2002

Synthesizing Distributed Controllers for the Safe Operation of ConnectedSpaces

Baskar Sridharan

Aditya P. Mathur
*Purdue University, apm@cs.purdue.edu*

Kai-Yuan Cai

Report Number:
02-024


This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epub@purdue.edu for additional information.
Synthesizing Distributed Controllers for the Safe Operation of ConnectedSpaces

Baskar Sridharan* Aditya P. Mathur†
Software Engineering Research Center
Department of Computer Sciences
Purdue University
West Lafayette, IN - 47907, USA
{baskars, apm}@cs.purdue.edu

Kai-Yuan Cai‡
Department of Automatic Control
Beijing University of Aeronautics and Astronautics
Beijing 100083, China
kyc@ns.dept3.buaa.edu.cn

Abstract

A collection of one or more devices, each described by its Digital Device Manual and reachable over a network, is a ConnectedSpace. A set of safety policies may be enforced on a ConnectedSpace to ensure the safety of the environment in which the ConnectedSpace is deployed. The enforcement of these safety policies by one or more safety controllers governs the behavior of the devices within the ConnectedSpace. We propose a policy-based partitioning scheme for synthesizing distributed safety controllers such that (a) each device is guaranteed to be controlled by no more than two controllers, and (b) each policy is guaranteed to be enforced by exactly one controller. We present an experimental evaluation of our scheme. The experimental results show that the scheme is scalable with respect to the number of devices and the number of policies. We also show how safety controllers, that are correct with respect to the policies, are synthesized using the theory of supervisory control.

1. Introduction

Devices controlled via embedded software are becoming widespread. Many such devices are capable of connecting to a local network or the Internet. It may also be possible to monitor and control such devices from a remote location. Such devices are sometimes referred to as "smart" devices. A ConnectedSpace is a collection of one or more such devices. ConnectedSpaces are often deployed in environments such as aircraft, hospitals and health care units, automobiles, and homes. In such environments, interactions of two or more devices may lead to an unsafe situation. For example, consider the cabin of a passenger aircraft. NASA's Air Safety Reporting System lists 52 incidents where a portable electronic device was suspected to interfere with the aircraft navigation and control systems [2, 3]. For example, the headphones used by a passenger on a Boeing 757 was suspected to have caused all three autopilot systems on the aircraft to stop functioning. The use of a cellular phone inside a Cessna 340A was reported as the likely cause for erroneous readings of the cockpit meters. Similar examples may be found in hospital and health care environments. For example, the interference of an Magnetic Resonance Imaging device with a cardiac Pacemaker is well known [18]. Such interference may even be fatal [1].

In each of the above examples, the unsafe situation might arise due to the interaction amongst two or more devices that are individually safe. Such unsafe situations may be prevented by enforcing safety policies on the ConnectedSpace to which the devices belong. Several analysis tools such as fault-trees and Petri nets may be used to obtain the safety policies [6, 7]. Though safety policies exist for many environments [5], to our knowledge there do not seem to exist any safety standard for the automatic enforcement of such policies on an arbitrary collection of devices. The ability to remotely monitor and control the devices provides opportunities for the automatic enforcement of the policies. Also, the environments in which these devices exist may be modeled as a ConnectedSpace. For instance, an imaging room in a health care unit, or an aircraft, may be modeled as a ConnectedSpace. The safety policies for such environments may then be enforced on the ConnectedSpace by one or more safety controllers.

The theory of supervisory control of Discrete Event Systems (DES), based on a framework of automata and formal languages, was proposed by Ramadge and Wonham [12, 13]. This theory (RW theory) can be used to synthesize a controller when the DES and the control specifications (policies) are both modeled as generators of formal...
languages. The controller is guaranteed to be correct with respect to the specification.

Ostroff [9, 10] proposed a framework based on Timed Transition Model (TTM) and Real-Time Temporal Logic (RTTL) for synthesizing controllers for Real-time DES (RDES). Given an RDES modeled using TTM, and the control specifications expressed using RTTL, the controller may be synthesized using the Controller Design Procedure (CDP) described in [9]. Brandin and Wonham [4] show how the RW theory may be used for RDES. Sridharan et al. [15] describe CS-CDP, a modified CDP, for synthesizing a monolithic safety controller for ConnectedSpaces. It is important to emphasize that these techniques employ a constructive approach in contrast to a verification-based approach for synthesizing controllers. A constructive approach aims at the synthesis of a controller that is correct with respect to a given specification. In a verification-based approach, the controller is synthesized and then verified for its correctness. The automated synthesis of correct safety controllers is critical in ConnectedSpaces. A constructive approach is useful in such situations.

