International Journal of Control

ISSN: 0020-7179 (Print) 1366-5820 (Online) Journal homepage: http://www.tandfonline.com/loi/tcon20

Delay-robustness in distributed control of timed discrete-event systems based on supervisor localisation

Renyuan Zhang, Kai Cai, Yongmei Gan & W. M. Wonham

To cite this article: Renyuan Zhang, Kai Cai, Yongmei Gan & W. M. Wonham (2016): Delay-robustness in distributed control of timed discrete-event systems based on supervisor localisation, International Journal of Control, DOI: 10.1080/00207179.2016.1147606

To link to this article: http://dx.doi.org/10.1080/00207179.2016.1147606

Accepted author version posted online: 05 Feb 2016.
Published online: 25 Feb 2016.

INTERNATIONAL JOURNAL OF CONTROL, 2016
http://dx.doi.org/10.1080/00207179.2016.1147606

Taylor & Francis
Taylor & Francis Group

Delay-robustness in distributed control of timed discrete-event systems based on supervisor localisation

Renyuan Zhang [a], Kai Cai[b], Yongmei Gan[c] and W. M. Wonham[d]

[a]School of Automation, Northwestern Polytechnical University, Xi'an, Shaanxi, China; [b]Urban Research Plaza, Osaka City University, Osaka, Japan; [c]School of Electrical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi, China; [d]Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada

ABSTRACT
Recently, we studied communication delay in distributed control of untimed discrete-event systems based on supervisor localisation. We proposed a property called delay-robustness: the overall system behaviour controlled by distributed controllers with communication delay is logically equivalent to its delay-free counterpart. In this paper, we extend our previous work to timed discrete-event systems, in which communication delays are counted by a special clock event *tick*. First, we propose a timed channel model and define timed delay-robustness; for the latter, a verification procedure is presented. Next, if the delay-robust property does not hold, we introduce *bounded* delay-robustness, and present an algorithm to compute the *maximal* delay bound (measured by number of *ticks*) for transmitting a channelled event. Finally, we demonstrate delay-robustness on the example of an underload tap-changing transformer.

ARTICLE HISTORY
Received 17 July 2015
Accepted 24 January 2016

KEYWORDS
Timed discrete-event systems; distributed supervisory control; supervisor localisation; delay-robustness

1. Introduction

For distributed control of discrete-event systems (DES), supervisor localisation was recently proposed (Cai & Wonham, 2010a, 2010b, 2015; Cai, Zhang, & Wonham, 2013; Zhang, Cai, Gan, Wang, & Wonham, 2013), which decomposes a monolithic supervisor or a heterarchical array of modular supervisors into local controllers for individual agents. Collective local controlled behaviour is guaranteed to be globally optimal and nonblocking, assuming that the shared events among local controllers are communicated instantaneously, i.e. with no delay. In practice, however, local controllers are linked by a physical communication network in which delays may be inevitable. Hence, for correct implementation of the local controllers obtained by localisation, it is essential to model and appraise communication delays.

In Zhang, Cai, Gan, Wang, and Wonham (2015) and its conference precursor (Zhang, Cai, Gan, Wang, & Wonham, 2012), we studied communication delays among local controllers for untimed DES. In particular, we proposed a new concept called *delay-robustness*, meaning that the system's behaviour of local controllers interconnected by communication channels subject to unbounded delays is logically equivalent to its delay-free counterpart. Moreover, we designed an efficient procedure to verify for which channelled events the system is delay-robust. If for a channelled event r the system fails to be delay-robust, there may still exist a finite bound for which the system can tolerate a delay in r. In untimed DES, however, there lacks a *temporal* measure for the delay bound (except for counting the number of occurrences of untimed events).

In this paper and its conference antecedent (Zhang, Cai, & Wonham, 2014), we extend our study on delay-robustness to the timed DES (or TDES) framework proposed in Brandin and Wonham (1994) and Wonham (2015). In this framework, the special clock event *tick* provides a natural way of modelling communication delay as temporal behaviour. We first propose a timed channel model for transmitting each channelled event, which effectively measures communication delay by the number of *tick* occurrences, with no *a priori* upper bound, so that the channel models *unbounded* delay. We then define timed delay-robustness with respect to the timed channel, thus extending its untimed counterpart (Zhang et al., 2015, 2012) in two respects: (1) the system's temporal behaviour is accounted for, and (2) timed controllability is required. An algorithm is presented to verify timed delay-robustness according to this new definition.

If the delay-robust property fails to hold, we introduce *bounded* delay-robustness and present a corresponding verification algorithm. In particular, the algorithm computes the *maximal* delay bound (in terms of number of

CONTACT Renyuan Zhang ryzhang@nwpu.edu.cn

ticks) for transmitting a channelled event, i.e. the largest delay that can be tolerated without violating the system specifications. These concepts and the corresponding algorithms are illustrated for the case of an under-load tap-changing transformer (ULTC).

Distributed/decentralised supervisory control with communication delay has been widely studied for untimed DES (e.g. Barrett & Lafortune, 2000; Darondeau & Ricker, 2012; Hiraishi, 2009; Kalyon, Gall, Marchand, & Massart, 2011; Lin, 2014; Sadid, Ricker, & Hashtrudi-Zad, 2015; Schmidt, Schmidt, & Zaddach, 2007; Tripakis, 2004; Xu & Kumar, 2008). In particular, in Tripakis (2004) and Kalyon et al. (2011), the existence of distributed controllers in the unbounded delay case is proved to be undecidable; and in Tripakis (2004), Schmidt et al. (2007), Xu and Kumar (2008), Hiraishi (2009) and Lin (2014), distributed controllers are synthesised under the condition that communication delay is bounded. We also note that Sadid et al. (2015) propose a way to verify robustness of a given synchronous protocol with respect to a fixed or a finitely bounded delay, as measured by the number of untimed events occurring during the transmitting process. We refer to Zhang et al. (2015, 2012) for a detailed review of these works and their differences from our approach. Communication delay in timed DES, on the other hand, has (to our knowledge) received little attention. The present work is based on our previous research on timed supervisor localisation (Cai & Wonham, 2015; Cai et al., 2013; Zhang et al., 2013); the new approach may be applied to different settings, e.g. timed automata, in a suitable manner (the details still need to be studied in future work).

The paper is organised as follows. Section 2 provides a review of the Brandin–Wonham TDES framework and recalls supervisor localisation for TDES. In Section 3, we introduce a timed channel model, and present the concept and verification algorithm for timed delay-robustness. In Section 4, we define bounded delay-robustness, and present an algorithm to compute the maximal delay bound. These concepts and the corresponding algorithms are demonstrated in Section 5 on the distributed control problem for a ULTC with communications. Conclusions are presented in Section 6.

2. Distributed control by supervisor localisation of TDES

2.1 Preliminaries on TDES

The TDES model proposed in Brandin and Wonham (1994) is an extension of the untimed DES generator model of the Ramadge–Wonham framework (Wonham, 2015). A TDES is given by

$$\mathbf{G} := (Q, \Sigma, \delta, q_0, Q_m). \tag{1}$$

Here, Q is the finite set of *states*; Σ is the finite set of events, including the special event *tick*, which represents 'tick of the global clock'; δ: $Q \times \Sigma \to Q$ is the (partial) *state transition function* (this is derived from the corresponding activity transition function; the reader is referred to the detailed transition rules given in Brandin and Wonham (1994) and Wonham (2015)); q_0 is the *initial state*; and $Q_m \subseteq Q$ is the set of *marker states*. The transition function is extended to δ: $Q \times \Sigma^* \to Q$ in the usual way. The *closed behaviour* of $\mathbf{G}$ is the language $L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_0, s)!\}$ and the *marked behaviour* is $L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \delta(q_0, s) \in Q_m\} \subseteq L(\mathbf{G})$. We say that $\mathbf{G}$ is *nonblocking* if $\bar{L}_m(\mathbf{G}) = L(\mathbf{G})$, where $\bar{\cdot}$ denotes *prefix closure* (Wonham, 2015).

Let Σ^* be the set of all finite strings, including the empty string ϵ. For $\Sigma' \subseteq \Sigma$, the *natural projection* P: $\Sigma^* \to \Sigma'^*$ is defined by

$$\begin{aligned} P(\epsilon) &= \epsilon; \\ P(\sigma) &= \begin{cases} \epsilon, & \text{if } \sigma \notin \Sigma', \\ \sigma, & \text{if } \sigma \in \Sigma'; \end{cases} \\ P(s\sigma) &= P(s)P(\sigma), \quad s \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{2}$$

As usual, P is extended to P: $Pwr(\Sigma^*) \to Pwr(\Sigma'^*)$, where $Pwr(\cdot)$ denotes powerset. Write P^{-1}: $Pwr(\Sigma'^*) \to Pwr(\Sigma^*)$ for the *inverse-image function* of P.

To adapt the TDES $\mathbf{G}$ in (1) for supervisory control, we first designate a subset of events, denoted by $\Sigma_{hib} \subseteq \Sigma$, to be the *prohibitible* events which can be disabled by an external supervisor. Next, and specific to TDES, we bring in another category of events, called the *forcible* events, which can *preempt* event *tick*; let $\Sigma_{for} \subseteq \Sigma$ denote the set of forcible events. Note that $tick \notin \Sigma_{hib} \cup \Sigma_{for}$. Now it is convenient to define the *controllable* event set $\Sigma_c := \Sigma_{hib} \dot{\cup} \{tick\}$. The *uncontrollable* event set is $\Sigma_u := \Sigma - \Sigma_c$.

We introduce the notion of (timed) controllability as follows. For a string $s \in L(\mathbf{G})$, define $Elig_{\mathbf{G}}(s) := \{\sigma \in \Sigma | s\sigma \in L(\mathbf{G})\}$ to be the subset of events 'eligible' to occur (i.e. defined) at the state $q = \delta(q_0, s)$. Consider an arbitrary language $F \subseteq L(\mathbf{G})$ and a string $s \in \overline{F}$; similarly, define the eligible event subset $Elig_F(s) := \{\sigma \in \Sigma | s\sigma \in \overline{F}\}$. We say F is *controllable* with respect to $\mathbf{G}$ if, for all $s \in \overline{F}$,

$$Elig_F(s) \supseteq \begin{cases} Elig_{\mathbf{G}}(s) \cap (\Sigma_u \dot{\cup} \{tick\}) \\ \qquad \text{if } Elig_F(s) \cap \Sigma_{for} = \emptyset, \\ Elig_{\mathbf{G}}(s) \cap \Sigma_u \\ \qquad \text{if } Elig_F(s) \cap \Sigma_{for} \neq \emptyset. \end{cases} \tag{3}$$

Whether or not F is controllable, we denote by $\mathcal{C}(F)$ the set of all controllable sublanguages of F. Then, $\mathcal{C}(F)$ is nonempty, closed under arbitrary set unions, and thus contains a unique supremal (largest) element denoted by $sup\mathcal{C}(F)$ (Brandin and Wonham, 1994; Wonham, 2015). Now consider a specification language $E \subseteq \Sigma^*$ imposed on the timed behaviour of **G**; E may represent a logical and/or temporal requirement. Let the TDES

$$\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m) \tag{4}$$

be the corresponding *monolithic supervisor* that is optimal (i.e. maximally permissive) and nonblocking in the following sense: **SUP**'s marked language $L_m(\mathbf{SUP})$ satisfies

$$L_m(\mathbf{SUP}) = sup\mathcal{C}(E \cap L_m(\mathbf{G})) \subseteq L_m(\mathbf{G})$$

and, moreover, its closed language $L(\mathbf{SUP}) = \overline{L}_m(\mathbf{SUP})$.

2.2 Supervisor localisation of TDES

In this section, we introduce the supervisor localisation procedure, which was initially proposed in the untimed DES framework (Cai and Wonham, 2010a) and then adapted to the TDES framework (Cai & Wonham, 2015; Cai et al., 2013; Zhang et al., 2013). By this procedure, a set of *local controllers* and *local preemptors* is obtained and shown to be 'control-equivalent' to the monolithic supervisor **SUP** in (4). By allocating these constructed local controllers and preemptors to each component agent, we build a distributed supervisory control architecture.

Let TDES **G** in (1) be the plant to be controlled and E be a specification language. As in Wonham (2015), synthesise the monolithic optimal and nonblocking supervisor **SUP**.[1] Supervisor **SUP**'s control action includes (1) disabling prohibitible events in Σ_{hib} and (2) preempting *tick* via forcible events in Σ_{for}. By the supervisor localisation procedure, a set of local controllers $\{\mathbf{LOC}_\alpha^C = (Y_\alpha, \Sigma_\alpha, \eta_\alpha, y_{0,\alpha}, Y_{m,\alpha}) | \alpha \in \Sigma_{hib}\}$ and a set of local preemptors $\{\mathbf{LOC}_\beta^P = (Y_\beta, \Sigma_\beta, \eta_\beta, y_{0,\beta}, Y_{m,\beta}) | \beta \in \Sigma_{for}\}$ are constructed. In each local controller $\mathbf{LOC}_\alpha^C$, each state $y_\alpha \in Y_\alpha$ corresponds to a cell of a *control cover* on **SUP**'s state set. The initial state is $y_{0,\alpha}$ and the marker state set $Y_{m,\alpha}$. The transition function η_α is derived from the control cover and the transitions defined on the corresponding states. The event set Σ_α is the set of events that cause state changes in η_α. A local preemptor is constructed similarly, but based on a *preemption cover* on **SUP**'s state set (Cai & Wonham, 2010a; Zhang et al., 2013). The supervisor localisation procedure localises **SUP**'s control action with respect to each prohibitible event α (resp. forcible event β) into a local controller $\mathbf{LOC}_\alpha^C$ (resp. local preemptor $\mathbf{LOC}_\beta^P$). These $\mathbf{LOC}_\alpha^C$ and $\mathbf{LOC}_\beta^P$ are all generalised TDES,[2] and proved to be control-equivalent to **SUP** (with respect to **G**) in the following sense:

$$L(\mathbf{G}) \cap \Big(\bigcap_{\alpha \in \Sigma_{hib}} P_\alpha^{-1} L(\mathbf{LOC}_\alpha^C) \Big) \cap \Big(\bigcap_{\beta \in \Sigma_{for}} P_\beta^{-1} L(\mathbf{LOC}_\beta^P) \Big) = L(\mathbf{SUP}), \tag{5}$$

$$L_m(\mathbf{G}) \cap \Big(\bigcap_{\alpha \in \Sigma_{hib}} P_\alpha^{-1} L_m(\mathbf{LOC}_\alpha^C) \Big) \cap \Big(\bigcap_{\beta \in \Sigma_{for}} P_\beta^{-1} L_m(\mathbf{LOC}_\beta^P) \Big) = L_m(\mathbf{SUP}). \tag{6}$$

Here, $P_\alpha : \Sigma^* \to \Sigma_\alpha^*$ and $P_\beta : \Sigma^* \to \Sigma_\beta^*$ are the natural projections as in (2).

