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

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

Version  Post-Print/Accepted Manuscript


Publisher's Statement  This is an Accepted Manuscript of an article published by Taylor & Francis in International Journal of Control on 2016, available online: http://www.tandfonline.com/10.1080/00207179.2016.1147606.

How to cite TSpace items

Always cite the published version, so the author(s) will receive recognition through services that track citation counts, e.g. Scopus. If you need to cite the page number of the author manuscript from TSpace because you cannot access the published version, then cite the TSpace version in addition to the published version using the permanent URI (handle) found on the record page.

This article was made openly accessible by U of T Faculty. Please tell us how this access benefits you. Your story matters.
Delay-Robustness in Distributed Control of Timed Discrete-Event Systems Based on Supervisor Localization

Renyuan Zhang\textsuperscript{a}\textsuperscript{*}, Kai Cai\textsuperscript{b}, Yongmei Gan\textsuperscript{c}, W.M. Wonham\textsuperscript{d}

\textsuperscript{a}School of Automation, Northwestern Polytechnical University, China; \textsuperscript{b}Urban Research Plaza, Osaka City University, Japan; \textsuperscript{c}School of Electrical Engineering, Xian Jiaotong University, China; \textsuperscript{d}Department of Electrical and Computer Engineering, University of Toronto, Canada

(Received 00 Month 20XX; accepted 00 Month 20XX)

Recently we studied communication delay in distributed control of untimed discrete-event systems based on supervisor localization. We proposed a property called delay-robustness: the overall system behavior 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 \textit{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 \textit{bounded} delay-robustness, and present an algorithm to compute the maximal delay bound (measured by number of \textit{ticks}) for transmitting a channeled event. Finally, we demonstrate delay-robustness on the example of an under-load tap-changing transformer.

\textbf{Keywords:} Timed Discrete-Event Systems; Distributed Supervisory Control; Supervisor Localization; Delay-robustness.

1. Introduction

For distributed control of discrete-event systems (DES), supervisor localization was recently proposed (Cai & Wonham, 2010b, 2010a, 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 behavior 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 localization, 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 \textit{delay-robustness}, meaning that the system’s behavior 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 channeled events the system is delay-robust. If for a channeled 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 \textit{temporal} measure for the delay bound (except

*Corresponding author. Email: ryzhang@nwpu.edu.cn.
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); Wonham (2015). In this framework, the special clock event tick provides a natural way of modeling communication delay as temporal behavior. We first propose a timed channel model for transmitting each channeled 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., 2012, 2015) in two respects: (1) the system’s temporal behavior 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 ticks) for transmitting a channeled 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/decentralized supervisory control with communication delay has been widely studied for untimed DES (e.g. Barrett and Lafortune (2000); Darondeau and Ricker (2012); Hiraishi (2009); Kalyon, Gall, Marchand, and Massart (2011); Lin (2014); Sadid, Ricker, and Hashtrudi-Zad (2015); Schmidt, Schmidt, and Zaddach (2007); Tripakis (2004); Xu and Kumar (2008)). In particular in (Kalyon et al., 2011; Tripakis, 2004), the existence of distributed controllers in the unbounded delay case is proved to be undecidable; and in (Hiraishi, 2009; Lin, 2014; Schmidt et al., 2007; Tripakis, 2004; Xu & Kumar, 2008), distributed controllers are synthesized 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. (2012, 2015) 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 localization Cai and 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 organized as follows. Sect. 2 provides a review of the Brandin-Wonham TDES framework and recalls supervisor localization for TDES. In Sect. 3 we introduce a timed channel model, and present the concept and verification algorithm for timed delay-robustness. In Sect. 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 Sect. 5 on the distributed control problem for a ULTC with communications. Conclusions are presented in Sect. 6.

2. Distributed Control by Supervisor Localization of TDES

2.1 Preliminaries on TDES

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

\[ G := (Q, \Sigma, \delta, q_0, Q_m). \]  

Here \( Q \) is the finite set of states; \( \Sigma \) is the finite set of events including the special event tick, which represents “tick of the global clock”; \( \delta : Q \times \Sigma \rightarrow Q \) 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 & Wonham, 1994; Wonham, 2015); \( q_0 \) is the \textit{initial state}; and \( Q_m \subseteq Q \) is the set of \textit{marker states}. The transition function is extended to \( \delta : Q \times \Sigma^* \rightarrow Q \) in the usual way. The \textit{closed behavior} of \( G \) is the language \( L(G) := \{ s \in \Sigma^* | \delta(q_0, s)! \} \) and the \textit{marked behavior} is \( L_m(G) := \{ s \in L(G) | \delta(q_0, s) \in Q_m \} \subseteq L(G) \). We say that \( G \) is \textit{nonblocking} if \( L_m(G) = L(G) \), where \( \supseteq \) denotes prefix closure (Wonham, 2015).

Let \( \Sigma^* \) be the set of all finite strings, including the empty string \( \epsilon \). For \( \Sigma' \subseteq \Sigma \), the \textit{natural projection} \( P : \Sigma^* \rightarrow \Sigma'^* \) is defined by

\[
P(\epsilon) = \epsilon; \quad P(\sigma) = \begin{cases} \epsilon, & \text{if } \sigma \notin \Sigma', \\ \sigma, & \text{if } \sigma \in \Sigma'. \end{cases} \quad (2)
\]

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

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

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

\[
\text{Elig}_F(s) \supseteq \begin{cases} \text{Elig}_G(s) \cap (\Sigma_u \cup \{ \text{tick} \}) & \text{if } \text{Elig}_F(s) \cap \Sigma_{for} = \emptyset, \\ \text{Elig}_G(s) \cap \Sigma_u & \text{if } \text{Elig}_F(s) \cap \Sigma_{for} \neq \emptyset. \end{cases} \quad (3)
\]