A. Our Contributions

1. A policy-based partitioning scheme (PPS) is proposed for suitably partitioning a given set of safety policies into disjoint sets.
2. PPS is evaluated experimentally and the results are presented.
3. The procedure for synthesizing the safety controllers, using the partitions generated by the PPS, is described.

B. Organization of this Paper

The remainder of this paper is organized as follows. ConnectedSpaces are described in Section 2. Safety policies are defined in Section 3. The criteria for the synthesis of the controllers are enumerated in Section 4. Section 5 presents the policy-based partitioning scheme. The results of an experimental evaluation of PPS is presented in Section 6. Section 7 describes the procedure for synthesizing the safety controllers using CS-CDP. Possible extensions of the work are discussed in Section 8.

2. ConnectedSpaces

A collection of one or more devices, each described by its Digital Device Manual and reachable over a network, is a ConnectedSpace. The behavior, services, and attributes of each device is specified in the form of a Digital Device Manual (DDM) [15, 16, 17]. A device may carry either its DDM or a pointer to the location of its DDM. The DDM for a cardiac Pacemaker is shown in Figure 1.

Formally, a ConnectedSpace is defined as $CS = \{D_1, D_2, \ldots, D_m\}$ where each $D_i = 1 \ldots m$, is a DDM. A DDM $D$ is given by $(CSM, F, LS, O, A)$ where

- $CSM$ is an augmented finite state automaton for the device. $CSM$ is given by a 6-tuple $(Q, \Sigma, \delta, \Omega, q_0, SD)$ where $Q = Q_u \cup Q_s$ is a finite set of states, $\Sigma$ is a possibly empty set of events, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, $\Omega : \Sigma \rightarrow \mathbb{N}$ gives the cost associated with a transition, $q_0$ is the initial state of the device, and $SD : Q \rightarrow \text{string}$ is the semantic description function. $Q_u$ and $Q_s$ are non-empty sets of states such that $8s \in Q_u$, there is a $r \in \Sigma$ such that $\delta(s, r) \in Q_a$. In practice, $Q_u$ may consist of states that may potentially lead to an unsafe situation, and $Q_s$ consists of states that are guaranteed not to cause an unsafe situation. For the purpose of this paper, we assume that $\Omega$ gives the maximum time duration required to complete a transition.

- $F = \{f_0, f_1, f_2, \ldots, f_b\}$ is a set of mapping functions. $f_0 : O \rightarrow \Sigma$ must be defined by all devices. $f_0$ maps the functionality of the device to an event in the finite state machine.

- $LS = \{l_0, l_1, \ldots, l_r\}$ is a set of labels where each $l_i, i = 1 \ldots r$ identifies a particular geographic location.

- $O = \{o_1, o_2, \ldots, o_u\}$ is the set of services provided by the device. Invocation of $o_i, i = 1 \ldots u$ generates an event $\sigma \in \Sigma$.

- $A = \{a_1, a_2, \ldots, a_h, \text{curState}, \text{curLoc}\}$ is the set of attributes exported by the device. $\text{curState} \in Q$ and $\text{curLoc} \in LS$ specify the current state and location of the device, respectively, and are exported by all devices. The initial values of $\text{curState}$ and $\text{curLoc}$ are set to $q_0$ and $l_0$, respectively.

The safe behavior of a ConnectedSpace is governed by the correct enforcement of the associated safety policies. The policies are assumed to reside in a Policy Server (PS). The policies may be added, deleted, or modified at any time during the operation of the ConnectedSpace. In addition, the number and type of the constituent devices may also change. Such a change may be detected by a Device Discovery Service (DDS). DDS may utilize an appropriate discovery protocol. Each device must adhere to the discovery protocol. The policies in PS may be enforced by a Safety Controller (SC). An architecture for enforcing a set of safety policies would consist of the ConnectedSpace, the policy server, the device discovery service, and one or more safety controllers. Figure 2 shows an enforcement architecture with distributed safety controllers. The PPS approach proposed in this work addresses the issue of synthesizing safety controllers for such an architecture.
that differ in their safety criticality requirements. For example, it is possible for a laboratory to require a low safety criticality for certain policies, i.e., it may tolerate violations of these policies for a brief period, whereas a healthcare unit may not tolerate any safety violation. The notion of policy relaxation duration helps specify and enforce such a requirement. It is possible that the devices constituting the ConnectedSpace may have varying safety criticality. Also, the same type of device may have differing safety criticality depending on the environment and the safety policies that govern its behavior. The notion of safety criticality rank provides the ability to specify and enforce such requirements.

