# Controller Aggregation for Distributed Discrete-Event Supervisors on a Shared-medium Network

**Klaus Schmidt** *

* *Chair of Automatic Control, University of Erlangen-Nuremberg, Germany (e-mail: klaus.schmidt@rt.eei.uni-erlangen.de).*

**Abstract:** In our previous work, a *communication protocol* for the reliable communication of discrete event supervisors that are implemented on physically distinct controller devices on a shared-medium network was developed. Here, the required data exchange is captured by *communication models* that are algorithmically computed from an underlying hierarchical and decentralized supervisor synthesis. These communication models are particularly efficient if all synthesized supervisors are implemented on distinct controller devices. In this paper, the general case is considered, where multiple supervisors can be *aggregated* on each controller device. To this end, the algorithmic communication model computation is adapted in order to remove communication among supervisors on the same controller device. The benefit of the controller aggregation is illustrated by a manufacturing system case study.

*Keywords:* Hierarchical discrete-event systems, distributed control, shared-medium networks.

## 1. INTRODUCTION

Several approaches enable the efficient supervisor synthesis for large-scale manufacturing systems modeled as discrete event systems (DES) (de Queiroz and Cury, 2000; Leduc et al., 2005; Hill and Tilbury, 2006; Feng and Wonham, 2008; Schmidt et al., 2008a). As a common feature, they result in a set of *modular* or *decentralized* supervisors that interact by *synchronizing* the occurrence of *shared events*. These methods ensure the reliable operation of the DES plant, and are particularly beneficial if the supervisors can be implemented on a single *centralized* controller device such that the shared event synchronization can be handled internally, e.g., via shared memory.

However, in practical applications (e.g., on a factory floor), the supervisors are implemented on various controller devices in distinct physical locations that are connected by a communication medium. Hence, the synchronization of shared events has to be performed by exchanging information about their occurrences among the supervisors while preserving the reliable system operation. An initial *communication model* of the required information exchange for the approach in (Schmidt et al., 2008a) was developed in (Schmidt et al., 2008b), where a *fully distributed implementation* is assumed, i.e., each supervisor runs on a separate controller device.

In this work, we propose a communication model for the general case, where multiple supervisors can be executed on each controller device. This scenario for example addresses the implementation of multiple supervisors for a system component in an industrial application on a single controller device (e.g., Programmable Logic Controller). Then, communication is only required among supervisors that are located on different controller devices, while shared event occurrences can be synchronized internally among supervisors that are *aggregated* on the same controller device. Hence, smaller communication models can be computed compared to the fully distributed case. This result is illustrated by a manufacturing system case study that is performed for different supervisor aggregations.

The organization of the paper is as follows. Section 2 provides a brief overview of hierarchical and decentralized control for DES. The communication model construction for the general setup with multiple supervisors on each controller device is developed in Section 3 and applied to a manufacturing system example in Section 4. Conclusions are given in Section 5.

## 2. HIERARCHICAL AND DECENTRALIZED CONTROL

### 2.1 Architecture

This work is based on the hierarchical and decentralized control approach for DES in (Schmidt et al., 2008a), which is suitable for large-scale DES that are composed of several components. The hierarchical supervisor construction results in a set $\mathcal{R} = \{R_1, \ldots, R_n\}$ of $n$ supervisors that exhibit a hierarchical relationship as depicted in the example in Fig. 1 (a), where each supervisor is represented by a finite automaton $R_k = (X_k, \Sigma_k, \delta_k, x_{0,k}, X_{\mathrm{m},k})$ with the set of states $X_k$, the alphabet $\Sigma_k$, the transition function $\delta_k : X_k \times \Sigma_k \to X_k$, the initial state $x_{0,k}$ and the set of marked states $X_{\mathrm{m},k}$ following the notation in (Cassandras and Lafortune, 2006). W.l.o.g. $R_n$ denotes the highest-level supervisor. In this approach, interaction among the supervisors is represented by *shared events* in the set $\Sigma_\cap := \bigcup_{k=1}^{n} \bigcup_{j=1, j\neq k}^{n} (\Sigma_k \cap \Sigma_j)$ that have to occur synchronously in all supervisors where they appear.

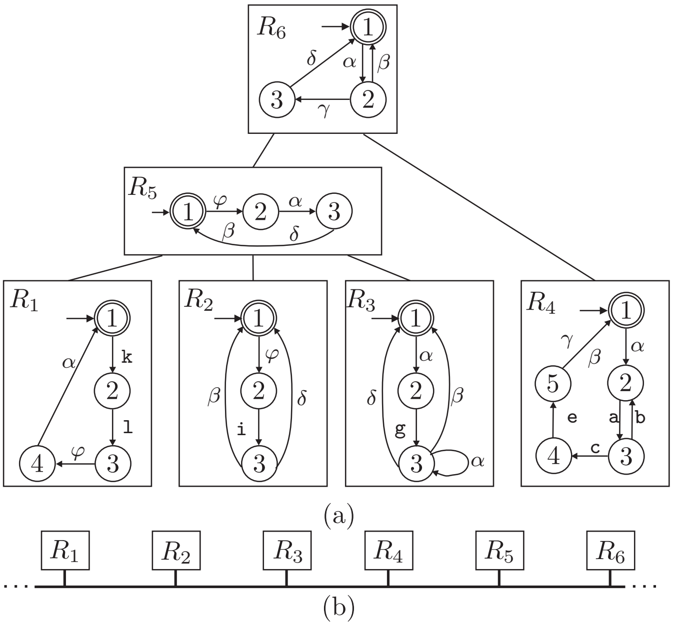


Fig. 1. (a) Supervisor hierarchy; (b) Shared-medium network.