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 & Wonham, 1994; Wonham, 2015). Now consider a specification language \( E \subseteq \Sigma^* \) imposed on the timed behavior of \( G \); \( E \) may represent a logical and/or temporal requirement. Let the TDES

\[
\text{SUP} = (X, \Sigma, \xi, x_0, X_m)
\]

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

\[
L_m(\text{SUP}) = \sup \mathcal{C}(E \cap L_m(G)) \subseteq L_m(G)
\]

and moreover its closed language \( L(\text{SUP}) = \overline{L_m(\text{SUP})} \).
2.2 Supervisor Localization of TDES

In this subsection, we introduce the supervisor localization procedure, which was initially proposed in the untimed DES framework (Cai & 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), synthesize the monolithic optimal and nonblocking supervisor SUP,\(^1\) Supervisor SUP’s control action includes (i) disabling prohibitive events in \(\Sigma_{hub}\) and (ii) pre-emting \(\text{tick}\) via forcible events in \(\Sigma_{far}\). By the supervisor localization procedure, a set of local controllers \(\{\text{LOC}^C_\alpha\}_{\alpha \in \Sigma_{hub}}\) and a set of local preemptors \(\{\text{LOC}^P_\beta\}_{\beta \in \Sigma_{far}}\) are constructed. In each local controller \(\text{LOC}^C_\alpha\), each state \(y_\alpha \in Y_\alpha\) and the marker state set \(Y_{\alpha,0}\). The transition function \(\eta_\alpha\) is derived from the control cover and the transitions defined on the corresponding states. The event set \(\Sigma_\alpha\) is the set of events that cause state changes in \(y_\alpha\). 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 localization procedure localizes SUP’s control action with respect to each prohibitive event \(\alpha\) (resp. forcible event \(\beta\)) into a local controller \(\text{LOC}^C_\alpha\) (resp. local preemptor \(\text{LOC}^P_\beta\)). These \(\text{LOC}^C_\alpha\) and \(\text{LOC}^P_\beta\) are all generalized TDES,\(^2\) and proved to be control-equivalent to SUP (with respect to G) in the following sense:

\[
L(G) \cap \left( \bigcap_{\alpha \in \Sigma_{hub}} P^{-1}_\alpha L(\text{LOC}^C_\alpha) \right) \cap \left( \bigcap_{\beta \in \Sigma_{far}} P^{-1}_\beta L(\text{LOC}^P_\beta) \right) = L(SUP),
\]

(5)

\[
L_m(G) \cap \left( \bigcap_{\alpha \in \Sigma_{hub}} P^{-1}_\alpha L_m(\text{LOC}^C_\alpha) \right) \cap \left( \bigcap_{\beta \in \Sigma_{far}} P^{-1}_\beta L_m(\text{LOC}^P_\beta) \right) = L_m(SUP).
\]

(6)

Here \(P_\alpha : \Sigma^* \rightarrow \Sigma^*_\alpha\) and \(P_\beta : \Sigma^* \rightarrow \Sigma^*_\beta\) 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 \(G_i\) (\(i \in N := \{1, 2, ..., N\}\)), each with event set \(\Sigma_i \ni \text{tick}\). For simplicity assume \(\Sigma_i \cap \Sigma_j = \{\text{tick}\}\), for all \(i \neq j \in N\); namely the agents \(G_i\) are independent except for synchronization on the global event \(\text{tick}\). As a result, the marked and closed behaviors of the composition of the \(G_i\) coincide with those of their synchronous product (Wonham, 2015), and thus we use synchronous product instead of composition to combine TDES together, i.e. \(G = ||_{i \in N} G_i\) where || denotes the synchronous product of TDES.\(^3\)

\(^1\)We remark that if the system and its supervisor are large scale, we first synthesize a set of decentralized supervisors to achieve global optimality and nonblocking, and then apply supervisor localization to decompose each decentralized supervisor in the set (as in Cai and Wonham (2010a, 2015)). Other decentralized/distributed control schemes, e.g. coordination technique (Seow, Pham, Ma, and Yokoo (2009) and Komenda, Masopust, and van Schuppen (2014)), are also candidates for computing the decentralized/distributed supervisors.

\(^2\)We refer to any DES over an alphabet which includes \(\text{tick}\) as a generalized TDES; it need not be a (strict) TDES constructed according to the rules in Wonham (2015, Sect. 9.2). Generalized TDES are needed to model specifications and supervisors (Wonham, 2015).

\(^3\)The closed and marked behaviors of TDES = TDES1 || TDES2 are \(L(TDES) = L(TDES1) || L(TDES2)\) and \(L_m(TDES) = L_m(TDES1) || L_m(TDES2)\), where || denotes the synchronous product of languages (Wonham, 2015). There are three cases where we use the plant component to compute the concurrent behavior of TDES. (i) The TDES that describe plant components share only event tick; in this case, synchronous product is equivalent to composition (Wonham, 2015). (ii) We compute the plant behavior restricted by a generalized TDES; in this case, the TDES synchronizes with the plant to form their concurrent behavior; it is not composed with the plant to generate a larger system. (iii) We compute the concurrent
Figure 1. Example of distributed control by supervisor localization: plant $G$ is composed of three agents $G_k$ with event sets $\Sigma_k$, $k \in \{1, 3\}$: $\sigma_1, \sigma_2 \in \Sigma_1$, $\sigma_3 \in \Sigma_2$, and $\sigma_4, \sigma_5 \in \Sigma_3$. First, by the supervisor localization procedure we construct from $SUP$ a set of local controllers $\{LOC_{\sigma_1}^C, LOC_{\sigma_2}^C, LOC_{\sigma_3}^C\}$ and a set of local preemptors $\{LOC_{\sigma_4}^P, LOC_{\sigma_5}^P\}$. Then, by our allocation policy we obtain a distributed control architecture in which each $G_k$ is controlled by its own local controllers/preemptors, while interacting with other agents through communication of shared events.

