Communication of Distributed Discrete-Event Supervisors on a Switched Network

Klaus Schmidt, Ece Güran Schmidt

***Abstract*— In order to tackle the controller synthesis problem for large-scale discrete-event systems, recent approaches suggest the design of interacting *modular* or *decentralized* supervisors. In these works, information exchange between the supervisors is either required implicitly by the synchronization of *shared events* or explicitly by the communication of events or symbols. However, it is not discussed how the communication can be realized if the supervisors are implemented in distributed controller devices that are connected by a communication network. In this paper, we study the synchronization of shared events among distributed supervisors on a *switched network*. In particular, we develop a communication model that accounts for possible transmission delays, and enables the correct operation of the communicating supervisors.**

I. Introduction

The efficient design of supervisors for discrete-event systems (DES) has been an area of intensive study in recent years. As a result, a variety of approaches that suggest the synthesis of *modular* and *decentralized* supervisors have been developed. In these works, interaction between supervisors is represented either implicitly by the use of *shared events* that have to occur synchronously in all supervisors [1], [2], [3], [4], [5], [6] or explicitly by introducing *communication channels* between supervisors as in [7], [8], [9].

The above representation of supervisor interaction is particularly beneficial for the supervisor design in order to determine *when* communication is required. However, the above approaches do not address *how* to model and realize the communication in the practical case, where the designed supervisors are placed in different physical locations and communicate over a network with possible delays.

A preliminary study of this problem is given in [10] in the framework of decentralized supervisory control. Furthermore, in our previous work in [11], [12], the *shared-medium* communication of supervisors synthesized as in [3], [4] has been investigated. Here, the potential *collisions* of messages on the shared medium are avoided by an appropriate scheduling strategy. In this paper, we discuss the implementation of distributed supervisors on a *switched network* which is inherently collision free. However, it now has to be considered that the order of message receptions can deviate from the order of message transmissions due to the network properties such as queuing delays in the switches. To this end, we propose *communication models* for each supervisor that capture the information about when data exchange with other supervisors is required. Together with the behavior of the switched network, these communication models ensure correct operation of the distributed supervisors.

The paper outline is as follows. In Section II, we describe the underlying hierarchical control approach. Our communication model is developed in Section III, and the correct operation of the communication system is established in Section IV. Section V discusses the proposed communication strategy, and we give conclusions in Section VI.

K. Schmidt is with the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, Germany, klaus.schmidt@rt.eei.uni-erlangen.de

E. G. Schmidt is with the Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, eguran@metu.edu.tr

II. Preliminaries

A. Basic Notation

We recall basic notions from [13].

For a finite alphabet Σ, the set of all finite strings over Σ is denoted Σ^*. We write $s_1 s_2 \in \Sigma^*$ for the concatenation of two strings $s_1,\ s_2 \in \Sigma^*$. We write $s_1 \leq s$ when s_1 is a *prefix* of s, i.e. if there exists a string $s_2 \in \Sigma^*$ with $s = s_1 s_2$. The empty string is denoted $\epsilon \in \Sigma^*$, i.e. $s\epsilon = \epsilon s = s$ for all $s \in \Sigma^*$. A *language* over Σ is a subset $M \subseteq \Sigma^*$. The *prefix closure* of M is defined by $\overline{M} := \{s_1 \in \Sigma^* | \exists s \in M \text{ s.t. } s_1 \leq s\}$. If M is *prefix closed*, i.e., $M = \overline{M}$, then for any string $s \in M$, $\Sigma_M(s) := \{\sigma | s\sigma \in M\}$ is the set of feasible events after s.

The *natural projection* $p_i : \Sigma^* \to \Sigma_i^*$, $i = 1, 2$, for the union $\Sigma = \Sigma_1 \cup \Sigma_2$ is defined iteratively: (1) let $p_i(\epsilon) := \epsilon$; (2) for $s \in \Sigma^*$, $\sigma \in \Sigma$, let $p_i(s\sigma) := p_i(s)\sigma$ if $\sigma \in \Sigma_i$, or $p_i(s\sigma) := p_i(s)$ otherwise. The set-valued inverse of p_i is denoted $p_i^{-1} : \Sigma_i^* \to 2^{\Sigma^*}$, $p_i^{-1}(t) := \{s \in \Sigma^* | p_i(s) = t\}$. The *synchronous product* $M_1 || M_2 \subseteq \Sigma^*$ of two languages $M_i \subseteq \Sigma_i^*$ is $M_1 || M_2 = p_1^{-1}(M_1) \cap p_2^{-1}(M_2) \subseteq \Sigma^*$.

A *finite automaton* is a tuple $R = (X, \Sigma, \delta, x_0, X_m)$, with the finite set of *states* X; the finite alphabet of *events* Σ; the partial *transition function* $\delta : X \times \Sigma \to X$; the *initial state* $x_0 \in X$; and the set of *marked states* $X_m \subseteq X$. We write $\delta(x, \sigma)!$ if δ is defined at (x, σ), and define $\Sigma_R : X \to 2^\Sigma$ s.t. for $x \in X$, $\Sigma_R(x) := \{\sigma \in \Sigma | \delta(x, \sigma)!\}$. In order to extend δ to a partial function on $X \times \Sigma^*$, recursively let $\delta(x, \epsilon) := x$ and $\delta(x, s\sigma) := \delta(\delta(x, s), \sigma)$, whenever both $x' = \delta(x, s)$ and $\delta(x', \sigma)!$. $L(R) := \{s \in \Sigma^* : \delta(x_0, s)!\}$ and $L_m(R) := \{s \in L(R) : \delta(x_0, s) \in X_m\}$ are the *closed* and *marked language* generated by the finite automaton R, respectively. R is denoted *nonblocking* if $L(R) = \overline{L_m(R)}$. A formal definition of the synchronous composition of two automata R_1 and R_2 can be taken from e.g. [14].