Hence, the overall closed-loop behavior is characterized by an automaton $R = (X, \Sigma, \delta, x_0, X_\mathrm{m})$ that is computed by evaluating the *parallel composition* of all supervisors.

$$R := ||_{k=1}^{n} R_k. \tag{1}$$

Note that *nonblocking control* is ensured, i.e., $L(R) = \overline{L_\mathrm{m}(R)}$, where $L(R)$ and $L_\mathrm{m}(R)$ denote the closed and marked language of $R$, respectively. In addition, (1) need not be evaluated explicitly in a practical implementation which avoids the *state space explosion* encountered by monolithic implementations.

The hierarchical relationship is formally described by a directed tree $T_\mathcal{R} = (\mathcal{R}, R_n, c_\mathcal{R}, p_\mathcal{R})$ (see e.g., Hopcroft and Ullman (1975)). 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*. Furthermore, we define the *descendant map* $d_\mathcal{R} : \mathcal{R} \to 2^\mathcal{R}$ and the *ancestor map* $a_\mathcal{R} : \mathcal{R} \to 2^\mathcal{R}$, where $d_\mathcal{R}(R_k)$ is the set of descendants and $a_\mathcal{R}(R_k)$ is the set of ancestors of $R_k$ in $\mathcal{R}$.

*Example 1.* A hierarchy with 3 levels and $n = 6$ supervisors is depicted in Fig. 1 (a). The lowest-level supervisors $R_1$, $R_2$, $R_3$ and $R_4$ constitute leafs of the tree $T_\mathcal{R}$, while the root is given by $R_6$. All events in the set $\Sigma_\cap = \{\alpha, \beta, \gamma, \delta, \varphi\}$ are synchronized when $R = ||_{k=1}^{6} R_k$ is computed. In this hierarchy, it holds that for example the set of children of $R_5$ is $c_\mathcal{R}(R_5) = \{R_1, R_2, R_3\}$ and the parent of $R_5$ is $p_\mathcal{R}(R_5) = R_6$.

### 2.2 Properties

In addition to the hierarchical structure, the approach in (Schmidt et al., 2008a) features further properties that are relevant in the course of this paper. We denote $\hat{\Sigma}_k := \Sigma_k \cap \Sigma_\cap$ as the set of events of $R_k$ that are shared with other supervisors, and introduce the natural projection $p_k : \Sigma_k^* \to \hat{\Sigma}_k^*$. The abstraction $\hat{R}_k = (\hat{X}_k, \hat{\Sigma}_k, \hat{\delta}_k, \hat{x}_{0,k}, \hat{X}_{\mathrm{m},k})$ is defined for each supervisor $R_k$ by

$$L(\hat{R}_k) := p_k(L(R_k)) \text{ and } L_\mathrm{m}(\hat{R}_k) := p_k(L_\mathrm{m}(R_k)) \tag{2}$$

Furthermore, the dependency of $R_k$ on its children supervisors is described as

- $\Sigma_k = \bigcup_{l, R_l \in c_\mathcal{R}(R_k)} \hat{\Sigma}_l$
- $L(R_k) \subseteq \|_{l, R_l \in c_\mathcal{R}(R_k)} L(\hat{R}_l)$

## 3. CONTROLLER AGGREGATION

We now consider the case of a practical implementation of the derived supervisors on controller devices that are potentially located in distinct physical locations, and that can communicate via a shared-medium network. Hence, at most one controller device can access the medium at a time. Such scenario arises for example on a factory floor with communicating programmable logic controllers (PLCs). In this work, we investigate the general case, where multiple supervisors can be assigned to the same controller device. Our goal is the construction of communication models (CMs) that enable the synchronization of shared events among supervisors on distinct controller devices via the shared-medium network.

### 3.1 Grouping of Controller Components

Formally, we introduce a *set of groups* $\mathcal{G}$ such that each group $G \in \mathcal{G}$ represents the supervisors assigned to a unique controller device. Each supervisor is associated to a group in $\mathcal{G}$ by the *group assignment map* $g : \mathcal{R} \to \mathcal{G}$, i.e., $g(R_k)$ denotes the group of the supervisor $R_k \in \mathcal{R}$.

*Example 2.* Fig. 2 shows two possible grouping scenarios, where gray boxes indicate supervisors that occupy the same group. For example, in Fig. 2 (a), $\mathcal{G} = \{G_1, G_2, G_3\}$ and $g(R_2) = g(R_3) = g(R_5) = G_3$.

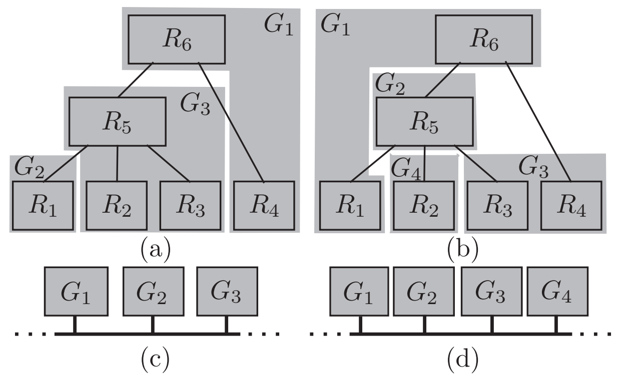


Fig. 2. (a) and (b): Grouping of supervisor components; (c) and (d): Communication relationship.

It can be observed from the example in Fig. 2 (a), that controllers that reside in the same group (i.e., on the same controller device) can perform the synchronization of shared events internally. Conversely, the synchronization of shared events among different groups still relies on communication as illustrated by the network scenario in Fig. 2 (c). Furthermore, it has to be noted that arbitrary aggregations of controllers are not desirable. Fig. 2 (b) depicts two situations that have to be avoided.