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

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

(7)

where $\Sigma_\alpha$ and $\Sigma_\beta$ are the event sets of $LOC_i^C$ and of $LOC_i^P$ respectively.

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 synthesized local control strategies and the corresponding overall system behavior.

3. Timed Delay-Robustness

Consider event communication between a pair of agents $G_i$ and $G_j$ ($i, j \in N$): specifically, $G_j$ sends an event $\sigma$ to $G_i$. Let $\Sigma_j$ be the event set of $G_j$ and $\Sigma_{com,i}$ as in (7) the set of communication events that $G_i$ imports from other agents. Then the set of events that $G_j$ sends to $G_i$ is

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

(8)

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

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

It should be stressed that the number of $tick$ occurrences between $\sigma$ and $\sigma'$ is unspecified, inasmuch as the selfloop $tick$ at state 1 may occur indefinitely. In this sense, $CH(j, \sigma, i)$ models unbounded communication delay. Note that $tick$ is also selflooped 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 synchronizing $CH(j, \sigma, i)$ with other TDES. The initial state 0 is marked, signaling each completion of event $\sigma$ transmission; state 1, on the other hand, is unmarked because the transmission is still ongoing.

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

As displayed in Fig. 3, after the delay is introduced, the transmission of $\sigma$ is replaced by the channel model $CH(j, \sigma, i)$, and the channeled behavior of the system is described as follows. Suppose given $G_k$, $k \in N$; by localization (see Sect. 2.2) $G_k$ acquires a set of local controllers $LOC^C_k := \{LOC^C_\alpha | \alpha \in \Sigma_k, hib\}$ and a set of local preemters $LOC^P_k := \{LOC^P_\beta | \beta \in \Sigma_{k, for}\}$. So the local

\[4\] At each state $x$ of each local controller $LOC^C_\alpha$, if a communication event $\sigma \in \Sigma_i - \Sigma_k$ is not defined, we add a $\sigma$-selfloop, i.e. transition $(x, \sigma, x)$ to $LOC^C_\alpha$; if $tick$ is not defined at $x$, we add a $tick$-selfloop, i.e. transition $(x, tick, x)$ to $LOC^C_\alpha$. At each state $y$ of each preemptor $LOC^P_\beta$, if communication event $\sigma \in \Sigma_j - \Sigma_k$ is not defined, we add a $\sigma$-selfloop, i.e. transition $(y, \sigma, y)$ to $LOC^P_\beta$; if $tick$ is not defined and not preempted, we add a $tick$-selfloop, i.e. transition $(y, tick, y)$ to $LOC^P_\beta$. With this modification, (1) the new local controllers $LOC^C_\alpha$ (resp. local preemptors $LOC^P_\beta$) are also control equivalent to SUP (because $LOC^C_\alpha$ (resp. $LOC^P_\beta$) does not disable events $\sigma$ from other components $G_j$; (2) the definition of $\sigma$ at every state of $LOC^C_\alpha$ (resp. $LOC^P_\beta$) is consistent with the assumption that $LOC^C_\alpha$ (resp. $LOC^P_\beta$) may receive $\sigma$ after indefinite communication delay; and (3) $tick$ is only preempted by forcible events, say $\beta$, at some states of the corresponding local preemptor $LOC^P_\beta$.

Figure 3. Example of distributed control architecture with communication delay: consider the example in Fig. 1 and assume that only event $\sigma_4$ in $G_3$ is transmitted to $G_1$ with unbounded delay. When the communication delay of $\sigma_4$ is negligible, $LOC^C_{\sigma_1}$ and $LOC^C_{\sigma_2}$ will 'synchronize' on the occurrence of $\sigma_4$ in $G_3$, namely $\sigma_4$ is communicated through 'synchronization'. When the communication delay of $\sigma_4$ is unbounded, the communication of $\sigma_4$ is modeled by a TDES channel model $CH(3, \sigma_4, 1)$.
controlled behavior of $G_k$ is

$$\text{SUP}_k := G_k \parallel \left( \big||_{\alpha \in \Sigma_{k, \text{ch}}} \text{LOC}_{\alpha}^C \right) \parallel \left( \big||_{\beta \in \Sigma_{k, \text{for}}} \text{LOC}_{\beta}^P \right). \quad (9)$$

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

$$\text{NSUP} := \text{SUP}_i' \parallel \big||_{k \in N, k \neq i} \text{SUP}_k \quad (10)$$

and then

$$\text{SUP}' := \text{NSUP} \parallel \text{CH}(j, \sigma, i). \quad (11)$$

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

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

**Definition 1:** Consider that $G_j$ sends event $\sigma$ to $G_i$ through channel $\text{CH}(j, \sigma, i)$.

The monolithic supervisor $\text{SUP}$ in (4) is delay-robust with respect to $\text{CH}(j, \sigma, i)$ if the following conditions hold:

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

$$PL(\text{SUP}') = L(\text{SUP}) \quad (12)$$
$$PL_m(\text{SUP}') = L_m(\text{SUP}) \quad (13)$$
$$(\forall s \in \Sigma^*)(\forall w \in \Sigma^*) s \in L(\text{SUP}') \& (Ps)w \in L_m(\text{SUP})$$
$$\Rightarrow (\exists v \in \Sigma^*) Pw = w \& sv \in L_m(\text{SUP}') \quad (14)$$

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

$$P_{\text{ch}}^{-1}L(\text{CH}(j, \sigma, i)) \cap L(\text{NSUP}) \subseteq P_{\text{ch}}^{-1}L(\text{CH}(j, \sigma, i)) \quad (15)$$

