_2' \tau) = \gamma_1$ to enable p_1 at q_{11}. Afterwards, string $\tau p_1' \tau p_2' \tau p_1 p_1' \tau$ will inevitably occur in S'/Ω, implying that p_1 and p_1' can be transmitted through observation channel oc_1 within one time unit. This violates the global observation channel bandwidth constraint. Consequently, the control delays and losses make the originally bandwidth-safe supervisor S' unsafe. □

B. The Over-Approximation Language of NTDESs

To deal with control delays and losses, we adopt the over-approximation technique, as introduced in [4]–[6], [11], to obtain the behavior of an NTDES $\Omega = (G, oc, cc)$ under the control of a supervisor S. Given a string $s \in \mathcal{L}(R(G))$, the notation $\#_t(s)$ denotes the number of τ events in s. Let $\Gamma_S(s) = \{\gamma \in \Gamma | (\exists \alpha', \alpha'' \in \Sigma_o^*) \alpha' \alpha'' = P(s) \wedge \#_t(\alpha'') \leq cc \wedge \gamma = S(\alpha')\}$ be the set of control decisions issued by S in the past cc time units after s, each of which may be used by Ω [11].

Definition 7: Let $\Omega = (G, oc, cc)$ be an NTDES and S be a supervisor. The over-approximation language generated by the closed-loop system S/Ω, denoted by $\mathcal{L}_{oa}(S/\Omega)$, is defined recursively as follows:

1) $\varepsilon \in \mathcal{L}_{oa}(S/\Omega)$;
2) For any string $s \in \mathcal{L}_{oa}(S/\Omega)$ and non-τ event $\sigma \in \Sigma$, $s\sigma \in \mathcal{L}_{oa}(S/\Omega)$ if and only if there exists $\gamma \in \Gamma_S(s)$ such that $\sigma \in \Sigma_{R(G)}(s) \cap \gamma[1]$;
3) For any $s \in \mathcal{L}_{oa}(S/\Omega)$, $s\tau \in \mathcal{L}_{oa}(S/\Omega)$ if and only if $\tau \in \Sigma_{R(G)}(s)$ and there exists $\gamma \in \Gamma_S(s)$ such that $\Sigma_{R(G)}(s) \cap \gamma[2] = \emptyset$. ◇

The language $\mathcal{L}_{oa}(S/\Omega)$ is an over-approximation of the actual behavior generated by S/Ω as it captures all possible

control decisions that may be currently used by Ω under control delays and losses [4], [6].

Example 7: Consider the NTDES $\Omega = (G, oc, cc)$ developed in Example 6. Define a supervisor S by $S(\tau p_1' \tau) = S(\tau p_1' \tau p_2') = (\{p_1, p_1', p_2'\}, \{p_1\})$ and $S(\alpha) = (\{p_1', p_2'\}, \emptyset)$ for all $\alpha \in P(\mathcal{L}(R(G)))\backslash\{\tau p_1'\tau, \tau p_1'\tau p_2'\}$. After string $\tau p_1' \tau p_2'$ occurs, the closed-loop system S/Ω enters state q_8 and S issues control decision $S(\tau p_1' \tau p_2')$. Moreover, $\Gamma_S(\tau p_1' \tau p_2') = \{S(\tau p_1'\tau), S(\tau p_1'\tau p_2')\}$ holds. This indicates that the control decisions issued in the past $cc = 0$ time unit are $S(\tau p_1'\tau)$ and $S(\tau p_1'\tau p_2')$. Regarding control delays and losses, two situations are considered:

- $S(\tau p_1'\tau p_2')$ is lost or delayed. In this case, S/Ω continues to operate under the previously issued control decision $S(\tau p_1'\tau)$ to enable and force event p_1 at state q_8 to preempt the timed event τ.
- $S(\tau p_1'\tau p_2')$ is received immediately by S/Ω. In this case, S/Ω uses $S(\tau p_1'\tau p_2')$ to enable and force p_1 at q_8 to preempt τ.

In contrast to the case in Example 6, under the control of S, state q_{11} that leads to violation of the global observation channel bandwidth constraint will not be reachable from q_8 through τ. This can be verified by the over-approximation language $\mathcal{L}_{oa}(S/\Omega)$ generated by the automaton shown in Fig. 5. The global observation channel bandwidth constraint is satisfied. □

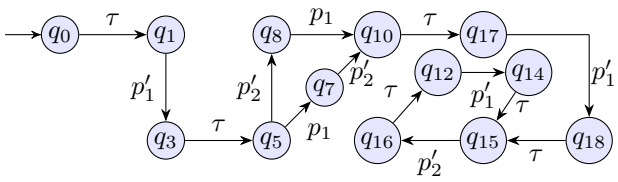

Fig. 5: Automaton generating $\mathcal{L}_{oa}(S/\Omega)$.

C. Problem Formulation

Let $\Omega = (G, oc, cc)$ be an NTDES and $I_o = \{1, 2, \ldots, k\}$ be the index set for k observation channels. To determine whether the global observation channel bandwidth constraint is violated, the notion of a global instantaneous throughput is introduced, which is a k-tuple $(n_1, n_2, \ldots, n_k)$ such that $n_i \in \mathbb{N}$ for $i \in I_o$ is the number of events transmitted through observation channel oc_i most recently. Let $\Lambda \subseteq \underbrace{\mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N}}_{k}$ be the set of all global instantaneous throughputs. To track global instantaneous throughputs, we introduce an operator $Count : \Lambda \times \Sigma_o \to \Lambda$ defined by: for any $\xi \in \Lambda$ and $\sigma \in \Sigma_o$,

1) if $\sigma = \tau$, then $Count(\xi, \sigma) = \underbrace{(0, 0, \ldots, 0)}_{k}$;
2) if $\sigma \in \Sigma_{o,i}$ for some $i \in I_o$, then $Count(\xi, \sigma) = \xi'$ such that for any $j \in I_o$,

$$\xi'[j] = \begin{cases} \xi[j], & \text{if } j \in I_o \setminus \{i\} \\ \xi[j] + 1, & \text{if } j = i. \end{cases} \tag{3}$$

In plain words, upon the occurrence of timed event τ, all entries in ξ are set to zero. After a non-τ event $\sigma \in \Sigma_{o,i}$ occurs, it is transmitted through observation channel oc_i and $\xi[i]$ is correspondingly increased by one, while keeping the other entries $\xi[j]$ unchanged for $j \in I_o \setminus \{i\}$. The global instantaneous throughput upon the occurrence of an observation $\alpha\sigma \in P(\mathcal{L}(R(G)))$ with $\alpha \in \Sigma_o^*$ and $\sigma \in \Sigma_o$, denoted by $\mathcal{H}(\alpha\sigma)$, is recursively computed by:

$$\begin{aligned} \mathcal{H}(\varepsilon) &= \underbrace{(0, 0, \ldots, 0)}_{k}, \\ \mathcal{H}(\alpha\sigma) &= Count(\mathcal{H}(\alpha), \sigma). \end{aligned} \tag{4}$$

Definition 8: Let $\Omega = (G, oc, cc)$ be an NTDES, $\alpha \in P(\mathcal{L}(R(G)))$ be an observation, and $\mathcal{H}(\alpha)$ be the current global instantaneous throughput. Observation α is said to be *bandwidth safe* if $\mathcal{H}(\alpha)[i] \leq AB_i$ for all $i \in I_o$. ◇

The strings $s \in \mathcal{L}(R(G))$ such that $P(s)$ is not bandwidth safe are those violating the global observation channel bandwidth constraint and need to be excluded from the over-approximation language $\mathcal{L}_{oa}(S/\Omega)$ under the control of a supervisor S. In this sense, we can determine whether or not $\mathcal{L}_{oa}(S/\Omega)$ is bandwidth safe.