B. Distributed Discrete Event Controllers

In this paper, we develop a networked implementation of distributed DES supervisors according to the hierarchical and

decentralized supervisory control approach in [3], [4]. First, we give a brief description of the control architecture and present several properties that will support our study.

The outcome of the chosen control method is a set $\mathcal{R} = \{R_1, \ldots, R_n\}$ of n supervisors in a hierarchical relationship as in Fig. 1 (a). Each supervisor is represented by a finite automaton $R_i = (X_i, \Sigma_i, \delta_i, x_{0,i}, X_{\mathrm{m},i})$ that recognizes the respective closed-loop behavior. The hierarchical relationship can be formally described by a directed tree $T_{\mathcal{R}} = (\mathcal{R}, R_n, c_{\mathcal{R}}, p_{\mathcal{R}})$ (see e.g., [15]). In this paper, $\mathcal{R}$ denotes the set of *vertices*, R_n is the *root vertex* and $c_{\mathcal{R}} : \mathcal{R} \to 2^{\mathcal{R}}$ and $p_{\mathcal{R}} : \mathcal{R} \to \mathcal{R}$ are the *children map* and the *parent map* such that $c_{\mathcal{R}}(R_k)$ is the *set of children* and $p_{\mathcal{R}}(R_k)$ is the parent of $R_k \in \mathcal{R}$, respectively. Note that the unique highest-level supervisor R_n does not have a parent, and any vertex without children is called a *leaf*.

With the above notation, the characteristic features of the control architecture are as follows. For each supervisor $R_k \in \mathcal{R} - \{R_n\}$, there is an *abstraction alphabet* $\hat{\Sigma}_k \subseteq \Sigma_k$, and it holds that $\hat{\Sigma}_k = \bigcup_{i=1, i\neq k}^{n} (\Sigma_i \cap \Sigma_k)$, i.e., $\hat{\Sigma}_k$ contains all *shared events* with other supervisors. Also let $\Sigma_{\mathrm{s}} := \bigcup_{k=1}^{n-1} \hat{\Sigma}_k$ be the overall set of shared events. The abstracted supervisor $\hat{R}_k = (\hat{X}_k, \hat{\Sigma}_k, \hat{\delta}_k, \hat{x}_{0,k}, \hat{X}_{\mathrm{m},k})$ is then defined such that $L(\hat{R}_k) = p_{\Sigma_k \to \hat{\Sigma}_k}(L(R_k))$ and $L_{\mathrm{m}}(\hat{R}_k) = p_{\Sigma_k \to \hat{\Sigma}_k}(L_{\mathrm{m}}(R_k))$ with the natural projection $p_{\Sigma_k \to \hat{\Sigma}_k} : \Sigma_k^* \to \hat{\Sigma}_k^*$. Consequently, for each controller $R_k \in \mathcal{R}$ that is not a leaf of $T_{\mathcal{R}}$, it holds that

$$\Sigma_k = \bigcup_{l, R_l \in c_{\mathcal{R}}(R_k)} \hat{\Sigma}_l \quad \text{and} \quad L(R_k) \subseteq \parallel_{l, R_l \in c_{\mathcal{R}}(R_k)} L(\hat{R}_l) \ , \tag{1}$$

i.e., each parent restricts the abstracted behavior of its children. Due to the control architecture in [3], [4], the overall closed-loop system is represented by a finite automaton $R := ||_{k=1}^{n} R_k$ over the alphabet $\Sigma := \bigcup_{k=1}^{n} \Sigma_k$. Furthermore, it is ensured that R is nonblocking, i.e.,

$$L(R) = \overline{L_{\mathrm{m}}(R)}. \tag{2}$$

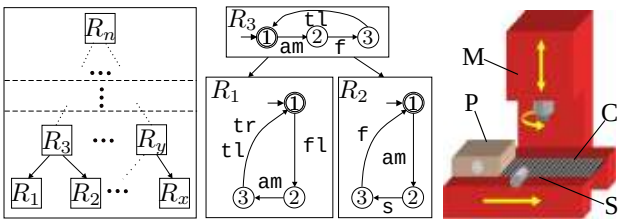

Fig. 1. (a) Control Architecture; (b) and (c) Example: Manufacturing Unit.

Example 1 illustrates the hierarchical control architecture.

Example 1: Fig. 1 (b) shows a hierarchical architecture with two levels and $n = 3$ automata. It describes the operation of a manufacturing unit with a conveyor belt C (R_1) and a machine M (R_2, see Fig. 1 (c)) that is controlled by a high-level supervisor R_3. At the conveyor belt, a product P can be transported from left (`fl`) before it arrives at the machine (sensor S), which is indicated by the shared event `am` (product at machine). Then, P can leave the conveyor belt to the left (`tl`) or to the right (`tr`). After `am`, the machine starts processing (`s`) and finishes processing (`f`) after some time. The high-level supervisor R_3 over $\Sigma_3 = \{\texttt{am}, \texttt{tr}, \texttt{tl}, \texttt{f}\}$ ensures that the shared events `am`, `f` and `tl` occur such that the product is not transported to the left before the machine finished processing, and `tr` is always disabled.

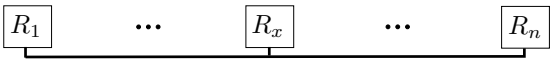

Fig. 2. Nodes on a Shared Medium.

III. Communication Model

A. Previous Work

In a large-scale DES (e.g., on a factory floor), the synthesized supervisors are usually not implemented on a single controller device (e.g., PLC) but rather distributed to several controller devices or *network nodes* that are placed in distinct physical locations and connected by a network. Accounting for the interaction of the supervisors, this means that the occurrence of each shared event $\sigma \in \Sigma_{\mathrm{s}}$ has to be synchronized among all supervisors that share σ via an appropriate communication strategy.

