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

Klaus Schmidt

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. Note that this paper is an extended version of [7].

I. Introduction

Several approaches enable the efficient supervisor synthesis for large-scale manufacturing systems modeled as discrete event systems (DES) [2], [6], [4], [3], [8]. 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 the shared event occurrences among the supervisors while it has to be guaranteed that the reliable system operation

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

is not affected. An initial *communication model* of the required information exchange for the approach in [8] was developed in [10], where a *fully distributed implementation* is assumed, i.e., each supervisor is implemented 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 II provides a brief overview of hierarchical and decentralized control for DES, and Section III establishes the communication model computation for the fully distributed case. The communication model construction for the general setup with multiple supervisors on each controller device is developed in Section IV and applied to a manufacturing system example in Section V. Conclusions are given in Section VI.

II. Hierarchical and Decentralized Control

A. Architecture

This work is based on the hierarchical and decentralized control approach for DES in [8], 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 Σ_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 [1]. 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. 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., [5]). In this paper, $\mathcal{R}$ denotes the set of *vertices*, R_n is the *root vertex* and $c_\mathcal{R} : \mathcal{R} \rightarrow 2^\mathcal{R}$ and $p_\mathcal{R} : \mathcal{R} \rightarrow \mathcal{R}$ 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* $d_\mathcal{R} : \mathcal{R} \rightarrow 2^\mathcal{R}$ and the *ancestor map* $a_\mathcal{R} : \mathcal{R} \rightarrow 2^\mathcal{R}$, 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 $p_\mathcal{R}(R_5) = R_6$.

B. Properties

In addition to the hierarchical structure, the approach in [8] 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 components, and introduce the natural projection $p_k : \Sigma_k^* \rightarrow \hat{\Sigma}_k^*$. Then, 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

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

III. Communication Models for Separate Controller Devices

In this section, we present results derived in our earlier work in [10].

A. Problem Setup

We assume that all supervisors in $\mathcal{R}$ are implemented on distinct distributed controller devices that can communicate via a *shared medium network*, where only one controller device can access the medium

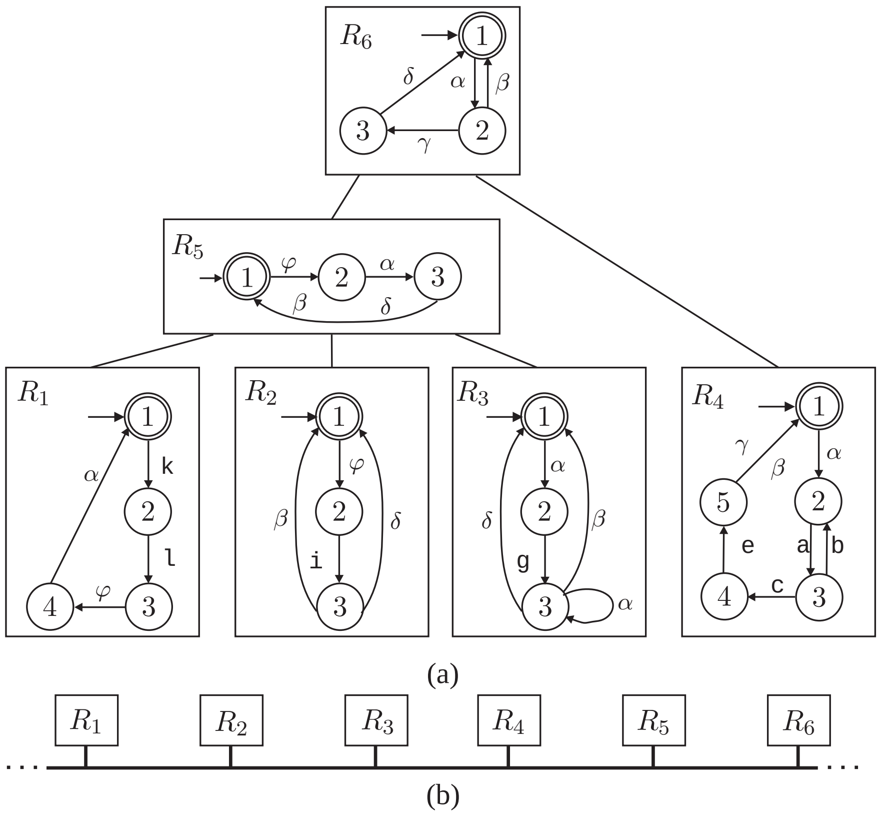

Fig. 1. (a) Supervisor hierarchy; (b) Shared-medium network.

at a time. Such scenario arises for example on a factory floor with communicating programmable logic controllers (PLCs). Fig. 1 (b) illustrates such implementation for the example in Fig. 1 (a).

In order to synchronize the shared events in $\Sigma_\cap$ in such a fully distributed implementation, communication among the supervisors is required. We now present the formal model of this communication on a shared-medium network as developed in [10]. It makes use of the hierarchical relationship between the supervisors so as to reduce the communication of each supervisor $R_k \in \mathcal{R}$ to information exchange with its children $R_l \in c_{\mathcal{R}}(R_k)$ about the shared events in $\hat{\Sigma}_l$ and with the parent $R_j = p_{\mathcal{R}}(R_k)$ about the shared events in $\hat{\Sigma}_k$. The basic communication strategy is outlined in Example 2.