In condition (i) above, ‘correctness’ of $\text{SUP}'$ means that no $P$-projection of anything $\text{SUP}'$ can do is disallowed by $\text{SUP}$, while ‘completeness’ means that anything $\text{SUP}$ can do is the $P$-projection of something $\text{SUP}'$ can do. In this sense, the channeled behavior $\text{SUP}'$ is ‘equivalent’ to its delay-free counterpart $\text{SUP}$. Thus if $\text{SUP}$ is determined to be delay-robust with respect to $\text{CH}(j, \sigma, i)$, then by (9)-(11) the composite plant under the control of the local controllers and preemptors with unbounded delay of event $\sigma$ transmission from $G_j$ to $G_i$ is equivalent to $\text{SUP}$. Note that here the local controllers and preemptors are constructed from the monolithic supervisor by the supervisor localization procedure in Cai and Wonham (2010a, 2015), and we don’t 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 behaviors between $\text{SUP}$ and the $P$-projection of $\text{SUP}'$; condition (14), which is required for ‘completeness’, states

and at those states, $\beta$ is enabled (otherwise, tick is defined).
that if $\text{SUP}'$ executes a string $s$ whose projection $P_s$ in $\text{SUP}$ can be extended by a string $w$ to a marked string of $\text{SUP}$, then $\text{SUP}'$ can further execute a string $v$ whose projection $Pv$ is $w$ and such that $sv$ is marked in $\text{SUP}'$. Roughly, an observationally consistent inference about coreachability at the “operating” level of $\text{SUP}'$ can be drawn from coreachability at the abstract (projected) level of $\text{SUP}$.

Condition (ii) of Definition 1 imposes a basic requirement that channel $\text{CH}(j, \sigma, i)$, when combined with $\text{NSUP}$ in (10) to form $\text{SUP}'$, should not entail uncontrollability with respect to $\sigma$. We impose condition (ii) no matter whether $\sigma$ is controllable or uncontrollable. This is because we view the channel $\text{CH}(j, \sigma, i)$ as a purely passive adjunction to the original system, and therefore $\text{CH}(j, \sigma, i)$ cannot exercise control on $\sigma$. 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 $\sigma$ in the channel (i.e. $\text{CH}(j, \sigma, i)$ at state 1), then reoccurrence of $\sigma$ 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 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 $\text{CH}(j, \sigma, i)$ 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. Notice that when (12) and (13) hold, then (14) is identical with the $L_m(\text{SUP}')$-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 $\text{SUP}'$) by computing the supremal quasi-congruence of a nondeterministic automaton derived from $\text{SUP}'$ 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 $L_m(\text{SUP}')$-observer. If not, return $false$.
2. Check if $PL(\text{SUP}') = L(\text{SUP})$ and $PL_m(\text{SUP}') = L_m(\text{SUP})$. If not, return $false$.
3. Check if $P^{-1}_c(L(\text{CH}(j, \sigma, i)))$ is controllable with respect to $L(\text{NSUP})$ 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 $L_m(\text{SUP}')$-observer, then Step 2 of computing $PL(\text{SUP}')$ and $PL_m(\text{SUP}')$ is of polynomial complexity $O(m^4)$ (Feng & 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 $\text{SUP}$ is required, and in that case the complexity of Algorithm 1 would be exponential. The following result is straightforward.

**Proposition 1:** Consider that $G_j$ sends event $\sigma$ to $G_i$ through channel $\text{CH}(j, \sigma, i)$. The monolithic supervisor $\text{SUP}$ is delay-robust with respect to $\text{CH}(j, \sigma, i)$ if and only if Algorithm 1 returns $true$.

We note here that coerciveness, as defined in Wonham (2015, Chapt. 9), is preserved when $\text{SUP}$ 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 Footnote 4, $tick$ is only preempted by a forcible event $\beta$ at some state $y$ of local preemptor $\text{LOC}_j^P$, and at state $y$, $\beta$ is enabled; at any other states, $tick$ is defined. After the communication delay is introduced, if

---

5We 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.
**Remark 1:** (Multi-capacity channel model) So far we have considered the 1-capacity channel model $\text{CH}(j,\sigma,i)$, and defined delay-robustness with respect to it. We now consider the more general $C$-capacity channel model $\text{NCH}(j,\sigma,i)$, $C \geq 1$ a positive integer, displayed in Fig. 4. The sender $G_j$ may send at most $C$ instances of event $\sigma$ to $\text{NCH}(j,\sigma,i)$, with each instance subject to indefinite delay. With channel $\text{NCH}(j,\sigma,i)$, one may proceed just as before, by replacing $\text{CH}(j,\sigma,i)$ by $\text{NCH}(j,\sigma,i)$ throughout, to define the corresponding delay-robustness property with respect to $\text{NCH}(j,\sigma,i)$, and then revising Algorithm 1 correspondingly to verify delay-robustness.

It is worth noting that when $\text{NCH}(j,\sigma,i)$ reaches its maximum capacity, and $G_j$ sends yet another instance of $\sigma$, then $\sigma$ is ‘blocked’ by $\text{NCH}(j,\sigma,i)$, implying uncontrollability of the channeled behavior. 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) of Definition 1) is more easily satisfied for larger capacity channels (simply because more instances of $\sigma$ may be sent to the channel).

4. Bounded Delay-Robustness and Maximal Delay Bound

Consider again the situation that agent $G_j$ sends an event $\sigma$ to $G_i$. If the monolithic supervisor $\text{SUP}$ is verified (by Algorithm 1) to be delay-robust, then we will use channel $\text{CH}(j,\sigma,i)$ in Fig. 2 to transmit $\sigma$ subject to unbounded delay, and the system’s behavior will not be affected. If, however, $\text{SUP}$ fails to be delay-robust, there are two possible implications: (1) $\sigma$ must be transmitted without delay (as in the original setup of localization (Cai & Wonham, 2015; Cai et al., 2013; Zhang et al., 2013)); or (2) there exists a delay bound $d \geq 1$ for $\sigma$ such that if each transmission of $\sigma$ is completed within $d$ occurrences of tick, the system’s behavior 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 $\text{CH}_d(j,\sigma,i)$ in Fig. 5, with parameter $d \geq 1$. $\text{CH}_d(j,\sigma,i)$ is a $(d+2)$-state TDES with event set $\{\sigma, \text{tick}, \sigma'\}$. After an occurrence of $\sigma$ (state 0 to 1), $\text{CH}_d(j,\sigma,i)$ counts up to $d$ occurrences of tick (state 1 through $d+1$) by which time the signal event $\sigma'$ must occur. That is, the occurrence of $\sigma'$ ($G_i$ receives $\sigma$) is bounded by $d$ ticks. Note that the tick selfloop at state 0 is again for the technical requirement to prevent the blocking of event tick when synchronizing $\text{CH}_d(j,\sigma,i)$ with other TDES.