(i) On the one hand, $R_6$ can internally synchronize with $R_1$ as both supervisors belong to the same group $G_1$, while on the other hand, $R_6$ communicates with $R_1$ via the intermediate supervisor $R_5$ that resides in a different group. Hence, we require in Definition 3.1 (i) that all supervisors on the path between two supervisors $R_i \in \mathcal{R}$ to $R_j \in \mathcal{R}$ in the same branch of $T_\mathcal{R}$ must be in the same group if $R_i$ and $R_j$ belong to the same group.

(ii) The group $G_3$ has two different parent groups $G_1$ and $G_2$. Similar to (i), this implies that there are different communication paths from $G_1$ to $G_3$ (direct and via $G_2$). Consequently, we require that each group must have a unique parent group in Definition 3.1 (ii).

*Definition 3.1.* (Compatibility). Let $T_\mathcal{R}$ be a directed tree of supervisors, let $\mathcal{G}$ be a set of groups, and let $g : \mathcal{R} \to \mathcal{G}$ be a group assignment map. $g$ is said to be *compatible to* $T_\mathcal{R}$ if the following holds.

(i) $R_i, R_j \in \mathcal{R}$ s.t. $g(R_i) = g(R_j)$ and $R_l \in (a_\mathcal{R}(R_i) \cap d_\mathcal{R}(R_j)) \cup (d_\mathcal{R}(R_i) \cap a_\mathcal{R}(R_j)) \Rightarrow g(R_l) = g(R_i)$.

(ii) $R_i, R_j \in \mathcal{R}$ s.t. $g(R_i) = g(R_j)$, while $g(p_\mathcal{R}(R_i)) \neq g(R_i)$ and $g(p_\mathcal{R}(R_j)) \neq g(R_j) \Rightarrow g(p_\mathcal{R}(R_i)) = g(p_\mathcal{R}(R_j))$.

It it readily verified that compatibility of a group assignment map $g : \mathcal{R} \to \mathcal{G}$ as introduced in Definition 3.1 ensures that the groups in $\mathcal{G}$ again constitute a tree structure. We denote this tree by $T_\mathcal{G} = (\mathcal{G}, G, c_\mathcal{G}, p_\mathcal{G})$ with the associated root $G = g(R_n)$, children map $c_\mathcal{G} : \mathcal{G} \to 2^\mathcal{G}$ and parent map $p_\mathcal{G} : \mathcal{G} \to \mathcal{G}$. Again, $c^\sigma_\mathcal{G} : \mathcal{G} \to 2^\mathcal{G}$ denotes the restriction of $c^\sigma_\mathcal{G}$ to groups that contain the event $\sigma$. In the sequel, our goal is to adopt the communication strategy in (Schmidt et al., 2008b) to the grouped case. [1]

In this context, the basic idea is to introduce *question events*, *answer events*, and a *command event* for each event and each group, where the event appears. The synchronized occurrence of such event is then determined by questions that are propagated from parent groups to children groups and answers that are sent by children groups and collected by parent groups. In this framework, an event occurs, if all possible answers arrived at the highest-level parent group that shares the event. In that case, the command is issued. Furthermore, it is required that all supervisors in a group agree on their questions and answers. Example 3 substantiates this idea.

*Example 3.* We consider the situation in Fig. 2 (a) with the supervisor hierarchy in Fig. 1 (a). Initially, the group $G_1$ would ask a question $?\alpha_{G_1}$ to $G_3$. Then, $G_3$ would first inquire about the event $\varphi$ ($?\varphi_{G_3}$ to $G_2$). After the answer $!\varphi_{G_2}$ from $G_2$, the command $\varphi$ is given by $G_3$ if $\varphi$ is feasible in both $R_5$ and $R_2$. Note that no communication between $R_5$ and $R_2$ is necessary as they share the same group. Next, $G_3$ asks the question $?\alpha_{G_3}$ to the group $G_2$. After receiving the answer $!\alpha_{G_2}$, $G_3$ can locally decide about the answer $!\alpha_{G_3}$ to $G_1$ if $\alpha$ is feasible in $R_3$ and $R_5$. The execution of $\alpha$ is then locally decided by $G_1$ if the answer $!\alpha_{G_3}$ is received and also $\alpha$ is feasible in $R_4$. After the execution of $\alpha$, communication for the events $\beta$ and $\gamma$ is initiated by $G_1$ asking $?\beta_{G_1}$ and $?\gamma_{G_1}$ as soon as $R_4$ reaches state 5.

[1] We refer to Schmidt (2009) for a detailed discussion.

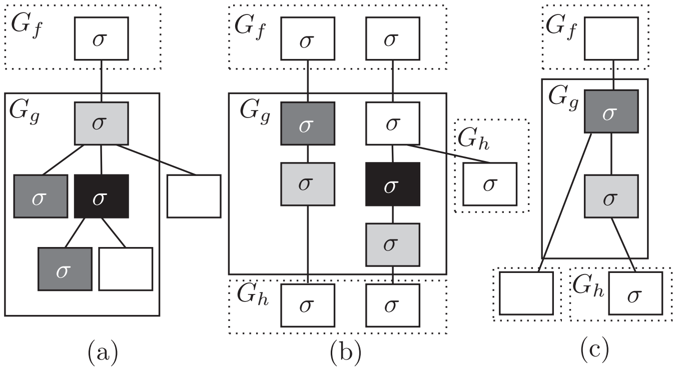


Fig. 3. (a) Low-level group; (b) Intermediate-level group; (c) High-level group.