Example 2: Assume that each of the supervisors in Fig. 1 (a) is in its initial state. We first propagate information about the occurrence of shared events from the high-level supervisors to the low-level supervisors. Since α is the first feasible event in the root supervisor R_6, it asks the question "is α possible?" to the children R_4 and R_5, which is expressed by a *communication event* $?\alpha_{R_6}$. Then, R_4 can directly answer "α is possible!" ($!\alpha_{R_4}$), while R_5 first has to execute φ. Hence, R_5 collects the answers $!\varphi_{R_1}$ and $!\varphi_{R_2}$ to the question $?\varphi_{R_5}$ before it commands the execution of φ to the participating supervisors R_1 and R_2. Note that R_1 can only answer $!\varphi_{R_1}$ after the local string $\mathtt{kl}$ occurred. Then, R_5 continues asking $?\alpha_{R_5}$ to R_1 and R_3 and forwards their answers $!\alpha_{R_1}$ and $!\alpha_{R_3}$ to the R_6 by sending $!\alpha_{R_5}$. After receiving all answers for α, R_6 can give the command α such that all participating supervisors execute α. The subsequent communication for the events γ and β is again initiated by R_6 and follows the same strategy.

B. *Formal Description*

In this section, we summarize the formal CM computation according to the strategy in Section III-A as developed in our previous work [10].[1] For notational convenience, we introduce the map $c^{\sigma}_{\mathcal{R}} : \mathcal{R} \to 2^{\mathcal{R}}$ that restricts $c_{\mathcal{R}}$ to supervisors that contain the event σ, i.e., $c^{\sigma}_{\mathcal{R}}(R_k) := \{R_l \in c_{\mathcal{R}}(R_k) | \sigma \in \Sigma_l\}$. Similarly, we use the maps $a^{\sigma}_{\mathcal{R}} : \mathcal{R} \to 2^{\mathcal{R}}$ and $d^{\sigma}_{\mathcal{R}} : \mathcal{R} \to 2^{\mathcal{R}}$.

The alphabet of each supervisor R_k is divided into 4 subsets s.t. $\Sigma_k = \Sigma_{k,\mathrm{L}} \dot{\cup} \Sigma_{k,\mathrm{I}} \dot{\cup} \Sigma_{k,\mathrm{H}} \dot{\cup} \Sigma_{k,\mathrm{N}}$. The subsets characterize events that are communicated with only the parent ($\Sigma_{k,\mathrm{L}}$), only with children ($\Sigma_{k,\mathrm{H}}$), with the parent and with children ($\Sigma_{k,\mathrm{I}}$) or not at all ($\Sigma_{k,\mathrm{N}}$):

$$\Sigma_{k,\mathrm{L}} := \{\sigma \in \hat{\Sigma}_k | c^{\sigma}_{\mathcal{R}}(R_k) = \emptyset\} \tag{3}$$

$$\Sigma_{k,\mathrm{I}} := \{\sigma \in \hat{\Sigma}_k | c^{\sigma}_{\mathcal{R}}(R_k) \neq \emptyset\} \tag{4}$$

$$\Sigma_{k,\mathrm{H}} := \{\sigma \in \Sigma_k - \hat{\Sigma}_k | c^{\sigma}_{\mathcal{R}}(R_k) \neq \emptyset\} \tag{5}$$

$$\Sigma_{k,\mathrm{N}} := \{\sigma \in \Sigma_k - \hat{\Sigma}_k | c^{\sigma}_{\mathcal{R}}(R_k) = \emptyset\} \tag{6}$$

Example 3: For R_3: $\Sigma_{3,\mathrm{L}} = \{\alpha, \beta, \delta\}$, $\Sigma_{3,\mathrm{N}} = \{\mathtt{g}\}$ and $\Sigma_{3,\mathrm{I}} = \Sigma_{3,\mathrm{H}} = \emptyset$. For R_5, $\Sigma_{5,\mathrm{I}} = \{\alpha, \beta, \delta\}$, $\Sigma_{5,\mathrm{H}} = \{\varphi\}$ and $\Sigma_{5,\mathrm{L}} = \Sigma_{5,\mathrm{N}} = \emptyset$. For R_6, $\Sigma_{6,\mathrm{H}} = \{\alpha, \beta, \gamma, \delta\}$, $\Sigma_{6,\mathrm{I}} = \Sigma_{6,\mathrm{L}} = \Sigma_{6,\mathrm{N}} = \emptyset$.

Based on the event classification in (3) - (6), we now construct components of the CM for each supervisor $R_k \in \mathcal{R}$ and each event $\sigma \in (\Sigma_k - \Sigma_{k,\mathrm{N}})$. In this context, a CM component represents the communication of R_k for the event σ with a neighboring supervisor.

1) Events in $\Sigma_{k,\mathrm{L}}$:: Events $\sigma \in \Sigma_{k,\mathrm{L}}$ are communicated with the parent component $R_j = p_{\mathcal{R}}(R_k)$, which requires the reception of $?\sigma_{R_j}$ from R_j and the answer $!\sigma_{R_k}$ from R_k before σ is actually feasible. We define the CM component $L^{\sigma}_{j,k}$ to capture this behavior. It 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 $?\sigma_{R_j}!\sigma_{R_k}$ must occur before σ is feasible, while the transition structure of $\hat{R}_k$ for events

[1]Note that we use a slightly different representation from [10] in order to support the discussion in Section IV.

different from σ remains unchanged. Algorithm 1 performs this computation with the input parameters $\hat{R}_i = \hat{R}_k$ and $\Delta = \hat{\Sigma}_k^\sigma \cup \{?\sigma_{R_j}, !\sigma_{R_k}\}$. As an example, $L_{5,1}^\alpha$ and $L_{5,3}^\alpha$ are depicted in Fig. 2.

Furthermore, it has to be ensured that the answer $!\sigma_{R_k}$ for $\sigma \in \Sigma_{k,\mathrm{L}}$ is only given by the CM when σ is feasible in the original supervisor R_k. To this end, the automaton $L_k^{\sigma\prime}$ is computed from R_k by adding a selfloop with $!\sigma_{R_k}$ in each state where $\delta_k(x,\sigma)$ exists as illustrated by $L_1^{\alpha\prime}$ in Fig. 2.