This issue was first addressed in [11], [12] under the assumption of *shared-medium* communication, i.e., all network nodes use the same communication medium to exchange information (see Fig. 2). In these works, two main characteristics of shared-medium communication were considered.

First, it had to be taken into account that *collisions* can happen on a shared medium, i.e., several nodes might want to transmit a communication *message* at the same time. Hence, in [11], a *question-answer-command* strategy was introduced to synchronize each shared event occurrence while avoiding collisions. Starting from the highest-level supervisor, *questions* are propagated along the hierarchy to the lower-level supervisors. These, in turn, provide the information if the shared event is currently possible in the form of an *answer* to their respective higher-level supervisors. If all answers are present at the highest-level supervisor, a *command* is issued that triggers the occurrence of the shared event. Formally, this communication idea was embedded into the individual supervisors, and a finite automata representation of the resulting *communication models* was derived.

Second, the possibility of *synchronous broadcast* could be exploited by assuming that each message transmitted by a network node is received by the other nodes synchronously. Hence, the *synchronous composition* of the communication models could be used to verify that the behavior of the distributed and networked control system complies with the behavior of the original closed-loop system.

B. Communication Strategy for Switched Networks

In this paper, we employ the basic ideas of [11] in order to model the required communication in the case of a *switched network* as depicted in Fig. 3. Here, we assume that each supervisor R_k is implemented in an individual network node

k that has a distinct *full-duplex* connection to a unique switch S_m, i.e., there are separate lines for incoming and outgoing messages. Hence, different from the shared-medium case, the switched network is collision free. As sending a *question* in our previous communication model was solely required to resolve collisions, we suggest an adapted communication model with only *notifications* and *commands*.

Our communication strategy is based on the idea that information about the feasibility of each shared event $\sigma \in \Sigma_{\mathrm{s}}$ is propagated along the supervisor hierarchy starting from the lowest-level supervisors that share σ and towards the highest-level supervisor that shares σ. That is, each supervisor $R_l \in \mathcal{R}$ s.t. $\sigma \in \hat{\Sigma}_l$ sends a *notification* event σ_{R_l} ("σ is feasible in R_l") to its parent supervisor $R_k = p_{\mathcal{R}}(R_l)$, whenever σ becomes feasible. The parent R_k, in turn, collects the notifications from all children that share σ. As soon as all such notifications have been received, there are two possibilities. If $\sigma \in \hat{\Sigma}_k$, i.e., the parent $R_j = p_{\mathcal{R}}(R_k)$ of R_k also shares σ, then R_k sends its own notification σ_{R_k} to R_j. If $\sigma \notin \hat{\Sigma}_k$, then R_k is the highest-level supervisor that shares σ. Hence, R_k can command the execution of σ.

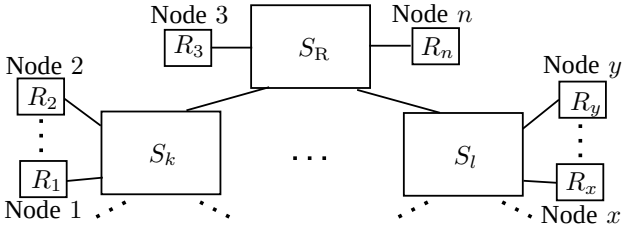

Fig. 3. Nodes on a Switched Network.

Note that the proposed communication strategy makes use of the hierarchical control architecture. Each supervisor only needs to know its respective parent supervisor for sending notifications about the feasibility of shared events $\sigma \in \Sigma_{\mathrm{s}}$. Likewise, receiving notifications for a shared event $\sigma \in \Sigma_{\mathrm{s}}$, only requires information about the respective children supervisors. This property will be used to model the communication behavior in Section III-C.

At this point, it has to be emphasized that, different from the shared medium case, synchronous broadcast cannot be assumed for a switched network as messages can for example be delayed at the switch output ports while traversing the network. Hence, we describe the receiving action of a previously sent notification σ_{R_k} at R_j by a different event $!\sigma_{R_k}$. If σ is feasible in all supervisors that share σ, R_j can command the execution of σ, where it has to be made sure that still each supervisor R_k that shares σ can participate in its execution. We first illustrate this concept in the following example and then develop the communication model.

Example 2: We assume that each supervisor in Fig. 1 (b) is in its initial state. Then, the first shared event that can occur is `am`. While R_2 can immediately send a notification $\mathtt{am}_{R_2}$, R_1 has to wait until either `fr` or `fl` occurs before stating the feasibility of `am` by sending $\mathtt{am}_{R_1}$. After receiving the notifications $!\mathtt{am}_{R_2}$ and $!\mathtt{am}_{R_1}$, R_3 can confirm the occurrence of `am` by sending a command to both R_1 and R_2 which causes a state change from 2 to 3, 1 to 2, and 1 to 2 in R_1, R_2, and R_3, respectively. Note that in both R_1 and R_2, the command `am` is accepted as there are no possible event occurrences that could make `am` infeasible. Analogously, the occurrences of `f` and `tr` are synchronized among the participating supervisors.

As stated above, the communication idea relies on the fact that whenever a lower-level supervisor R_k confirms that an event σ is feasible, the event must stay feasible until a higher-level supervisor commands the execution of σ or another event. We express this requirement by an additional condition that has to be met by each automaton $R_k \in \mathcal{R}$. Whenever a high-level event $\sigma \in \hat{\Sigma}_k$ is possible in a state of R_k, then no low-level event in $\Sigma_k - \hat{\Sigma}_k$ is allowed to occur. If this condition is fulfilled, we denote R_k *communication consistent*. (A further discussion of this condition is provided in Section V-A.)