Definition 9: Let $\Omega = (G, oc, cc)$ be an NTDES and S be a supervisor. The over-approximation language $\mathcal{L}_{oa}(S/\Omega)$ is said to be bandwidth safe if for any $\alpha \in P(\mathcal{L}_{oa}(S/\Omega))$, α is bandwidth safe. ◇

If $\mathcal{L}_{oa}(S/\Omega)$ is bandwidth safe, then the actual number of events transmitted through an observation channel per time unit must not exceed the corresponding channel bandwidth. Consequently, the global observation channel bandwidth constraint is satisfied. In contrast, if $\mathcal{L}_{oa}(S/\Omega)$ is not bandwidth safe, then the global observation channel bandwidth constraint may not be satisfied. This motivates us to formulate the following problem.

Problem 1: *Let $\Omega = (G, oc, cc)$ be an NTDES. Find a supervisor S such that the over-approximation language $\mathcal{L}_{oa}(S/\Omega)$ is bandwidth safe.* ■

Example 8: Consider the NTDES $\Omega = (G, oc, cc)$ developed in Example 6. The initial global instantaneous throughput is $\mathcal{H}(\varepsilon) = (0, 0)$. Upon the occurrence of timed event τ, the global instantaneous throughput becomes $\mathcal{H}(\tau) = Count(\mathcal{H}(\varepsilon), \tau) = (0, 0)$. After $p_1 \in \Sigma_{o,1} = \{p_1, p_1'\}$ occurs, it holds $\mathcal{H}(\tau p_1) = Count(\mathcal{H}(\tau), p_1) = (1, 0)$. Next, the occurrence of p_1' leads to $\mathcal{H}(\tau p_1 p_1') = Count(\mathcal{H}(\tau p_1), p_1') = (2, 0)$. Observation $\tau p_1 p_1'$ is not bandwidth safe, due to $\mathcal{H}(\tau p_1 p_1')[1] = 2 > AB_1 = 1$. In fact, p_1 and p_1' occur within the same time unit, thus violating the observation channel bandwidth constraint on oc_1.

For the closed-loop system S/Ω developed in Example 7, its over-approximation language $\mathcal{L}_{oa}(S/\Omega)$ is generated by the automaton shown in Fig. 5. One can easily verify that $\mathcal{L}_{oa}(S/\Omega)$ is bandwidth safe. The global observation channel bandwidth constraint on S/Ω is satisfied. □

IV. Online Over-Approximation State Estimation

To solve Problem 1, the over-approximation state estimate upon the occurrence of an observation is defined.

Definition 10: Let $\Omega = (G, oc, cc)$ be an NTDES and S be a supervisor. The over-approximation state estimate of the closed-loop system S/Ω upon the occurrence of an observation $\alpha \in P(\mathcal{L}_{oa}(S/\Omega))$ is defined by

$$E_S(\alpha) = \{q \in Q| \\ (\exists s \in \mathcal{L}_{oa}(S/\Omega))P(s) = \alpha \wedge q = \delta(q_0, s)\}. \quad (5)$$

◇

The over-approximation state estimate $E_S(\alpha)$ is an over-approximation of all possible states that S/Ω may reach after observing α by S. For short, an over-approximation state estimate will be called state estimate. To compute $E_S(\alpha)$, it is necessary to track all possible delayed control decisions using the notion of control channel configurations [6].

Definition 11: Let $\Omega = (G, oc, cc)$ be an NTDES. A control channel configuration is defined by a set $\theta \subseteq \Gamma \times \{0, 1, \ldots, cc\}$. Denote by $\Theta \subseteq 2^{\Gamma \times \{0,1,\ldots,cc\}}$ the set of all control channel configurations. ◇

Specifically, if $\theta = \emptyset$, no control decision is delayed in the control channel; otherwise, any element $(\gamma, n) \in \theta$ represents that a control decision γ is delayed and can be used by Ω for up to n occurrences of the timed event τ [6].

After a supervisor issues a control decision γ, we add the tuple (γ, cc) to the current control channel configuration θ, which is formalized by an operator $Add_c : \Theta \times \Gamma \to \Theta$ such that for any $\theta \in \Theta$ and $\gamma \in \Gamma$, $Add_c(\theta, \gamma) = \theta \cup \{(\gamma, cc)\}$. Upon the occurrence of timed event τ, we decrease all the delay tolerances n of control decisions in θ by one and delete all the updated tuples (γ, n') with $n' < 0$. This process is modeled by an operator $Next_c : \Theta \to \Theta$ such that for any $\theta \in \Theta$, $Next_c(\theta) = \{(\gamma, n-1) \in \Gamma \times \mathbb{N} | (\gamma, n) \in \theta \wedge n \geq 1\}$.

The control channel configuration upon observing a string $\alpha\sigma \in P(\mathcal{L}(R(G)))$ with $\alpha \in \Sigma_o^*$ and $\sigma \in \Sigma_o$ by a supervisor S, denoted by $\Theta_S(\alpha\sigma)$, is recursively defined by:

$$\Theta_S(\varepsilon) = \{(S(\varepsilon), cc)\}, \\ \Theta_S(\alpha\sigma) = \begin{cases} Add_c(\Theta_S(\alpha), S(\alpha\sigma)), \text{ if } \sigma \neq \tau \\ Add_c(Next_c(\Theta_S(\alpha)), S(\alpha\sigma)), \text{ if } \\ \qquad\qquad\qquad\qquad \sigma = \tau. \end{cases} \quad (6)$$

Specifically, $\{(S(\varepsilon), cc)\}$ is the initial control channel configuration immediately after S issues $S(\varepsilon)$. Upon the occurrence of $\alpha\sigma$, S issues $S(\alpha\sigma)$. If $\sigma \neq \tau$, the operator Add_c is used to add $S(\alpha\sigma)$ into the previous control channel configuration $\Theta_S(\alpha)$. If $\sigma = \tau$, we first update $\Theta_S(\alpha)$ using the $Next_c$ operator and then add $S(\alpha\sigma)$ to the updated control channel configuration through Add_c.

Considering a string $s \in \mathcal{L}(R(G))$, we denote by $\Gamma_{\Theta,S}(s) = \{\gamma \in \Gamma | (\exists n \in \{0, 1, \ldots, cc\})(\gamma, n) \in \Theta_S(P(s))\}$ the set of control decisions currently delayed in the control channel. It holds $\Gamma_{\Theta,S}(s) = \Gamma_S(s)$, i.e., $\Gamma_{\Theta,S}(s)$ is also the set of control decisions issued by S in the past cc time units, each of which may be used by an NTDES in the over-approximation sense.

Given a control channel configuration $\theta \in \Theta$, the set of non-τ events enabled by θ is defined as $\Sigma_e(\theta) = \bigcup_{(\gamma,n)\in\theta} \gamma[1]$. We say that a non-$\tau$ event $\sigma \in \Sigma$ can occur at a state $q \in Q$ under θ if $\delta(q, \sigma)!$ and σ is enabled by θ, i.e., $\sigma \in \Sigma_e(\theta)$. Further, we say that the timed event τ can occur at q under θ if $\delta(q, \tau)!$ and there exists $(\gamma, n) \in \theta$ such that γ does not force any events that are active at q to preempt τ, i.e., $\gamma[2] \cap \{\sigma' \in \Sigma_R | \delta(q, \sigma')!\} = \emptyset$.