Together, the automaton $L_k^\sigma = L_{j,k}^\sigma || L_k^{\sigma\prime}$ characterizes the communication of R_k for the event $\sigma \in \Sigma_{k,\mathrm{L}}$.

2) Events in $\Sigma_{k,\mathrm{I}}$:: On the one hand, events $\sigma \in \Sigma_{k,\mathrm{I}}$ are communicated with the parent supervisor $R_j = p_{\mathcal{R}}(R_k)$ using $?\sigma_{R_j}$ and $!\sigma_{R_k}$. On the other hand, such events are communicated with all children $R_l \in c_{\mathcal{R}}^\sigma(R_k)$ which requires the question $?\sigma_{R_k}$ and the respective answer $!\sigma_{R_l}$. We characterize the communication with R_j by the automaton $IU_{j,k}^\sigma$ that results from Algorithm 1 with the input parameters $\hat{R}_i = \hat{R}_k$ and $\Delta = \hat{\Sigma}_k^\sigma \cup \{?\sigma_{R_j}, ?\sigma_{R_k}, !\sigma_{R_k}\}$. Note that the question $?\sigma_{R_k}$ to $R_l \in c_{\mathcal{R}}^\sigma(R_k)$ has to be asked between the occurrence of $?\sigma_{R_j}$ and $!\sigma_{R_k}$ according to the communication strategy in Section III-A. In addition, the communication with $R_l \in c_{\mathcal{R}}^\sigma(R_k)$ is captured by the automaton $ID_{k,l}^\sigma$ that results from Algorithm 1 with the input parameters $\hat{R}_i = \hat{R}_l$ and $\Delta = \hat{\Sigma}_k^\sigma \cup \{?\sigma_{R_k}, !\sigma_{R_l}, !\sigma_{R_k}\}$. Here, $\hat{R}_l$ is adapted such that the string $?\sigma_{R_k}!\sigma_{R_l}$ happens before the answer $!\sigma_{R_k}$ can be given to the parent R_j. Fig. 2 shows the automata $IU_{6,5}^\alpha$, $ID_{5,3}^\alpha$ and $ID_{5,1}^\alpha$.

The communication of R_k for $\sigma \in \Sigma_{k,\mathrm{I}}$ is hence represented by the automaton $I_k^\sigma = IU_{j,k}^\sigma || (||_{l, R_l \in c_{\mathcal{R}}^\sigma(R_k)} ID_{k,l}^\sigma)$, where $R_j = p_{\mathcal{R}}(R_k)$ is the parent of R_k.

3) Events in $\Sigma_{k,\mathrm{H}}$:: Events $\sigma \in \Sigma_{k,\mathrm{H}}$ are communicated with all children $R_l \in c_{\mathcal{R}}^\sigma(R_k)$. The CM component $H_{k,l}^\sigma$ is constructed for each $\hat{R}_l$. It has the same structure as $L_{j,k}^\sigma$ in (i), since the same types of events $?\sigma_{R_k}$ and $!\sigma_{R_l}$ in the same sequential order are involved. The input parameters for Algorithm 1 are $\hat{R}_i = \hat{R}_l$ and $\Delta = \hat{\Sigma}_l \cup \{?\sigma_{R_k}, !\sigma_{R_l}\}$. $H_{6,5}^\alpha$ in Fig. 2 is an example for this computation.

The overall communication for the event $\sigma \in \Sigma_{k,\mathrm{H}}$ is then modeled by the automaton $H_k^\sigma = ||_{l, R_l \in c_{\mathcal{R}}^\sigma(R_k)} H_{k,l}^\sigma$.

4) Communication Model CM_k:: 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.

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

Example 4: It holds that $CM_2 = L_2^\varphi || L_2^\beta || L_2^\delta || R_2 = (L_{5,2}^\varphi || L_2^{\varphi\prime}) || (L_{5,2}^\beta || L_2^{\beta\prime}) || (L_{5,2}^\delta || L_2^{\delta\prime}) || R_2$, $CM_5 = I_5^\alpha || I_5^\beta || I_5^\delta || H_5^\varphi || R_5$ and $CM_6 = H_6^\alpha || H_6^\beta || H_6^\gamma || H_6^\delta || R_6$, with the respective alphabets in Example 3.