### 3.2 Aggregated Communication Models

Formalizing the strategy presented in the previous section, we now develop an aggregation method that avoids unnecessary communication among supervisors in the same group while synchronizing shared events via communication among different groups. To this end, we divide the alphabet of each supervisor $R_k$ into 4 subsets s.t. $\Sigma_k = \Sigma_{k,\mathrm{L}} \dot\cup \Sigma_{k,\mathrm{I}} \dot\cup \Sigma_{k,\mathrm{H}} \dot\cup \Sigma_{k,\mathrm{N}}$ based on the position of $R_k$ in the tree $T_\mathcal{G}$. Here, we assume that $g(R_k) = G_g$.

$$\Sigma_{k,\mathrm{L}} := \{\sigma \in \hat\Sigma_k | \exists R_j \in a^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_j) \neq G_g \wedge \nexists R_l \in d^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_l) \neq G_g\} \quad (3)$$

$$\Sigma_{k,\mathrm{I}} := \{\sigma \in \hat\Sigma_k | \exists R_j \in a^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_j) \neq G_g \wedge \exists R_l \in d^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_l) \neq G_g\} \quad (4)$$

$$\Sigma_{k,\mathrm{H}} := \{\sigma \in \Sigma_k | \nexists R_j \in a^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_j) \neq G_g \wedge \exists R_l \in d^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_l) \neq G_g\} \quad (5)$$

$$\Sigma_{k,\mathrm{N}} := \{\sigma \in \Sigma_k | \nexists R_j \in a^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_j) \neq G_g \wedge \nexists R_l \in d^\sigma_\mathcal{R}(R_k) \text{ s.t. } g(R_l) \neq G_g\} \quad (6)$$

That is, $\Sigma_{k,\mathrm{L}}$ contains the events that are communicated with a parent group, $\Sigma_{k,\mathrm{H}}$ represents the events that are only synchronized with children groups and $\Sigma_{k,\mathrm{I}}$ consists of events that are shared with parent and children groups. The events in $\Sigma_{k,\mathrm{N}}$ are not communicated at all.

*Example 4.* Referring to the aggregation in Fig. 2 (a), it holds that $\Sigma_{3,\mathrm{L}} = \{\alpha, \beta, \delta\}$, $\Sigma_{3,\mathrm{N}} = \{\mathtt{g}\}$, $\Sigma_{3,\mathrm{H}} = \Sigma_{3,\mathrm{I}} = \emptyset$ and $\Sigma_{5,\mathrm{L}} = \{\beta, \delta\}$, $\Sigma_{5,\mathrm{I}} = \{\alpha\}$, $\Sigma_{5,\mathrm{H}} = \{\varphi\}$, $\Sigma_{5,\mathrm{N}} = \emptyset$. Furthermore, $\Sigma_{6,\mathrm{H}} = \{\alpha, \beta, \delta\}$, $\Sigma_{6,\mathrm{N}} = \{\gamma\}$, $\Sigma_{6,\mathrm{L}} = \Sigma_{6,\mathrm{I}} = \emptyset$. It is interesting to note that no communication is required for the shared events $\beta, \delta$ within $G_3$ and for the shared event $\gamma$ in $G_1$.

We now propose an algorithmic computation of the CM for each group. In the following sections, we discuss how this CM has to be constructed for events in $\Sigma_{k,\mathrm{L}}$, $\Sigma_{k,\mathrm{I}}$ and $\Sigma_{k,\mathrm{H}}$. The algorithmic CM computation is then summarized in Algorithm 3.1. In all cases, we consider a supervisor $R_k$ in the group $G_g = g(R_k)$ and the event $\sigma$ in the respective alphabet $\Sigma_{k,\mathrm{L}}$, $\Sigma_{k,\mathrm{I}}$ or $\Sigma_{k,\mathrm{H}}$. In addition, we denote $R_j = p_\mathcal{R}(R_k)$ and $G_f = p_\mathcal{G}(G_g)$ as the parent supervisor of $R_k$ and the parent group of $G_g$, respectively, if they exist.

*Communication Model for* $\Sigma_{k,\mathrm{L}}$*:* We want to compute the CM component $L^\sigma_{j,k}$, where three different positions of $R_k$ in $G_g$ are possible as depicted in Fig. 3 (a).

L1 $R_j$ belongs to $G_f$, i.e., $g(R_j) = G_f$ (see the light gray box in Fig. 3 (a)). Then, $R_k$ receives the question $?\sigma_{G_f}$ from and provides the answer $!\sigma_{G_g}$ to the parent supervisor that is located in the different group $G_f$. Hence, $L^{\sigma}_{j,k}$ is computed from $\hat{R}_k$ by inserting two states $\tilde{x}$ and $\bar{x}$ for each state $x$ where $\hat{\delta}_k(x, \sigma)$ exists. These additional states are then connected such that the string $?\sigma_{G_f}!\sigma_{G_g}$ must occur before $\sigma$ is feasible, while the transition structure of $\hat{R}_k$ for events different from $\sigma$ remains unchanged. Algorithm 3.1 with the input parameters $\hat{R}_i = \hat{R}_k$, $\Delta = \hat{\Sigma}_k \cup \{?\sigma_{G_f}, !\sigma_{G_g}\}$ describes this computation.

L2 $R_j$ belongs to $G_g$, i.e., $g(R_j) = G_g$ and $c_{\mathcal{R}}(R_k) = \emptyset$ s.t. $R_k$ is a leaf supervisor (see the dark gray box in Fig. 3 (a)). Then, $R_k$ does not receive $?\sigma_{G_f}$ but has to give the answer $!\sigma_{G_g}$ since $!\sigma_{G_g}$ is only provided if all supervisors in $G_g$ agree that $\sigma$ is feasible. Thus, $\hat{R}_i = \hat{R}_k$, $\Delta = \hat{\Sigma}_k \cup \{!\sigma_{G_g}\}$ for computing $L^{\sigma}_{j,k}$.