Now, using the constructed local controllers and local preemptors, we build a distributed supervisory control architecture (without communication delay) for a multi-agent TDES plant. Consider that the plant **G** consists of N component TDES $\mathbf{G}_i$ ($i \in \mathcal{N} := \{1, 2, \ldots, N\}$), each with event set $\Sigma_i \ni tick$. For simplicity, assume $\Sigma_i \cap \Sigma_j = \{tick\}$, for all $i \neq j \in \mathcal{N}$; namely the agents $\mathbf{G}_i$ are independent except for synchronisation on the global event *tick*. As a result, the marked and closed behaviours of the composition of the $\mathbf{G}_i$ coincide with those of their synchronous product (Wonham, 2015), and thus we use synchronous product instead of composition to combine TDES together, i.e. $\mathbf{G} = \underset{i \in \mathcal{N}}{||} \mathbf{G}_i$, where $||$ denotes the synchronous product of TDES.[3]

A convenient *allocation policy* of local controllers/preemptors is as follows. For a fixed agent $\mathbf{G}_i$, let $\Sigma_{i,for}$, $\Sigma_{i,hib} \subseteq \Sigma_i$ be its forcible event set and prohibitible event set, respectively. As exemplified in Figure 1, allocate to $\mathbf{G}_i$ the set of local controllers $\mathbf{LOC}_i^C := \{\mathbf{LOC}_\alpha^C | \alpha \in \Sigma_{i,hib}\}$ and the set of local preemptors $\mathbf{LOC}_i^P := \{\mathbf{LOC}_\beta^P | \beta \in \Sigma_{i,for}\}$. This allocation creates a distributed control architecture for the multi-agent plant **G**, in which each agent $\mathbf{G}_i$ is controlled by its own local controllers/preemptors, while interacting with other agents through communication of shared events. For agent $\mathbf{G}_i$, the set of *communication events* that need to be imported from other agents is

$$\Sigma_{com,i} := \Big(\bigcup_{\alpha \in \Sigma_{i,hib}} \Sigma_\alpha - \Sigma_i \Big) \cup \Big(\bigcup_{\beta \in \Sigma_{i,for}} \Sigma_\beta - \Sigma_i \Big) \tag{7}$$

where Σ_α and Σ_β are the event sets of $\mathbf{LOC}_\alpha^C$ and of $\mathbf{LOC}_\beta^P$, respectively.

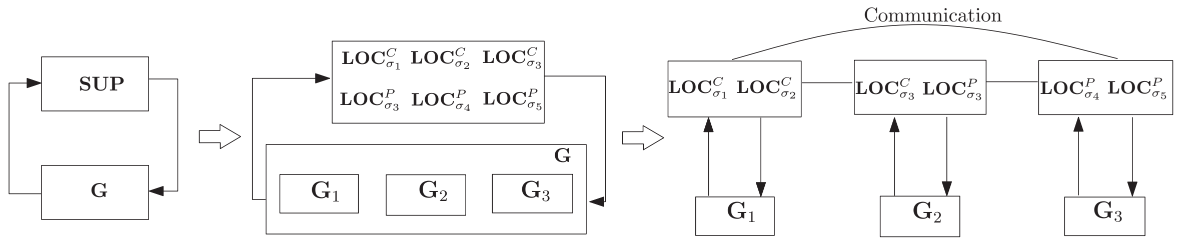
Communication
SUP
G
$\mathbf{LOC}^C_{\sigma_1}$ $\mathbf{LOC}^C_{\sigma_2}$ $\mathbf{LOC}^C_{\sigma_3}$
$\mathbf{LOC}^P_{\sigma_3}$ $\mathbf{LOC}^P_{\sigma_4}$ $\mathbf{LOC}^P_{\sigma_5}$
G
$\mathbf{G}_1$ $\mathbf{G}_2$ $\mathbf{G}_3$

Figure 1. Example of distributed control by supervisor localisation: plant **G** is composed of three agents $\mathbf{G}_k$ with event sets Σ_k, $k \in [1, 3]$; $\sigma_1, \sigma_2 \in \Sigma_1, \sigma_3 \in \Sigma_2$, and $\sigma_4, \sigma_5 \in \Sigma_3$; $\sigma_1, \sigma_2, \sigma_3 \in \Sigma_{hib}$ and $\sigma_3, \sigma_4, \sigma_5 \in \Sigma_{for}$. First, by the supervisor localisation procedure, we construct from **SUP** a set of local controllers $\{\mathbf{LOC}^C_{\sigma_1}, \mathbf{LOC}^C_{\sigma_2}, \mathbf{LOC}^C_{\sigma_3}\}$ and a set of local preemptors $\{\mathbf{LOC}^P_{\sigma_3}, \mathbf{LOC}^P_{\sigma_4}, \mathbf{LOC}^P_{\sigma_5}\}$. Then, by our allocation policy, we obtain a distributed control architecture in which each $\mathbf{G}_k$ is controlled by its own local controllers/preemptors, while interacting with other agents through communication of shared events.

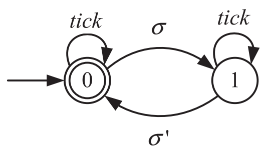

Figure 2. Timed channel model $\mathbf{CH}(j, \sigma, i)$ for transmitting event σ from $\mathbf{G}_j$ to $\mathbf{G}_i$ with indefinite (i.e. unspecified) time delay.

However, this distributed control architecture is built under the assumption that the communication delay of communication events is negligible. While simplifying the design of distributed controllers, this assumption may be unrealistic in practice, where controllers are linked by a physical network subject to delay. In the rest of this paper, we investigate how the communication delay affects the synthesised local control strategies and the corresponding overall system behaviour.

3. Timed delay-robustness

Consider event communication between a pair of agents $\mathbf{G}_i$ and $\mathbf{G}_j$ ($i, j \in \mathcal{N}$): specifically, $\mathbf{G}_j$ sends an event σ to $\mathbf{G}_i$. Let Σ_j be the event set of $\mathbf{G}_j$ and $\Sigma_{com,i}$ as in (7) the set of communication events that $\mathbf{G}_i$ imports from other agents. Then, the set of events that $\mathbf{G}_j$ sends to $\mathbf{G}_i$ is

$$\Sigma_{j,com,i} := \Sigma_j \cap \Sigma_{com,i}. \tag{8}$$

We thus have event $\sigma \in \Sigma_{j,com,i}$.

Now consider the timed channel model $\mathbf{CH}(j, \sigma, i)$ for transmission of σ displayed in Figure 2. $\mathbf{CH}(j, \sigma, i)$ is a 2-state TDES with event set $\{\sigma, \sigma', tick\}$. The transition from state 0 to 1 by σ means that $\mathbf{G}_j$ has sent σ to channel, while the transition from state 1 back to 0 by σ' means that $\mathbf{G}_i$ has received σ from channel. We refer to σ' as the *signal event* of σ, and assign its controllability status to be the same as σ (i.e. σ' is controllable if σ is controllable). The self-loop transition *tick* at state 1 therefore counts communication delay of σ transmission: the number of *ticks* that elapse between σ and σ'. Measuring the delay by *tick* events is a major improvement compared to the untimed channel model we used in Zhang et al. (2015, 2012) where no suitable measure existed to count the delay. With the aid of this *tick* measure, we shall later (Section 4) compute useful delay bounds for event communication.

It should be stressed that the number of *tick* occurrences between σ and σ' is unspecified, inasmuch as the self-loop *tick* at state 1 may occur indefinitely. In this sense, $\mathbf{CH}(j, \sigma, i)$ models *unbounded* communication delay. Note that *tick* is also self-looped at state 0; this is not used to count delay, but rather for the technical necessity of preventing the event *tick* from being blocked when synchronising $\mathbf{CH}(j, \sigma, i)$ with other TDES. The initial state 0 is marked, signalling each completion of event σ transmission; state 1, on the other hand, is unmarked because the transmission is still ongoing.

The capacity of channel $\mathbf{CH}(j, \sigma, i)$ is 1, meaning that only after the latest occurrence of event σ is received by its recipient $\mathbf{G}_i$, will the channel accept a fresh instance of σ from $\mathbf{G}_j$. Hence, $\mathbf{CH}(j, \sigma, i)$ permits reoccurrence of σ (i.e. $\mathbf{G}_j$ sends σ again) only when it is idle, namely at state 0. The capacity constraint of $\mathbf{CH}(j, \sigma, i)$ can be easily relaxed to allow *multi-capacity* channel models, as we shall see in Remark 3.1. We, nevertheless, adopt $\mathbf{CH}(j, \sigma, i)$ for its structural simplicity and suitability for clarifying the concept of delay-robustness presented next.

As displayed in Figure 3, after the delay is introduced, the transmission of σ is replaced by the channel model $\mathbf{CH}(j, \sigma, i)$, and the *channelled behaviour* of the system is described as follows. Suppose given $\mathbf{G}_k$, $k \in \mathcal{N}$; by localisation (see Section 2.2), $\mathbf{G}_k$ acquires a set of local controllers $\mathbf{LOC}^C_k := \{\mathbf{LOC}^C_\alpha | \alpha \in \Sigma_{k,hib}\}$ and a set of local preemptors $\mathbf{LOC}^P_k := \{\mathbf{LOC}^P_\beta | \beta \in \Sigma_{k,for}\}$. [4] So, the local controlled behaviour of $\mathbf{G}_k$ is

$$\mathbf{SUP}_k := \mathbf{G}_k \,||\, \Big(\mathop{||}_{\alpha\in\Sigma_{k,hib}} \mathbf{LOC}^C_\alpha\Big) \,||\, \Big(\mathop{||}_{\beta\in\Sigma_{k,for}} \mathbf{LOC}^P_\beta\Big). \tag{9}$$

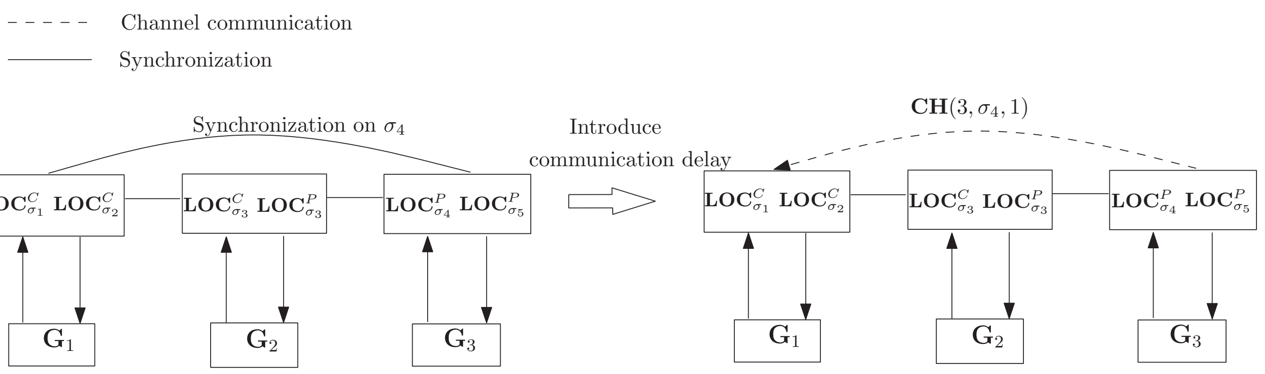
Channel communication
Synchronization
Synchronization on σ_4
Introduce communication delay
$\mathbf{CH}(3,\sigma_4,1)$
$\mathbf{LOC}^C_{\sigma_1}$ $\mathbf{LOC}^C_{\sigma_2}$
$\mathbf{LOC}^C_{\sigma_3}$ $\mathbf{LOC}^P_{\sigma_3}$
$\mathbf{LOC}^P_{\sigma_4}$ $\mathbf{LOC}^P_{\sigma_5}$
$\mathbf{G}_1$
$\mathbf{G}_2$
$\mathbf{G}_3$

Figure 3. Example of distributed control architecture with communication delay: consider the example in Figure 1 and assume that only event σ_4 in $\mathbf{G}_3$ is transmitted to $\mathbf{G}_1$ with unbounded delay. When the communication delay of σ_4 is negligible, $\mathbf{LOC}^C_{\sigma_1}$ and $\mathbf{LOC}^C_{\sigma_2}$ will 'synchronise' on the occurrence of σ_4 in $\mathbf{G}_3$, namely σ_4 is communicated through 'synchronisation'. When the communication delay of σ_4 is unbounded, the communication of σ_4 is modelled by a TDES channel model $\mathbf{CH}(3,\sigma_4,1)$.