Consider a ConnectedSpace that consists of a Magnetic Resonance Imaging (MRI) device and a cardiac Pacemaker (PM). For such a ConnectedSpace, a safety policy may be specified as PMSAFE = (\text{curState}_{PM} = \text{PACING} \land \text{curState}_{MRI} = \text{IMAGING}) \langle\text{PM}, \text{MRI}\rangle(25). The policy is interpreted as follows. The policy is violated when PM and MRI are in PACING and IMAGING states, respectively. Also, PM has a higher safety criticality rank than MRI. Hence, when PM is in PACING, MRI must not be allowed to move to IMAGING. If MRI is already in IMAGING, and if there is a request to move PM to PACING, then the safety controller has the option of moving MRI to a different state in order to permit the request. Also, during this move, the boolean expression must not remain false for more than 25 time units.

4. Criteria for safety controller synthesis

In safety-critical systems such as a ConnectedSpace, several criteria constrain the synthesis of the controller. The following criteria are addressed in this work.

R1. The termination of a controller must only affect the behavior of a subset of devices in the ConnectedSpace.
R2. Across ConnectedSpaces, each device must be controlled by a fixed number of controllers.
R3. The controller must be correct with respect to the safety policies.

R1 is important for ensuring that the safe behavior of the ConnectedSpace is enforced in a fault-tolerant manner. R2 is important in the design of devices that have a limited memory. Such devices may enter and leave multiple ConnectedSpaces. When such a device exists in a ConnectedSpace, and is controlled by one or more safety controllers, all requests arriving at the device may be sent to a controller for authorization. The addresses of these controllers may be stored in the device's memory. Typically, the amount of memory for storing these addresses is fixed. If the maximum number of controllers for each device is
not fixed, then it is possible for the device to cause an unsafe situation. The inability of the device to contact all its controllers, due to the insufficient memory to store all the addresses, may cause such a situation. R3 is vital in safety-critical systems such as ConnectedSpaces. To enforce a safe behavior on such systems, the safety policies must correctly model the safe behavior. Also, the controller must be correct with respect to the policies. An incorrect controller may lead to unsafe situations that are difficult to detect and diagnose.

5. Policy-based partitioning scheme

R1 may be satisfied using distributed safety controllers, and by assigning only a subset of policies and devices to each controller. The assignment of policies and devices to controllers must also satisfy R2. The proposed policy-based partitioning scheme (PPS) produces partitions consisting of policies and devices, that satisfy R1 and R2. PPS produces non-empty partitions such that each device is present in no more than 2 partitions, and where each policy is assigned to a unique partition.

Given a ConnectedSpace CS, a set of safety policies SP, and the suggested number of partitions \( g \in \mathbb{N} \), PPS computes a set of partitions \( \{ \mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k \} \), \( 2 \leq k \leq g \), where \( \mathcal{G}_i = (V_i, E_i) \), \( V_i \subseteq SP \) and \( E_i \subseteq CS \), \( i = 1 \ldots k \). The following properties are valid for the partitions produced by PPS.

**Prop1.** \( V_i \cap V_j = \emptyset, i \neq j, i, j = 1 \ldots k \).

**Prop2.** \( v \in E_i \cap E_j, e \not\in E_i \cap E_p \) and \( e \not\in E_q \cap E_j \), where \( \mathcal{G}_p, \mathcal{G}_q \in pp \).

PPS constructs a matrix, called a policy partition matrix (PPM) using CS and SP. PPM is similar to an adjacency matrix for an undirected graph. An edge-weighted, undirected graph \( G \) is then constructed from PPM. A heuristic, that exploits the presence of bridges in \( G \), is used to assign weights to the edges of \( G \). PPM uses the heuristic to maximize \( k \). A minimum-weight \( k \)-cut of \( G \) is then obtained using the SPLIT [14] algorithm. The cut is used to obtain the corresponding partitions of CS and SP.

PPS is implemented by executing procedures Proc1 [16], Proc2, and Proc3 in sequence. Proc1 constructs the PPM. Proc2 constructs \( G \) from PPM. The set of partitions \( pp = \{ \mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k \} \) is then computed by Proc3. Proc2 is shown in Figure 3 and Figure 4. Step 4 and Step 5 of Proc2 implement the heuristic. Proc3 is shown in Figure 5.