Definition 1 (Communication Consistency): Let $R_k \in \mathcal{R} - \{R_n\}$ and $\sigma \in \hat{\Sigma}_k$. R_k is communication consistent for σ if for all $s \in L(R_k)$ s.t. $\sigma \in \Sigma_{L(R_k)}(s)$

$$\Sigma_{L(R_k)}(s) \cap (\Sigma_k - \hat{\Sigma}_k) = \emptyset. \tag{3}$$

R_k is communication consistent if (3) holds for all $\sigma \in \hat{\Sigma}_k$.

C. Communication Model for Switched Networks

Assuming communication consistency for all supervisors $R_k \in \mathcal{R} - \{R_n\}$, the presented communication strategy is now embedded in a communication model.

1) Leaf Supervisors: First, we consider *leaf supervisors* R_k, i.e., $c_{\mathcal{R}}(R_k) = \emptyset$. As no event $\sigma \in \hat{\Sigma}_k$ is shared with a lower-level supervisor, no commands are issued by R_k. Hence, the desired operation of R_k is to send a notification σ_{R_k} to the parent supervisor $R_j = p_{\mathcal{R}}(R_k)$, whenever an event $\sigma \in \hat{\Sigma}_k$ becomes feasible. To this end, we define two types of automata that capture the desired behavior of R_k.

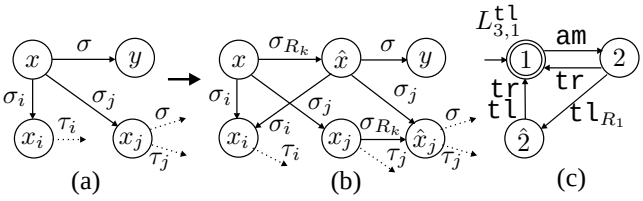

Fig. 4. (a) Subautomaton of $\hat{R}_k$; (b) Subautomaton of $L^{\sigma}_{j,k}$; (c) $L^{\mathtt{tl}}_{3,1}$.

The first automaton $L^{\sigma}_{j,k} = (X^{\sigma}_{j,k}, \Lambda^{\sigma}_{j,k}, \delta^{\sigma}_{j,k}, x^{\sigma}_{0,j,k}, X^{\sigma}_{\mathrm{m},j,k})$ over the alphabet $\Lambda^{\sigma}_{j,k} = \hat{\Sigma}_k \cup \{\sigma_{R_k}\}$ is constructed based on the abstracted supervisor $\hat{R}_k$. It expresses that exactly one notification σ_{R_k} is sent when σ becomes feasible. A procedure to determine $L^{\sigma}_{j,k}$ from $\hat{R}_k$ is outlined in Algorithm 1. For each state of $\hat{R}_k$, where σ is feasible, a state $\hat{x}$ is added in $L^{\sigma}_{j,k}$. It memorizes the transmission of the notification σ_{R_k} before σ can occur. Furthermore, Fig. 4 graphically illustrates the construction of $L^{\sigma}_{j,k}$. Fig. 4 (a) represents a subautomaton of $\hat{R}_k$, where σ is feasible in state x with $y = \hat{\delta}_k(x, \sigma)$, and $\sigma_i, \sigma_j, \tau_i, \tau_j \in \hat{\Sigma}_k$ are other events such that σ_i, σ_j are feasible in x with $x_i = \hat{\delta}_k(x, \sigma_i)$, $x_j = \hat{\delta}_k(x, \sigma_j)$. Also, τ_i

and σ, τ_j are feasible in x_i and x_j, respectively. Then $L^{\sigma}_{j,k}$ is obtained from $\hat{R}_k$ by replacing each subautomaton as in Fig. 4 (a) by the subautomaton as in Fig. 4 (b). Analogous to Algorithm 1, the new states $\hat{x}$ and $\hat{x}_j$ are added to record exactly one occurrence of σ_{R_k} before σ itself is feasible even if another event (such as σ_j) occurs. As an example, $L^{\mathtt{tl}}_{3,1}$ for R_1 and R_3 in Fig. 1 is shown in Fig. 4 (c).

Algorithm 1 (Computation of $L^{\sigma}_{j,k}$): Let $\hat{R}_k$ be given as in Section II-B, and set $X^{\sigma}_{j,k} := \hat{X}_k$, $\Lambda^{\sigma}_{j,k} := \hat{\Sigma}_k \cup \{\sigma_{R_k}\}$, $x^{\sigma}_{0,j,k} := \hat{x}_{0,k}$, and $X^{\sigma}_{\mathrm{m},j,k} := \hat{X}_{\mathrm{m},k}$. The transition function $\delta^{\sigma}_{j,k}$ is defined as follows.

for all $x \in \hat{X}_k$ s.t. $\hat{\delta}_k(x,\sigma)!$
  $X^{\sigma}_{j,k} := X^{\sigma}_{j,k} \cup \{\hat{x}\}$
  $\delta^{\sigma}_{j,k}(x, \sigma_{R_k}) := \hat{x}$ and $\delta^{\sigma}_{j,k}(\hat{x}, \sigma) := \delta(x, \sigma)$
for all $x \in \hat{X}_k$
  for all $\hat{\sigma} \in \Sigma_{\hat{R}_k}(x) - \{\sigma\}$
    $y := \delta(x, \hat{\sigma})$ and $\delta^{\sigma}_{j,k}(x, \hat{\sigma}) := y$
    if $\hat{x} \in X^{\sigma}_{j,k}$ and $\hat{y} \in X^{\sigma}_{j,k}$
      $\delta^{\sigma}_{j,k}(\hat{x}, \hat{\sigma}) := \hat{y}$
    if $\hat{x} \in X^{\sigma}_{j,k}$ and $\hat{y} \notin X^{\sigma}_{j,k}$
      $\delta^{\sigma}_{j,k}(\hat{x}, \hat{\sigma}) := y$