As exemplified in Figure 3, after the delay is introduced, the transmission of σ is replaced by the channel model $\mathbf{CH}(j,\sigma,i)$. Observe that when $\mathbf{G}_j$ sends σ to $\mathbf{G}_i$ through $\mathbf{CH}(j,\sigma,i)$, only the recipient $\mathbf{G}_i$'s local behaviour $\mathbf{SUP}_i$ is affected because $\mathbf{G}_i$ receives σ' instead of σ due to delay. Hence, each transition σ of $\mathbf{SUP}_i$ must be replaced by its signal event σ'; we denote by $\mathbf{SUP}'_i$ the resulting new local behaviour of $\mathbf{G}_i$ (the corresponding local controllers are denoted by $\mathbf{LOC}^{C'}_\alpha$ and the local preemptors $\mathbf{LOC}^{P'}_\beta$). Now let

$$\mathbf{NSUP} := \mathbf{SUP}'_i \,||\, (\mathop{||}_{k\in\mathcal{N},k\neq i} \mathbf{SUP}_k) \tag{10}$$

and then

$$\mathbf{SUP}' := \mathbf{NSUP} \,||\, \mathbf{CH}(j,\sigma,i). \tag{11}$$

So, $\mathbf{SUP'}$ is the channelled behaviour of the system with respect to $\mathbf{CH}(j,\sigma,i)$. Note that both $\mathbf{SUP'}$ and $\mathbf{NSUP}$ are defined over $\Sigma' := \Sigma \cup \{\sigma'\}$.

Let P: $\Sigma'^* \to \Sigma^*$ and P_{ch}: $\Sigma'^* \to \{\sigma, tick, \sigma'\}^*$ be natural projections (as in (2)). We define delay-robustness as follows.

Definition 3.1: Consider that $\mathbf{G}_j$ sends event σ to $\mathbf{G}_i$ through channel $\mathbf{CH}(j,\sigma,i)$. The monolithic supervisor $\mathbf{SUP}$ in (4) is *delay-robust* with respect to $\mathbf{CH}(j,\sigma,i)$ if the following conditions hold:

(i) $\mathbf{SUP'}$ in (11) is *correct* and *complete*, i.e.

$$PL(\mathbf{SUP}') = L(\mathbf{SUP}) \tag{12}$$

$$PL_m(\mathbf{SUP}') = L_m(\mathbf{SUP}) \tag{13}$$

$$\begin{aligned}&(\forall s \in \Sigma'^*)(\forall w \in \Sigma^*) s \in L(\mathbf{SUP}')\&(Ps)w\\ &\in L_m(\mathbf{SUP}) \Rightarrow (\exists v \in \Sigma'^*)Pv\\ &= w\&sv \in L_m(\mathbf{SUP}')\end{aligned} \tag{14}$$

(ii) $P_{ch}^{-1}(L(\mathbf{CH}(j,\sigma,i)))$ is controllable with respect to $L(\mathbf{NSUP})$ and $\{\sigma\}$, i.e.

$$\begin{aligned}&P_{ch}^{-1}L(\mathbf{CH}(j,\sigma,i))\{\sigma\}\\ &\cap L(\mathbf{NSUP}) \subseteq P_{ch}^{-1}L(\mathbf{CH}(j,\sigma,i))\end{aligned} \tag{15}$$

In condition (i) above, 'correctness' of $\mathbf{SUP'}$ means that no P-projection of anything $\mathbf{SUP}'$ can do is disallowed by $\mathbf{SUP}$, while 'completeness' means that anything $\mathbf{SUP}$ can do is the P-projection of something $\mathbf{SUP}'$ can do. In this sense, the channelled behaviour $\mathbf{SUP'}$ is 'equivalent' to its delay-free counterpart $\mathbf{SUP}$. Thus, if SUP is determined to be delay-robust with respect to $\mathbf{CH}(j,\sigma,i)$, then by (9)–(11), the composite plant under the control of the local controllers and preemptors with unbounded delay of event σ transmission from $\mathbf{G}_j$ to $\mathbf{G}_i$ is equivalent to $\mathbf{SUP}$. Note that here the local controllers and preemptors are constructed from the monolithic supervisor by the supervisor localisation procedure in Cai and Wonham (2010a, 2015), and we do not design a new supervisor with delay-prone channels from scratch, but to verify delay-robustness of the original supervisor.

Specifically, conditions (12) and (13) state the equality of closed and marked behaviours between $\mathbf{SUP}$ and the P-projection of $\mathbf{SUP}'$; condition (14), which is required for 'completeness', states that if $\mathbf{SUP}'$ executes a string s whose projection Ps in $\mathbf{SUP}$ can be extended by a string w to a marked string of $\mathbf{SUP}$, then $\mathbf{SUP}'$ can further execute a string v whose projection Pv is w and such that *sv* is marked in **SUP**′. Roughly, an *observationally consistent inference* about coreachability at the 'operating' level of **SUP**′ can be drawn from coreachability at the abstract (projected) level of **SUP**.

Condition (ii) of Definition 3.1 imposes a basic requirement that channel **CH**(*j*, σ, *i*), when combined with **NSUP** in (10) to form **SUP′**, should not entail uncontrollability with respect to σ. We impose condition (ii) no matter whether σ is controllable or uncontrollable. This is because we view the channel **CH**(*j*, σ, *i*) as a purely passive adjunction to the original system, and therefore **CH**(*j*, σ, *i*) cannot exercise control on σ. In other words, the channel has to 'accept' any event that the rest of the system might execute, whether that event is controllable or uncontrollable. Otherwise, if there were already an instance of σ in the channel (i.e. **CH**(*j*, σ, *i*) at state 1), then reoccurrence of σ would be (unintentionally) 'blocked', causing condition (ii) to fail. This issue persists, albeit in milder form, even if we use channel models of multiple (finite) capacities (see Remark 3.1 below).

We note that delay-robustness as defined above is an extension, from untimed DES to timed DES, of the concept proposed under the same name in Zhang et al. (2012) . In particular, the channel model **CH**(*j*, σ, *i*) used in the definition is capable of measuring transmission delay by counting *tick* occurrences; and condition (ii) in the definition requires controllability for timed DES.

Finally, we present an algorithm to verify the delay-robustness property. Note that when (12) and (13) hold, then (14) is identical with the L_m(**SUP**′)-*observer* property of *P* (Feng & Wonham, 2008; Wong & Wonham, 2004). The latter may be verified in polynomial time ($O(m^4)$, *m* the state size of **SUP**′) by computing the *supremal quasi-congruence* of a nondeterministic automaton derived from **SUP**′ and *P* (Feng & Wonham, 2010; Wong & Wonham, 2004).[5] The following is the delay-robustness verification algorithm.

Algorithm 1:

(1) Check if *P* is an L_m(**SUP**′)-observer. If not, return *false*.
(2) Check if $PL(\mathbf{SUP'}) = L(\mathbf{SUP})$ and $PL_m(\mathbf{SUP'}) = L_m(\mathbf{SUP})$. If not, return *false*.
(3) Check if $P_{ch}^{-1}(L(\mathbf{CH}(j,\sigma,i)))$ is controllable with respect to L(**NSUP**) and $\{\sigma\}$. If not, return *false*.
(4) Return *true*.

If Step 1 above, ($O(m^4)$ complexity) is successful, i.e. *P* is indeed an L_m(**SUP**′)-observer, then Step 2 of computing $PL(\mathbf{SUP'})$ and $PL_m(\mathbf{SUP'})$ is of polynomial complexity $O(m^4)$ (Feng and Wonham, 2010). Then, checking the two equalities in Step 2 is of $O(m^2)$ complexity. Finally, in Step 3, controllability may be checked using a standard algorithm (Brandin & Wonham, 1994) in linear time $O(m)$. Therefore, Algorithm 1 terminates and is of polynomial complexity $O(m^4)$. Note that if we change the order of Steps 1 and 2, the subset construction algorithm for computing projection on **SUP** is required, and in that case, the complexity of Algorithm 1 would be exponential. The following result is straightforward.

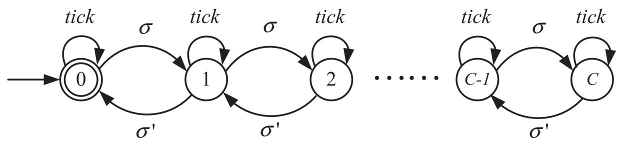

Figure 4. *C*-capacity channel model **NCH**(*j*, σ, *i*).

Proposition 3.1: *Consider that* $\mathbf{G}_j$ *sends event* σ *to* $\mathbf{G}_i$ *through channel* **CH**(*j*, σ, *i*). *The monolithic supervisor* **SUP** *is delay-robust with respect to* **CH**(*j*, σ, *i*) *if and only if Algorithm* 1 *returns true.*

We note here that *coerciveness*, as defined in Wonham (2015, Chapter 9), is preserved when **SUP** is delay-robust. *Coerciveness* says that if *tick* is preempted by one of the local controllers and preemptors, then a forcible event must remain enabled in the plant. According to endnote 4, *tick* is only preempted by a forcible event β at some state *y* of local preemptor $\mathbf{LOC}_\beta^P$, and at state *y*, β is enabled; at any other states, *tick* is defined. After the communication delay is introduced, if *tick* is preempted in **SUP**′, there must exist a local preemptor $\mathbf{LOC}_\beta^P$ and its state *y* such that at *y*, *tick* is preempted by β, and β is enabled, since after the modification on *tick*-selfloop, *tick* will not be prevented by synchronous product. Thus, coerciveness is preserved despite the fact that delay is introduced.

Remark 3.1 (Multi-capacity channel model): So far, we have considered the 1-capacity channel model **CH**(*j*, σ, *i*), and defined delay-robustness with respect to it. We now consider the more general *C*-capacity channel model **NCH**(*j*, σ, *i*), $C \geq 1$ a positive integer, displayed in Figure 4 . The sender $\mathbf{G}_j$ may send at most *C* instances of event σ to **NCH**(*j*, σ, *i*), with each instance subject to indefinite delay. With channel **NCH**(*j*, σ, *i*), one may proceed just as before, by replacing **CH**(*j*, σ, *i*) by **NCH**(*j*, σ, *i*) throughout, to define the corresponding delay-robustness property with respect to **NCH**(*j*, σ, *i*), and then revising Algorithm 1 correspondingly to verify delay-robustness.

It is worth noting that when **NCH**(*j*, σ, *i*) reaches its maximum capacity, and $\mathbf{G}_j$ sends yet another instance of σ, then σ is 'blocked' by **NCH**(*j*, σ, *i*), implying uncontrollability of the channelled behaviour. Hence, the uncontrollability problem always exists as long as the channel model is of finite capacity and delay is indefinite, although the controllability condition (cf. condition (ii)

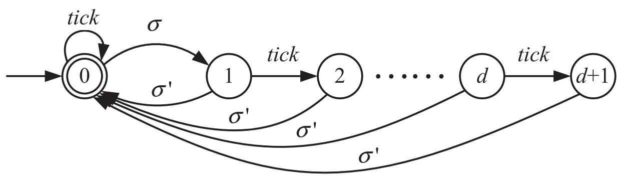

Figure 5. Timed channel model $\mathbf{CH}_d(j, \sigma, i)$, $d \geq 1$, for transmitting event σ from $\mathbf{G}_j$ to $\mathbf{G}_i$ with delay bound d.

of Definition 3.1) is more easily satisfied for larger capacity channels (simply because more instances of σ may be sent to the channel).

4. Bounded delay-robustness and maximal delay bound

Consider again the situation that agent $\mathbf{G}_j$ sends an event σ to $\mathbf{G}_i$. If the monolithic supervisor **SUP** is verified (by Algorithm 1) to be delay-robust, then we will use channel $\mathbf{CH}(j, \sigma, i)$ in Figure 2 to transmit σ subject to unbounded delay, and the system's behaviour will not be affected. If, however, **SUP** fails to be delay-robust, there are two possible implications: (1) σ must be transmitted without delay (as in the original setup of localisation (Cai & Wonham, 2015; Cai et al., 2013; Zhang et al., 2013)); or (2) there exists a delay bound d ($\geq$1) for σ such that if each transmission of σ is completed within d occurrences of *tick*, the system's behaviour will remain unaffected. This section aims to identify the latter case, which we call 'bounded delay-robustness', and, moreover, to determine the bound d.

To that end, consider the channel model $\mathbf{CH}_d(j, \sigma, i)$ in Figure 5, with parameter $d \geq 1$. $\mathbf{CH}_d(j, \sigma, i)$ is a $(d + 2)$-state TDES with event set $\{\sigma, tick, \sigma'\}$. After an occurrence of σ (state 0 to 1), $\mathbf{CH}_d(j, \sigma, i)$ counts up to d ($\geq$0) occurrences of *tick* (state 1 through $d + 1$) by which time the signal event σ' must occur. That is, the occurrence of σ' ($\mathbf{G}_i$ receives σ) is bounded by d *tick*s. Note that the *tick* self-loop at state 0 is again for the technical requirement to prevent the blocking of event *tick* when synchronising $\mathbf{CH}_d(j, \sigma, i)$ with other TDES.

Now with $\mathbf{CH}_d(j, \sigma, i)$, the channelled behaviour of the system is

$$\mathbf{SUP}'_d := \mathbf{NSUP} \,||\, \mathbf{CH}_d(j, \sigma, i) \tag{16}$$

where **NSUP** is given in (10). The event set of $\mathbf{SUP'}_d$ is $\Sigma' = \Sigma \cup \{\sigma'\}$, and we recall the natural projections P: $\Sigma'^* \to \Sigma^*$ and P_{ch}: $\Sigma'^* \to \{\sigma, tick, \sigma'\}^*$.