L3 $R_j$ belongs to $G_g$ and $c_{\mathcal{R}}(R_k) \neq \emptyset$ (see the black box in Fig. 3 (a)). Then neither $?\sigma_{G_f}$ is received nor the answer $!\sigma_{G_g}$ needs to be given, since there must be a descendant in $d^{\sigma}_{\mathcal{R}}(R_k)$ that already gives the answer according to L2.[2] Consequently, $L^{\sigma}_{j,k} = \hat{R}_k$.

*Example 5.* Fig. 4 shows the component $L^{\alpha}_{5,1}$ for $R_1$ and $\alpha$ (type L1) and $L^{\alpha}_{5,3}$ for $R_3$ and $\alpha$ (type L2).

*Communication Model for $\Sigma_{k,\mathrm{I}}$:* The CM component $IU^{\sigma}_{j,k}$ is computed. In I1 and I2, we address the case, where there exists an $R_l \in c^{\sigma}_{\mathcal{R}}(R_k)$ that lies in a different group than $R_k$, i.e., $G_h := g(R_l) \neq G_g$.

IU1 It is further assumed that $G_f = g(R_j) \neq G_g$ (see the white box with $\sigma$ in $G_g$ in Fig. 3 (b)). Here, $IU^{\sigma}_{j,k}$ includes the question $?\sigma_{G_f}$ from $R_j$, the question $?\sigma_{G_g}$ to $R_l$ and the answer $!\sigma_{G_g}$ to $R_j$. Note that the question $?\sigma_{G_g}$ to $R_l$ has to be asked between the occurrence of $?\sigma_{G_f}$ and $!\sigma_{G_g}$ according to our communication strategy. This is captured by $\hat{R}_i = \hat{R}_k$ and $\Delta = \hat{\Sigma}_k \cup \{?\sigma_{G_f}, ?\sigma_{G_g}, !\sigma_{G_g}\}$ in Algorithm 3.1.

IU2 Now, $g(R_j) = G_g$ (see the light gray boxes in Fig 3 (b)). Then, $R_k$ does not receive a question from its parent, while it has to agree on asking the question $?\sigma_{G_g}$ to the group $G_h$ and answering $!\sigma_{G_g}$ to the parent group $G_f$.[3] Thus, $\hat{R}_i = \hat{R}_k$ and $\Delta = \hat{\Sigma}_k \cup \{?\sigma_{G_g}, !\sigma_{G_g}\}$ for computing $IU^{\sigma}_{j,k}$.

Next, we assume that all $R_l \in c^{\sigma}_{\mathcal{R}}(R_k)$ are in the same group with $R_k$, i.e., $g(R_l) = G_g$. There are again two cases for the computation of $IU^{\sigma}_{j,k}$.

IU3 If $G_f = g(R_j) \neq G_g$ (see the dark gray box in Fig. 3 (b)), then $R_k$ receives the question $?\sigma_{G_f}$ from $R_j$ and asks the question $?\sigma_{G_g}$. However, the answer $!\sigma_{G_g}$ does not have to be given by $R_k$ as there is at least one descendant in $d^{\sigma}_{\mathcal{R}}(R_k)$ that already gives this answer. It holds that $\hat{R}_i = \hat{R}_k$ and $\Delta = \hat{\Sigma}_k \cup \{?\sigma_{G_f}, ?\sigma_{G_g}\}$.

[2] It can be shown that the feasibility of $\sigma$ in a descendant in $d^{\sigma}_{\mathcal{R}}(R_k)$ implies the feasibility of $\sigma$ in $R_k$.

[3] $?\sigma_{G_g}$ has to be agreed on by all supervisors in the group since there is no implication from the feasibility of $\sigma$ in an ancestor in $a^{\sigma}_{\mathcal{R}}(R_k)$ on the feasibility of $\sigma$ in $R_k$.

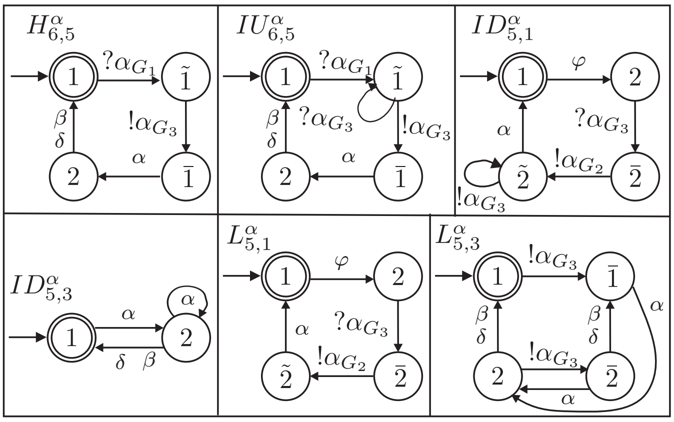


Fig. 4. Grouped communication model components.

IU4 If $g(R_j) = G_g$ (see the black box in Fig. 3 (b)), then $R_k$ neither receives $?\sigma_{G_f}$ nor participates in $!\sigma_{G_g}$. Only the question $?\sigma_{G_g}$ has to be asked. Hence, $\hat{R}_i = \hat{R}_k$ and $\Delta = \hat{\Sigma}_k \cup \{?\sigma_{G_g}\}$.

The computation of $ID^{\sigma}_{k,l}$ for $R_l \in c^{\sigma}_{\mathcal{R}}(R_k)$ involves two cases.