Algorithm 1 (Computation of CM components): The algorithm returns an automaton $G = (Q, \Delta, \nu, q_0, Q_{\mathrm{m}})$ that represents the CM component $H_{j,k}^\sigma$, $IU_{j,k}^\sigma$ or $ID_{k,l}^\sigma$ depending on the input alphabet Δ.

```
1 Given: $\hat{R}_i$, $\sigma$, $\Delta$.
2 Initialize: $Q = \hat{X}_i$; $q_0 = \hat{x}_{0,k}$; $Q_\mathrm{m} = \hat{X}_{\mathrm{m},k}$
%% Introduce states needed for communication
3 for each $x \in \hat{X}_i$ s.t. $\hat{\delta}_i(x,\sigma)!$
4       $Q = Q \cup \{\tilde{x}, \bar{x}\}$
5       $\nu(\bar{x},\sigma) = \hat{\delta}_i(x,\sigma)$
6       if $\{?\sigma_{R_j}, !\sigma_{R_k}\} \subseteq \Delta$ and $?\sigma_{R_k} \notin \Delta$ %% $L^\sigma_{j,k}$
7           $\nu(x, ?\sigma_{R_j}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{R_k}) = \bar{x}$
8       if $\{?\sigma_{R_j}, ?\sigma_{R_k}, !\sigma_{R_k}\} \subseteq \Delta$ %% $IU^\sigma_{j,k}$
9           $\nu(x, ?\sigma_{R_j}) = \tilde{x}$; $\nu(\tilde{x}, ?\sigma_{R_k}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{R_k}) = \bar{x}$
10      if $\{?\sigma_{R_k}, !\sigma_{R_l}, !\sigma_{R_k}\} \subseteq \Delta$ %% $ID^\sigma_{k,l}$
11          $\nu(x, ?\sigma_{R_k}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{R_l}) = \bar{x}$; $\nu(\bar{x}, !\sigma_{R_k}) = \bar{x}$
12      if $\{?\sigma_{R_k}, !\sigma_{R_l}\} \subseteq \Delta$ and $?\sigma_{R_k} \notin \Delta$ %% $H^\sigma_{j,k}$
13          $\nu(x, ?\sigma_{R_k}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{R_l}) = \bar{x}$
%% Add transition structure of original automaton $\hat{R}_i$
14 for each $x \in \hat{X}_i$
15      for each $\tau \in \hat{\Sigma}_i(x) - \{\sigma\}$
16          $\nu(x,\tau) = \hat{\delta}_i(x,\tau) =: x'$
17          if $\sigma \in \hat{\Sigma}_i(x) \wedge \sigma \in \hat{\Sigma}_i(\hat{\delta}_i(x,\tau))$
18              $\nu(\tilde{x},\tau) = \tilde{x}'$; $\nu(\bar{x},\tau) = \bar{x}'$
19          else if $\sigma \in \hat{\Sigma}_i(x) \wedge \sigma \notin \hat{\Sigma}_i(\hat{\delta}_i(x,\tau))$
20              $\nu(\tilde{x},\tau) = x'$; $\nu(\bar{x},\tau) = x'$
21 return $G$
```

C. Properties of the Communication Model

We analyze the automaton $CM := ||_{k=1}^n CM_k$ over the alphabet $\mathcal{J} := \bigcup_{k=1}^n \mathcal{J}_k$ that characterizes all possible communication sequences. The following result has been stated in our previous work [10].

Theorem 1 (Communication Equivalence): Let $T_\mathcal{R} = (\mathcal{R}, R_n, c_\mathcal{R}, p_\mathcal{R})$ be a hierarchical tree of distributed supervisors according to Section II-A, let $CM_1, \ldots, CM_n$ be the corresponding communication models in (7) and define the natural projection $\theta : \mathcal{J}^* \to \Sigma^*$. Then

$$\overline{||_{k=1}^n L_\mathrm{m}(CM_k)} = ||_{k=1}^n L(CM_k)$$

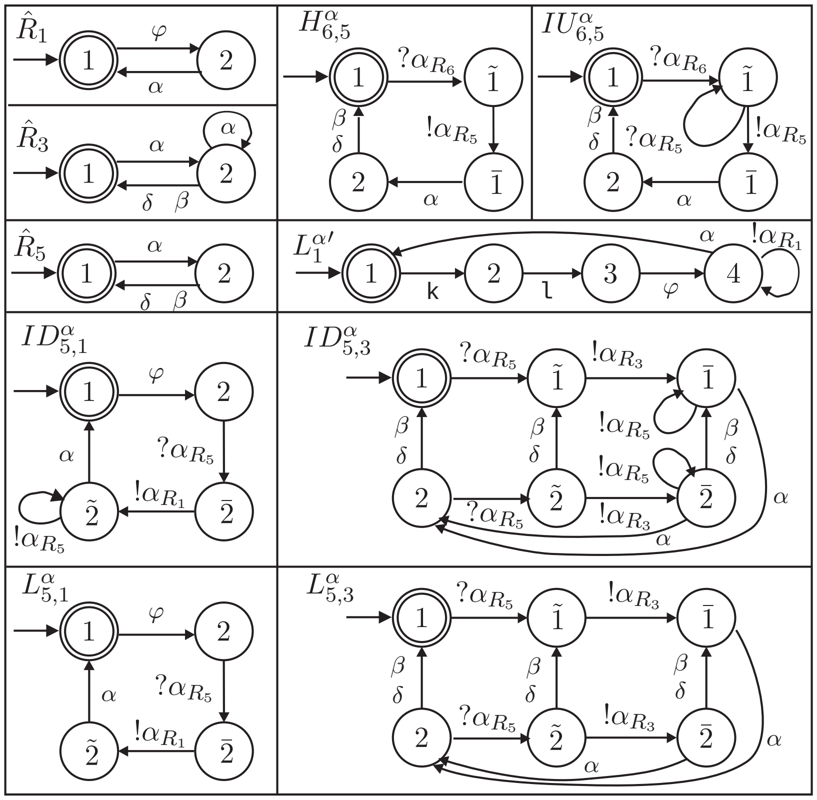

Fig. 2. Communication model components.

$$\theta(||_{k=1}^{n}L(CM_k)) \;=\; ||_{k=1}^{n}L(R_k)$$

That is, the overall communication behavior modeled by CM is nonblocking and enables the same sequences of events as the original architecture of hierarchical supervisors. Note that communication events (questions or answers) are shared between neighboring CMs if and only if they correspond to neighboring supervisors in the supervisor hierarchy. Hence, our CM supports communication according to the strategy described in Section III-A.

IV. Controller Aggregation

The communication models presented in the previous section have been derived with the assumption that all supervisors are implemented on controller devices in distinct physical locations. However, in a practical implementation, it is potentially the case that multiple supervisors are assigned to the same controller device.

A. 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 5: Fig. 3 shows two possible grouping scenarios, where gray boxes indicate supervisors that occupy the same group. For example, in Fig. 3 (a), $\mathcal{G} = \{G_1, G_2, G_3\}$ and $g(R_2) = g(R_3) = g(R_5) = G_3$.