Definition 4.1: Consider that $\mathbf{G}_j$ sends event σ to $\mathbf{G}_i$ through channel $\mathbf{CH}_d(j, \sigma, i)$, $d \geq 1$. The monolithic supervisor **SUP** in (4) is *bounded delay-robust* with respect to $\mathbf{CH}_d(j, \sigma, i)$ (or *d-bounded delay-robust*) if the following conditions hold:

(i) $\mathbf{SUP'}_d$ in (16) is *correct* and *complete*, i.e.

$$PL(\mathbf{SUP'}_d) = L(\mathbf{SUP}) \tag{17}$$

$$PL_m(\mathbf{SUP'}_d) = L_m(\mathbf{SUP}) \tag{18}$$

$$\begin{aligned}(\forall s \in \Sigma'^*)(\forall w \in \Sigma^*) s \in L(\mathbf{SUP'}_d) \& (Ps)w \\ \in L_m(\mathbf{SUP}) \Rightarrow (\exists v \in \Sigma'^*) Pv \\ = w \& sv \in L_m(\mathbf{SUP'}_d)\end{aligned} \tag{19}$$

(ii) $P_{ch}^{-1}(L(\mathbf{CH}_d(j, \sigma, i)))$ is controllable with respect to $L(\mathbf{NSUP})$ and $\{\sigma\}$, i.e.

$$\begin{aligned}P_{ch}^{-1} L(\mathbf{CH}_d(j, \sigma, i))\{\sigma\} \cap L(\mathbf{NSUP}) \\ \subseteq P_{ch}^{-1} L(\mathbf{CH}_d(j, \sigma, i))\end{aligned} \tag{20}$$

Bounded delay-robustness is defined in the same way as (unbounded) delay-robustness in Definition 3.1, but with respect to the new channel model $\mathbf{CH}_d(j, \sigma, i)$ with delay bound d. As a result, d-bounded delay-robustness may be verified by Algorithm 1 with corresponding modifications. For later reference, we state here the modified algorithm.

Algorithm 2:

(1) Check if P is an $L_m(\mathbf{SUP}'_d)$-observer. If not, return *false*.
(2) Check if $PL(\mathbf{SUP'}_d) = L(\mathbf{SUP})$ and $PL_m(\mathbf{SUP'}_d) = L_m(\mathbf{SUP})$. If not, return *false*.
(3) Check if $P_{ch}^{-1}(L(\mathbf{CH}_d(j, \sigma, i)))$ is controllable with respect to $L(\mathbf{NSUP})$ and $\{\sigma\}$. If not, return *false*.
(4) Return *true*.

We note here that according to endnote 4, the coerciveness is also preserved.

Now, if the monolithic supervisor **SUP** fails to be (unbounded) delay-robust with respect to channel $\mathbf{CH}(j, \sigma, i)$, we would like to verify whether **SUP** is bounded delay-robust with respect to $\mathbf{CH}_d(j, \sigma, i)$ for some $d \geq 1$. In that case, we compute the *maximal* delay bound, i.e. the largest delay (number of *tick*s) that can be tolerated without changing the system's logical behaviour. For this, we need the following lemma.

Lemma 4.1: *Consider that $\mathbf{G}_j$ sends event σ to $\mathbf{G}_i$ through channel $\mathbf{CH}_d(j, \sigma, i)$, $d \geq 1$. If **SUP** is not d-bounded delay-robust, then it is not $(d + 1)$-bounded delay-robust.*

The result of Lemma 4.1 is intuitive: if **SUP** cannot tolerate a σ transmission delay of d, neither can it tolerate a delay $(d + 1)$. By induction, in fact, **SUP** cannot tolerate any delay larger than d. The proof of Lemma 4.1 is in

Appendix 1. This fact suggests the following algorithm for identifying bounded delay-robustness as well as computing the maximal delay bound.

Algorithm 3:[6]

(1) Set $d = 1$.
(2) Check by Algorithm 2 if **SUP** is d-bounded delay-robust relative to channel $\mathbf{CH}_d(j, \sigma, i)$. If not, let $d = d - 1$ and go to Step 3. Otherwise, advance d to $d + 1$ and repeat Step 2.
(3) Output $d_{\max} := d$.

Lemma 4.2: *If* **SUP** *is not delay-robust with respect to* **CH**(j, σ, i)*, then Algorithm 3 terminates in at most $2^{m*}m$ steps, i.e. $d_{max} \le 2^{m*}m$, where m is the state size of* **SUP**$'$ *in* (11).

The proof of Lemma 4.2 is given in Appendix 2. In Algorithm 3, we work upwards starting from the minimal delay $d = 1$. If **SUP** is *not* 1-bounded delay-robust with respect to $\mathbf{CH}_1(j, \sigma, i)$, then by Lemma 4.1, **SUP** is *not* d-bounded delay-robust for any $d > 1$. Therefore, **SUP** is *not* bounded delay-robust and σ must be transmitted without delay. Note that in this case, Algorithm 3 outputs $d_{\max} = 0$.

If **SUP** is 1-bounded delay-robust, we next check if it is 2-bounded delay-robust with respect to $\mathbf{CH}_2(j, \sigma, i)$. If **SUP** fails to be 2-bounded delay-robust, then again by Lemma 4.1, **SUP** fails to be d-bounded delay-robust for any $d > 2$. Hence, **SUP** is bounded delay-robust, with the maximal delay bound $d_{\max} = 1$.

If **SUP** is shown to be 2-bounded delay-robust, the iterative process continues until **SUP** fails to be $(d + 1)$-bounded delay-robust for some $d \ge 2$; this happens in finitely many steps according to Lemma 4.2. Then, **SUP** is bounded delay-robust, with the maximal delay bound $d_{\max} = d$. The following result is immediate.

Proposition 4.1: *Consider that* $\mathbf{G}_j$ *sends event* σ *to* $\mathbf{G}_i$ *through channel* $\mathbf{CH}_d(j, \sigma, i)$, $d \ge 1$. *The monolithic supervisor* **SUP** *is bounded delay-robust with respect to* $\mathbf{CH}_d(j, \sigma, i)$ *if and only if the output d_{max} of Algorithm 3 satisfies $d_{max} > 0$. Moreover, if* **SUP** *is bounded delay-robust, then d_{max} is the maximal delay bound for σ transmission.*

To summarise, when an event σ is sent from $\mathbf{G}_j$ to $\mathbf{G}_i$, we determine unbounded or bounded delay-robustness and choose the corresponding channel as follows.

Algorithm 4:

(1) Check by Algorithm 1 if **SUP** is (unbounded) delay-robust. If so, terminate, set the maximal delay bound $d_{\max} = \infty$, and use channel $\mathbf{CH}(j, \sigma, i)$ in Figure 2.
(2) Check by Algorithm 3 if **SUP** is bounded delay-robust. If so (i.e. $d_{max} \ge 1$), terminate and use channel $\mathbf{CH}_d(j, \sigma, i)$ in Figure 5 with $d = d_{\max}$.

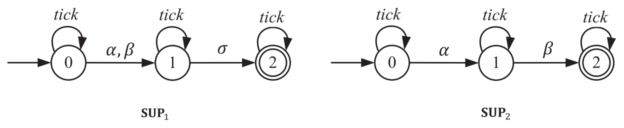

Figure 6. Example 4.1: $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$.

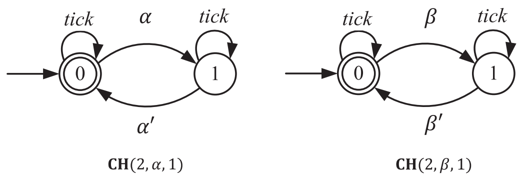

Figure 7. Example 4.1: **CH**$(2, \alpha, 1)$ and **CH**$(2, \beta, 1)$.

(3) In this case, $d_{\max} = 0$. Terminate and use no channel: σ must be transmitted without delay.

Remark 4.1 (Multiple channelled events): So far, we have considered a single event communication: agent $\mathbf{G}_j$ sends event σ to $\mathbf{G}_i$. Using this as a basis, we consider the case that multiple events are transmitted simultaneously. In this case, we build different channel models to transmit each event separately. Note here that our model does not impose any particular order on transmitted events, nor does delay-robustness require that the order of received events be the same as that of the corresponding transmitted events. The following example shows that even if the order of reception, say of events α and β, differs from that of their transmission, **SUP** may still be delay-robust with respect to $\{\alpha, \beta\}$.

Example 4.1: Let $\mathbf{SUP}_1$ and $\mathbf{SUP}_2$ be the generators shown in Figure 6 (here we do not consider the preemption of *tick*); assume events α and β in $\mathbf{SUP}_2$ are exported to $\mathbf{SUP}_1$ and both controllable. First, we create channels **CH**$(2, \alpha, 1)$ and **CH**$(2, \beta, 1)$, as shown in Figure 7, to transmit event α and β, respectively. Then, replacing all instances of α and β by α' and β' correspondingly, we obtain $\mathbf{SUP}_1'$. Finally, with **CH**$(2, \alpha, 1)$ and **CH**$(2, \beta, 1)$ hard-wired into the system, the overall system behaviour is represented by **SUP**$'$, as shown in Figure 8. Defining natural projection P: $\{\alpha, \beta, \sigma, tick, \alpha', \beta'\}^* \to \{\alpha, \beta, \sigma, tick\}^*$, we can verify that conditions (12), (13) and (14) hold for **SUP**, P and **SUP**$'$, and condition (15) holds for both **CH**$(2, \alpha, 1)$ and **CH**$(2, \beta, 1)$ (events α and β are both controllable). By inspection of Figure 8, we find that in string $s = \alpha\beta\beta'\alpha'\sigma \in L(\mathbf{SUP}')$, the order of arrivals of α and β differ from the order of their occurrence, without affecting the delay-robustness of **SUP**.

In the following, we present an approach to the general case of multiple channelled events, as is common in distributed control, such that the overall system behaviour with all the delays is optimal and nonblocking. We will

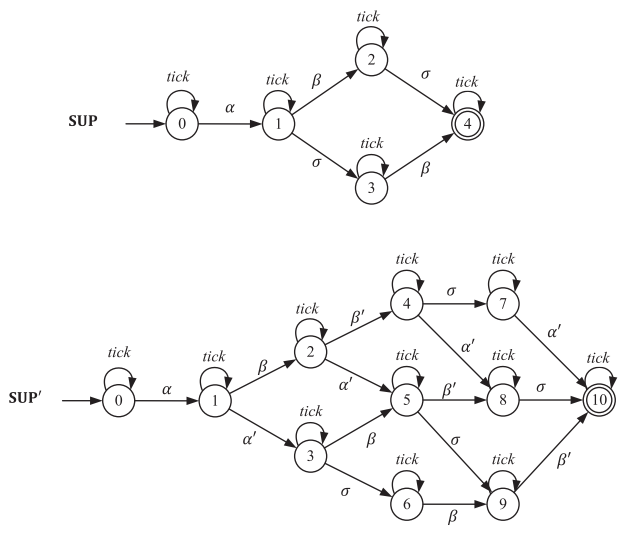

Figure 8. Example 4.1: **SUP** and **SUP**′.

consider that each fixed triple (sender, channelled event, receiver) is assigned with its own communication channel, and the assigned channels operate concurrently. Our goal is to obtain these channels, ensuring unbounded or bounded delay-robustness, one for each triple (sender, channelled event, receiver).

First fix $i, j \in \mathcal{N}$, and recall from (8) that $\Sigma_{j,com,i}$ is the set of events that $\mathbf{G}_j$ sends to $\mathbf{G}_i$. Write $\Sigma_{j,com,i} = \{\sigma_1, \ldots, \sigma_r\}$, $r \geq 1$, and treat the channelled events σ_1, σ_2,...*sequentially*, in order of indexing.

Algorithm 5:

(1) Set $p = 1$.
(2) For event $\sigma_p \in \Sigma_{j,com,i}$ apply Algorithm 4 to obtain the maximal delay bound $d_{\max}$.
(2.1) If $d_{\max} = \infty$, the case of unbounded delay-robustness, choose channel $\mathbf{CH}(j, \sigma_p, i)$, and let $\mathbf{NSUP} := \mathbf{NSUP}||\mathbf{CH}(j, \sigma_p, i)$.
(2.2) If $d_{\max} \geq 1$ is finite, the case of bounded delay-robustness, choose channel $\mathbf{CH}_d(j, \sigma_p, i)$, and let $\mathbf{NSUP} := \mathbf{NSUP}||\mathbf{CH}_d(j, \sigma_p, i)$.
(2.3) If $d_{\max} = 0$, then no channel is chosen and σ_p must be transmitted without delay.
(1) If $p < r$, advance p to $p + 1$ and repeat Step 2.
(3) Output a set of channels used for sending events from $\mathbf{G}_j$ to $\mathbf{G}_i$.

Note that at Step 2 of Algorithm 5, if a channel is chosen for event σ_p, then **NSUP** must be reset to be the synchronous product of **NSUP** and the channel, so that in choosing a channel for the next event σ_{p+1}, the previously chosen channel is considered together. This ensures that when the derived channels operate concurrently, the system's behaviour is not affected. It is worth noting that a different ordering of the set $\Sigma_{j,com,i}$ may result in a different set of channels; if no priority of transmission delay

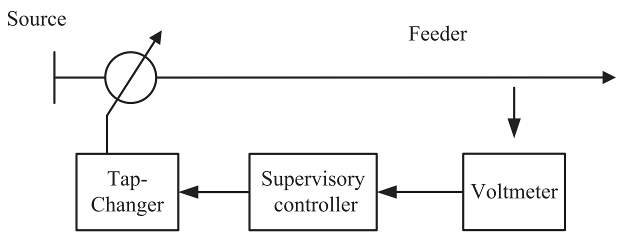

Figure 9. ULTC: components and controller.

is imposed on the communication events, we may choose an ordering randomly.

Finally, since the set of all communication events is $\Sigma_{com} := \bigcup_{i,j \in \mathcal{N}} \Sigma_{j,com,i}$, we simply apply Algorithm 5 for each (ordered) pair $i, j \in \mathcal{N}$ to derive all communication channels. Again, a different ordering of the set $\mathcal{N} \times \mathcal{N}$ generally results in a different set of channels, because the channels chosen for a pair (i, j) will be used to decide channels for all subsequent (i', j'). For convenience, we will simply order the pairs (i, j) sequentially first on j, then on i.