ID1 If $g(R_l) \neq G_g$, the automaton $ID^{\sigma}_{k,l}$ is computed from $\hat{R}_l$ such that the question $?\sigma_{G_g}$ and the answer $!\sigma_{G_h}$ have to be exchanged before the answer $!\sigma_{G_g}$ can be given to the parent group. Hence, $\hat{R}_i = \hat{R}_l$ and $\Delta = \hat{\Sigma}_l \cup \{?\sigma_{G_g}, !\sigma_{G_h}, !\sigma_{G_g}\}$.

ID2 If $g(R_l) = G_g$, there is no direct communication with a child of $R_k$ outside the group $G_g$. Hence, we set $ID^{\sigma}_{k,l} = \hat{R}_l$.

*Example 6.* Fig. 4 shows the CM components $IU^{\alpha}_{6,5}$ (type IU1), $ID^{\alpha}_{5,1}$ (ID1) and $ID^{\alpha}_{5,3}$ (ID2) for $R_5$ and $\alpha$.

*Communication Model for $\Sigma_{k,\mathrm{H}}$:* The computation of the CM component $H^{\sigma}_{k,l}$ for a child supervisor $R_l \in c^{\sigma}_{\mathcal{R}}(R_k)$ involves two different cases as shown in Fig. 3 (c).

H1 We assume that $G_h := g(R_l) \neq G_g$ (see the light gray box in Fig. 3 (c)). Then, $H^{\sigma}_{k,l}$ is constructed for $\hat{R}_l$. It has the same structure as $L^{\sigma}_{j,k}$, since the same types of events $?\sigma_{G_g}$ and $!\sigma_{G_h}$ in the same sequential order are involved. The input parameters for Algorithm 3.1 are $\hat{R}_i = \hat{R}_l$ and $\Delta = \hat{\Sigma}_l \cup \{?\sigma_{G_g}, !\sigma_{G_h}\}$.

H2 If $g(R_l) = G_g$ (see the dark gray box in Fig. 3 (c)), then no answer is received by $R_k$. Hence, $!\sigma_{G_h}$ is removed from $\Delta$ compared to H1 for computing $H^{\sigma}_{k,l}$.

*Example 7.* The type H1 is illustrated in Fig. 4 by $H^{\alpha}_{6,5}$ for $R_6$ and $\alpha$.

*Algorithm 3.1.* (Computation of Aggregated CMs). We compute an automaton $G = (Q, \Delta, \nu, q_0, Q_{\mathrm{m}})$ for $L^{\sigma}_{j,k}$, $IU^{\sigma}_{j,k}$ or $H^{\sigma}_{k,l}$ depending on the input alphabet $\Delta$.

```
Given: R̂_i, σ, Δ.
Initialize: Q = X̂_i; q_0 = x̂_{0,k}; Q_m = X̂_{m,k}
%% Introduce states needed for communication
for each x ∈ X̂_i s.t. δ̂_i(x, σ)!
    if !σ_{G_g} ∈ Δ and ?σ_{G_g} ∉ Δ
        Q = Q ∪ {x̄}; ν(x̄, σ) = δ̂_i(x, σ)
        if ?σ_{G_f} ∈ Δ %% Case L1
```

$Q = Q \cup \{\tilde{x}\}$
$\nu(x, ?\sigma_{G_f}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{G_g}) = \bar{x}$
**else** %% Case L2
$\nu(x, !\sigma_{G_g}) = \bar{x}$
**else if** $\{?\sigma_{G_g}, !\sigma_{G_g}\} \subseteq \Delta$
$Q = Q \cup \{\bar{x}\}$; $\nu(\bar{x}, \sigma) = \hat{\delta}_i(x, \sigma)$
**if** $?\sigma_{G_f} \in \Delta$ %% Case I1
$Q = Q \cup \{\tilde{x}\}$; $\nu(x, ?\sigma_{G_f}) = \tilde{x}$
$\nu(\tilde{x}, ?\sigma_{G_g}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{G_g}) = \bar{x}$
**else** %% Case I2
$\nu(x, ?\sigma_{G_g}) = x$; $\nu(x, !\sigma_{G_g}) = \bar{x}$
**else if** $?\sigma_{G_g} \in \Delta$ and $!\sigma_{G_g} \notin \Delta$
**if** $?\sigma_{G_f} \in \Delta$ %% Case I3
$Q = Q \cup \{\tilde{x}\}$; $\nu(\tilde{x}, \sigma) = \hat{\delta}_i(x, \sigma)$
$\nu(x, ?\sigma_{G_f}) = \tilde{x}$; $\nu(\tilde{x}, ?\sigma_{G_g}) = \tilde{x}$;
**else** %% Case I4
$\nu(x, ?\sigma_{G_g}) = x$; $\nu(x, \sigma) = \hat{\delta}_i(x, \sigma)$
**else if** $?\sigma_{G_g} \in \Delta$ and $!\sigma_{G_g} \notin \Delta$
$Q = Q \cup \{\bar{x}\}$; $\nu(\bar{x}, \sigma) = \hat{\delta}_i(x, \sigma)$
**if** $!\sigma_{G_h} \in \Delta$ %% Case H1
$Q = Q \cup \{\tilde{x}\}$
$\nu(x, ?\sigma_{G_g}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{G_h}) = \bar{x}$
**else** %% Case H2
$\nu(x, ?\sigma_{G_g}) = \bar{x}$
*%% Add transition structure of original automaton* $\hat{R}_i$
**for each** $x \in \hat{X}_i$
**for each** $\tau \in \hat{\Sigma}_i(x) - \{\sigma\}$
$\nu(x, \tau) = \hat{\delta}_i(x, \tau) =: x'$
**if** $\sigma \in \hat{\Sigma}_i(x) \wedge \sigma \in \hat{\Sigma}_i(\hat{\delta}_i(x, \tau))$
**if** $\tilde{x} \in Q$
$\nu(\tilde{x}, \tau) = \tilde{x}'$
**if** $\bar{x} \in Q$
$\nu(\bar{x}, \tau) = \bar{x}'$
**else if** $\sigma \in \hat{\Sigma}_i(x) \wedge \sigma \notin \hat{\Sigma}_i(\hat{\delta}_i(x, \tau))$
**if** $\tilde{x} \in Q$
$\nu(\tilde{x}, \tau) = x'$
**if** $\bar{x} \in Q$
$\nu(\bar{x}, \tau) = x'$
**return** $G$