The second automaton L_k over the alphabet $\Gamma_k := \Sigma_k \cup \{\sigma_{R_k} | \sigma \in \hat{\Sigma}_k\}$ expresses that for each $\sigma \in \hat{\Sigma}_k$, the notification σ_{R_k} can only be given when σ is feasible in the original supervisor R_k. L_k is obtained by replacing the subautomaton of R_k if Fig. 5 (a) by the subautomaton in Fig. 5 (b), i.e., a selfloop with σ_{R_k} is added in each state where σ is feasible in R_k. L_1 for R_1 in Fig. 1 (b) illustrates this concept in Fig. 5 (c).[1]

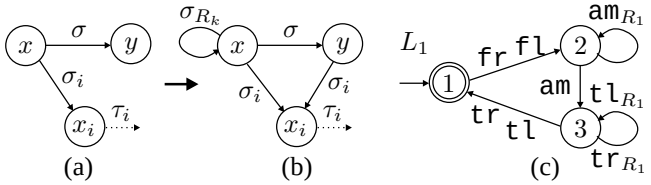

Fig. 5. (a) Subautomaton of R_k; (b) Subautomaton of L_k; (c) L_1.

Based on L_k and $L^{\sigma}_{j,k}$ for $\sigma \in \Sigma_k - \hat{\Sigma}_k$, we now define the *communication model* (CM) $C_k = (Q_k, \Gamma_k, \nu_k, q_{0,k}, Q_{\mathrm{m},k})$ over the alphabet Γ_k for any leaf supervisor R_k as follows.

$$C_k := L_k || (||_{\sigma \in \hat{\Sigma}_k} L^{\sigma}_{j,k}). \tag{4}$$

Semantically, C_k expresses that for each event $\sigma \in \hat{\Sigma}_k$, it holds that exactly one notification σ_{R_k} will be sent to the parent supervisor as soon as σ becomes feasible in R_k.

Example 3: The CMs $C_1 := L_1 || L^{\mathtt{am}}_{3,1} || L^{\mathtt{tr}}_{3,1} || L^{\mathtt{tl}}_{3,1}$ and $C_2 := L_2 || L^{\mathtt{am}}_{3,2} || L^{\mathtt{f}}_{3,2}$ for the leaf supervisors R_1 and R_2 in Fig. 1 (b), respectively, are depicted in Fig. 6.

2) Non-Leaf Supervisors: For notational convenience, we introduce the map $c^{\sigma}_{\mathcal{R}} : \mathcal{R} \to 2^{\mathcal{R}}$ s.t. $c^{\sigma}_{\mathcal{R}}(R_k) := \{R_l \in c_{\mathcal{R}}(R_k) | \sigma \in \Sigma_l\}$. It denotes all children of R_k that share the event σ. If $R_k \in \mathcal{R}$ is not a leaf supervisor, then for each $\sigma \in \Sigma_k$, there can be two situations: (i) R_k is the highest-level supervisor for σ, i.e., $\sigma \in \Sigma_k - \hat{\Sigma}_k$ or (ii) R_k is an intermediate-level supervisor for σ, i.e., $\sigma \in \hat{\Sigma}_k$.

[1]Algorithms for the remaining automata in this paper are given in [16].

(i): R_k receives notifications $!\sigma_{R_l}$ from the supervisors $R_l \in c^{\sigma}_{\mathcal{R}}(R_k)$ and can send the command for σ if all notifications have arrived. At this point, it has to be addressed that synchronous broadcast is not possible in a switched network. Consider that R_l sends σ_{R_l} as described in Fig. 4 (b). Then it can happen that R_k commands the execution of another event $\sigma_i \neq \sigma$, that is also feasible before the notification $!\sigma_{R_l}$ is received, i.e., the state of R_k changes according to the occurrence of σ_i. If σ is no longer feasible in R_k, this means that $!\sigma_{R_l}$ can no longer be accepted by R_k. As we intend to provide a CM that accepts all notifications, we incorporate the possible delay in our model. We describe the communication behavior of R_k w.r.t. R_l and the event $\sigma \in \Sigma_k - \hat{\Sigma}_k$ by the automaton $H^{\sigma}_{k,l}$ over the alphabet $\Pi^{\sigma}_{k,l} := \hat{\Sigma}_l \cup \{!\sigma_{R_l}\}$, where each subautomaton $\hat{R}_l$ as in Fig. 7 (a) is replaced by the subautomaton in Fig. 7 (b). Note that the receiving behavior in Fig. 7 (b) almost identically mirrors the sending behavior in Fig. 4 (b). The only difference is that a delayed notification $!\sigma_{R_l}$ is accepted in state x_i even if σ is no longer feasible. The additional state $\hat{x}_i$ is a copy of x_i in Fig. 7 (a), i.e., it has the same transitions as x_i. Also note that a possible delay is irrelevant for commands $\sigma \in \Sigma_k - \hat{\Sigma}_k$ that are sent by R_k since all supervisors that receive the command must be in a state where σ is feasible due to communication consistency in Definition 1. Hence, commands need not be split into distinct sending and receiving actions. $H^{\mathtt{tl}}_{3,1}$ for R_3 in Fig. 1 (b) and $\mathtt{tl}$ is shown in Fig. 7 (c).