We note that to verify delay-robustness in Algorithm 5, **SUP**′ becomes more complex when more channels are introduced. The computation may be expensive when there are a large number of communication channels. Nevertheless, **SUP**′ is implemented in a purely distributed fashion: distributed supervisors and communication channels. We shall investigate the computational issue for **SUP**′ in future work, one promising approach being to use *State Tree Structures* (Ma & Wonham, 2005).

5. Case study: under-load tap-changing transformer

In this section, we demonstrate timed delay-robustness and associated verification algorithms on an under-load tap-changing transformer (ULTC) system.

5.1 Model description and supervisor localisation

Transformers with tap-changing facilities constitute an important means of controlling voltage at all levels throughout electrical power systems. We consider a ULTC as displayed in Figure 9, which consists of two components: voltmeter and tap-changer (Afzalian, Saadatpoor, & Wonham, 2008).

This ULTC is operated in two modes: automatic and manual. In the automatic mode, the tap-changer works according to the following logic. (1) If the voltage deviation is greater than some threshold value, then a timer will start; when the timer times out, a 'tap increase (or decrease) event' will occur and the timer will reset; a tap increase or decrease should only occur if the voltage

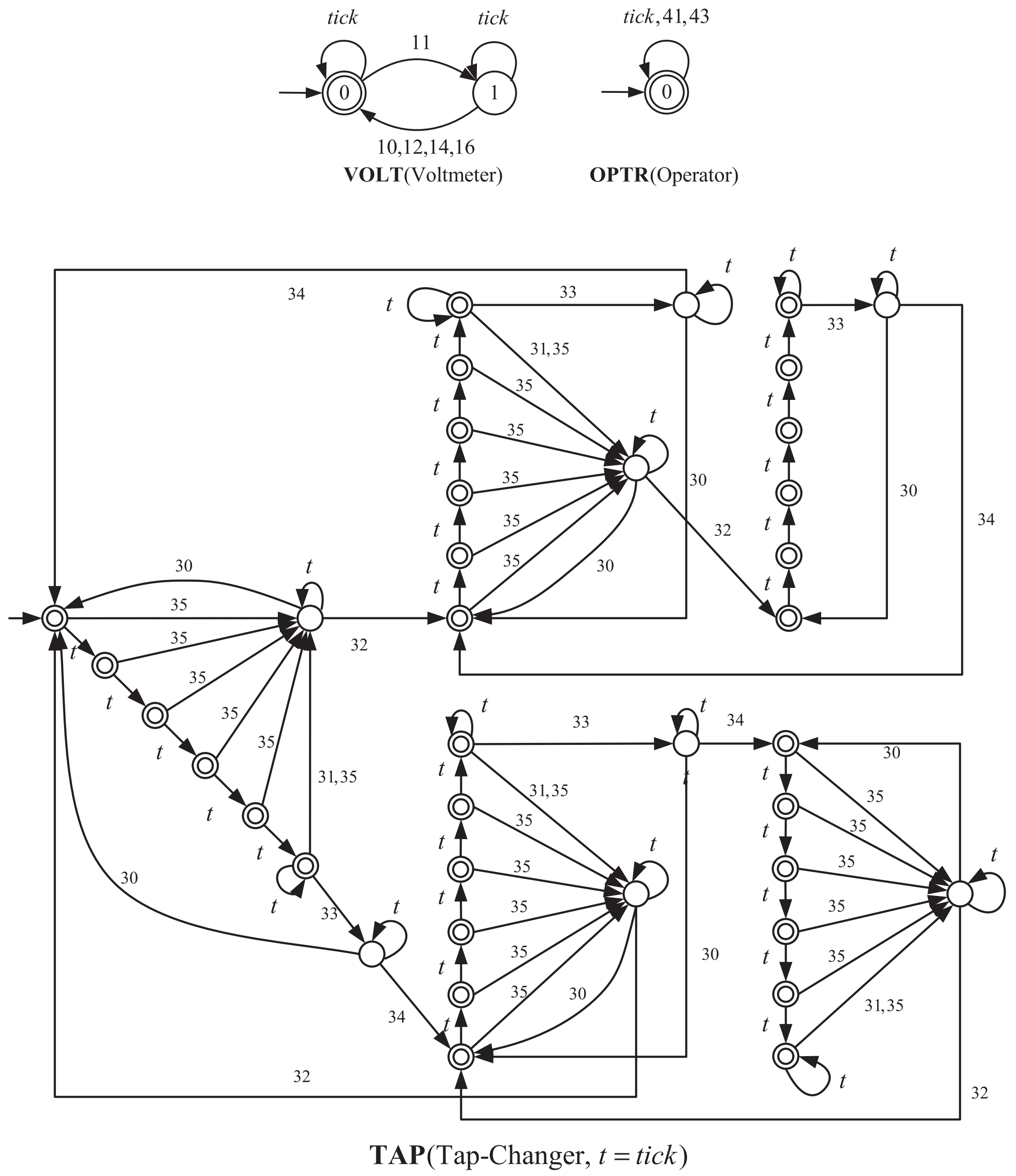

Figure 10. Timed transition graph of ULTC components.

change continues to exceed threshold after the time out – this is to avoid tap changes in response to merely occasional random fluctuations of brief duration. (2) If the voltage returns to the dead-band, because of a tap change or some other reason, then no tap change will occur. (3) If the voltage exceeds the maximally allowed value $V_{\max}$, then lowering of the tap command without delay occurs instantaneously. In the manual mode, the system is waiting for 'tap-up', 'tap-down', or 'automatic' commands. An operator can change the operation mode from one to the other, and thus the operator is included with the plant components to be controlled.

Each plant component is modelled as a TDES displayed in Figure 10, and associated events are listed in Table 1. The plant to be controlled is the synchronised behaviour of voltmeter (**VOLT**), tap-changer (**TAP**) and operator (**OPTR**), i.e.

$$\mathbf{PLANT} = \mathbf{VOLT}||\mathbf{TAP}||\mathbf{OPTR}. \qquad (21)$$

We consider a voltage control problem of the ULTC: when the voltage is not 'normal', design controllers to recover the voltage through controlling tap ratio after a time delay to recover the voltage. Figure 11 displays the TDES model **SPEC** for the control specification in automatic/manual mode.

Table 1 Physical interpretation of events.

Event	Physical interpretation	Time bounds (lower, upper)	(hib/for)
11	Initialise voltmeter	$(0, \infty)$	hib
10	Report $\lvert\Delta V\rvert > ID$ and $\Delta V > 0$	$(0, \infty)$	
12	Report $\lvert\Delta V\rvert < ID$, i.e. voltage recovered	$(0, \infty)$	
14	Report $\lvert\Delta V\rvert > ID$ and $\Delta V < 0$	$(0, \infty)$	
16	Report voltage exceeds V_{max}	$(0, \infty)$	
30	Tap-up/down failed	$(0, \infty)$	
31	Tap-down command with 5 *tick* delay	$(5, \infty)$	hib & for
32	Tap-down successful	$(0, \infty)$	
33	Tap-up command with 5 *tick* delay	$(5, \infty)$	hib & for
34	Tap-up successful	$(0, \infty)$	
35	Tap-down command without delay	$(0, \infty)$	hib & for
41	Enter automatic mode	$(0, \infty)$	hib
43	Enter manual mode	$(0, \infty)$	hib

Note that since the tap increase (decrease) and lowering tap commands would preempt the occurrence of *tick*, the corresponding events 31, 33 and 35 are designated as forcible events. In the following, we synthesise the monolithic supervisor **SUP** by the standard TDES supervisory control theory (Brandin & Wonham, 1994; Wonham, 2015) and the local controllers by TDES supervisor localisation (Cai et al., 2013; Zhang et al., 2013).

First, synthesise the monolithic supervisor TDES **SUP** in the usual sense that its marked behaviour

$$L_m(\mathbf{SUP}) = Sup\mathcal{C}(L_m(\mathbf{SPEC}) \cap L_m(\mathbf{PLANT})) \qquad (22)$$

and its closed behaviour $L(\mathbf{SUP}) = \overline{L_m(\mathbf{SUP})}$. **SUP** has 231 states and 543 transitions, and embodies disabling actions for all the prohibitible events and preempting actions relative to *tick* for all the forcible events.

Next, by the supervisor localisation, we obtain a set of local controllers $\mathbf{LOC}^C_{11}$, $\mathbf{LOC}^C_{31}$, $\mathbf{LOC}^C_{33}$, $\mathbf{LOC}^C_{35}$ $\mathbf{LOC}^C_{41}$ and $\mathbf{LOC}^C_{43}$ for controllable events 11, 31, 33, 35, 41 and 43, respectively, and a set of local preemptors $\mathbf{LOC}^P_{31}$, $\mathbf{LOC}^P_{33}$ and $\mathbf{LOC}^P_{35}$ for forcible events 31, 33 and 35, respectively; their transition diagrams are shown in Figure 12.

Here, the local controllers and local preemptors are coincidentally described by the same automaton. However, in general, there is no particular relationship between $\mathbf{LOC}^c_\sigma$ and $\mathbf{LOC}^P_\sigma$ for the prohibitible and forcible event σ; an illustrative example can be found in Section 5 in Zhang et al. (2013).

Finally, using these constructed local controllers/preemptors, we build a distributed control architecture without communication delays for ULTC as displayed in Figure 13. It is guaranteed by supervisor localisation of TDES (Cai et al., 2013; Zhang et al., 2013) that the ULTC under the control of these local controllers and preemptors without communication delay has closed and marked behaviour identical to **SUP** in (22).

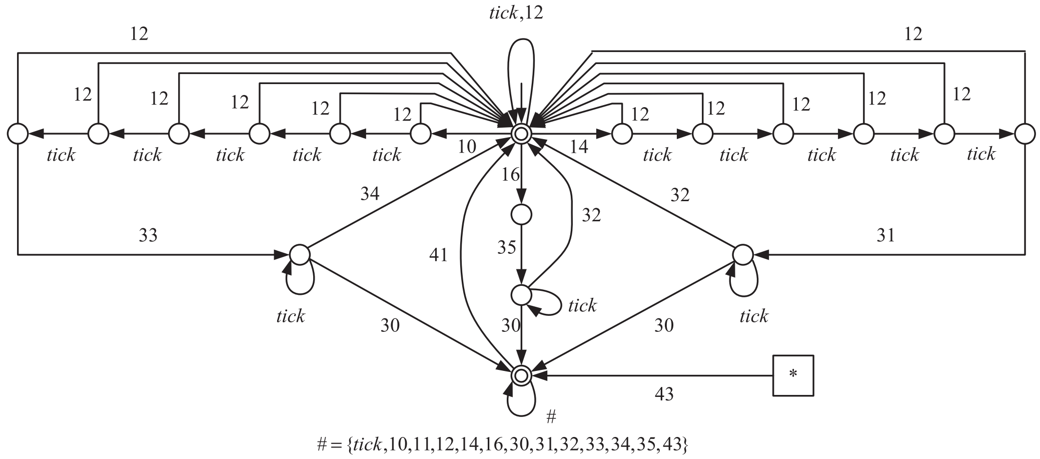

Figure 11. Control specification **SPEC** in automatic/manual mode. The transition 43 from the square with '*' represents similar transitions from all states to the 'manual operation mode'.

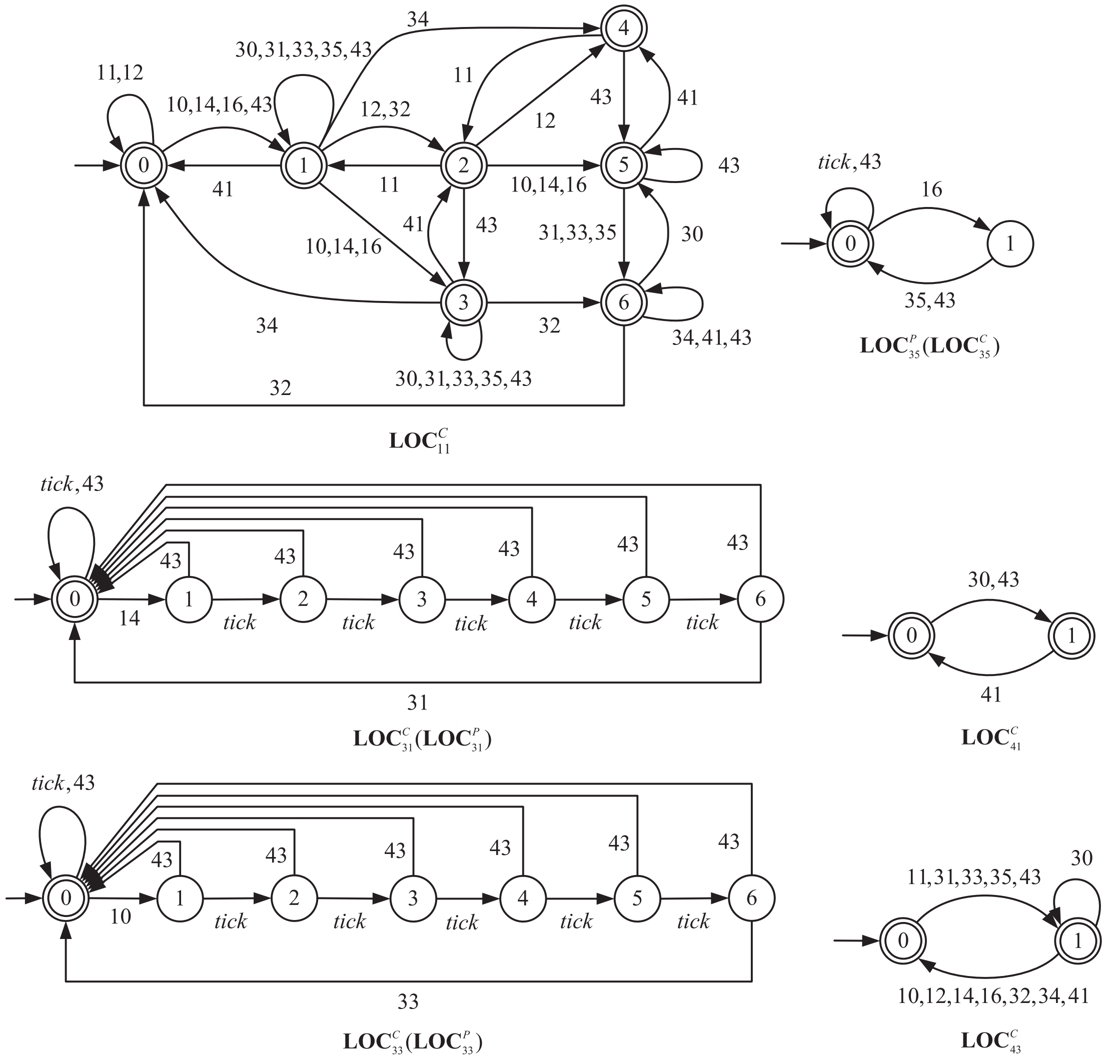
LOC$^C_{11}$
LOC$^P_{35}$(LOC$^C_{35}$)
LOC$^C_{31}$(LOC$^P_{31}$)
LOC$^C_{41}$
LOC$^C_{33}$(LOC$^P_{33}$)
LOC$^C_{43}$
tick, 43
tick