### *3.3 Communication Model for Groups*

The CM $CM_k = (Q_k, \mathcal{J}_k, \nu_k, q_{0,k}, Q_{\mathrm{m},k})$ for each supervisor $R_k$ is composed of the CM components as constructed above, where the composition with $R_k$ introduces the supervisor action of $R_k$ in the model.

$$CM_k = (||_{\sigma \in \Sigma_{k,\mathrm{L}}} L_k^\sigma) || (||_{\sigma \in \Sigma_{k,\mathrm{I}}} I_k^\sigma) || (||_{\sigma \in \Sigma_{k,\mathrm{H}}} H_k^\sigma) || R_k. \quad (7)$$

Then, the CMs of supervisors that belong to the same group can be executed in parallel on the same controller device. That is, for each $G_g \in \mathcal{G}$, we arrive at

$$CG_g = ||_{k, g(R_k) = G_g} CM_k. \quad (8)$$

Implementing the aggregated CMs computed in this section, it holds that the desired reliable operation of the DES plant which is achieved by the hierarchical supervisor design is still realized after introducing communication among the supervisors that are grouped on distinct controller devices.

*Theorem 3.1.* (Aggregation). Let $T_\mathcal{R} = (\mathcal{R}, R_n, c_\mathcal{R}, p_\mathcal{R})$ be a hierarchical tree of distributed supervisors, let $T_\mathcal{G} = (\mathcal{G}, G, c_\mathcal{G}, p_\mathcal{G})$ be a tree of groups with the group assignment map $g : \mathcal{R} \to \mathcal{G}$, and let $CG_1, \ldots, CG_{|\mathcal{G}|}$ be the group CMs defined above. Also let $\theta : \mathcal{K}^* \to \Sigma^*$ be the natural projection, where $\mathcal{K} = \bigcup_{g=1}^{|\mathcal{G}|} \mathcal{K}_g$. Then

$$\overline{||_{g=1}^{|\mathcal{G}|} L_\mathrm{m}(CG_g)} = ||_{i=k}^{|\mathcal{G}|} L(CG_g)$$

$$\theta(||_{k=1}^{|\mathcal{G}|} L(CG_g)) = ||_{k=1}^{n} L(R_k)$$

*Remark 1.* In our previous work (Schmidt et al., 2008b), a less general version of Theorem 3.1 was stated based on CMs that are computed as if all supervisors were located in distinct groups. This *simple aggregation* potentially leads to larger CMs, since the removal of unnecessary communication among supervisors in the same group as performed in Section 3.2 is not taken into account.

## 4. APPLICATION EXAMPLE

### *4.1 General Setup*

The presented ideas are applied to the *distribution system* (ds) in Fig. 5. Its purpose is to deliver parts entering from a *stack feeder* (sf) to a larger manufacturing system via the conveyor belts c2 and c3. As further components of ds, there are two *pushers* p1 and p2 that push parts traveling along the long conveyor belt c1 to c2 and c3, respectively. In our models, c1 is divided into the 3 subcomponents c1a (at p2), c1b (at p1) and con (connecting c1a and c1b).

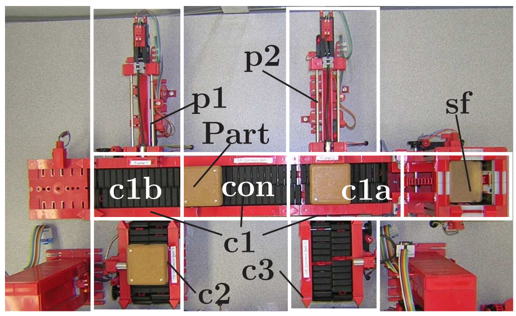


Fig. 5. Distribution system overview.

The supervisor synthesis for ds has been performed analogous to (Schmidt et al., 2008a). Fig. 6 shows the resulting hierarchy with 4 levels and 12 supervisors, whose respective state counts are listed in Table 1. Together, the supervisors have a sum of 218 states, which represents the size of the supervisor required for a centralized implementation on a single controller device.

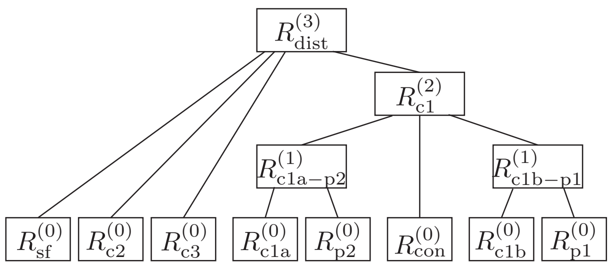


Fig. 6. Supervisor hierarchy of the distribution system.

| | SUP | CM | | SUP | CM |
|---|---|---|---|---|---|
| $R^{(0)}_{\text{sf}}$ | 9 | 19 | $R^{(0)}_{\text{c1a}}$ | 12 | 61 |
| $R^{(0)}_{\text{c2}}$ | 9 | 21 | $R^{(0)}_{\text{con}}$ | 23 | 588 |
| $R^{(0)}_{\text{c3}}$ | 9 | 21 | $R^{(0)}_{\text{c1b}}$ | 9 | 37 |
| $R^{(0)}_{\text{p2}}$ | 10 | 22 | $R^{(1)}_{\text{c1a-p2}}$ | 11 | 484 |
| $R^{(0)}_{\text{p1}}$ | 10 | 22 | $R^{(1)}_{\text{c1b-p1}}$ | 9 | 237 |
| $R^{(2)}_{\text{c1}}$ | 47 | 1359 | $R^{(3)}_{\text{dist}}$ | 60 | 724 |

Table 1. Supervisor and CM state counts.