Now with $\text{CH}_d(j,\sigma,i)$, the channeled behavior of the system is

$$\text{SUP}'_d := \text{NSUP} \parallel \text{CH}_d(j,\sigma,i)$$

where $\text{NSUP}$ is given in (10). The event set of $\text{SUP}'_d$ is $\Sigma' = \Sigma \cup \{\sigma'\}$, and we recall the natural projections $P : \Sigma'^* \rightarrow \Sigma^*$ and $P_{\text{ch}} : \Sigma'^* \rightarrow \{\sigma, \text{tick}, \sigma'\}^*$.
Definition 2: Consider that $\mathbf{G}_j$ sends event $\sigma$ to $\mathbf{G}_i$ through channel $\mathbf{CH}_d(j, \sigma, i)$, $d \geq 1$. The monolithic supervisor $\mathbf{SUP}$ 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.

\[
PL(\mathbf{SUP}'_d) = L(\mathbf{SUP})
\]
\[
PL_m(\mathbf{SUP}'_d) = L_m(\mathbf{SUP})
\]
\[
(\forall s \in \Sigma^*) (\forall w \in \Sigma^*) \ s \in L(\mathbf{SUP}'_d) \& (Ps)w \in \Sigma_m(\mathbf{SUP}) \Rightarrow (\exists v \in \Sigma^*) \ Pv = w \& sv \in \Sigma_m(\mathbf{SUP}'_d)
\]

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

\[
P_{ch}^{-1}(L(\mathbf{CH}_d(j, \sigma, i))) \cap L(\mathbf{NSUP}) \subseteq P_{ch}^{-1}L(\mathbf{CH}_d(j, \sigma, i))
\]

Bounded delay-robustness is defined in the same way as (unbounded) delay-robustness in Definition 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 Footnote 4, the coerciveness is also preserved.

Now if the monolithic supervisor $\mathbf{SUP}$ fails to be (unbounded) delay-robust with respect to channel $\mathbf{CH}_d(j, \sigma, i)$, we would like to verify if $\mathbf{SUP}$ 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 ticks) that can be tolerated without changing the system’s logical behavior. For this we need the following lemma.

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

The result of Lemma 1 is intuitive: if $\mathbf{SUP}$ cannot tolerate a $\sigma$ transmission delay of $d$, neither can it tolerate a delay $(d + 1)$. By induction, in fact, $\mathbf{SUP}$ cannot tolerate any delay larger than $d$. The proof of Lemma 1 is in Appendix A. This fact suggests the following algorithm for identifying bounded delay-robustness as well as computing the maximal delay bound.

Algorithm 3

1. Set $d = 1$.

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 1 and easier to follow.
2. Check by Algorithm 2 if SUP is $d$-bounded delay-robust relative to channel $\text{CH}_d(j, \sigma, i)$. 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 2:** If SUP is not delay-robust with respect to $\text{CH}(j, \sigma, i)$, then Algorithm 3 terminates in at most $2^n \times m$ steps, i.e. $d_{max} \leq 2^n \times m$, where $m$ is the state size of SUP’ in (11).

The proof of Lemma 2 is given in Appendix B. In Algorithm 3, we work upwards starting from the minimal delay $d = 1$. If SUP is not 1-bounded delay-robust with respect to $\text{CH}_1(j, \sigma, i)$, then by Lemma 1 SUP is not $d$-bounded delay-robust for any $d > 1$. Therefore SUP is not bounded delay-robust and $\sigma$ 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 $\text{CH}_2(j, \sigma, i)$. If SUP fails to be 2-bounded delay-robust, then again by Lemma 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 $d \geq 2$; this happens in finitely many steps according to Lemma 2. Then SUP is bounded delay-robust, with the maximal delay bound $d_{max} = d$. The following result is immediate.

**Proposition 2:** Consider that $G_j$ sends event $\sigma$ to $G_i$, through channel $\text{CH}_d(j, \sigma, i)$, $d \geq 1$. The monolithic supervisor SUP is bounded delay-robust with respect to $\text{CH}_d(j, \sigma, i)$ 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 $\sigma$ transmission.

To summarize, when an event $\sigma$ is sent from $G_j$ to $G_i$, 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 $\text{CH}(j, \sigma, i)$ in Fig. 2.

2. Check by Algorithm 3 if SUP is bounded delay-robust. If so (i.e. $d_{max} \geq 1$), terminate and use channel $\text{CH}_d(j, \sigma, i)$ in Fig. 5 with $d = d_{max}$.

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

**Remark 2:** (Multiple channeled events) So far we have considered a single event communication: agent $G_j$ sends event $\sigma$ to $G_i$. 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 $\alpha$ and $\beta$, differs from that of their transmission, SUP may still be delay-robust with respect to $\{\alpha, \beta\}$.

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

In the following, we present an approach to the general case of multiple channeled events, as is common in distributed control, such that the overall system behavior with all the delays is optimal and nonblocking. We will consider that each fixed triple (sender, channeled 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, channeled event, receiver).

First fix $i, j \in \mathcal{N}$, and recall from (8) that $\Sigma_{j,\text{com},i}$ is the set of events that $G_j$ sends to $G_i$. Write $\Sigma_{j,\text{com},i} = \{\sigma_1, \ldots, \sigma_r\}$, $r \geq 1$, and treat the channeled events $\sigma_1, \sigma_2, \ldots$ sequentially, in order of indexing.