Example 9: Consider the closed-loop system S/Ω developed in Example 7. Initially, supervisor S issues $S(\varepsilon) = \gamma_1 = (\{p_1', p_2'\}, \emptyset)$. The current control channel configuration is $\Theta_S(\varepsilon) = \{(S(\varepsilon), cc)\} = \{(\gamma_1, 0)\}$. It follows $\Gamma_{\Theta,S}(\varepsilon) = \{\gamma_1\} = \Gamma_S(\varepsilon)$. The set of non-$\tau$ events enabled by $\Theta_S(\varepsilon)$ is $\Sigma_e(\Theta_S(\varepsilon)) = \{p_1', p_2'\}$. Thanks to Fig. 2, no non-τ event can occur at state q_0 under $\Theta_S(\varepsilon)$. Since $\delta(q_0, \tau)!$ and $\gamma_1[2] = \emptyset$, the timed event τ can occur at q_0 under $\Theta_S(\varepsilon)$. After this occurrence, S/Ω enters state q_1 and supervisor S issues $S(\tau) = \gamma_1$. Then, we update $\Theta_S(\varepsilon)$ to $Next_c(\Theta_S(\varepsilon)) = \emptyset$ and $Next_c(\Theta_S(\varepsilon))$ to $\Theta_S(\tau) = Add_c(\emptyset, S(\tau)) = \{(\gamma_1, 0)\}$. One has $\Gamma_{\Theta,S}(\tau) = \{\gamma_1\} = \Gamma_S(\tau)$. The set of non-$\tau$ events enabled by $\Theta_S(\tau)$ is $\{p_1', p_2'\}$. By $\delta(q_1, p_1')!$, event p_1' can occur at q_1 under $\Theta_S(\tau)$. Upon this occurrence, S/Ω reaches state q_3 and S issues $S(\tau p_1') = \gamma_1$. Subsequently, $\Theta_S(\tau)$ is updated to $\Theta_S(\tau p_1') = Add_c(\Theta_S(\tau), S(\tau p_1')) = \{(\gamma_1, 0)\}$. We have $\Gamma_{\Theta,S}(\tau p_1') = \{\gamma_1\} = \Gamma_S(\tau p_1')$.

The timed event τ can occur at q_3 under $\Theta_S(\tau p_1')$. Then, S/Ω arrives at state q_5 and S issues $S(\tau p_1' \tau) = (\{p_1, p_1', p_2'\}, \{p_1\}) = \gamma_2$. We update $\Theta_S(\tau p_1')$ to $Next_c(\Theta_S(\tau p_1')) = \emptyset$ and $Next_c(\Theta_S(\tau p_1'))$ to $\Theta_S(\tau p_1' \tau) = Add_c(\emptyset, S(\tau p_1' \tau)) = \{(\gamma_2, 0)\}$. It follows $\Gamma_{\Theta,S}(\tau p_1' \tau) = \{\gamma_2\} = \Gamma_S(\tau p_1' \tau)$. The set of non-$\tau$ events enabled by $\Theta_S(\tau p_1' \tau)$ is $\{p_1, p_1', p_2'\}$. Due to $\delta(q_5, p_1)!$ and $\delta(q_5, p_2')!$, events p_1 and p_2' can occur at q_5 under $\Theta_S(\tau p_1' \tau)$. All the derived system behavior thus far is consistent with the over-approximation language $\mathcal{L}_{oa}(S/\Omega)$ generated by the automaton shown in Fig. 5. □

Let us define an information state of the form $\tilde{q} = (x, \theta, \xi)$ to simultaneously track state estimate $x \subseteq Q$, control channel configuration $\theta \in \Theta$, and global instantaneous throughput $\xi \in \Lambda$. Let $\tilde{Q} \subseteq 2^Q \times \Theta \times \Lambda$ be the set of all information states. In the game perspective [3], control decisions and observable events take effect on information states. In this sense, a supervisor state $y = (\tilde{q}, \Gamma)$ is introduced such that $\tilde{q} \in \tilde{Q}$ is the current information state and Γ is used to inform us that a supervisor can make a control decision in Γ at y. Let $Y \subseteq \tilde{Q} \times \{\Gamma\}$ be the set of all supervisor states. Next, a networked system state, denoted by $z = (\tilde{q}, \Sigma_o)$, is created such that $\tilde{q} \in \tilde{Q}$ represents the current information state and Σ_o informs us that the observable events in Σ_o can occur at z. Let $Z \subseteq \tilde{Q} \times \{\Sigma_o\}$ be the set of all networked system states. Not all events can occur at a networked system state.

Definition 12: Let $\Omega = (G, oc, cc)$ be an NTDES. An observable event $\sigma \in \Sigma_o$ can occur at a networked system state $z = ((x, \theta, \xi), \Sigma_o) \in Z$ if there exists a state $q \in x$ such that σ can occur at q under θ. ◇

Example 10: Consider the NTDES $\Omega = (G, oc, cc)$ developed in Example 6 and a networked system state $z = ((\{q_1\}, \theta, (0, 0)), \Sigma_o) \in Z$ with $\theta = \{(\gamma_1, 0)\}$, where $\gamma_1 = (\{p_1', p_2'\}, \emptyset)$. The observable event p_1' can occur at z, since p_1' can occur at state q_1 under θ. Although p_1 is active at q_1, it cannot occur at z, as p_1 cannot occur at q_1 under θ. □

The effect of control decisions on a supervisor state is captured by an "unobservable reach" operator $U_r : Y \times \Gamma \to Z$

by: for any $y = (\tilde{q} = (x, \theta, \xi), \Gamma) \in Y$ and $\gamma \in \Gamma$,

$$U_r(y, \gamma) = z = (\tilde{q}' = (x', \theta', \xi'), \Sigma_o), \tag{7}$$

where $\theta' = Add_c(\theta, \gamma)$, $\xi = \xi'$, and $x' = \{q' \in Q | (\exists q \in x)(\exists s \in (\Sigma_e(\theta') \cap \Sigma_{uo})^*) q' = \delta(q, s)\}$. After control decision γ is issued, supervisor state y transitions to networked system state z. During this process, the control channel configuration θ is first updated to $\theta' = Add_c(\theta, \gamma)$. As no observable event is sent to observation channels, global instantaneous throughput ξ keeps unchanged. The state estimate x' is the set of states reachable from the states in x via unobservable strings enabled by θ'.

Example 11: Consider the NTDES $\Omega = (G, oc, cc)$ developed in Example 6. Assume that a supervisor issues a control decision $\gamma_1 = (\{p_1, p_1', p_2', p_3'\}, \{p_1\})$ at a supervisor state $y = ((\{q_5\}, \emptyset, (0, 0)), \Gamma) \in Y$. This leads to a networked system state $z = U_r(y, \gamma_1) = (\tilde{q}' = (x', \theta', \xi'), \Sigma_o)$, where $\theta' = \emptyset \cup \{(\gamma_1, cc)\} = \{(\gamma_1, 0)\}$ and $\xi' = (0, 0)$. Due to $\Sigma_e(\theta') = \{p_1, p_1', p_2', p_3'\}$, the unobservable event p_3' is enabled by θ'. Since $\delta(q_5, p_3') = q_8$, it holds $x' = \{q_5, q_8\}$. □

To model the effect of observable events on networked system states, an "observable reach" operator $O_r : Z \times \Sigma_o \to Y$ is defined such that for any $z = (\tilde{q} = (x, \theta, \xi), \Sigma_o) \in Z$ and $\sigma \in \Sigma_o$,

- if $\sigma \neq \tau$, we have

$$O_r(z, \sigma) = y = (\tilde{q}' = (x', \theta, Count(\xi, \sigma)), \Gamma), \tag{8}$$

where $x' = \{q' \in Q | (\exists q \in x) \delta(q, \sigma) = q'\}$.

- if $\sigma = \tau$, we have

$$O_r(z, \tau) = y = (\tilde{q}' = (x', Next_c(\theta), Count(\xi, \tau)), \Gamma), \tag{9}$$

where $x' = \{q' \in Q | (\exists q \in x)(\exists (\gamma, n) \in \theta) \delta(q, \tau) = q' \wedge \gamma[2] \cap \{\sigma' \in \Sigma_R | \delta(q, \sigma')!\} = \emptyset\}$. Upon the occurrence of observable event σ, networked system state z reaches supervisor state y. Correspondingly, global instantaneous throughput ξ is updated to $Count(\xi, \sigma)$. If $\sigma \neq \tau$, control channel configuration θ does not change and the updated state estimate x' collects all the states that are reachable from the states in x through σ. If $\sigma = \tau$, even though τ is active at a state in x, τ may not occur, as other events may preempt τ. To this end, we first compute all the states in x at which τ can occur under θ. Then, x' is the set of states reachable from the computed states through τ. Subsequently, θ is updated to $Next_c(\theta)$.