### *4.2 Controller Aggregation*

For comparison, we first evaluate the CMs of the simple aggregation in Remark 1, where all CMs are computed as if their corresponding supervisors were implemented on different controller devices. The state counts of the CMs are shown in Table 1.

We now illustrate the grouping idea by two scenarios. In Fig. 7, it is assumed that each of the functional entities c1, c2, c3, sf, p1 and p2 is controlled by a local controller device, while the components are coordinated by the superposed supervisor $R^{(3)}_{\text{dist}}$ on a separate controller device. The state counts of the CMs for the corresponding 7 groups computed according to Section 3.3 are depicted in Table 2. The reduction from 3595 to 1120 states compared to the simple aggregation can be explained by the removal of internal communication in the group $G_5$.

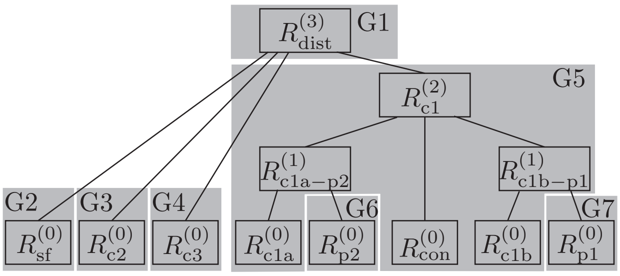


Fig. 7. Grouping of functional entities.

| | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ | $G_8$ | $\sum$ |
|---|---|---|---|---|---|---|---|---|---|
| Fig. 7 | 527 | 21 | 21 | 19 | 428 | 22 | 82 | — | 1120 |
| Fig. 8 | 727 | 21 | 21 | 19 | 61 | 22 | 37 | 22 | 930 |

Table 2. State counts for Fig. 7 and 8.

The second scenario in Fig. 8 considers that local controllers are used for the supervisors c2, c3, c1a, c1b, p1, p2 and sf that exchange sensor and actuator information with the plant. All remaining supervisors perform coordination on a separate controller device ($G_1$). Again, a reduction to 930 states due to the avoidance of internal communication in $G_1$ can be seen in Table 2. In our study, it could be determined that it is favorable to group supervisors on different hierarchical levels that share multiple events.

## 5. CONCLUSION

In this paper, the implementation of hierarchical and decentralized supervisors on distributed controller devices that are connected by a shared-medium network is investigated. Extending previous work that addresses a fully distributed implementation, communication models for the general case, where multiple supervisors can be aggregated on a single controller device, are computed algorithmically. These communication models capture the required information exchange among supervisors in order to synchronize the occurrences of their respective shared events in order to achieve reliable operation of the DES plant. A manufacturing system case study illustrates that the communication can be reduced by supervisor aggregation.

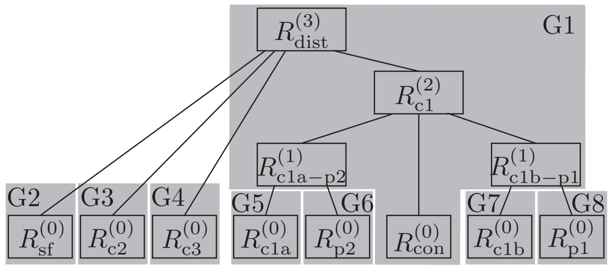


Fig. 8. Grouping with high-level coordination.

In future work, it will be evaluated how the reduced communication affects the communication behavior of the distributed supervisors both analytically and by simulation analogous to (Schmidt et al., 2008b). Furthermore, the fully distributed communication models for switched networks in (Schmidt and Schmidt, 2008) will be adapted to the general case with supervisor aggregation.

## REFERENCES

Cassandras, C.G. and Lafortune, S. (2006). *Introduction to Discrete Event Systems.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

de Queiroz, M.H. and Cury, J.E.R. (2000). Modular supervisory control of large scale discrete event systems. In *Workshop on Discrete Event Systems.*

Feng, L. and Wonham, W. (2008). Supervisory control architecture for discrete-event systems. *Automatic Control, IEEE Transactions on*, 53(6), 1449–1461. doi: 10.1109/TAC.2008.927679.

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

Hopcroft, J.E. and Ullman, J.D. (1975). *The design and analysis of computer algorithms.* Addison-Wesley.

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

Schmidt, K. (2009). Controller aggregation for distributed discrete event supervisors on a shared-medium network. *Technical Report, Lehrstuhl fr Regelungstechnik, Universitt Erlangen-Nrnberg.*

Schmidt, K., Moor, T., and Perk, S. (2008a). Nonblocking hierarchical control of decentralized discrete event systems. *Automatic Control, IEEE Transactions on*, 53(10), 2252–2265.

Schmidt, K. and Schmidt, E.G. (2008). Communication of distributed discrete-event supervisors on a switched network. In *Workshop on Discrete Event Systems.*

Schmidt, K., Schmidt, E.G., and Zaddach, J. (2008b). Reliable and safe operation of distributed discrete-event controllers: A networked implementation with real-time guarantees. In *IFAC World Congress.*