**Algorithm 5**
1. Set \( p = 1 \).
2. For event \( \sigma_p \in \Sigma_{j,\text{com},i} \) apply Algorithm 4 to obtain the maximal delay bound \( d_{\text{max}} \).
   2.1. If \( d_{\text{max}} = \infty \), the case of unbounded delay-robustness, choose channel \( \text{CH}(j, \sigma_p, i) \), and let \( \text{NSUP} := \text{NSUP} \parallel \text{CH}(j, \sigma_p, i) \).
   2.2. If \( d_{\text{max}} \geq 1 \) is finite, the case of bounded delay-robustness, choose channel \( \text{CH}_d(j, \sigma_p, i) \), and let \( \text{NSUP} := \text{NSUP} \parallel \text{CH}_d(j, \sigma_p, i) \).
   2.3 If \( d_{\text{max}} = 0 \), then no channel is chosen and \( \sigma_p \) must be transmitted without delay.
3. Output a set of channels used for sending events from \( G_j \) to \( G_i \).

Note that at Step 2 of Algorithm 5, if a channel is chosen for event \( \sigma_p \), then \( \text{NSUP} \) must be reset to be the synchronous product of \( \text{NSUP} \) and the channel, so that in choosing a channel for the next event \( \sigma_{p+1} \) the previously chosen channel is considered together. This ensures that when the derived channels operate concurrently, the system’s behavior is not affected. It is worth noting that a different ordering of the set \( \Sigma_{j,\text{com},i} \) may result in a different set of channels; if no priority of transmission delay is imposed on the communication events, we may choose an ordering randomly.

Finally, since the set of all communication events is \( \Sigma_{\text{com}} := \bigcup_{i,j \in \mathbb{N}} \Sigma_{j,\text{com},i} \), we simply apply Algorithm 5 for each (ordered) pair \( i,j \in \mathbb{N} \) to derive all communication channels. Again, a different ordering of the set \( \mathbb{N} \times \mathbb{N} \) 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, \( \text{SUP} \) becomes more complex when more channels are introduced. The computation may be expensive when there are a large number of communication channels. Nevertheless \( \text{SUP} \) is implemented in a purely distributed fashion: distributed supervisors and communication channels. We shall investigate the computational issue for \( \text{SUP} \) 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 Localization

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 Fig. 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 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 \( V_{\text{max}} \), 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 modeled as a TDES displayed in Fig. 10, and associated events are listed in Table 1. The plant to be controlled is the synchronized behavior of Voltmeter (VOLT), Tap-changer (TAP) and Operator (OPTR), i.e.

\[ \text{PLANT} = \text{VOLT} \parallel \text{TAP} \parallel \text{OPTR}. \]  

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. Fig. 11 displays the TDES model SPEC for the control specification in Automatic/Manual mode.

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 synthesize the monolithic supervisor SUP by the standard TDES supervisory control theory (Brandin & Wonham, 1994; Wonham, 2015) and the local controllers by TDES supervisor localization (Cai et al., 2013; Zhang et al., 2013).

First, synthesize the monolithic supervisor TDES SUP in the usual sense that its marked behavior

\[ \text{L}_m(\text{SUP}) = \text{SupC}(\text{L}_m(\text{SPEC}) \cap \text{L}_m(\text{PLANT})) \]  

and its closed behavior \( \text{L}(\text{SUP}) = \overline{\text{L}_m(\text{SUP})} \). 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 localization, we obtain a set of local controllers $\text{LOC}_{11}^C$, $\text{LOC}_{31}^C$, $\text{LOC}_{33}^C$, $\text{LOC}_{35}^C$, $\text{LOC}_{41}^C$, and $\text{LOC}_{43}^C$ for controllable events 11, 31, 33, 35, 41 and 43 respectively, and a set of local preemptors $\text{LOC}_{31}^P$, $\text{LOC}_{33}^P$, and $\text{LOC}_{35}^P$ for forcible events 31, 33 and 35 respectively; their transition diagrams are shown in Fig. 12.

Finally, using these constructed local controllers/preemptors, we build a distributed control architecture without communication delays for ULTC as displayed in Fig. 13. It is guaranteed by supervisor localization 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

---

8Here the local controllers and local preemptors are coincidentally described by the same automaton. However, in general, there is no particular relationship between $\text{LOC}_\sigma^C$ and $\text{LOC}_\sigma^P$ for the prohibitive and forcible event $\sigma$; an illustrative example can be found in Section 5 in Zhang et al. (2013).
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’.

marked behavior identical to SUP in (22).

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 Fig. 14.

For illustration, consider the case that the voltmeter reported an increase in voltage (in VOLT as displayed in Fig. 10, events 11 and 10 have occurred), and the tap has received a tap-up command, but the tap-up operation failed (in TAP as displayed in Fig. 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) LOC$_{11}$ disables event 11; (2) LOC$_{35}$ disables event 35; (3) LOC$_{41}$ will disable or enable event 41 depending on the communication delay of event 30; (4) LOC$_{43}$ 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 LOC$_{41}$ 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 LOC$_{41}$ displayed in Fig. 12, only after LOC$_{41}$ has received the occurrence of event 30, will it enable event 41, and bring the system into the automatic mode. Hence, the overall system behavior 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 Fig. 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 Fig. 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 $s' := 11.10.tick.tick.tick.tick.tick.33$. $s$
We add necessary self-loops of communication events and event tick to the local controllers and preempts according to Footnote 4. Let \( * (s) \) be the set of events whose selfloops need to be added at state \( s \). In \( \text{LOC}_{CH}^{i} \), \( * (0) = * (2) = * (4) = \{ \text{tick}, 30, 31, 32, 33, 34, 35, 41 \} \), \( * (1) = \{ \text{tick}, 43 \} \), \( * (5) = \{ \text{tick}, 30, 32, 34 \} \), and \( * (6) = \{ \text{tick}, 31, 33, 35 \} \); in \( \text{LOC}_{max}^{i} \), \( * (1) = * (2) = * (3) = * (4) = * (5) = \{ 14 \} \) and \( * (6) = \{ \text{tick}, 14 \} \); in \( \text{LOC}_{SUP}^{i} \), \( * (1) = * (2) = * (3) = * (4) = * (5) = \{ 10 \} \) and \( * (6) = \{ \text{tick}, 10 \} \); in \( \text{LOC}_{O}^{i} \), \( * (0) = \{ \text{tick} \} \); in \( \text{LOC}_{P}^{i} \), \( * (0) = \{ \text{tick}, 30 \} \); in \( \text{LOC}_{V}^{i} \), \( * (0) = \{ \text{tick}, 10, 12, 14, 16, 30, 32, 34 \} \), and \( * (1) = \{ \text{tick}, 11, 31, 33, 35 \} \).