It can be observed from the example in Fig. 3 (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. 3 (c). Furthermore, it has to be noted that arbitrary aggregations of controllers are not desirable. Fig. 3 (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 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 1 (ii).

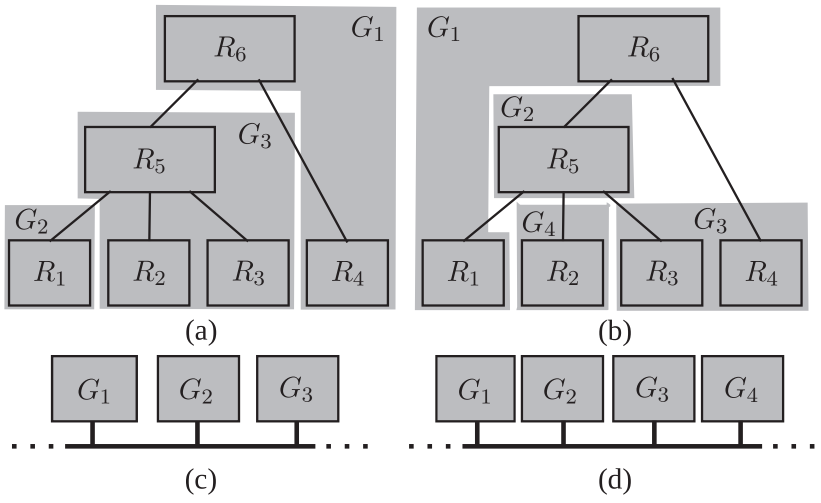

Fig. 3. (a) and (b): Grouping of supervisor components; (c) and (d): Communication relationship.

Definition 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 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, $c_{\mathcal{G}}^{\sigma} : \mathcal{G} \to 2^{\mathcal{G}}$ denotes the restriction of $c_{\mathcal{G}}^{\sigma}$ to groups that contain the event σ. In the sequel, our goal is to adopt the communication strategy described in Section III-A for the grouped case.

In this context, the basic idea is to associate questions and answers to groups instead of supervisors. As a consequence, we require that all supervisors in a group agree on the questions and answers of each group. Example 6 substantiates this idea.

Example 6: We consider the situation in Fig. 3 (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 φ ($?\varphi_{G_3}$ to G_2). After the answer $!\varphi_{G_2}$ from G_2, the command φ is given by G_3 if φ 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 α is feasible in R_3, R_5. The execution of α is then locally decided by G_1 if the answer $!\alpha_{G_3}$ is received and also α is feasible in R_4.

B. Simple Aggregation Method

The goal of this section is the computation of CMs for grouped supervisors. Formally, we denote the CM of each group $G_g \in \mathcal{G}$ as CG_g over the alphabet $\mathcal{K}_g$.

The straightforward approach to realize the communication in the case of grouping is to first compute the communication model CM_k as described in Section III-B for each $R_k \in \mathcal{R}$. Then, the CMs of supervisors that belong to the same group can be executed in parallel on the same controller device, while the result in Theorem 1 remains valid. That is, for each $G_g \in \mathcal{G}$, we arrive at

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

Although the automaton CG_g in (8) need not be evaluated explicitly in a practical implementation, it holds that the component CMs of CG_g contain redundant information that can lead to unnecessarily large CMs. In particular, the component CMs in the same group still introduce questions and answers for the synchronization of shared events among each other although such synchronization could be handled internally. As a result, the number of CM states for the simple aggregation is equal to the number of CM states of a fully distributed implementation.

Example 7: We consider R_3 and R_5 with $g(R_3) = g(R_5) = G_3$ in Fig. 3 (a). If CM_3 and CM_5 are computed according to (8), CM_3 contains the question $?\alpha_{R_5}$ and the answer $!\alpha_{R_3}$ (see $L^{\alpha}_{5,3}$ in Fig. 2) and CM_5 contains the answer $!\alpha_{R_3}$ ($ID^{\alpha}_{5,3}$). Hence, the communication for the event α is included in the CMs although the related supervisors R_3 and R_5 belong to the same group.

C. Aggregated Communication Models

Addressing the issues raised in the previous section, we now develop an aggregation method that avoids unnecessary communication among supervisors in the same group while retaining the communication strategy described in Section III-A and IV-A.

To this end, we redefine the division $\Sigma_k = \Sigma_{k,\mathrm{L}} \dot{\cup} \Sigma_{k,\mathrm{I}} \dot{\cup} \Sigma_{k,\mathrm{H}} \dot{\cup} \Sigma_{k,\mathrm{N}}$ as introduced in Section III-B based on the position of the related supervisor R_k in the tree $T_{\mathcal{G}}$. Here, we assume that $g(R_k) = G_g$.

$$\begin{aligned}\Sigma_{k,\mathrm{L}} := \{\sigma \in \hat{\Sigma}_k | \exists R_j \in a^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_j) \neq G_g \\ \wedge \nexists R_l \in d^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_l) \neq G_g\}\end{aligned} \tag{9}$$

$$\begin{aligned}\Sigma_{k,\mathrm{I}} := \{\sigma \in \hat{\Sigma}_k | \exists R_j \in a^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_j) \neq G_g \\ \wedge \exists R_l \in d^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_l) \neq G_g\}\end{aligned} \tag{10}$$

$$\begin{aligned}\Sigma_{k,\mathrm{H}} := \{\sigma \in \Sigma_k | \nexists R_j \in a^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_j) \neq G_g \\ \wedge \exists R_l \in d^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_l) \neq G_g\}\end{aligned} \tag{11}$$

$$\begin{aligned}\Sigma_{k,\mathrm{N}} := \{\sigma \in \Sigma_k | \nexists R_j \in a^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_j) \neq G_g \\ \wedge \nexists R_l \in d^{\sigma}_{\mathcal{R}}(R_k) \text{ s.t. } g(R_l) \neq G_g\}\end{aligned} \tag{12}$$

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.