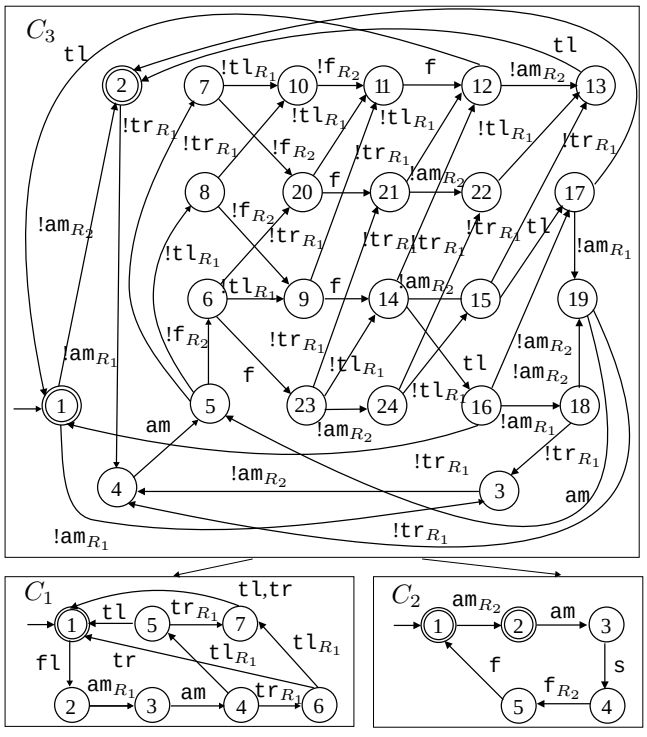

Fig. 6. Communication Models for the Manufacturing Unit.

For notational convenience, we combine all communication behaviors for an event σ and a supervisor R_k to form H^{σ}_k over the alphabet Π^{σ}_k.

$$H^{\sigma}_k := ||_{l, R_l \in c^{\sigma}_{\mathcal{R}}(R_k)} H^{\sigma}_{k,l} \text{ and } \Pi^{\sigma}_k := ||_{l, R_l \in c^{\sigma}_{\mathcal{R}}(R_k)} \Pi^{\sigma}_{k,l}. \tag{5}$$

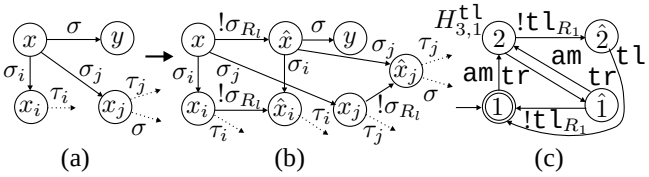

Fig. 7. (a) Subautomaton of $\hat{R}_l$; (b) Subautomaton of $H^{\sigma}_{k,l}$; (c) $H^{\mathtt{tl}}_{3,1}$.

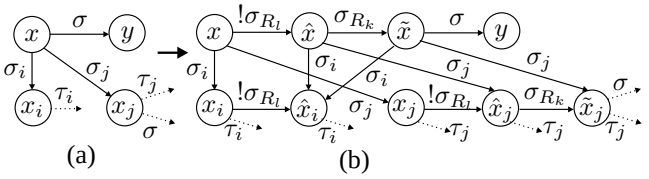

Fig. 8. (a) Subautomaton of $\hat{R}_l$; (b) Subautomaton of $I^{\sigma}_{k,l}$.

In case (ii), in addition to receiving $!\sigma_{R_l}$ from each $R_l \in c^{\sigma}_{\mathcal{R}}(R_k)$, R_k also has to send a notification σ_{R_k} to the parent supervisor $R_j = p_{\mathcal{R}}(R_k)$. In order to use the hierarchy efficiently, we suggest that R_k first collects all notifications $!\sigma_{R_l}$ from the children supervisors before sending its own notification σ_{R_k} to the parent. Hence, instead of $H^{\sigma}_{k,l}$ as in case (i), we propose to use an automaton $I^{\sigma}_{k,l}$ over the alphabet $\Theta^{\sigma}_{k,l} := \hat{\Sigma}_k \cup \{!\sigma_{R_l}, \sigma_{R_k}\}$ by replacing each subautomaton of $\hat{R}_l$ in Fig. 8 (a) by the subautomaton in Fig. 8 (b). Here, the reception of $!\sigma_{R_l}$ is analogous to Fig. 7 (b). The additional states $\tilde{x}$, $\tilde{x}_j$ express the fact that σ_{R_k} is only sent if $!\sigma_{R_l}$ has been received and σ is feasible.

Then, the communication behavior I^{σ}_k over the alphabet Θ^{σ}_k for R_k and $\sigma \in \hat{\Sigma}_k$ can be determined as

$$I^{\sigma}_k := \mathop{||}_{l, R_l \in c^{\sigma}_{\mathcal{R}}(R_k)} I^{\sigma}_{k,l} \text{ and } \Theta^{\sigma}_k := \bigcup_{l, R_l \in c^{\sigma}_{\mathcal{R}}(R_k)} \Theta^{\sigma}_{k,l}. \quad (6)$$

Using (5) and (6), the CM $C_k = (Q_k, \Gamma_k, \nu_k, q_{0,k}, Q_{\mathrm{m},k})$ over the alphabet $\Gamma_k := (\bigcup_{\sigma \in \hat{\Sigma}_k} \Theta^{\sigma}_k) \cup (\bigcup_{\sigma \in \Sigma_k - \hat{\Sigma}_k} \Pi^{\sigma}_k)$ for a supervisor R_k that is not a leaf in the hierarchical tree $T_{\mathcal{R}}$ is computed as the synchronous composition of its component communication behaviors and the original supervisor R_k.

$$C_k := (||_{\sigma \in \hat{\Sigma}_k} I^{\sigma}_k) || (||_{\sigma \in \Sigma_k - \hat{\Sigma}_k} H^{\sigma}_k) || R_k. \quad (7)$$

As all automata I^{σ}_k and H^{σ}_k are computed based on $\hat{R}_l$ for l such that $R_l \in c_{\mathcal{R}}(R_k)$, and $L(R_k) \subseteq ||_{l, R_l \in c_{\mathcal{R}}(R_k)} \hat{R}_l$ according to Section II-B, the synchronous composition with R_k in (7) is required to implement the actual control action. Also, the functionality of L_k for the computation of C_k in (4) is already captured by I^{σ}_k and H^{σ}_k, respectively.