Figure 12. Local controllers and local preemptors for ULTC. We add necessary self-loops of communication events and event *tick* to the local controllers and preemptors according to endnote 4. Let $*(x)$ be the set of events whose self-loops need to be added at state x. In $\mathbf{LOC}^C_{11}$, $*(0) = *(2) = *(4) = \{tick, 30, 31, 32, 33, 34, 35, 41\}$, $*(1) = \{tick, 43\}$, $*(5) = \{tick, 30, 32, 34\}$, and $*(6) = \{tick, 31, 33, 35\}$; in $\mathbf{LOC}^C_{31}$, $*(1) = *(2) = *(3) = *(4) = *(5) = \{14\}$ and $*(6) = \{tick, 14\}$; in $\mathbf{LOC}^P_{31}$, $*(1) = *(2) = *(3) = *(4) = *(5) = *(6) = \{14\}$; in $\mathbf{LOC}^C_{33}$, $*(1) = *(2) = *(3) = *(4) = *(5) = \{10\}$ and $*(6) = \{tick, 10\}$; in $\mathbf{LOC}^P_{33}$, $*(1) = *(2) = *(3) = *(4) = *(5) = *(6) = \{10\}$; in $\mathbf{LOC}^C_{35}$, $*(1) = \{tick, 16\}$; in $\mathbf{LOC}^P_{35}$, $*(1) = \{16\}$; in $\mathbf{LOC}^C_{41}$, $*(0) = \{tick\}$ $*(1) = \{tick, 30\}$; in $\mathbf{LOC}^C_{43}$, $*(0) = \{tick, 10, 12, 14, 16, 30, 32, 34\}$, and $*(1) = \{tick, 11, 31, 33, 35\}$.

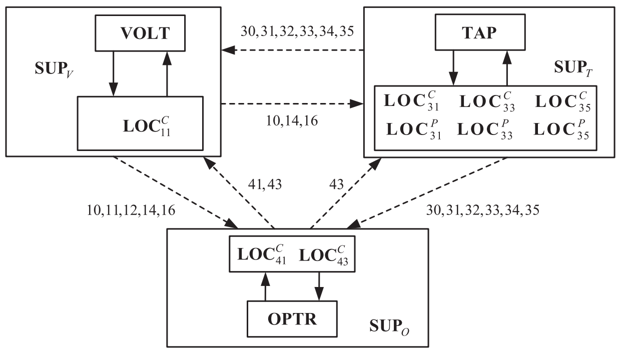

Figure 13. Distributed control architecture of ULTC: the labels on the dashed arrows denote the events to be communicated.

5.2 Delay-robustness verification

Now, we investigate the timed delay-robustness property for ULTC. For illustration, we consider the following three cases.

(1) Event 30 in $\Sigma_{T,com,O}$

Applying Algorithm 4, at Step 1, we verify by Algorithm 1 that **SUP** is delay-robust with respect to the communication channel **CH**$(T, 30, O)$ transmitting event 30, as displayed in Figure 14.

For illustration, consider the case that the voltmeter reported an increase in voltage (in **VOLT** as displayed in Figure 10, events 11 and 10 have occurred), and

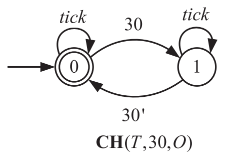

Figure 14. Communication Channel **CH**(T, 30, O).

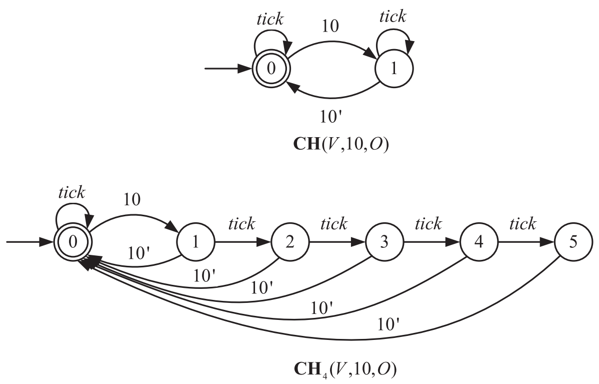

Figure 15. Communication Channels **CH**(V, 10, O) and **CH**$_4$(V, 10, O).

the tap has received a tap-up command, but the tap-up operation failed (in **TAP** as displayed in Figure 10, events *tick*, *tick*, *tick*, *tick*, *tick*, 33 and 30 have occurred in sequence). The events that are eligible to occur are 11, 35, 41, 43, and *tick*. However, (1) $\mathbf{LOC}^C_{11}$ disables event 11; (2) $\mathbf{LOC}^C_{35}$ disables event 35; (3) $\mathbf{LOC}^C_{41}$ will disable or enable event 41 depending on the communication delay of event 30; (4) $\mathbf{LOC}^C_{43}$ disables event 43; (5) *tick* will not be preempted, since no forcible event is enabled. If 30 is transmitted instantly, event 41 is enabled by $\mathbf{LOC}^C_{41}$ and the system will enter the automatic mode. If the transmission of 30 is delayed, only event *tick* is enabled, and other events will not be enabled until the system enters the automatic mode. However, according to the transition diagram of $\mathbf{LOC}^C_{41}$ displayed in Figure 12, only after $\mathbf{LOC}^C_{41}$ has received the occurrence of event 30, will it enable event 41, and bring the system into the automatic mode. Hence, the overall system behaviour will not be affected even if the communication of event 30 is delayed.

(2) Event 10 in $\Sigma_{V,\, com,\, O}$

Applying Algorithm 4, at Step 1, we verify by Algorithm 1 that **SUP** fails to be delay-robust with respect to the channel **CH**(V, 10, O), as displayed in Figure 15; then at Step 2, we check by Algorithm 3 that the maximal delay bound for event 10 is 4, i.e. **SUP** is bounded delay-robust with respect to the channel **CH**$_4$(V, 10, O), as displayed in Figure 15.

For illustration, consider the case that an increase in the voltage is reported (i.e. events 11 and 10 in **VOLT** have occurred sequentially). The events that are eligible to occur are 11, 35, 41, 43, and *tick*. And if **OPTR** knows the voltage increase before the fifth *tick* occurs, the tap-changer will generate a tap-up command and the operator can switch the system into manual mode; otherwise, the tap-changer will also generate a tap-up command, but the system cannot enter the manual mode. So, event 43 will be enabled after the event sequence s ≔ 11.10.*tick*.*tick*.*tick*.*tick*.*tick*.310.33 (where event 310 is the signal event of 10), but is disabled after s' ≔ 11.10.*tick*.*tick*.*tick*.*tick*.*tick*.33. s and s' cannot be distinguished under the projection P that erases the signal event 310. However, the system can enter the manual mode after the sequence s, but not after s'. Namely, the system cannot 'complete' the behaviour of entering manual mode after s', but this behaviour can be finished in its delay-free counterpart **SUP**. So, the observer property (19) required by bounded delay-robustness is violated when the delay bound d exceeds 4 *tick*s, and we conclude that the maximal delay bound for event 10 is 4.

Similarly, one can verify by Algorithm 4 that **SUP** is bounded delay-robust with respect to **CH**$_4$(V, 14, O), as displayed in Figure 16, and any other events except 10, 14 and 30 must be transmitted without delay.

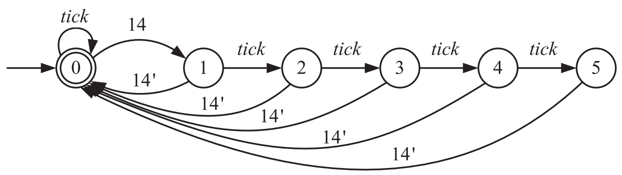

Figure 16. Communication Channel **CH**$_4$(V, 14, O).

(3) All communication events

Applying Algorithm 5 to each of the sets of communication events displayed in Figure 13 in sequence, we obtain that $d'_{\max}(T, 30, O) = \infty$, $d'_{\max}(V, 10, O) = d'_{\max}(V, 14, O) = 4$, and for the remaining events, $d'_{\max} = 0$. In the following, we verify that if all the communication events are communicated within their corresponding delay bounds, the overall system behaviour will still not be affected.

First, use **CH**(T, 30, O), **CH**$_4$(V, 10, O) and **CH**$_4$(V, 14, O) to transmit events 30, 10 and 14 respectively. Second, connected by these channels, the overall system behaviour is

$$\begin{aligned}\mathbf{SUP}'_{com} = {}& \mathbf{SUP}_V||\mathbf{SUP}_T||\mathbf{CH}_4(V, 10, O)|| \\ & \mathbf{CH}_4(V, 14, O)||\mathbf{CH}(T, 30, O)||\mathbf{SUP}'''_O)\end{aligned}$$

over the augmented alphabet $\{10, 11, \ldots, 43, 10', 14', 30'\}$, where $\mathbf{SUP}'''_O$ is obtained by replacing 10, 14, and 30 by $10'$, $14'$ and $30'$, respectively. Third, one can verify that: (1) $\mathbf{SUP}'_{com}$ is correct and complete, and (2) **CH**(T, 30, O),

$\mathbf{CH}_4(V, 10, O)$ and $\mathbf{CH}_4(V, 14, O)$ will not cause uncontrollability with respect to the uncontrollable communication events. Finally, we conclude that the overall system behaviour is still optimal and nonblocking.

6. Conclusions

In this paper, we have studied communication delays among local controllers obtained by supervisor localisation in TDES. First, we have identified properties of 'timed delay-robustness' which guarantee that the specification of our delay-free distributed control continues to be enforced in the presence of (unbounded) delay, and presented a verification algorithm to determine delay-robustness. Second, for those events that fail to be delay-robust, we have proposed an algorithm to determine their maximal delay bound $d_{\max}$ such that the system is $d_{\max}$-bounded delay-robust. Finally, a ULTC example has exemplified these results, showing how to verify delay-robustness; in addition, we obtained a set of maximal delay bounds, one for each communication event, under the condition that the overall system behaviour is still optimal and nonblocking.

When the system and its supervisor are of large-scale, the verification procedures of the delay-robustness may well face computational difficulty. To overcome this difficulty, we aim in our future work to study delay-robustness of large-scale systems from the following two approaches. The first approach is to base our verification procedures on State Tree Structures (Ma & Wonham, 2005), which has been applied to DES to synthesise the monolithic supervisor efficiently. The second approach is to employ the decentralised control architecture. Instead of computing the monolithic supervisor, we first synthesise a set of decentralised supervisors to achieve global optimality and nonblocking (Cai & Wonham, 2010b; Feng & Wonham, 2008), and then verify delay-robustness of each decentralised supervisor. By this reasoning, the overall system is delay-robust if all the decentralised supervisors are delay-robust.

Notes

1. We remark that if the system and its supervisor are of large scale, we first synthesise a set of decentralised supervisors to achieve global optimality and nonblocking, and then apply supervisor localisation to decompose each decentralised supervisor in the set (as in Cai & Wonham, 2010a, 2015). Other decentralised/distributed control schemes, e.g. coordination technique (Komenda, Masopust, & van Schuppen, 2014 and Seow, Pham, Ma, & Yokoo, 2009), are also candidates for computing the decentralised/distributed supervisors.
2. We refer to any DES over an alphabet which includes *tick* as a generalised TDES; it need not be a (strict) TDES constructed according to the rules in Wonham (2015, Section 9.2). Generalised TDES are needed to model specifications and supervisors (Wonham, 2015).
3. The closed and marked behaviours of $\mathbf{TDES} = \mathbf{TDES1} \,||\, \mathbf{TDES2}$ are $L(\mathbf{TDES}) = L(\mathbf{TDES1}) \,||\, L(\mathbf{TDES2})$ and $L_m(\mathbf{TDES}) = L_m(\mathbf{TDES1}) \,||\, L_m(\mathbf{TDES2})$, where $||$ denotes the synchronous product of languages (Wonham, 2015). There are three cases where we use synchronous product to compute the concurrent behaviour of TDES. (1) The TDES that describe plant components share only event 'tick' in this case, synchronous product is equivalent to composition (Wonham, 2015). (2) We compute the plant behaviour restricted by a generalised TDES; in this case, the TDES synchronises with the plant to form their concurrent behaviour; it is not composed with the plant to generate a larger system. (3) We compute the concurrent behaviour of generalised TDES.
4. At each state x of each local controller $\mathbf{LOC}^C_\alpha$, if a communication event $\sigma \in \Sigma_\alpha - \Sigma_k$ is not defined, we add a σ-self-loop, i.e. transition (x, σ, x) to $\mathbf{LOC}^C_\alpha$; if *tick* is not defined at x, we add a *tick*-self-loop, i.e. transition $(x, tick, x)$ to $\mathbf{LOC}^C_\alpha$. At each state y of each preemptor $\mathbf{LOC}^P_\beta$, if communication event $\sigma \in \Sigma_\beta - \Sigma_k$ is not defined, we add a σ-self-loop, i.e. transition (y, σ, y) to $\mathbf{LOC}^P_\beta$; if *tick* is not defined and not preempted, we add a *tick*-self-loop, i.e. transition $(y, tick, y)$ to $\mathbf{LOC}^P_\beta$. With this modification, (1) the new local controllers $\mathbf{LOC}^C_\alpha$ (resp. local preemptors $\mathbf{LOC}^P_\beta$) are also control equivalent to SUP (because $\mathbf{LOC}^C_\alpha$ (resp. $\mathbf{LOC}^P_\beta$) does not disable events σ from other components $\mathbf{G}_j$; (2) the definition of σ at every state of $\mathbf{LOC}^C_\alpha$ (resp. $\mathbf{LOC}^P_\beta$) is consistent with the assumption that $\mathbf{LOC}^C_\alpha$ (resp. $\mathbf{LOC}^P_\beta$) may receive σ after indefinite communication delay; and (3) *tick* is only preempted by forcible events, say β, at some states of the corresponding local preemptor $\mathbf{LOC}^P_\beta$, and at those states, β is enabled (otherwise, *tick* is defined).
5. We note *en passant* that Bravo, da Cunha, Pena, Malik, and Cury (2012) reports an algorithm with quadratic time complexity for verifying the observer property alone; that does not, however, yield structural information which (if the observer property is not satisfied) might be useful for remedial design.
6. The algorithm can be transformed into a binary search version, which can reduce the complexity. Here, the linear search version we used is directly derived from Lemma 4.1 and is easier to follow.
7. The concept 'simple string' is derived from the 'simple path' in graph theory, where a path is called *simple* if no vertex is traversed more than once (Danielson, 1968). Here, string t is called *simple* if no state is traversed more than once.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was partially supported by the National Nature Science Foundation of China [grant number 61403308]; the Fundamental Research Funds for the Central Universities, China [grant number 3102014JCQ01069]; the Program to Disseminate Tenure Tracking System, MEXT, Japan; the Natural Sciences and Engineering Research Council, Canada [grant number 7399].