Remark 1: Note that the situation in Section III-B corresponds to the special case of the above definition, where all supervisors belong to different groups.

Example 8: Referring to the aggregation in Fig. 3 (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, $\Sigma_{6,\mathrm{H}} = \{\alpha, \beta, \delta\}$,

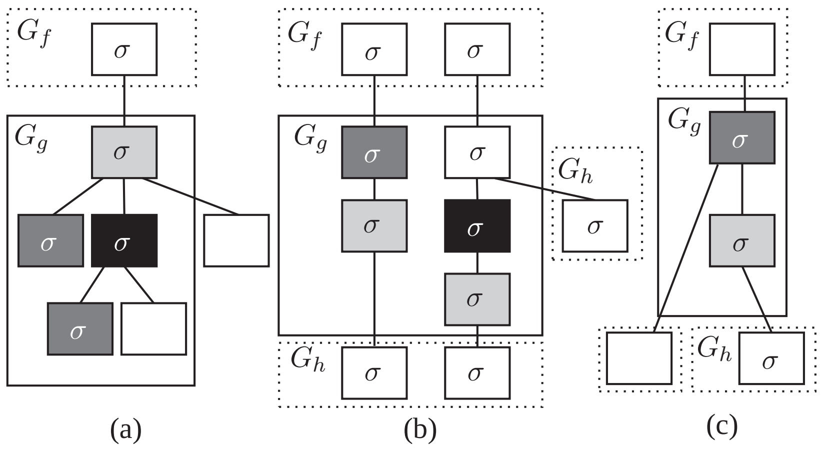

Fig. 4. (a) Low-level group; (b) Intermediate-level group; (c) High-level group.

$\Sigma_{6,\mathrm{N}} = \{\gamma\}$, $\Sigma_{6,\mathrm{L}} = \Sigma_{6,\mathrm{I}} = \emptyset$. It is interesting to note that no communication is required for the shared events β, δ within G_3 and for the shared event γ in G_1.

In principle, we propose to determine the CM for each group according to (8) with an adapted computation for each component CM. In the following sections, we discuss how this CM has to be constructed for events in $\Sigma_{k,\mathrm{L}}$ (Section IV-D), $\Sigma_{k,\mathrm{I}}$ (Section IV-E) and $\Sigma_{k,\mathrm{H}}$ (Section IV-F). The algorithmic CM computation is then summarized in Algorithm 2. In all cases, we consider a supervisor R_k in the group $G_g = g(R_k)$ and the event σ 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.

D. 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 have to be taken into account as depicted in Fig. 4 (a).

L1 R_j belongs to G_f, i.e., $g(R_j) = G_f$ (see the light gray box in Fig. 4 (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 analogous to Section III-B. Hence, $L^{\sigma}_{j,k}$ is computed by Algorithm 2 with the input parameters $\hat{R}_i = \hat{R}_k$, $\Delta = \hat{\Sigma}_k \cup \{?\sigma_{G_f}, !\sigma_{G_g}\}$.

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. 4 (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 by G_g if all supervisors in G_g agree that σ is feasible. Thus, $\hat{R}_i = \hat{R}_k$, $\Delta = \hat{\Sigma}_k \cup \{!\sigma_{G_g}\}$

for the computation of $L^{\sigma}_{j,k}$.

L3 R_j belongs to G_g and $c_{\mathcal{R}}(R_k) \neq \emptyset$ (see the black box in Fig. 4 (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 9: Fig. 5 shows the component $L^{\alpha}_{5,1}$ for R_1 and α (type L1) and $L^{\alpha}_{5,3}$ for R_3 and α (type L2).

E. 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 σ in G_g in Fig. 4 (b)). This situation is analogous to the situation without grouping in Section III-B. Hence, $IU^{\sigma}_{j,k}$ 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. 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 2.

IU2 Now, $g(R_j) = G_g$ (see the light gray boxes in Fig 4 (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. 4 (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 $\Delta = \hat{\Sigma}_k \cup \{?\sigma_{G_f}, ?\sigma_{G_g}\}$.

IU4 If $g(R_j) = G_g$ (see the black box in Fig. 4 (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 $\hat{R}_l$ analogous to the case without aggregation in Section III-B 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 $\Delta = \hat{\Sigma}_l \cup \{?\sigma_{G_g}, !\sigma_{G_h}, !\sigma_{G_g}\}$.

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

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 10: Fig. 5 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 α.

F. 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. 4 (c).

H1 We assume that $G_h := g(R_l) \neq G_g$ (see the light gray box in Fig. 4 (c)). Then, $H^{\sigma}_{k,l}$ is computed as in Section III-B with $\hat{R}_i = \hat{R}_l$ 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. 4 (c)), then no answer is received by R_k. Hence, $!\sigma_{G_h}$ is removed from Δ compared to H1 for the computation of $H^{\sigma}_{k,l}$.

Example 11: The type H1 is illustrated in Fig. 5 by $H^{\alpha}_{6,5}$ for R_6 and α.