Example 4: The CM $C_3 = H^{\mathtt{am}}_3 || H^{\mathtt{tr}}_3 || H^{\mathtt{tl}}_3 || H^{\mathtt{f}}_3$ for R_3 according to (7) is shown in Fig. 6. Note that in the states 16, 17, 18, and 19, the notification $!\mathtt{tr}_{R_1}$ is still accepted although the alternative event $\mathtt{tl}$ already occurred.

IV. Communication Operation

A. Behavior of the Communication System

In addition to the CM of the network nodes, the behavior of the switched network itself has to be considered. Assume R_k and R_l are supervisors with $\sigma \in \hat{\Sigma}_l$ and $R_k = p_{\mathcal{R}}(R_l)$. Then, R_l can send notifications σ_{R_l} according to C_l while R_k receives the notifications $!\sigma_{R_l}$ according to C_k. Furthermore, the construction of $H^{\sigma}_{k,l}$ in Fig. 7 suggests that whenever a

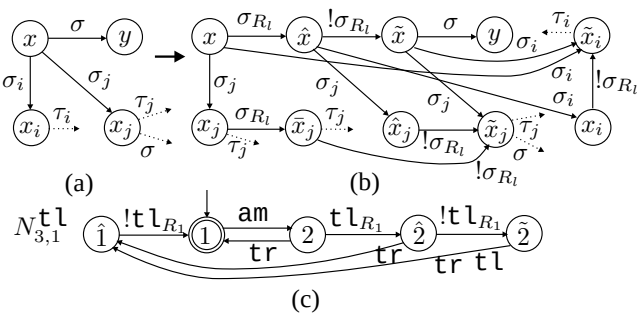

Fig. 9. (a) Subautomaton of $\hat{R}_l$; (b) Subautomaton of $N^{\sigma}_{k,l}$; (c) $N^{\mathtt{tl}}_{3,1}$.

command, e.g., σ_j is issued by R_k, an immediate subsequent command for another event can only be issued if that event was already feasible (e.g., σ in state $\hat{x}$) as otherwise first the notifications have to arrive at R_k (e.g., for τ_j). Hence, a delayed notification (σ_{R_l}) will always arrive before the next command (τ_j) can be issued. The *switched network model* $N^{\sigma}_{k,l}$ over $\Omega^{\sigma}_{k,l} := \hat{\Sigma}_l \cup \{\sigma_{R_l}, !\sigma_{R_l}\}$ for each event σ and the supervisors R_k and R_l captures this behavior. It is obtained from $\hat{R}_l$ by replacing each subautomaton in Fig. 9 (a) by the subautomaton in Fig. 9 (b). It expresses the alternate sending and receiving action for σ_{R_l} and $!\sigma_{R_l}$. In particular, delayed notifications ($!\sigma_{R_l}$) preempt subsequent commands (τ_j) as characterized in state $\hat{x}_j$. $N^{\mathtt{tl}}_{3,1}$ is depicted in Fig. 9 (c).

Combining the behavior of the network nodes represented by their respective CMs C_k and the behavior of the network characterized by the automata $N^{\sigma}_{k,l}$, the behavior of the overall communication system can be determined as an automaton C over the alphabet $\Gamma := \bigcup_{k=1}^{n} \Gamma_k$ as follows.

$$C := (||_{k=1}^{n} C_k) || (||_{l=1}^{n} (||_{\sigma \in \hat{\Sigma}_l} N^{\sigma}_{j,l})), \text{ and } j = p_{\mathcal{R}}(R_l). \quad (8)$$

Example 5: For the communication system as in Fig. 6, the overall communication behavior is represented by

$$C = (||_{k=1}^{3} C_k) || N^{\mathtt{am}}_{3,1} || N^{\mathtt{am}}_{3,2} || N^{\mathtt{f}}_{3,2} || N^{\mathtt{tr}}_{3,1} || N^{\mathtt{tl}}_{3,1}.$$

B. Correct Operation of the Communication System

The CMs in (4) and (7) were constructed in order to model the required information exchange for the synchronization of shared events in the hierarchical control system described in Section II-B. In this section, we verify that the CMs indeed express the same nonblocking behavior as the original control system. To this end, it has to be shown that for any communication sequence $c \in L(C)$, (i) its corresponding original string $s := p_{\Gamma \to \Sigma}(c)$ is an element of $L(R)$ (the communication system complies with the designed closed-loop behavior), and (ii) there is an extension $d \in \Gamma^*$ s.t. $cd \in L(C)$ and $p_{\Gamma \to \Sigma}(cd) \in L_{\mathrm{m}}(R)$ (the communication system is always able to reach a marked state in the original closed-loop behavior). Theorem 1 states the desired result. A proof of Theorem 1 is provided in [16].

Theorem 1: Let $c \in L(C)$ and $s := p_{\Gamma \to \Sigma}(c)$. Then $s \in L(R)$ and $\exists d \in \Gamma^*$ s.t. $cd \in L(C)$ and $p_{\Gamma \to \Sigma}(cd) \in L_{\mathrm{m}}(R)$.

In this section, a CM C_k that is intended for communication on a switched network was determined for each supervisor R_k. Furthermore, it has been shown in Theorem 1 that communication according to the CMs C_k, $k = 1, \ldots, n$ does not affect the original behavior of the supervisors R_k,

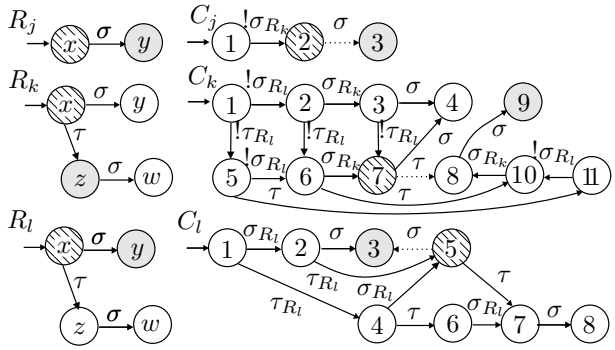

Fig. 10. Supervisors R_j, R_k and R_l; CMs C_j, C_k and C_l.