Example 12: Consider the NTDES $\Omega = (G, oc, cc)$ developed in Example 6. Let $z = (\tilde{q} = (\{q_3, q_4\}, \theta, (0, 0)), \Sigma_o) \in Z$, where $\theta = \{(\gamma_1, 0)\}$ with $\gamma_1 = (\{p_1, p_1', p_2', p_3'\}, \{p_1\})$, be a networked system state. If event p_1 occurs at z, one obtains a supervisor state $y_1 = O_r(z, p_1) = (\tilde{q}' = (x', \theta, \xi'), \Gamma)$, where $\xi' = Count((0, 0), p_1) = (1, 0)$. Thanks to $\delta(q_3, p_1) = q_4$, it holds $x' = \{q_4\}$. If the timed event τ occurs at z, we first compute all the states in $\{q_3, q_4\}$ at which τ can occur under θ. Clearly, although τ is active at state q_3, τ cannot occur at q_3 under θ. In contrast, τ can occur at q_4 under θ, leading to $\delta(q_4, \tau) = q_6$. Hence, the occurrence of τ at z results in a supervisor state $y_2 = O_r(z, \tau) = (\tilde{q}'' = (x'', \theta', \xi''), \Gamma)$, where $x'' = \{q_6\}$, $\theta' = Next_c(\theta) = \emptyset$, and $\xi'' = Count((0, 0), \tau) = (0, 0)$. □

Let $y_0 = ((\{q_0\}, \emptyset, \underbrace{(0, 0, \ldots, 0)}_{k}), \Gamma)$ be the initial supervisor state. The networked system state after an observation $\alpha\sigma \in P(\mathcal{L}_{oa}(S/\Omega))$ with $\alpha \in \Sigma_o^*$ and $\sigma \in \Sigma_o$, denoted by $L_{S,Z}(\alpha\sigma)$, is recursively defined by:

$$\begin{aligned} &L_{S,Z}(\varepsilon) = U_r(y_0, S(\varepsilon)), \\ &L_{S,Z}(\alpha\sigma) = U_r(O_r(L_{S,Z}(\alpha), \sigma), S(\alpha\sigma)). \end{aligned} \tag{10}$$

In plain words, $L_{S,Z}(\varepsilon)$ is the networked system state after ε. Assume that networked system state $L_{S,Z}(\alpha)$ is reached. After σ occurs, the observable reach $O_r(L_{S,Z}(\alpha), \sigma)$ is obtained. Then, S issues $S(\alpha\sigma)$. This results in unobservable reach $L_{S,Z}(\alpha\sigma) = U_r(O_r(L_{S,Z}(\alpha), \sigma), S(\alpha\sigma))$. It is not difficult to verify that the networked system state $L_{S,Z}(\alpha) = (\tilde{q} = (x, \theta, \xi), \Sigma_o)$ after an observation $\alpha \in P(\mathcal{L}_{oa}(S/\Omega))$ satisfies $x = E_S(\alpha)$, $\theta = \Theta_S(\alpha)$, and $\xi = \mathcal{H}(\alpha)$.

A supervisor S solves Problem 1 if and only if any observation $\alpha \in P(\mathcal{L}_{oa}(S/\Omega))$ is bandwidth safe, that is, the networked system state $L_{S,Z}(\alpha) = ((x, \theta, \xi), \Sigma_o)$ after α satisfies $\xi[i] \leq AB_i$ for all $i \in I_o$. This motivates us to make the following notion.

Definition 13: Given an NTDES $\Omega = (G, oc, cc)$, a networked system state $z = ((x, \theta, \xi), \Sigma_o) \in Z$ is said to be *safe* if $\xi[i] \leq AB_i$ for all $i \in I_o$. ◇

Let $\Omega = (G, oc, cc)$ be an NTDES and S be a supervisor. The supervisor S solves Problem 1 if and only if $L_{S,Z}(\alpha)$ is safe for all $\alpha \in P(\mathcal{L}_{oa}(S/\Omega))$.

Remark 3: In the supervisory control community, besides enforcing the control specification of bandwidth safety, a supervisor S may also need to ensure compliance with another control specification represented by $K \subseteq \mathcal{L}(R(G))$ such that $\mathcal{L}_{oa}(S/\Omega) \subseteq K$ [1], [5]. Using the notion of strict subautomaton [6], [11], [13], there exists a set of states $Q_H \subseteq Q$ such that for all $s \in \mathcal{L}(R(G))$, $\delta(q_0, s) \in Q_H$ if and only if $s \in K$. When forcing K is of importance, we extend Definition 13 to additionally require that the state estimate x should satisfy $x \subseteq Q_H$. Then, by synthesizing a supervisor S such that $L_{S,Z}(\alpha)$ is safe for all $\alpha \in P(\mathcal{L}_{oa}(S/\Omega))$, we can ensure not only bandwidth safety, but also $\mathcal{L}_{oa}(S/\Omega) \subseteq K$. ■

Example 13: Consider the NTDES $\Omega = (G, oc, cc)$ developed in Example 6. Define a supervisor S by $S(\varepsilon) = S(\tau) = \gamma_1 = (\{p_1', p_2', p_3'\}, \emptyset)$ and $S(\alpha) = \gamma_2 = (\{p_1, p_1', p_2', p_3'\}, \{p_1\})$ for all $\alpha \in P(\mathcal{L}(R(G))) \setminus \{\varepsilon, \tau\}$. The networked system state after ε is $z_0 = L_{S,Z}(\varepsilon) = U_r(y_0, S(\varepsilon)) = ((\{q_0\}, \{(\gamma_1, 0)\}, (0, 0)), \Sigma_o)$. Only the timed event τ can occur at z_0, leading to supervisor state $y_1 = O_r(z_0, \tau) = ((\{q_1\}, \emptyset, (0, 0)), \Gamma)$. The networked system state after τ is $z_1 = L_{S,Z}(\tau) = U_r(y_1, S(\tau)) = ((\{q_1\}, \{(\gamma_1, 0)\}, (0, 0)), \Sigma_o)$. Event p_1 cannot occur at z_1, while p_1' can. This occurrence results in supervisor state $y_2 = O_r(z_1, p_1') = ((\{q_3\}, \{(\gamma_1, 0)\}, (1, 0)), \Gamma)$. The networked system state after $\tau p_1'$ is $z_2 = L_{S,Z}(\tau p_1') = U_r(y_2, S(\tau p_1')) = ((\{q_3\}, \{(\gamma_1, 0), (\gamma_2, 0)\}, (1, 0)), \Sigma_o)$. Event p_1 can occur at z_2. This induces supervisor state $y_3 = O_r(z_2, p_1) = ((\{q_4\}, \{(\gamma_1, 0), (\gamma_2, 0)\}, (2, 0)), \Gamma)$. The networked system state after $\tau p_1' p_1$ is $z_3 = L_{S,Z}(\tau p_1' p_1) = U_r(y_3, S(\tau p_1' p_1)) =$

$((\{q_4\},\{(\gamma_1,0),(\gamma_2,0)\},(2,0)),\Sigma_o)$ and it is unsafe. As a result, S cannot solve Problem 1. □

V. Networked Timed Bipartite Transition Systems

Using the observable and unobservable reach operators defined in this paper, the bipartite transition system in [3] is extended to a networked timed bipartite transition system.