Algorithm 2 (Computation of Aggregated CMs): We compute an automaton $G = (Q, \Delta, \nu, q_0, Q_{\mathrm{m}})$ that represents $L^{\sigma}_{j,k}$, $IU^{\sigma}_{j,k}$ or $H^{\sigma}_{k,l}$ depending on the input alphabet Δ.

```
1 Given: $\hat{R}_i$, $\sigma$, $\Delta$.
2 Initialize: $Q = \hat{X}_i$; $q_0 = \hat{x}_{0,k}$; $Q_{\mathrm{m}} = \hat{X}_{\mathrm{m},k}$
%% Introduce states needed for communication
3 for each $x \in \hat{X}_i$ s.t. $\hat{\delta}_i(x, \sigma)!$
4       if $!\sigma_{G_g} \in \Delta$ and $?\sigma_{G_g} \notin \Delta$
5           $Q = Q \cup \{\bar{x}\}$; $\nu(\bar{x}, \sigma) = \hat{\delta}_i(x, \sigma)$
6           if $?\sigma_{G_f} \in \Delta$ %% Case L1
7               $Q = Q \cup \{\tilde{x}\}$
8               $\nu(x, ?\sigma_{G_f}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{G_g}) = \bar{x}$
9           else %% Case L2
10              $\nu(x, !\sigma_{G_g}) = \bar{x}$
11      else if $\{?\sigma_{G_g}, !\sigma_{G_g}\} \subseteq \Delta$
12          $Q = Q \cup \{\bar{x}\}$; $\nu(\bar{x}, \sigma) = \hat{\delta}_i(x, \sigma)$
13          if $?\sigma_{G_f} \in \Delta$ %% Case I1
14              $Q = Q \cup \{\tilde{x}\}$; $\nu(x, ?\sigma_{G_f}) = \tilde{x}$
15              $\nu(\tilde{x}, ?\sigma_{G_g}) = \tilde{x}$; $\nu(\tilde{x}, !\sigma_{G_g}) = \bar{x}$
16          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** $G$
```

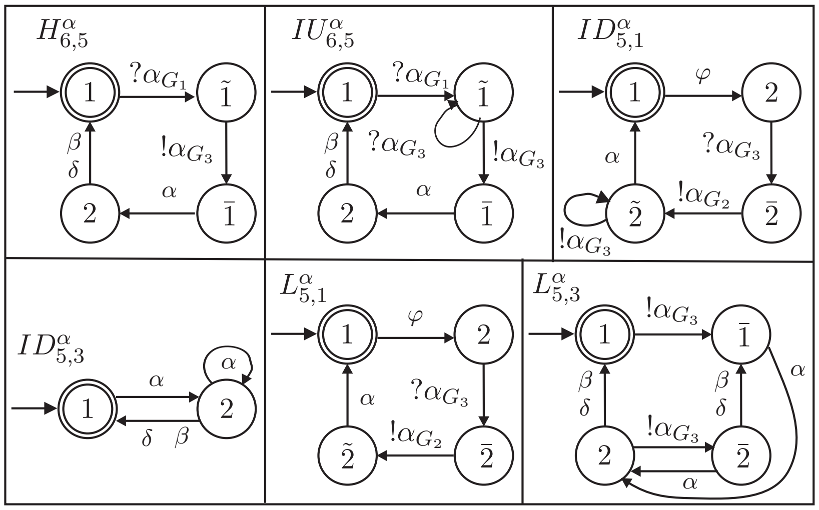

Fig. 5. Grouped communication model components.

G. Communication Model for Groups

Using the CM components in Section IV-D to IV-F, the CM CM_k for each supervisor $R_k \in \mathcal{R}$ is then again computed using (7). The evaluation of CG_g for each group G_g can is then performed with (8), where unnecessary communication among supervisors in the same group has been removed as described in Section IV-D to IV-F.

The aggregated CMs computed in this section retain the desirable properties introduced in Theorem 1.[4]

Theorem 2 (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

$$\begin{aligned}\overline{||_{g=1}^{|\mathcal{G}|} L_{\mathrm{m}}(CG_g)} &= ||_{i=k}^{|\mathcal{G}|} L(CG_g)\\ \theta(||_{k=1}^{|\mathcal{G}|} L(CG_g)) &= ||_{k=1}^{n} L(R_k)\end{aligned}$$

That is, 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.

[4]The proof of Theorem 1 is not in the scope of this paper.

V. Application Example

A. General Setup

The presented ideas are applied to the *distribution system* (ds) in Fig. 6. 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 (connection between c1a and c1b).

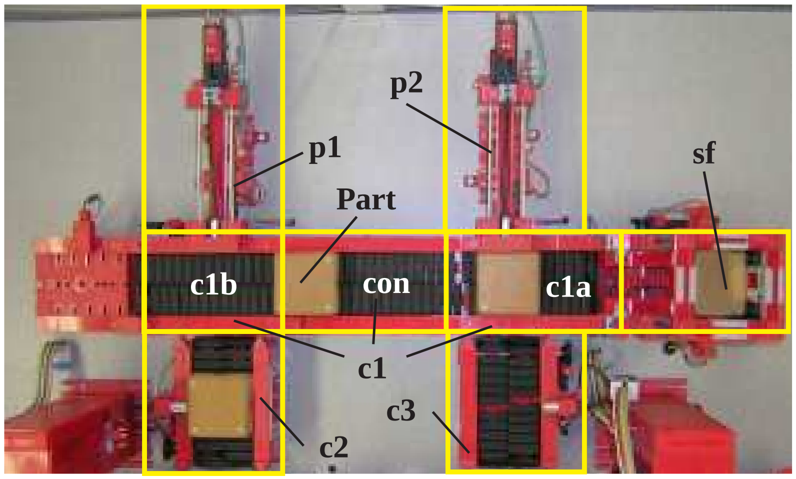

Fig. 6. Distribution system overview.

The supervisor synthesis for ds has been performed analogous to [8]. Fig. 7 shows the resulting hierarchy with 4 levels and 12 supervisors, whose respective state counts are listed in Table I. 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.