The characteristics of the results produced by SPLIT may be used to show that Prop1 and Prop2 are indeed true. By definition, the partitions produced by SPLIT satisfy \( V_i \cap V_j = \emptyset, i \neq j \). Each \( P_i \in SP \) corresponds to exactly one vertex \( v_i \in V \) of \( G \). Hence, \( V_i \cap V_j = \emptyset, i \neq j \) is satisfied for the sets \( V_i, i = 1 \ldots k \) producted by Step 6 of Proc3. Hence, Prop1 is true. Also, SPLIT produces partitions \( E_{ij} \cap E_{ij} = \emptyset, i \neq j, i, j = 1 \ldots k \), and \( E - \{ E_1, E_2, \ldots, E_k \} = S \). Hence, \( E_i \cap E_j = \emptyset, i \neq j, i, j = 1 \ldots k \) is true. In Step 7 of Proc3, for each edge \( D_{j,p,q} \in S \), the corresponding \( D_{j} \in SP \) is added to exactly two sets \( E_p \) and \( E_q \). Hence, Prop2 is true.

5.1. An example

Consider a ConnectedSpace CSP\( ASPACE = \{ D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10} \} \) where \( D_1, i=1 \ldots 10, \) is the DDM for device \( D_i \). Let pol set = \{ \( \{ P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8 \} \) where \( P_i = \{ I_1, O_1, r_1\} \), \( i = 1 \ldots 8 \) \} be the set of safety policies. Let \( I_1 = \neg (\text{curState}_1 = q_{a,i} \land \text{curState}_2 = q_{a,j}) \), \( I_2 = \neg (\text{curState}_2 = q_{a,2} \land \text{curState}_3 = q_{a,3}) \), \( I_3 = \neg (\text{curState}_1 = q_{a,1}) \), \( I_4 = \neg (\text{curState}_3 = q_{a,3} \land \text{curState}_3 = q_{a,4}) \), \( I_5 = \neg (\text{curState}_4 = q_{a,5}) \), \( I_6 = \neg (\text{curState}_4 = q_{a,6}) \), \( I_7 = \neg (\text{curState}_5 = q_{a,7}) \), \( I_8 = \neg (\text{curState}_5 = q_{a,8}) \), \( I_9 = \neg (\text{curState}_5 = q_{a,9}) \), \( I_{10} = \neg (\text{curState}_5 = q_{a,10}) \).
PROCEDURE Proc2 (continued).

4. let $B = \{b_1, b_2, \ldots, b_l\}$, $0 \leq l \leq |E|$ be the set of bridges of G.
   for $j = 1 \ldots m$ do
     if $\sum_{i=1}^{n} PPM_{i,j} < 2$ then
       initialize $ts_j$ to 1.
     else initialize $ts_j$ to 0.
   for $i = 1 \ldots l$ do
     let $b_i = (v_p, d_j)$.
     if $ts_j = 0$ then
       set $ts_j = 1$ and $w(b_i) = 1$.

5. for $j = 1 \ldots m$ do
   sort the vertices in $N_j$ in the ascending order of their degree.
   if $\sum_{i=1}^{n} PPM_{i,j} > 2$ and $ts_j = 0$ then
     for $z = 1 \ldots |N_j|$ do
       let $N_z = U_1 \cup U_2$ such that $\forall u = (w_z, v) \in U_1$, $w(u) = 1$, and $\forall u = (w_z, v) \in U_2$, $w(u) = \infty$.
       if $|U_1| = \text{degree}(w_z) - 1$ then
         set $w(u) = 1$, $U_2 = \{u\}$, and $ts_j = 1$.
   if $ts_j = 1$ then
     let $V \in N_j$ be a vertex such that $\text{degree}(u) \leq \text{degree}(u)$.
     set $w(e) = 1$, where $e = (v, d_j) \in E$, and $ts_j = 1$.

Figure 4. Continuation of procedure Proc2 to express a policy partition matrix as an edge-weighted, undirected graph $G$.

PROCEDURE Proc3.

INPUT: (a) An edge-weighted, undirected graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$, $0 \leq b \leq m$, and $E = \{(e_1, e_2, \ldots, e_r)\}$, $r \geq m$ are the set of vertices and edges, respectively. (b) A weight function $w: E \rightarrow \mathbb{R}^+$. (c) The suggested number of partitions $g \leq n$.