$k = 1, \ldots, n$. In particular, the nonblocking behavior that is guaranteed by the original supervisors is preserved.

V. Discussion

In this section we provide a brief discussion of the communication consistency condition as required in Definition 1 and the level-wise propagation of notifications.

A. Communication Consistency

Communication consistency implies that whenever a high-level event $\sigma \in \hat{\Sigma}_k$ is feasible in a state of the supervisor R_k, then no low-level event in $\Sigma_k - \hat{\Sigma}_k$ is allowed to occur. In [11], a less restrictive condition was employed for shared-medium networks: for any event $\sigma \in \hat{\Sigma}_k$ that is feasible in a state of R_k, there must not be a local string such that σ is no longer possible, i.e. $\forall s \in L(R_k), \sigma \in \hat{\Sigma}_k$ with $s\sigma \in L(R_k)$

$$\nexists u \in (\Sigma_k - \hat{\Sigma}_k)^* \text{ s.t. } su \in L(R_k) \wedge su\sigma \notin L(R_k). \quad (9)$$

Requiring (9), the following undesirable situation can occur in the switched network case due to the lack of synchronous broadcast. Assume that the three supervisors R_j, R_k and R_l are given as in Fig. 10, where $R_j = p_{\mathcal{R}}(R_k)$ and $R_k = p_{\mathcal{R}}(R_l)$. R_j is the highest-level supervisor for σ, and R_k is the highest-level supervisor for τ. Now assume that each of the corresponding CMs is in the hatched state (compare also the corresponding states x in the supervisors), i.e., all notifications for σ and τ have been received by the respective CMs. Then, it can happen that R_k issues the command τ, while at the same time R_j commands σ. As the order in which the commands arrive at R_l is arbitrary due to the switched network, it is possible that first the command for σ arrives. Hence, R_l has to ignore the subsequent command τ, and the supervisors transition the incompatible states shaded in gray in Fig. 10.

B. Propagation of Notifications

In Section III-C, we propose to propagate the information about the feasibility of shared events $\sigma \in \Sigma_s$ from one level of the supervisor hierarchy to the next level. An alternative solution could be to send notifications for σ directly to the highest-level supervisor that shares σ. However, it turns out that this implies that the CM of the respective highest-level supervisor has to include the communication behavior of all lower-level supervisors that share σ, which compromises the computational savings of the hierarchical control approach.

VI. Conclusions

In this paper, the *distributed* implementation of hierarchical discrete-event supervisors on a *switched network* is addressed. In particular, we develop a communication strategy that synchronizes the occurrence of *shared events* while respecting the properties of the switched network and the hierarchical control architecture. Hence, our *communication model* suggests the level-wise propagation of information about the feasibility of shared events, and it accommodates messages that arrive out of sequence. We introduce *communication consistency*, and show that the behavior of the resulting networked system complies with the behavior of the originally designed supervisors if this condition is fulfilled.

In future work, we will combine the results in this paper with the results in [17], in order to derive bounds for the message delays that are encountered by distributed discrete-event supervisors on switched networks. Furthermore, a study of the communication behavior of a large-scale manufacturing system with 50 distributed supervisors is under way.

References

[1] M. H. de Queiroz and J. E. R. Cury, "Modular supervisory control of large scale discrete event systems," in *Workshop on Discrete Event Systems*, 2000.

[2] R. J. Leduc, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control-Part II: Parallel case," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1336–1348, 2005.

[3] K. Schmidt, *Hierarchical Control of Decentralized Discrete Event Systems Theory and Application*. Ph.D. Thesis, Technische Fakultät der Universität Erlangen-Nürnberg, 2005.

[4] K. Schmidt, T. Moor, and S. Perk, "Nonblocking hierarchical control of decentralized discrete event systems," *accepted to appear in IEEE Transactions on Automatic Control*, 2008.

[5] R. Hill and D. Tilbury, "Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction," *Workshop on Discrete Event Systems*, 2006.

[6] L. Feng and W. M. Wonham, "Computationally efficient supervisor design: Abstraction and modularity," in *Workshop on Discrete Event Systems*, 2006.

[7] K. C. Wong and J. H. van Schuppen, "Control of discrete-event systems with communication," in *Workshop on Discrete Event Systems*, 1996.

[8] G. Barrett and S. Lafortune, "Decentralized supervisory control with communicating controllers," *IEEE Transactions on Automic Control*, vol. 45, no. 9, pp. 1620–1638, 2000.

[9] S. L. Ricker and B. Caillaud, "Mind the gap: Expanding communication options in decentralized discrete-event control," in *Conference on Decision and Control*, 2007.

[10] P. Gohari and A. Mannani, "Discrete-event control over communication networks," in *European Control Conference*, 2007.

[11] K. Schmidt, E. G. Schmidt, and J. Zaddach, "A shared-medium communication architecture for distributed discrete event systems," in *Mediterranean Conference on Control and Automation*, 2007.

[12] ——, "Reliable and safe operation of distributed discrete-event controllers: A networked implementation with real-time guarantees," in *IFAC World Congress*, 2008.

[13] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, 1979.

[14] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[15] J. E. Hopcroft and J. D. Ullman, *The design and analysis of computer algorithms*. Addison-Wesley, 1975.

[16] K. Schmidt and E. G. Schmidt, "Communication of distributed discrete-event supervisors on a switched network," *Technical Report, Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg*, 2008.

[17] ——, "A shortest-path problem for evaluating the worst case message delay of switched Ethernet," *to be submitted to IEEE Transactions on Industrial Electronics*, 2008.