B. Controller Aggregation

For comparison, we first evaluate the CMs of the simple aggregation in Section IV-B, 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 I.

We now illustrate the grouping idea by two scenarios. In Fig. 8, 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

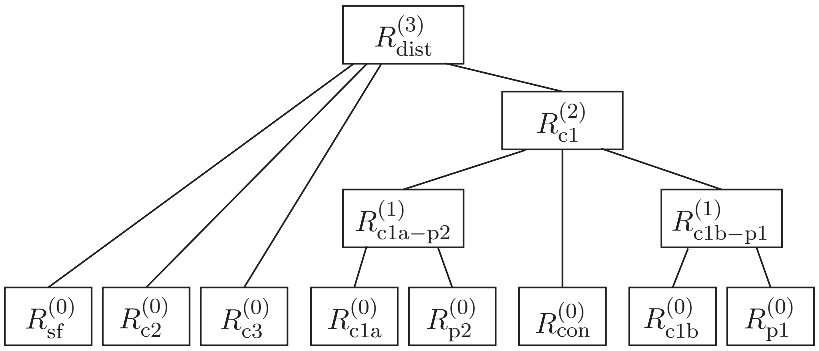

Fig. 7. Supervisor hierarchy of the distribution system.

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

TABLE I
NUMBER OF SUPERVISOR (SUP) AND CM STATES FOR DIFFERENT COMPONENTS.

coordinated by the superposed supervisor $R^{(3)}_{\mathrm{dist}}$ on a separate controller device. The state counts of the CMs for the corresponding 7 groups computed according to Section IV-G are depicted in Table II. It can be seen that a reduction from 3595 to 1120 states is achieved compared to the simple aggregation, which can be explained by the removal of internal communication in the group G_5.

	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	$\sum$
Fig. 8	527	21	21	19	428	22	82	—	1120
Fig. 9	727	21	21	19	61	22	37	22	930

TABLE II
NUMBER OF STATES FOR THE CONFIGURATIONS IN FIG. 8 AND 9.

The second scenario in Fig. 9 considers that local controllers are used for the components c2, c3,

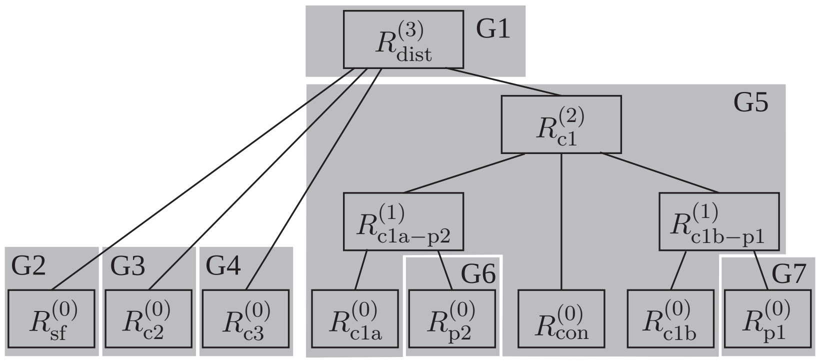

Fig. 8. Grouping of functional entities.

c1a, c1b, p1, p2 and sf that exchange sensor and actuator information with the plant. All remaining supervisors are employed for coordination on a separate controller device (group G_1). Again, a reduction to 930 states due to the avoidance of internal communication in G_1 can be seen in Table II. In our study, it could be determined that it is favorable to group supervisors on different hierarchical levels that share multiple events.

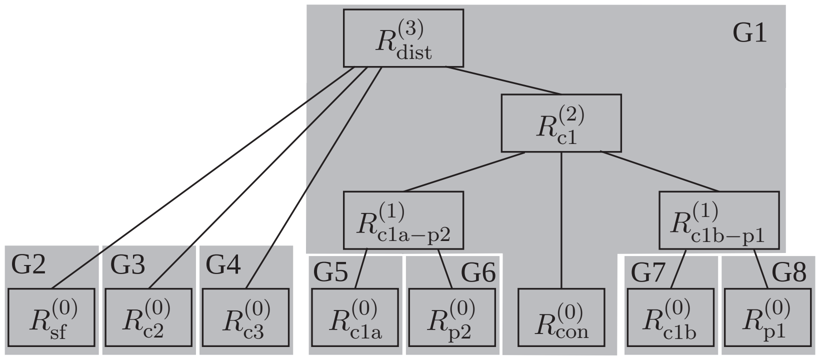

Fig. 9. Grouping with high-level coordination.

VI. 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 if our supervisor aggregation idea is employed.

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 [10]. Furthermore, the fully distributed communication models for switched networks in [9] will be adapted to the general case with supervisor aggregation.

REFERENCES

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

[2] 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.

[3] L. Feng and W. Wonham, “Supervisory control architecture for discrete-event systems,” *Automatic Control, IEEE Transactions on*, vol. 53, no. 6, pp. 1449–1461, July 2008.

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

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

[6] 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.

[7] K. Schmidt, “Controller aggregation for distributed discrete event supervisors on a shared-medium network,” in *Workshop on Dependable Control of Discrete Event Systems*, 2009.

[8] K. Schmidt, T. Moor, and S. Perk, “Nonblocking hierarchical control of decentralized discrete event systems,” *Automatic Control, IEEE Transactions on*, vol. 53, no. 10, pp. 2252–2265, Nov. 2008.

[9] K. Schmidt and E. G. Schmidt, “Communication of distributed discrete-event supervisors on a switched network,” in *Workshop on Discrete Event Systems*, 2008.

[10] K. Schmidt, E. G. Schmidt, and J. Zaddach, “Reliable and safe operation of distributed discrete-event controllers: A networked implementation with real-time guarantees,” in *IFAC World Congress*, 2008.