Definition 14: Let $\Omega = (G, oc, cc)$ be an NTDES. A networked timed bipartite transition system (NTBTS) for Ω, denoted by Δ, is a four-tuple

$$\Delta = (Q_B, \delta_{Y,Z}, \delta_{Z,Y}, y_0), \tag{11}$$

where

- Q_B is the set of states, partitioned into $Q_B = Y_B \dot\cup Z_B$;
- $Y_B \subseteq Y$ is the set of supervisor states of Δ;
- $Z_B \subseteq Z$ is the set of networked system states of Δ;
- $\delta_{Y,Z} : Y_B \times \Gamma \to Z_B$ is the partial transition function such that

$$\begin{aligned}&(\forall y \in Y_B)(\forall \gamma \in \Gamma)(\forall z \in Z_B)\delta_{Y,Z}(y,\gamma) = z\\ &\Rightarrow U_r(y,\gamma) = z \wedge z \text{ is safe.}\end{aligned} \tag{12}$$

- $\delta_{Z,Y} : Z_B \times \Sigma_o \to Y_B$ is the partial transition function such that for any $z \in Z_B$, $y \in Y_B$, and $\sigma \in \Sigma_o$,

$$\delta_{Z,Y}(z,\sigma) = y \Rightarrow O_r(z,\sigma) = y \wedge \sigma \text{ can occur at } z. \tag{13}$$

- $y_0 = (\{q_0\},\emptyset,\underbrace{(0,0,\ldots,0)}_{k}),\Gamma) \in Y_B$ is the initial supervisor state. ◇

The NTBTS Δ can be viewed as a bipartite game played by an NTDES Ω and a supervisor S. At a supervisor state $y \in Y_B$, S makes a control decision γ defined at y (i.e., $\delta_{Y,Z}(y,\gamma)!$). Then, the game enters a networked system state $z = U_r(y,\gamma)$ through the transition $(y,\gamma,z) \in \delta_{Y,Z}$. Note that Eq. (12) requires that z should be safe. This constraint ensures that any networked system state $L_{S,Z}(\alpha)$ reached by $\alpha \in P(\mathcal{L}_{oa}(S/\Omega))$ under S is safe, to solve Problem 1.

At a networked system state $z \in Z_B$, Ω chooses an observable event $\sigma \in \Sigma_o$ defined at z (i.e., $\delta_{Z,Y}(z,\sigma)!$) and the game enters a supervisor state $y = O_r(z,\sigma)$ through the transition $(z,\sigma,y) \in \delta_{Z,Y}$. Eq. (13) guarantees that only the observable events that can occur at z can be chosen.

Algorithm 1 is presented to construct an NTBTS. Thanks to the depth-first search implemented by Procedure $DoDFS$ in Lines 7–18, for any first-encountered supervisor state, all the control decisions in Γ are considered to compute corresponding successor networked system states; for any first-encountered safe networked system state, all the observable events in Σ_o that can occur are taken into account for deriving corresponding successor supervisor states. Then, Procedure $Prune$ in Lines 19–20 iteratively prunes the supervisor states that have no successors and their predecessor networked system states. Subsequently, if $y_0 \in Y_B$, Algorithm 1 returns an NTBTS Δ. Let $\mathbb{C}_\Delta(y) = \{\gamma \in \Gamma | \delta_{Y,Z}(y,\gamma)!\}$ be the set of control decisions defined at a supervisor state y in Δ. The returned NTBTS Δ satisfies $\mathbb{C}_\Delta(y) \neq \emptyset$ for all $y \in Y_B$, which is usually called complete NTBTS [3]. If, on the other hand, $y_0 \notin Y_B$, Algorithm 1 returns "No solution", implying that no complete NTBTS exists.

Algorithm 1: Construction of an NTBTS Δ.

Input: An NTDES $\Omega = (G, oc, cc)$.
Output: An NTBTS Δ.

1 $y_0 \leftarrow ((\{q_0\},\emptyset,\underbrace{(0,0,\ldots,0)}_{k}),\Gamma)$, $Y_B \leftarrow \{y_0\}$, $Z_B \leftarrow \emptyset$, $\delta_{Y,Z} \leftarrow \emptyset$, $\delta_{Z,Y} \leftarrow \emptyset$;
2 **DoDFS**(y_0, Δ);
3 $\Delta \leftarrow$ **Prune**(Δ);
4 **if** $y_0 \notin Y_B$ **then**
5 **return** "No solution";
  else
6 **return** Δ;
  procedure `DoDFS` (y, Δ) **;**
7 **for** $\gamma \in \Gamma$ **do**
8 $z \leftarrow U_r(y,\gamma)$;
9 **if** z is safe **then**
10 $\delta_{Y,Z} \leftarrow \delta_{Y,Z} \cup \{(y,\gamma,z)\}$;
11 **if** $z \notin Z_B$ **then**
12 $Z_B \leftarrow Z_B \cup \{z\}$;
13 **for** $\sigma \in \Sigma_o$ such that it can occur at z **do**
14 $y' \leftarrow O_r(z,\sigma)$;
15 $\delta_{Z,Y} \leftarrow \delta_{Z,Y} \cup \{(z,\sigma,y')\}$;
16 **if** $y' \notin Y_B$ **then**
17 $Y_B \leftarrow Y_B \cup \{y'\}$;
18 **DoDFS** (y', Δ);
  procedure `Prune` (Δ) **;**
19 **while** exists $y \in Y_B$ that has no successor **do**
20 delete y from Y_B and delete all its predecessor networked system states z for which $y = \delta_{Z,Y}(z,\sigma)$ for some $\sigma \in \Sigma_o$;

Remark 4: In Algorithm 1, a supervisor state without a successor will inevitably lead an NTDES, via the unobservable reach defined in Eq. (7), to an unsafe networked system state, which violates the global observation channel bandwidth constraint. This indicates that when the NTDES reaches that supervisor state, the global observation channel bandwidth constraint has already been violated, as during the unobservable reach, no observable events occur to affect observation channels. Hence, such a supervisor state must be deleted. ■

Theorem 1: Given an NTDES $\Omega = (G, oc, cc)$, Problem 1 has a solution if and only if Algorithm 1 returns an NTBTS.

Proof: If Algorithm 1 returns "No solution", Ω under the control of any supervisor will inevitably reach an unsafe networked system state, implying that Problem 1 has no solution. If, on the other hand, Algorithm 1 returns an NTBTS, there exists a supervisor S such that the closed-loop system S/Ω will not transition to an unsafe networked system state, thus solving the problem. ■

After Algorithm 1 outputs a complete NTBTS Δ, Algorithm 2 is designed to decode from Δ an online Δ-based supervisor S_Δ to solve Problem 1. Initially, S_Δ issues a control decision defined at the initial supervisor state y_0, leading Δ to a networked system state. Thereafter, upon observing an observable event by S_Δ, Δ enters a supervisor state following the event and S_Δ makes a control decision defined at the supervisor state. Then, Δ transitions to a new networked system state and waits for the occurrence of new observable events.

Algorithm 2: Construction of a Supervisor S_Δ.

Input: A complete NTBTS $\Delta = (Q_B, \delta_{Y,Z}, \delta_{Z,Y}, y_0)$.
Output: A control decision.

$\alpha \leftarrow \varepsilon$, $y \leftarrow y_0$;
Choose a control decision $\gamma \in \mathbb{C}_\Delta(y)$;
Output control decision $S_\Delta(\alpha) \leftarrow \gamma$;
$z \leftarrow \delta_{Y,Z}(y, \gamma)$;
while S_Δ observes a new observable event $\sigma \in \Sigma_o$ **do**
  $y \leftarrow \delta_{Z,Y}(z, \sigma)$;
  $\alpha \leftarrow \alpha\sigma$;
  Choose a control decision $\gamma \in \mathbb{C}_\Delta(y)$;
  Output control decision $S_\Delta(\alpha) \leftarrow \gamma$;
  $z \leftarrow \delta_{Y,Z}(y, \gamma)$;

VI. Case Study

In this section, a motivating example is presented to illustrate the formulated supervisory control problem under bandwidth constraints and the method for solving it. Consider an automated guided robot (AGR) system as depicted in Fig. 6, consisting of two zones, three paths, and two sensors. There are two AGRs: AGR 1 and AGR 2. They are initially in Zone 0 and can reach Zone 1 via Path 1. Then, only AGR 2 can return to Zone 0 through Paths 2 or 3.