ORCID

Renyuan Zhang http://orcid.org/0000-0002-4142-5166

References

Afzalian, A., Saadatpoor, A., & Wonham, W. (2008). Systematic supervisory control solutions for under-load tap-changing transformers. *Control Engineering Practice, 16*, 1035–1054.

Barrett, G., & Lafortune, S. (2000). Decentralized supervisory control with communicating controllers. *IEEE Transactions on Automatic Control, 45*(9), 1620–1638.

Brandin, B., & Wonham, W. (1994). Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control, 39*(2), 329–342.

Bravo, H., da Cunha, A., Pena, P., Malik, R., & Cury, J. (2012). *Generalised verification of observer property in discrete event systems*. Proceedings of 11th international workshop on discrete event systems (WODES2012) (pp. 337–342), Guadalajara, Mexico.

Cai, K., & Wonham, W.M. (2010a). Supervisor localization: A top-down approach to distributed control of discrete-event systems. *IEEE Transactions on Automatic Control, 55*(3), 605–618.

Cai, K., & Wonham, W. (2010b). Supervisor localization for large discrete-event systems: Case study production cell. *International Journal of Advanced Manufacturing Technology, 50*(9–12), 1189–1202.

Cai, K., & Wonham, W.M. (2015). *Supervisor localization: A top-down approach to distributed control of discrete-event systems*. Lecture Notes in Control and Information Sciences (Vol. 459). Switzerland: Springer.

Cai, K., Zhang, R., & Wonham, W. (2013). *Supervision localization of timed discrete-event systems*. Proceedings of 2013 American Control conference (pp. 5666–5671), Washington, DC.

Danielson, G. (1968). On finding simple paths and circuits in a graph. *IEEE Transactions on Circuit Theory, 15*(3), 294–295.

Darondeau, P., & Ricker, L. (2012). Distributed control of discrete-event systems: A first step. *Transactions on Petri Nets and Other Models of Concurrency, 6*, 24–45.

Feng, L., & Wonham, W.M. (2008). Supervisory control architecture for discrete-event systems. *IEEE Transactions on Automatic Control, 53*(6), 1449–1461.

Feng, L., & Wonham, W. (2010). On the computation of natural observers in discrete-event systems. *Discrete Event Dynamic Systems, 20*(1), 63–102.

Hiraishi, K. (2009). On solvability of a decentralized supervisory control problem with communication. *IEEE Transactions on Automatic Control, 54*(3), 468–480.

Kalyon, G., Gall, T.L., Marchand, H., & Massart, T. (2011). *Synthesis of communicating controllers for distributed systems*. Proceedings of 2011 50th IEEE conference on Decision and Control and European Control conference (CDC-ECC), Orlando, FL, USA.

Komenda, J., Masopust, T., & van Schuppen, J. (2014). Coordination control of discrete-event systems revisited. *Discrete Event Dynamic Systems, 25*(1–2), 65–94.

Lin, F. (2014). Control of networked discrete event systems: Dealing with communication delays and losses. *SIAM Journal on Control and Optimization, 52*(2), 1276–1298.

Ma, C., & Wonham, W.M. (2005). *Nonblocking supervisory control of state tree structures*. Berlin: Springer-Verlag.

Sadid, W., 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.

Schmidt, K., Schmidt, E., & Zaddach, J. (2007). *A shared-medium communication architecture for distributed discrete event systems*. Proceedings of Mediterranean conference on Control and Automation (pp. 1–6), Athens, Greece.

Seow, K.T., Pham, M.T., Ma, C., & Yokoo, M. (2009). Coordination planning: Applying control synthesis methods for a class of distributed agents. *IEEE Transactions on Control Systems Technology, 17*(2), 405–415.

Takai, S., & Ushio, T. (2003). Effective computation of Lm(g)-closed, controllable, and observable sublanguage arising in supervisory control. *Systems & Control Letters, 49*(3), 191–200.

Tripakis, S. (2004). Decentralized control of discrete-event systems with bounded or unbounded delay communication. *IEEE Transactions on Automatic Control, 49*(9), 1489–1501.

Wong, K., & Wonham, W. (2004). On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems, 14*(1), 55–107.

Wonham, W. (2015). *Supervisory control of discrete-event systems*. Toronto: Systems Control Group, ECE Department, University of Toronto. Retrieved from http://www.control.utoronto.ca/DES

Xu, S., & Kumar, R. (2008). *Asynchronous implementation of synchronous discrete event control*. Proceedings of 9th international workshop on Discrete Event Systems (WODES'08) (pp. 181–186), Göteborg, Sweden.

Zhang, R., Cai, K., Gan, Y., Wang, Z., & Wonham, W. (2012). *Checking delay-robustness of distributed supervisors of discrete-event systems*. Proceedings of international conference on Information Science and Control Engineering (pp. 350–355), Shenzhen, China.

Zhang, R., Cai, K., Gan, Y., Wang, Z., & Wonham, W. (2013). Supervision localization of timed discrete-event systems. *Automatica, 49*(9), 2786–2794.

Zhang, R., Cai, K., Gan, Y., Wang, Z., & Wonham, W. (2015). Distributed supervisory control of discrete-event systems with communication delay. *Discrete Event Dynamic Systems*. doi:10.1007/s10626-014-0208-4

Zhang, R., Cai, K., & Wonham, W. (2014). *Delay-robustness in distributed control of timed discrete-event systems based on supervisor localization*. Proceedings of 53rd IEEE conference on Decision and Control (pp. 6719–6724), Los Angeles, CA.

Appendices

Appendix 1. Proof of Lemma 4.1

To prove Lemma 4.1, we need the following Lemmas A.1 and A.2.

Lemma A.1: *For any delay bound $d \geq 1$, there hold*

$$L(\mathbf{SUP}) \subseteq PL(\mathbf{SUP}'_d) \tag{A1}$$

$$L_m(\mathbf{SUP}) \subseteq PL_m(\mathbf{SUP}'_d) \tag{A2}$$

Proof: Note that for different delay bounds d, the alphabets of $\mathbf{SUP}'_d$ and $\mathbf{CH}_d(j, \sigma, i)$ are $\Sigma' = \Sigma \cup \{\sigma'\}$ and $\{\sigma, tick, \sigma'\}$, respectively. Here, we only prove that $L(\mathbf{SUP}) \subseteq PL(\mathbf{SUP}'_d)$; (A2) can be proved in the same way by replacing L by L_m throughout.

Let $s \in L(\mathbf{SUP})$; we must show that there exists a string $t \in L(\mathbf{SUP}'_d)$ such that $P(t) = s$. We first consider that only one instance of σ appears in s, and write $s = x_1\sigma x_2$, where x_1, x_2 are free of σ. By (16) and observing that $\mathbf{SUP}'_i$ is obtained by replacing each instance of σ by σ', we obtain that $t := x_1\sigma\sigma' x_2 \in L(\mathbf{SUP}'_d)$. Furthermore, $P(t) = s$. So, $L(\mathbf{SUP}) \subseteq PL(\mathbf{SUP}'_d)$. This result can be easily extended to the general case where s has multiple instances of σ, because σ is transmitted by the channel model and the reoccurrence of σ is permitted only when transmission of the previous σ is completed. Namely, if $s = x_1\sigma x_2\sigma \ldots, x_{k-1}\sigma x_k$, there exists a string $t = x_1\sigma\sigma' x_2\sigma\sigma' \ldots, x_{k-1}\sigma\sigma' x_k$ such that $t \in L(\mathbf{SUP}'_d)$ and $Pt = s$. Hence, we declare that $L(\mathbf{SUP}) \subseteq PL(\mathbf{SUP}'_d)$. □

Lemma A.2: *Let $t = x_1\sigma x_2 x_3\sigma' x_4 \in L_m(\mathbf{SUP}'_d)$, where x_1, x_2, x_3 and x_4 are strings free of σ and σ', i.e. $x_1, x_2, x_3, x_4 \in (\Sigma - \{\sigma\})^*$. Then, $t' := x_1\sigma x_2\sigma' x_3 x_4 \in L_m(\mathbf{SUP}'_d)$.*

Proof of Lemma A.2: Recall that $\mathbf{SUP}'_i$ is $\mathbf{SUP}_i$ with transitions labelled σ relabelled σ'. By definition of synchronous product, x_2, x_3 and σ' can be re-ordered without affecting the membership of t in $L_m(\mathbf{SUP}'_d)$, namely the strings t' formed from t by the successive replacement

$$\begin{aligned} x_1\sigma x_2 x_3\sigma' x_4 &\rightarrow x_1\sigma\sigma' x_2 x_3 x_4 \\ &\rightarrow x_1\sigma x_2\sigma' x_3 x_4 \end{aligned}$$

will belong to $L_m(\mathbf{SUP}'_d)$ as well. In other words, if the transmission of σ is completed in a shorter time (the number of *ticks* in x_2 will be smaller than that in x_2x_3), the behaviour is still legal. □

Proof of Lemma 4.1: We prove Lemma 4.1 by contraposition, i.e. if **SUP** is $(d+1)$-bounded delay-robust, then it is also d-bounded delay-robust. To that end, we must verify (17)–(20).

(1) For (17), we prove that $PL(\mathbf{SUP}_{d'}) \supseteq L(\mathbf{SUP})$ and $PL(\mathbf{SUP}_{d'}) \subseteq L(\mathbf{SUP})$ in sequence. $PL(\mathbf{SUP}_{d'}) \supseteq L(\mathbf{SUP})$ is obtained from Lemma A.1 immediately. By inspection of the transition diagram of $\mathbf{CH}_d(j, \sigma, i)$ in Figure 5, we get that $L(\mathbf{CH}_d(j, \sigma, i)) \subseteq L(\mathbf{CH}_{d+1}(j, \sigma, i))$. So, according to (16),

$$L(\mathbf{SUP}'_d) \subseteq L(\mathbf{SUP}'_{d+1}). \tag{A3}$$

Since **SUP** is $(d+1)$ $-$bounded delay-robust, $PL(\mathbf{SUP}'_{d+1}) \subseteq L(\mathbf{SUP})$. Hence, $PL(\mathbf{SUP}_{d'}) \subseteq L(\mathbf{SUP})$.

(2) Condition (18) can be confirmed from the proof of (17) by replacing L by L_m throughout.

(3) For (19), assume that $s \in L(\mathbf{SUP}'_d)$ and $(Ps)w \in L_m(\mathbf{SUP})$; we must show that there exists a string $v \in \Sigma'^*$ such that $Pv = w$ and $sv \in L_m(\mathbf{SUP}'_d)$.

By (A3), we have $s \in L(\mathbf{SUP}'_{d+1})$. Since **SUP** is $(d+1)$-bounded delay-robust, there exists a string $u \in \Sigma'^*$ such that $Pu = w$ and $su \in L_m(\mathbf{SUP}'_d)$. Here, we consider the case that only one instance of σ exists in su; the general cases can be confirmed similarly (since the transmission of multiple instances of σ does not result in mutual interference). In the following, we prove (19) from these three cases: (i) $su = s_1\sigma s_2\sigma' s_3 u_1 u_2$, (ii) $su = s_1\sigma s_2 u_1\sigma' u_2$, and (iii) $s_1 s_2 u_1\sigma u_2\sigma' u_3$, where s_1, s_2, s_3, u_1, u_2, u_3 are free of σ and σ'.

(i) $su = s_1\sigma s_2\sigma' s_3 u_1 u_2$. By (16), we have $su \in L_m(\mathbf{NSUP})$. Similarly, since $s \in L(\mathbf{SUP}'_d)$, $s \in P_{ch}^{-1}L(\mathbf{CH}_d(j, \sigma, i))$. Furthermore, $s = s_1\sigma s_2\sigma' s_3$, which means that after string s, σ' has reset the channel $\mathbf{CH}_d(j, \sigma, i)$. Thus, $s \in P_{ch}^{-1}L_m(\mathbf{CH}_{d-1}(j, \sigma, i))$. On the other hand, because u is free of σ, $su \in P_{ch}^{-1}L_m(\mathbf{CH}_d(j, \sigma, i))$. Hence, $su \in L_m(\mathbf{SUP}'_d)$. Define $v = u$; then, $Pv = Pu = w$ and $sv \in L_m(\mathbf{SUP}'_d)$, as required by (19).