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 can not ‘complete’ the behavior of entering manual mode after \( s' \), but this behavior can be finished in its delay-free counterpart \( \text{SUP} \). So, the observer property (19) required by bounded delay-robustness is violated when the delay bound \( d \) exceeds 4 ticks, and we conclude that the maximal delay bound for event 10 is 4.

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

(3) All communication events

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

First, use \( \text{CH}(T, 30, O) \), \( \text{CH}_{4}(V, 10, O) \) and \( \text{CH}_{4}(V, 14, O) \) to transmit events 30, 10 and 14
Figure 13. Distributed Control Architecture of ULTC: the labels on the dashed arrows denote the events to be communicated.

![Diagram of Distributed Control Architecture of ULTC](image1)

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

![Diagram of Communication Channel $CH(T, 30, O)$](image2)

Figure 15. Communication Channels $CH(V, 10, O)$ and $CH_4(V, 10, O)$

![Diagram of Communication Channels $CH(V, 10, O)$ and $CH_4(V, 10, O)$](image3)

respectively. Second, connected by these channels, the overall system behavior is

$$SUP'_{com} = SUP_V \parallel SUP_T \parallel CH_4(V, 10, O) \parallel CH_4(V, 14, O) \parallel CH(T, 30, O) \parallel SUP''_O$$

over the augmented alphabet $\{10, 11, ..., 43, 10', 14', 30'\}$, where $SUP''_O$ is obtained by replacing 10, 14, and 30 by $10'$, $14'$ and $30'$ respectively. Third, one can verify that: (1) $SUP'_{com}$ is correct and complete, and (2) $CH(T, 30, O)$, $CH_4(V, 10, O)$ and $CH_4(V, 14, O)$ will not cause uncontrollability with respect to the uncontrollable communication events. Finally, we conclude that the overall system behavior is still optimal and nonblocking.
6. Conclusions

In this paper we have studied communication delays among local controllers obtained by supervisor localization 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 $d_{\text{max}}$ such that the system is $d_{\text{max}}$-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 behavior is still optimal and nonblocking.

When the system and its supervisor are 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 synthesize the monolithic supervisor efficiently. The second approach is to employ the decentralized control architecture. Instead of computing the monolithic supervisor, we first synthesize a set of decentralized supervisors to achieve global optimality and nonblocking (Cai & Wonham, 2010b; Feng & Wonham, 2008), and then verify delay-robustness of each decentralized supervisor. By this reasoning, the overall system is delay-robust if all the decentralized supervisors are delay-robust.

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].

Appendix A. Proof of Lemma 1

To prove Lemma 1, we need the following Lemmas 3 and 4.

**Lemma 3:** For any delay bound $d \geq 1$, there hold

\[ L(SUP) \subseteq PL(SUP_d') \quad (A1) \]
\[ L_m(SUP) \subseteq PL_m(SUP_d') \quad (A2) \]

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

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

**Lemma 4:** Let $t = x_1\sigma x_2\sigma' x_4 \in L_m(SUP_d')$ where $x_1, x_2, x_3$ and $x_4$ are strings free of $\sigma$ and $\sigma'$, i.e. $x_1, x_2, x_3, x_4 \in (\Sigma - \{\sigma\})^*$. Then $t' := x_1\sigma x_2\sigma' x_3 x_4 \in L_m(SUP_d')$.

**Proof of Lemma 4:** Recall that $SUP_d'$ is $SUP_i$ with transitions labeled $\sigma$ relabeled $\sigma'$. By definition of synchronous product, $x_2, x_3$ and $\sigma'$ can be re-ordered without affecting the membership of $t$ in $L_m(SUP_d')$, namely the strings $t'$ formed from $t$ by the successive replacement

$$x_1\sigma x_2 x_3\sigma' x_4 \rightarrow x_1\sigma' x_2 x_3 x_4$$

$$\rightarrow x_1 x_2 x_3\sigma' x_4$$

will belong to $L_m(SUP_d')$ as well. In other words, if the transmission of $\sigma$ is completed in a shorter time (the number of ticks in $x_2$ will be smaller than that in $x_2 x_3$), the behavior is still legal. ■

**Proof of Lemma 1:** We prove Lemma 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 $PL(SUP_d') \supseteq L(SUP)$ and $PL(SUP_d') \subseteq L(SUP_d)$ in sequence. $PL(SUP_d') \supseteq L(SUP)$ is obtained from Lemma 3 immediately. By inspection of the transition diagram of $CH_d(j, \sigma, i)$ in Fig. 5, we get that $L(CH_d(j, \sigma, i)) \subseteq L(CH_{d+1}(j, \sigma, i))$. So according to (16),

$$L(SUP_d') \subseteq L(SUP_{d+1}) \quad \text{(A3)}$$

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

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

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

By (A3), we have $s \in L(SUP_{d+1})$. Since $SUP$ is $(d + 1)$-bounded delay-robust, there exists a string $u \in \Sigma^*$ such that $Pv = w$ and $sv \in L_m(SUP_d')$. Here we consider the case that only one instance of $\sigma$ exists in $sv$; the general cases can be confirmed similarly (since the transmission of multiple instances of $\sigma$ does not result in mutual interference). In the following, we prove (19) from these three cases: (i) $su = s_1\sigma s_2\sigma' s_3 u_1 u_2$, (2) $su = s_1 s_2 u_1 \sigma' u_2$, and (iii) $s_1 s_2 u_1 \sigma s_2 u_2 u_3$, where $s_1, s_2, s_3, u_1, u_2, u_3$ are free of $\sigma$ and $\sigma'$.