AGR 1 either can choose to stay in Zone 0 forever without crossing Path 1, or it can linger in Zone 1 once it reaches there. In contrast, AGR 2 can only stay in a zone for up to one time unit. Sensors 1 and 2 monitor the movement of an AGR along Paths 1 and 2, respectively, while there is no sensor for observing Path 3. Both AGRs need to stay in Zone 0 for at least one time unit to load cargo before they decide to traverse Path 1 to Zone 1 for unloading their cargo. In Zone 1, AGR 2 requires at least one time unit to unload its cargo and then enter Zone 0 for loading cargo again, and so forth.

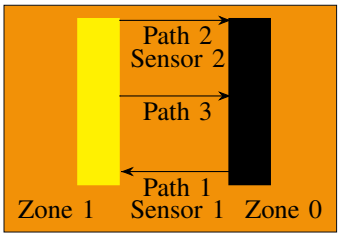

Fig. 6: An automated guided robot system.

The TDES G_1 for AGR 1 is shown in Fig. 1(a) with the sets of clocks $C_1 = \{c_s\}$, locations $L_1 = \{l_0, l_1\}$ with l_0 being the initial location, and events $\Sigma_1 = \{p_1\}$. The TDES G_2 for AGR 2 is shown in Fig. 1(b) with the sets of clocks $C_2 = \{c'_s\}$, locations $L_2 = \{l'_0, l'_1\}$, and events $\Sigma_2 = \{p'_1, p'_2, p'_3\}$. Location l_i (resp. l'_i), $i \in \{0, 1\}$, represents that AGR 1 (resp. AGR 2) has reached Zone i; event p_1 denotes that AGR 1 has passed through Path 1; event p'_j, $j \in \{1, 2, 3\}$, indicates that AGR 2 has gone through Path j; clock c_s (resp. c'_s) records the sojourn time units AGR 1 (resp. AGR 2) has spent in a zone.

The label $(c'_s \geq 1, p'_1, \{c'_s\})$ between locations l'_0 and l'_1 in G_2 signifies that G_2 can transition from l'_0 to l'_1, when the guard $c'_s \geq 1$ holds, which captures the fact that AGR 2 needs at least one time unit in Zone 0 to load cargo before passing through Path 1. After that, event p'_1 occurs and the clock c'_s is reset to zero, suggesting that upon entering Zone 1, c'_s is ready to measure the time AGR 2 sojourns in Zone 1. The label $[c'_s \leq 1]$ next to l'_1 denotes that the invariant of l'_1 is $c'_s \leq 1$, indicating that AGR 2 can stay at Zone 1 for up to one time unit. Note that, the invariants of l_0 and l_1 in G_1 equal true, meaning that AGR 1 can linger at Zone 0 or Zone 1 forever. We omit the labels of the two invariants for saving space. By the synchronous product operator defined in Definition 1, the synchronous product of G_1 and G_2 is the TDES $G = G_1||G_2 = (L, l_0, \Sigma, C, Inv, E)$ with the set of clocks $C = \{c_s, c'_s\}$, as shown in Fig. 1(c), where location (l_i, l'_j), $i, j \in \{0, 1\}$, represents that AGRs 1 and 2 have entered Zones i and j, respectively.

By Definition 5, the region automaton $R(G)$ of G is displayed in Fig. 2, where the states are displayed in Table I. To save space, we use simplified notations to denote clock regions. For instance, the notation $[0, 0]$ in the initial state q_0 refers to the initial clock region $[0_C]$ of all clock valuations $v \in V_C$ satisfying $v(c_s) = v(c'_s) = 0$. The notation $[> 1, 1]$ in q_5 represents the clock region of all clock valuations $v \in V_C$ satisfying $v(c_s) > 1$ and $v(c'_s) = 1$. After generating string τp_1, $R(G)$ enters state $q_2 = ((l_1, l'_0), [0, 1])$. Since p_1 indicates that AGR 1 goes through Path 1, τp_1 means that after 1 time unit, AGR 1 enters Zone 1 through Path 1, while AGR 2 still stays at Zone 0.

According to the assignment of sensors for the paths in the AGR system as depicted in Fig. 6, the events p_1, p'_1, p'_2, and τ are observable, while p'_3 is unobservable. Assume that a supervisor can only prevent AGR 1 from passing through Path 1 and AGR 2 from going through Path 3. Hence, both p_1 and p'_3 are controllable, but p'_1 and p'_2 are uncontrollable. Finally, let p_1 be forcible, that is, a supervisor can force AGR 1 to go through Path 1. As a result, for $R(G)$, one has $\Sigma_o = \{p_1, p'_1, p'_2, \tau\}$, $\Sigma_{uo} = \{p'_3\}$, $\Sigma_c = \{p_1, p'_3\}$, $\Sigma_{uc} = \{p'_1, p'_2\}$, and $\Sigma_{for} = \{p_1\}$.

AGR 1's task is to reach Zone 1 to unload its cargo. To this end, a supervisor must force event p_1 to occur. AGR 2's goal is to continuously travel between Zone 0 and Zone 1, loading cargo in Zone 0 and unloading it in Zone 1. Furthermore, we require that a supervisor should keep track of the passage of an AGR through a path to inform workers when to load and unload its cargo. For this purpose, the unobservable event p'_3 should be disabled all the time. For simplicity, the three objectives are collectively referred to as the AGR objectives.

We define a supervisor S by $S(\alpha) = (\{p_1, p_1', p_2'\}, \{p_1\})$ for all $\alpha \in P(\mathcal{L}(R(G)))$. It is a static supervisor which, upon observing an event, disables p_3', enables p_1, p_1', and p_2', and forces p_1. The closed-loop system S/G is shown in Fig. 3. As can be seen, AGR 1 will enter Zone 1 by executing p_1, AGR 2 continuously travels between Zone 0 and Zone 1 by generating p_1' and p_2', and the passage of the two AGRs through a path only causes the occurrence of observable events that can be observed by S. The AGR objectives are achieved in S/G.

Assume that Sensor 1 (resp. 2) transmits events p_1 and p_1' (resp. p_2') to the supervisor S through observation channel oc_1 (resp. oc_2). We assign bandwidths $AB_1 = 1$ and $AB_2 = 1$ for oc_1 and oc_2, respectively. Then, the number of events transmitted through an observation channel per time unit should not exceed one. Formally, by Definition 6, one has $oc = (oc_1, oc_2)$ with $oc_1 = (\Sigma_{o,1} = \{p_1, p_1'\}, AB_1 = 1)$ and $oc_2 = (\Sigma_{o,2} = \{p_2'\}, AB_2 = 1)$. Note that the occurrence of unobservable event p_3' will not be sent to an observation channel. In the closed-loop system S/G as shown in Fig. 3, the observation channel bandwidth constraint imposed on oc_2 is satisfied, as at most one event p_2' can be transmitted through oc_2 in one time unit. However, the occurrence of string $\tau p_1 p_1' \tau \in \mathcal{L}(S/G)$ indicates that two events p_1 and p_1' have been conveyed in oc_1 within the same time unit. Hence, the observation channel bandwidth constraint on oc_1 is violated. The global observation channel bandwidth constraint is not satisfied and the observation of p_1 and p_1' through oc_1 may be lost. Hence, supervisor S fails to keep track of the passage of an AGR through Path 1, which significantly impacts the subsequent loading and unloading tasks.