(ii) $su = s_1\sigma s_2 u_1\sigma' u_2$. By Lemma A.2, it results from $su \in L_m(\mathbf{SUP}'_d)$ that $s_1\sigma s_2\sigma' u_1 u_2 \in L_m(\mathbf{SUP}'_d)$. The rest is similar to case (1); in this case, $v = \sigma' u_1 u_2$.

(iii) $su = s_1 s_2 u_1\sigma u_2\sigma' u_3$. By Lemma A.2, we have $s_1 s_2 u_1\sigma\sigma' u_2 u_3 \in L_m(\mathbf{SUP}'_d)$. Also, the rest is similar to case (1); in this case, $v = u_1\sigma\sigma' u_2 u_3$.

(4) Let $s \in P_{ch}^{-1}L(\mathbf{CH}_d(j, \sigma, i))$ and $s\sigma \in L(\mathbf{NSUP})$; we show that $s\sigma \in P_{ch}^{-1}L(\mathbf{CH}_d(j, \sigma, i))$ by contraposition. Assume that $s\sigma \notin P_{ch}^{-1}L(\mathbf{CH}_d(j, \sigma, i))$. Write $\mathbf{CH}_d(j, \sigma, i) = (C_d, \Sigma_{ch}, \tau_d, c_{d,0}, \{c_{d,0}\})$, where $\Sigma_{ch} = \{\sigma, tick, \sigma'\}$. We claim that $\tau_d(c_{d,0}, P_{ch}s) \neq c_{d,0}$; otherwise, σ is defined at state $\tau_d(c_{d,0}, P_{ch}s)$ and $s\sigma \in P_{ch}^{-1}L(\mathbf{CH}_d(j, \sigma, i))$. By inspection of the transition diagrams of $\mathbf{CH}_d(j, \sigma, i)$ and $\mathbf{CH}_{d+1}(j, \sigma, i)$, it results from $\tau_d(c_{d,0}, P_{ch}s) \neq c_{d,0}$, that $\tau_{d+1}(c_{d+1,0}, P_{ch}s) \neq c_{d+1,0}$. Hence, $s\sigma \notin P_{ch}^{-1}L(\mathbf{CH}_{d+1}(j, \sigma, i))$, in contradiction to the fact that **SUP** is $(d+1)$ $-$bounded delay-robust. □

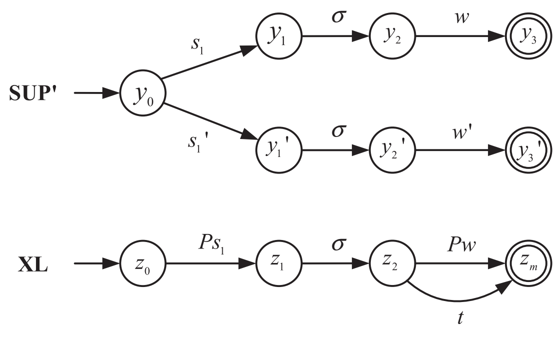

Figure B1. $Ps_1 = Ps_1'$, $Pw' = t$ and t is a simple string.

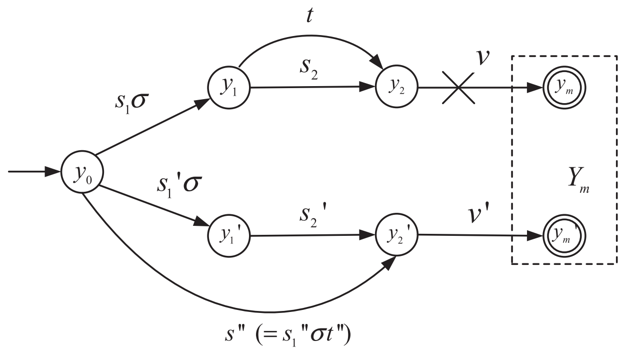

Figure B2. P-normality of **SUP**$'$

Appendix 2 Proof of Lemma 4.2

Since **SUP** is not delay-robust with respect to **CH**(j, σ, i), by Definition 3.1, one of the conditions (12)–(15) is violated. In the following, we prove that in each case, $d_{\max} \leq 2^{m*}m$, where m is the state size of **SUP**$'$ in (11).

(1) Condition (12) is violated. Since that $L(\mathbf{SUP}) \subseteq PL(\mathbf{SUP}')$ always holds (similar to Lemma A.1), we have $PL(\mathbf{SUP}') \not\subseteq L(\mathbf{SUP})$. So, there exists at least one string $s \in \Sigma'^*$ such that $s \in L(\mathbf{SUP}')$, but $Ps \notin L(\mathbf{SUP})$. We claim that s can be written as $s_1\sigma w$, where s_1, $w \in \Sigma'^*$; otherwise, s does not contain any σ, and it follows from the construction of **SUP**$'$ that $Ps \in L(\mathbf{SUP})$, a contradiction. As illustrated in Figure B1, we prove in the following that there exist strings $s_1' \in L(\mathbf{SUP}')$ and $w' \in \Sigma'^*$ such that $\#tick(w') \leq 2^m * m$ (where $\#tick(w')$ represents the number of events $tick$ appearing in string w'), $s'\sigma w' \in L(\mathbf{SUP}')$, but $P(s'\sigma w') \notin L(\mathbf{SUP})$, from which we can conclude: to prevent the occurrence of string $s_1'\sigma w'$, the maximal communication delay of σ must be less than $\#tick(w') \leq 2^m * m$, i.e. $d_{\max} \leq 2^{m*}(m' + 1)$.

By $s_1\sigma w \in L(\mathbf{SUP}')$ and $P(s_1\sigma w) \notin L(\mathbf{SUP})$, we have $P(s_1\sigma w) \in PL(\mathbf{SUP}') \cap (\Sigma^* - L(\mathbf{SUP}))$. To identify such strings, we build an TDES $\mathbf{XL} = (Z, \Sigma, \zeta, z_0, Z_m)$ such that

$$L_m(\mathbf{XL}) = PL(\mathbf{SUP}') \cap (\Sigma^* - L(\mathbf{SUP}))$$

and

$$L(\mathbf{XL}) = PL(\mathbf{SUP}'),$$

i.e. $P(s_1\sigma) \in L(\mathbf{XL})$, and $P(s_1\sigma w) \in L_m(\mathbf{XL})$.

First, we build **XA** such that $L_m(\mathbf{XA}) = PL(\mathbf{SUP}')$ and $L(\mathbf{XA}) = L_m(\mathbf{XA})$ by the following two steps: (i) construct **PSUP**$'$ by applying the subset construction algorithm on **SUP**$'$ with natural projection P, and (ii) obtain **XA** by marking all states of **PSUP**$'$. Second, we build **XB** such that $L_m(\mathbf{XB}) = \Sigma^* - L(\mathbf{SUP})$ and $L(\mathbf{XB}) = \Sigma^*$ by first adjoining a (non-marker) dump state $\hat{q}$ to the state set of **SUP** and transitions $(q, \sigma, \hat{q})$ for each state q of **SUP** if $\sigma \in \Sigma$ is not defined at q (i.e. $L(\mathbf{XB}) = \Sigma^*$), and secondly setting $\hat{q}$ to be the only marker state. Third, let $\mathbf{XL} = \mathbf{XA}||\mathbf{XB}$; then, $L_m(\mathbf{XL}) = PL(\mathbf{SUP}') \cap (\Sigma^* - L(\mathbf{SUP}))$, $L(\mathbf{XL}) = PL(\mathbf{SUP}')$. The state size $|Z| \leq 2^{m*}(m' + 1)$, since **XA** has at most 2^m states (due to the subset construction algorithm), and **XB** has $m' + 1$ states .

Finally, by $P(s_1\sigma) \in PL(\mathbf{SUP}') = L(\mathbf{XL})$, there exists a state $z_2 \in Z$ such that $z_2 = \zeta(z_0, P(s\sigma))$; by $P(s_1\sigma w) \in L_m(\mathbf{XL})$, there exists a marker state $z_m \in Z_m$ such that $z_m = \zeta(z_0, P(s_1\sigma w)) = \zeta(z_2, P(w))$. So, there exists at least a *simple string*[7] $t \in \Sigma^*$ joining z_2 and z_m such that $z_m = \zeta(z_2, t)$, and thus $P(s_1\sigma)t \in L_m(\mathbf{XL})$. It follows that $(Ps_1)\sigma t \in PL(\mathbf{SUP}') \cap (\Sigma^* - L(\mathbf{SUP}))$. So, there exist strings s_1', $w' \in \Sigma'^*$ such that $Ps_1' = Ps_1$, $Pw' = t$, $s_1'\sigma w' \in L(\mathbf{SUP}')$, and $P(s_1'\sigma w') \notin L(\mathbf{SUP})$, namely the occurrence of w' after $s_1'\sigma$ violates condition (12). Since t is simple, we have $\#tick(t) \leq |Z| \leq 2^m * (m' + 1)$. By $Pw' = t$, we have $\#tick(w') = \#tick(t) \leq 2^m * (m' + 1)$. Furthermore, since **SUP**$'$ represents the system behaviour with communication delay, we always have $m' + 1 \leq m$. Hence, $\#tick(w') \leq 2^m * m$, as required.

(2) Condition (13) is violated. $d_{\max} \leq m*2^m$ can be confirmed similar to case (1).

(3) Condition (14) is violated. Since delay-robustness of **SUP** is violated by the communication delay of σ, there must exist strings s_1, s_2, and w, such that $s_1\sigma s_2 \in L(\mathbf{SUP}')$ and $P(s_1\sigma s_2)w \in L_m(\mathbf{SUP})$, but no string v satisfies that $Pv = w$ and $s_1\sigma s_2 v \in L_m(\mathbf{SUP}')$. As illustrated in Figure B2, we prove in the following that condition (14) is also violated by the string pair $s_1\sigma t$ and $s_1''\sigma t''$, where $\#tick(t) \leq 2^m * m$ and $\#tick(t'') \leq 2^m * m$, from which we conclude: to prevent the occurrences of the strings $s_1\sigma t$ and $s_1''\sigma t''$, the communication delay of σ must be less than $min(\#tick(t), \#tick(t'')) \leq 2^m * m$, i.e. $d_{\max} \leq m*2^m$.

To that end, we need the concept 'normal automaton' (Takai & Ushio, 2003). For $\mathbf{SUP}' = (Y, \Sigma', \eta, y_0, Y_m)$, we say that $\mathbf{SUP}'$ is P-normal if

$$(\forall s, t \in L(\mathbf{SUP}'))R(s) \neq R(t) \Rightarrow R(s) \cap R(t) = \emptyset \tag{B1}$$

where $R(s) := \{y \in Y | y = \eta(y_0, s'), Ps = Ps'\}$. In case $\mathbf{SUP}'$ is not P-normal, replace $\mathbf{SUP}'$ by $\mathbf{SUP}'||\mathbf{PSUP}'$, where $\mathbf{PSUP}'$ is a deterministic generator over Σ obtained by the subset construction. $\mathbf{SUP}'||\mathbf{PSUP}'$ is always P-normal, and $L(\mathbf{SUP}') = L(\mathbf{SUP}'||\mathbf{PSUP}')$ and $L_m(\mathbf{SUP}') = L_m(\mathbf{SUP}'||\mathbf{PSUP}')$. The state size of the new $\mathbf{SUP}'$ is at most m^*2^m.

By $P(s_1\sigma s_2)w \in L_m(\mathbf{SUP}) \subseteq PL_m(\mathbf{SUP}')$, there must exist strings s_1', s_2', and v' such that $Ps_1' = Ps_1$, $Ps_2' = Ps_2$, $Pv' = w$, and $s_1'\sigma s_2'v' \in L_m(\mathbf{SUP}')$, as displayed in Figure B2. Let $y_1 = \eta(y_0, s_1\sigma)$, $y_2 = \eta(y_1, s_2)$, $y_1' = \eta(y_0, s_1'\sigma)$, and $y_2' = \eta(y_1', s_2')$. Joining y_1 and y_2, there must exist a simple string t such that $y_2 = \eta(y_1, t)$. So, $R(s_1\sigma s_2) \cap R(s_1\sigma t) = y_2$. By P-normality of $\mathbf{SUP}'$, there must exist a string $s'' \in L(\mathbf{SUP}')$ such that $y_2' = \eta(y_0, s'')$, $P(s_1\sigma t) = P(s'')$, and $y_2' \in R(s_1\sigma t)$. So, string s'' can be written as $s_1''\sigma t''$, where $Ps_1'' = Ps_1$ and $Pt'' = Pt$, and condition (14) is also violated by the string pair $s_1\sigma t$ and $s_1''\sigma t''$. Because t is simple, $\#tick(t) \leq m$, where m is the state size of P-normal form of $\mathbf{SUP}'$. So, when $\mathbf{SUP}'$ is not P-normal, $\#tick(t) \leq m * 2^m$. In addition, since $Pt'' = Pt$, $\#tick(t'') = \#tick(t) \leq m * 2^m$, as required.

(4) Condition (15) is violated. In this case, assume that σ is blocked at state y of $\mathbf{SUP}'$, and the last occurrence of σ occurs at state y' of $\mathbf{SUP}'$. From y' to y, there must exist a simple string t. We claim that the maximal communication delay of σ must be less than $\#tick(t)$; otherwise, the system will arrive at state y by string t. Hence, $d_{\max} \leq \#t(tick) \leq m$.

Finally, by comparing $d_{\max}$ in the above four cases, we conclude that if $\mathbf{SUP}$ is not delay-robust with respect to $\mathbf{CH}(j, \sigma, i)$, $d_{\max} \leq m^*2^m$.