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

(ii) $su = s_1 s_2 u_1 \sigma u_2$. By Lemma 4, it results from $su \in L_m(SUP_d')$ that $s_1 s_2 \sigma' u_1 u_2 \in
Appendix B. Proof of Lemma 2

Since SUP is not delay-robust wrt. CH(j, σ, i), by Definition 1, one of the conditions (12)-(15) is violated. In the following, we prove that in each case, \( d_{max} \leq 2^m \cdot m \), where \( m \) is the state size of SUP' in (11).

(1) Condition (12) is violated. Since that \( L(SUP) \subseteq PL(SUP') \) always holds (similar to Lemma 3), we have \( PL(SUP') \not\subseteq L(SUP) \). So, there exists at least one string \( s \in \Sigma^* \) such that \( s \in L(SUP') \), but \( Ps \not\in L(SUP) \). We claim that \( s \) can be written as \( s'1 \sigma w \) where \( s_1, w \in \Sigma^* \); otherwise, \( s \) does not contain any \( \sigma \), and it follows from the construction of SUP' that \( Ps \in L(SUP) \), a contradiction. As illustrated in Fig. B1, we prove in the following that there exist strings \( s'_1 \in L(SUP') \) and \( w' \in \Sigma^* \) such that \( \#tick(w') \leq 2^m \cdot m \) (where \( \#tick(w') \) represents the number of events tick appearing in string \( w' \)), \( s'1 \sigma w' \in L(SUP') \), but \( Ps'1 \sigma w' \not\in L(SUP) \), from which we can conclude: to prevent the occurrence of string \( s'_1 \sigma w' \), the maximal communication delay of \( \sigma \) must be less than \( \#tick(w') \leq 2^m \cdot m \), i.e. \( d_{max} \leq 2^m \cdot (m' + 1) \).

By \( s_1 \sigma w \in L(SUP') \) and \( Ps_1 \sigma w \not\in L(SUP) \), we have \( P(s_1 \sigma w) \in PL(SUP') \cap (\Sigma^* - L(SUP)) \). To identify such strings, we build an TDES XL = \((Z, \Sigma, \zeta, z_0, Z_m)\) such that

\[
L_m(XL) = PL(SUP') \cap (\Sigma^* - L(SUP))
\]

and

\[
L(XL) = PL(SUP'),
\]

i.e., \( P(s_1 \sigma) \in L(XL) \), and \( P(s_1 \sigma w) \in L_m(XL) \).
First, we build $X\mathcal{A}$ such that $L_m(X\mathcal{A}) = PL(\text{SUP}')$ and $L(X\mathcal{A}) = L_m(X\mathcal{A})$ by the following two steps: (i) construct $PSUP'$ by applying the subset construction algorithm on $\text{SUP}'$ with natural projection $P$, and (ii) obtain $X\mathcal{A}$ by marking all states of $PSUP'$. Second, we build $X\mathcal{B}$ such that $L_m(X\mathcal{B}) = \Sigma^* - L(\text{SUP})$ and $L(X\mathcal{B}) = \Sigma^*$ by first adjoining a (non-marker) dump state $\hat{q}$ to the state set of $\text{SUP}$ and transitions $(q, \sigma, \hat{q})$ for each state $q$ of $\text{SUP}$ if $\sigma \in \Sigma$ is not defined at $\hat{q}$ (i.e. $L(X\mathcal{B}) = \Sigma^*$), and secondly setting $\hat{q}$ to be the only marker state. Third, let $X\mathcal{B} = X\mathcal{A} \| X\mathcal{B}$; then $L_m(X\mathcal{B}) = PL(\text{SUP}') \cap (\Sigma^* - L(\text{SUP}))$, $L(X\mathcal{B}) = PL(\text{SUP}')$. The state size $|Z| \leq 2^m \times (m' + 1)$, since $X\mathcal{A}$ has at most $2^m$ states (due to the subset construction algorithm), and $X\mathcal{B}$ has $m' + 1$ states.

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

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

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

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

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

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

\section*{a}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.
By \(P(s_1\sigma s_2)w \in L_m(\text{SUP}) \subseteq PL_m(\text{SUP}^*)\), there must exist strings \(s'_1, s'_2, \) and \(v'\) such that \(Ps'_1 = Ps_1, Ps'_2 = Ps_2, P\nu' = w\), and \(s'_1\sigma s'_2 v' \in L_m(\text{SUP}^*)\), as displayed in Fig. B1. 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 \(\text{SUP}^*\), there must exist a string \(s'' \in L(\text{SUP}^*)\) 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 the condition (14) is also violated by the string pair \(s_1\sigma t\) and \(s'_1\sigma t''\). Because \(t\) is simple, \(#\text{tick}(t) \leq m\), where \(m\) is the state size of \(P\)-normal form of \(\text{SUP}^*\). So, when \(\text{SUP}^*\) is not \(P\)-normal, \(#\text{tick}(t) \leq m \times 2^m\). In addition, since \(Pt'' = Pt\), \(#\text{tick}(t'') = #\text{tick}(t) \leq m \times 2^m\), as required.

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

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

References