To this end, let us define another supervisor S' by $S'(\tau p_1' \tau p_2') = S'(\tau p_1' \tau p_2' \tau) = \gamma_1 = (\{p_1, p_1', p_2'\}, \{p_1\})$ and $S'(\alpha) = \gamma_2 = (\{p_1', p_2'\}, \emptyset)$ for all $\alpha \in P(\mathcal{L}(R(G))) \setminus \{\tau p_1' \tau p_2', \tau p_1' \tau p_2' \tau\}$. That is, S' enables p_1, p_1', and p_2', disables p_3', and forces p_1, upon observing strings $\tau p_1' \tau p_2'$ and $\tau p_1' \tau p_2' \tau$, while enabling p_1' and p_2', disabling p_3', but forcing nothing for other observations. The closed-loop system S'/G is shown in Fig. 4, where the occurrences of p_1 and p_1' are separated by one time unit. Now, the global observation channel bandwidth constraint is satisfied. Supervisor S' is bandwidth-safe. The observation of events p_1 and p_1' through oc_1 will not be lost. In S'/G, AGR 1 will enter Zone 1 by executing p_1, AGR 2 continuously travels between Zone 0 and Zone 1 by generating p_1' and p_2', and the passage of the two AGRs through a path only causes the occurrence of observable events that can be observed by S'. The AGR objectives are achieved.

Next, the control delays and losses are taken into account. Assume that the upper time bound of control delays and losses is $cc = 0$. By integrating the previous constructed observation channels oc, an NTDES $\Omega = (G, oc, cc)$ can be obtained. Then, when the supervisor S' observes string $\tau p_1' \tau p_2'$, the control decisions issued in the past 0 time unit are $S'(\tau p_1' \tau p_2') = \gamma_1$ and $S'(\tau p_1' \tau) = \gamma_2$. Thanks to Assumption A3, although S' issues γ_1 after $\tau p_1' \tau p_2'$, the control decision actually being used by the closed-loop system S'/Ω may be γ_2. Note that after $\tau p_1' \tau p_2'$ occurs, S'/Ω reaches state q_8. Since γ_2 does not force p_1 at q_8 to preempt τ, S'/Ω can enter q_{11} by executing τ. Subsequently, S' issues $S'(\tau p_1' \tau p_2' \tau) = \gamma_1$ to enable p_1 at q_{11}. Inevitably, the string $\tau p_1' \tau p_2' \tau p_1 p_1' \tau$ will occur in S'/Ω, implying that p_1 and p_1' can be transmitted through observation channel oc_1 within one time unit. This violates the observation channel bandwidth constraint imposed on oc_1. As can be seen, the control delays and losses render the original bandwidth-safe supervisor S' unsafe. As a result, the observation of p_1 and p_1' by S' through oc_1 may be lost. Then, S' fails to keep track of the passage of an AGR through Path 1. The AGR objectives in the closed-loop system S'/Ω cannot be achieved.

To ensure that a supervisor can keep track of the passage of an AGR through a path, the NTDES $\Omega = (G, oc, cc)$ must satisfy the global observation channel bandwidth constraint. Moreover, the unobservable event p_3' must be disabled at all times. This can be done by first applying Algorithm 1 to construct a complete NTBTS, where Ω will not reach unsafe networked system states, thus ensuring bandwidth safety. Then, we delete all the control decisions from the NTBTS that enable p_3'.

To ensure that AGR 1 will eventually reach Zone 1 to unload its cargo, it is necessary for the event p_1 to occur in Ω. Hence, we remove all the path branches from the NTBTS that prevent the generation of p_1. The final complete NTBTS, denoted by Δ, is displayed in Fig. 7, where the states and control decisions are stated in Table II.

Subsequently, by applying Algorithm 2, a Δ-based supervisor S_Δ can be decoded from Δ. Thanks to Fig. 7, S_Δ can be represented by $S_\Delta(\tau p_1' \tau) = S_\Delta(\tau p_1' \tau p_2') = S_\Delta(\tau p_1' \tau p_2' p_1) = \gamma_2 = (\{p_1, p_1', p_2'\}, \{p_1\})$ and $S_\Delta(\alpha) = \gamma_1 = (\{p_1', p_2'\}, \emptyset)$ for all $\alpha \in P(\mathcal{L}(R(G))) \setminus \{\tau p_1' \tau, \tau p_1' \tau p_2', \tau p_1' \tau p_2' p_1\}$. After string $\tau p_1' \tau p_2'$ occurs, the closed-loop system S_Δ/Ω enters state q_8 and S_Δ issues control decision $S_\Delta(\tau p_1' \tau p_2')$. The control decisions issued in the past 0 time unit are $S_\Delta(\tau p_1' \tau)$ and $S_\Delta(\tau p_1' \tau p_2')$. Hence, the control decision actually being used by S_Δ/Ω may be $S_\Delta(\tau p_1' \tau)$ or $S_\Delta(\tau p_1' \tau p_2')$. Regarding control delays and losses, two situations are considered:

- $S_\Delta(\tau p_1' \tau p_2')$ is lost or delayed. In this case, S_Δ/Ω continues to operate under the previously issued control decision $S_\Delta(\tau p_1' \tau)$ to enable and force event p_1 at state q_8 to preempt the timed event τ.
- $S_\Delta(\tau p_1' \tau p_2')$ is received immediately by S_Δ/Ω. In this case, S_Δ/Ω uses $S_\Delta(\tau p_1' \tau p_2')$ to enable and force p_1 at q_8 to preempt τ.

In contrast to the previous closed-loop system S'/Ω, under the control of S_Δ, state q_{11} that leads to violation of the global observation channel bandwidth constraint will not be reachable from q_8 through τ. The over-approximation language $\mathcal{L}_{oa}(S_\Delta/\Omega)$ is generated by the automaton as shown in Fig. 5. It follows that AGR 1 will enter Zone 1 by executing p_1, AGR 2 continuously travels between Zone 0 and Zone 1 by generating p_1' and p_2', and the passage of the two AGRs through a path causes the occurrence of observable events that can be observed by S_Δ. The AGR objectives in the closed-loop system S_Δ/Ω are achieved.

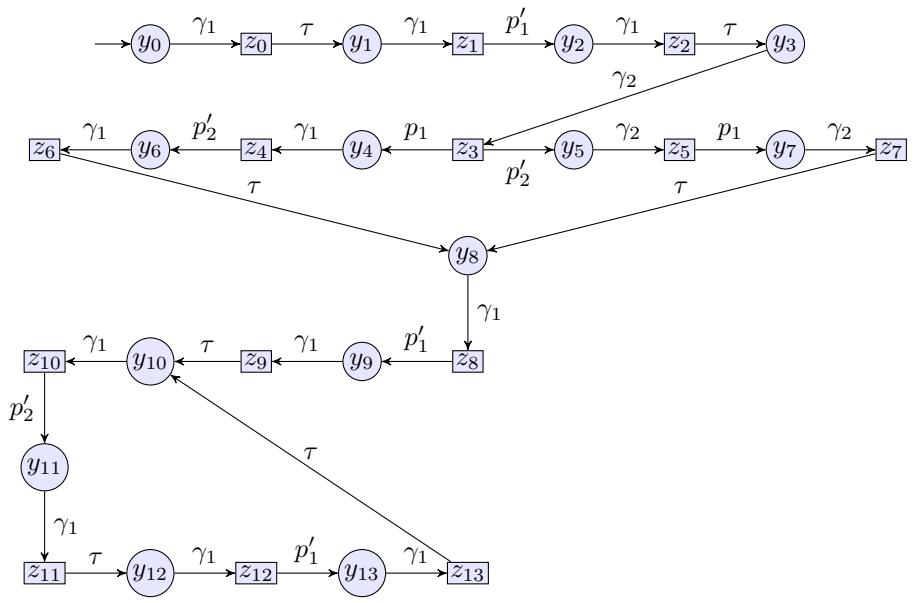

Fig. 7: A networked timed bipartite transition system Δ.

TABLE II: States and control decisions of the networked timed bipartite transition system Δ shown in Fig. 7.