OUTPUT: A set of partitions $pp = \{G_1, G_2, \ldots, G_k\}$, where $k \leq g$, $G_i = (\tilde{V}, \tilde{E})$, $\tilde{V} \subseteq SP$, and $\tilde{E} \subseteq CS$, $i = 1 \ldots k$.

1. let $E = E_1 \cup E_2$ such that $\forall e \in E_1$, $w(e) = 1$, and $\forall e \in E_2$, $w(e) = \infty$.
2. let $cc = \{G_1, G_2, \ldots, G_{k_{max}}\}$ be the connected components of $G' = (V, E')$.
3. if $g > k_{max}$ then
   set $k = k_{max}$.
4. set $k = g$.
5. determine the $k$-cut for $G$ using the SPLIT algorithm.
   let $p = \{G_1, G_2, \ldots, G_k\}$ be the partitions of $G$, and let $S = \{(v_1, v_2, \ldots, v_s)\}$ be the edges of the cut, produced by the algorithm. $p$ is obtained by removing the edges in $S$ from $E$.
   Each partition $G_i \in p$ is a connected component. $G_i$ is given by $G = (V_i, E_i)$ where $V_i \subseteq V$ and $E_i \subseteq E$.
6. for each $G_i \in p, i = 1 \ldots k$ do
   set $\tilde{V}_i = \emptyset$ and $\tilde{E}_i = \emptyset$.
   for each edge $e = (v_p, v_q) \in E_i$ labelled $D_{i,p,q}$ do
     add the corresponding $D_{i,p,q}$ to $\tilde{E}_i$.
   for each edge $e = (v_p, v_q) \in E_i$ labelled $D_{i,p,q}$ do
     set $w(u) = 1$, where $e = (v, d_j) \in E$, and $ts_j = 1$.
7. for each edge $e = (v_p, v_q) \in S$, $i = 1 \ldots s$ do
   a. $e \in D_{i,p,q}$ in $S$.
   b. $e \in D_{i,p,q}$ in $S$.
   c. $e \in D_{i,p,q}$ in $S$.
   d. $e \in D_{i,p,q}$ in $S$.
   e. $e \in D_{i,p,q}$ in $S$.
   f. $e \in D_{i,p,q}$ in $S$.
   g. $e \in D_{i,p,q}$ in $S$.
   h. $e \in D_{i,p,q}$ in $S$.
8. set $pp = \{\tilde{G}_1, \tilde{G}_2, \ldots, \tilde{G}_k\}$ where $\tilde{G}_i = (\tilde{V}_i, \tilde{E}_i)$, $i = 1 \ldots k$.

Figure 5. Procedure Proc3 for partitioning an edge-weighted, undirected graph G of a ConnectedSpace.

$k = g = 5$. Proc3 computes $p = \{G_1, G_2, G_3, G_4, G_5\}$ where $G_1 = \{(v_1, v_2), (D_{1,1,9}, D_{2,1,10})\}$, $G_2 = \{(v_2, v_3, v_10), (D_{2,1,10}, D_{2,2,10}, D_{3,2,10}, D_{3,3,3}, D_{3,3,3})\}$, $G_3 = \{(v_3, v_4, v_11), (D_{4,4,11}, D_{4,6,11}, D_{4,5,11}, D_{5,6,7}, D_{6,9,7}, D_{6,9,7})\}$, $G_4 = \{(v_5), (D_{4,5,11})\}$, and $G_5 = \{(v_6, v_{12}, v_{13})\}$, $D_{9,13,s}, D_{9,13,s})$. The set of partitions $pp = \{\tilde{G}_1, \tilde{G}_2, \tilde{G}_3, \tilde{G}_4, \tilde{G}_5\}$ produced by Step 8 is shown in Table 2.

6. Experimental evaluation of PPS

PPS was implemented in C++ using the LEDA [8] libraries. The implementation was used to experiment with PPS. The experiments were conducted on a Pentium-III machine, with 256MB of RAM, operating at 700MHz, and running the Linux operating system.