$y_0 = (\{q_0\}, \emptyset, (0,0)), \Gamma)$	$\gamma_1 = (\{p_1', p_2'\}, \emptyset)$
$z_0 = ((\{q_0\}, \{(\gamma_1, 0)\}, (0,0)), \Sigma_o)$	$y_1 = (\{q_1\}, \emptyset, (0,0)), \Gamma)$
$z_1 = ((\{q_1\}, \{(\gamma_1, 0)\}, (0,0)), \Sigma_o)$	$y_2 = (\{q_3\}, \{(\gamma_1, 0)\}, (1,0)), \Gamma)$
$z_2 = ((\{q_3\}, \{(\gamma_1, 0)\}, (1,0)), \Sigma_o)$	$y_3 = (\{q_5\}, \emptyset, (0,0)), \Gamma)$
$\gamma_2 = (\{p_1, p_1', p_2'\}, \{p_1\})$	$z_3 = ((\{q_5\}, \{(\gamma_2, 0)\}, (0,0)), \Sigma_o)$
$y_4 = (\{q_7\}, \{(\gamma_2, 0)\}, (1,0)), \Gamma)$	$y_5 = (\{q_8\}, \{(\gamma_2, 0)\}, (0,1)), \Gamma)$
$z_4 = (\{q_7\}, \{(\gamma_1, 0), (\gamma_2, 0)\}, (1,0)), \Sigma_o)$	$z_5 = ((\{q_8\}, \{(\gamma_2, 0)\}, (0,1)), \Sigma_o)$
$y_6 = (\{q_{10}\}, \{(\gamma_1, 0), (\gamma_2, 0)\}, (1,1)), \Gamma)$	$y_7 = (\{q_{10}\}, \{(\gamma_2, 0)\}, (1,1)), \Gamma)$
$z_6 = (\{q_{10}\}, \{(\gamma_1, 0), (\gamma_2, 0)\}, (1,1)), \Sigma_o)$	$z_7 = ((\{q_{10}\}, \{(\gamma_2, 0)\}, (1,1)), \Sigma_o)$
$y_8 = (\{q_{17}\}, \emptyset, (0,0)), \Gamma)$	$z_8 = ((\{q_{17}\}, \{(\gamma_1, 0)\}, (0,0)), \Sigma_o)$
$y_9 = (\{q_{18}\}, \{(\gamma_1, 0)\}, (1,0)), \Gamma)$	$z_9 = ((\{q_{18}\}, \{(\gamma_1, 0)\}, (1,0)), \Sigma_o)$
$y_{10} = (\{q_{15}\}, \emptyset, (0,0)), \Gamma)$	$z_{10} = ((\{q_{15}\}, \{(\gamma_1, 0)\}, (0,0)), \Sigma_o)$
$y_{11} = (\{q_{16}\}, \{(\gamma_1, 0)\}, (0,1)), \Gamma)$	$z_{11} = ((\{q_{16}\}, \{(\gamma_1, 0)\}, (0,1)), \Sigma_o)$
$y_{12} = (\{q_{12}\}, \emptyset, (0,0)), \Gamma)$	$z_{12} = ((\{q_{12}\}, \{(\gamma_1, 0)\}, (0,0)), \Sigma_o)$
$y_{13} = (\{q_{14}\}, \{(\gamma_1, 0)\}, (1,0)), \Gamma)$	$z_{13} = ((\{q_{14}\}, \{(\gamma_1, 0)\}, (1,0)), \Sigma_o)$

VII. Conclusion

In this paper, we address a supervisory control problem under bandwidth constraints for an NTDES. The proposed method for solving this problem depends on constructing a region automaton for a TDES modeled by a timed automaton. However, as the number of edges and clocks in the timed automaton increases, the structure of the resulting region automaton becomes complex, thus limiting the practical applicability of the proposed method. In the future, we plan to explore alternative methods that can either obviate the need for generating a region automaton or effectively reduce its structural complexity.

References

[1] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems,* 3nd ed. Springer, 2021, doi: https://doi.org/10.1007/978-3-030-72274-6.

[2] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Cont. Opt.,* vol. 25, no. 1, pp. 206–230, 1987.

[3] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially-observed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 61, no. 5, pp. 1239–1254, 2016.

[4] F. Lin, "Control of networked discrete event systems: Dealing with communication delays and losses," *SIAM J. Cont. Opt.,* vol. 52, no. 2, pp. 1276–1298, 2014.

[5] S. Shu and F. Lin, "Supervisor synthesis for networked discrete event systems with communication delays," *IEEE Trans. Autom. Control,* vol. 60, no. 8, pp. 2183–2188, 2015.

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

[7] B. A. Brandin and W. M. Wonham, "Supervisory control of timed discrete-event systems," *IEEE Trans. Autom. Control,* vol. 39, no. 2, pp. 329–342, 1994.

[8] S. Takai and T. Ushio, "A new class of supervisors for timed discrete event systems under partial observation," *Discrete Event Dyn. Syst.,* vol. 16, no. 2, pp. 257–278, 2006.

[9] M. V. S. Alves, L. K. Carvalho, and J. C. Basilio, "Supervisory control

of networked discrete event systems with timing structure," *IEEE Trans. Autom. Control,* vol. 66, no. 5, pp. 2206–2218, 2021.

[10] A. Rashidinejad, M. Reniers, and L. Feng, "Supervisory control of timed discrete-event systems subject to communication delays and non-fifo observations," in *14th International Workshop on Discrete Event Systems*, pp. 456–463, 2018.

[11] B. Zhao, F. Lin, C. Wang, X. Zhang, M. P. Polis, and L. Y. Wang, "Supervisory control of networked timed discrete event systems and its applications to power distribution networks," *IEEE Trans. Contr. Netw. Syst.,* vol. 4, no. 2, pp. 146–158, 2017.

[12] Y. F. Hou, Y. F. Ji, G. Wang, C. Y. Weng, and Q. D. Li, "Modeling and state estimation for supervisory control of networked timed discrete-event systems and their application in supervisor synthesis," *International Journal of Control*, vol. 97, no. 6, pp. 1262–1282, 2024.

[13] Z. Y. Xiang, Y. F. Chen, N. Q. Wu, and Z. W. Li, "On the existence of non-blocking bounded supervisors for discrete event systems," *IEEE Trans. Autom. Control,* pp. 1–8, 2024, doi: 10.1109/TAC.2024.3407612.

[14] A. Rashidinejad, M. Reniers, and M. Fabian, "Supervisory control synthesis of timed automata using forcible events," *IEEE Trans. Autom. Control,* vol. 69, no. 2, pp. 1074–1080, 2024.

[15] W. J. Dong, K. Z. Zhang, S. Y. Li, and X. Yin, "On the verification of detectability for timed discrete event systems," *Automatica,* vol. 164, Article 111644, 2024.

[16] J. Klein, P. Kogel, and S. Glesner, "Verifying opacity of discrete-timed automata," in *IEEE/ACM 12th International Conference on Formal Methods in Software Engineering*, pp. 55–65, 2024.

[17] H. Gruber, M. Holzer, A. Kiehn, and B. König, "On timed automata with discrete time—structural and language theoretical characterization," in *Lecture Notes in Computer Science*, pp. 272–283, 2005.

[18] C. Gao, D. Lefebvre, C. Seatzu, Z. W. Li, and A. Giua, "A region-based approach for state estimation of timed automata under no event observation," in *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 799–804, 2020.

[19] C. Baier and J. Katoen, *Principles of Model Checking,* 2008, doi: https://api.semanticscholar.org/CorpusID:5302889.

[20] A. Kunnappilly, P. Backeman, and C. Seceleanu, "From UML modeling to UPPAAL model checking of 5G dynamic service orchestration," in *7th Conference on the Engineering of Computer Based Systems*, no. 11, pp. 1–10, 2021, doi: https://dl.acm.org/doi/fullHtml/10.1145/3459960.3459965.

[21] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science,* vol. 126, no. 2, pp. 183–235, 1994.