A set of safety policies $SP$ is characterized using the notion of density. The density $d$ of $SP$ is defined using

$$d = \frac{\sum_{i=1}^{n} PPM_{i,j} \leq 2}{\sum_{i=1}^{n} PPM_{i,j}}.$$
the corresponding PPM. Specifically, density is calculated as

\[ d = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} PPM_{ij}}{nm} \]

The size of a ConnectedSpace \( CS \) is specified by the number of policies \( n = |SP| \) and the number of devices \( m = |CS| \). ConnectedSpaces are classified based on their size as (a) small, (b) medium, (c) large, and (d) huge. The CPU usage, in seconds, of Proce1, Proce2, and Proce3 are denoted by \( T_{Proce1} \), \( T_{Proce2} \), and \( T_{Proce3} \), respectively. The CPU usage for the combined execution of Proce1, Proce2, and Proce3, is denoted by \( T_{PPS} \).

For each of the four classes of ConnectedSpaces, experiments were conducted to study the impact of \( d \), \( n \), and \( m \), on (a) \( T_{Proce1} \), \( T_{Proce2} \), and \( T_{Proce3} \), (b) the maximum number of partitions \( k_{\max} \) produced by \( PPS \), (c) the number of edges \( |E| \) of \( G \), and (d) the average degree per vertex \( \text{deg}_{\text{avg}} \) of \( G \).

For each class of ConnectedSpace, representative values for \( n \), \( m \), and \( k \) are chosen. \( SP \) is then generated uniformly at random. A single run of the experiment consists of executing \( PPS \) for \( d = 0.1, 0.2, \ldots, 1 \), for each class. For each run of the experiment, values of \( k_{\max}, T_{Proce1}, T_{Proce2}, T_{Proce3}, |E|, |V|, \) and \( \text{deg}_{\text{avg}} \) are collected. The values are then averaged over 10 runs of the experiment.

Table 3 shows the representative values of \( n \), \( m \), \( k \), \(|E| \), and \(|V| \), for \( d = 0.4 \), for each of the four classes. Table 3 also shows the average values of \( T_{Proce2}, T_{Proce3}, \) and \( T_{PPS} \).

The effect of density on \( T_{PPS} \) is shown in Figure 7. As may be observed from Table 3 and Figure 7, \( T_{PPS} \) increases with increasing size of the ConnectedSpace and the density of \( SP \). As can be observed from Table 3, the percentage CPU usage of Proce3 increases with increase in the size of the ConnectedSpace. Also, the running time of \( PPS \) is dominated by Proce3.

The impact of \( d \) on \( k_{\max} \) is shown in Figure 8. As may be observed from the figure, the value of \( k_{\max} \) is higher for \( 0 < d < 0.5 \). For \( d \geq 0.5 \), \( k_{\max} \) drops to 2. This effect may be observed for each of the four classes. This is due to the increase in \( \text{deg}_{\text{avg}} \) with increasing \( d \). Figure 9 and Figure 10 show the impact of \( d \) on \( \text{deg}_{\text{avg}} \) and \(|E| \), respectively. As may be observed from the figures, both \( d_{\text{avg}} \) and \(|E| \) increase linearly with respect to \( d \).

### Table 3. The characteristic values of \( n, m, k, |V|, \) and \(|E| \) for each of the four classes of ConnectedSpaces are shown in this table. The average values of \( T_{Proce1}, T_{Proce3}, \) and \( T_{PPS} \) for \( d = 0.4 \) are also shown.

| \( n \) | \( m \) | \( k \) | \(|V| \) | \(|E| \) | \( T_{Proce1} \) | \( T_{Proce3} \) | \( T_{PPS} \) |
|------|------|------|------|------|----------|----------|----------|
| small | 100 | 100 | 1 | 1000 | 100 | 0.05 | 0.05 |
| medium | 200 | 200 | 2 | 2000 | 2000 | 0.15 | 0.15 |
| large | 300 | 300 | 3 | 3000 | 3000 | 0.25 | 0.25 |

7. Synthesizing the safety controllers

A safety controller for a ConnectedSpace may be synthesized using CS-CDP by expressing the ConnectedSpace and the safety policies using TTM and RTTL, respectively [15]. CS-CDP produces a safety controller \( SC \) and a control table. \( SC \) is specified using a finite state automaton. The control table defines the action that must be taken by the controller on receiving an event. Given the set of partitions \( PP = \{ G_1, G_2, \ldots, G_k \} \), a controller is assigned to each partition. The set of \( k \) controllers may then be obtained by executing \( \text{SynthProc} \) [16] on each partition in \( PP \). By Prop1, each policy is enforced by exactly one controller. By Prop2, a device is controlled by no more than two con-
controllers. Table 4 shows the assignment of devices and policies to safety controllers for CSPACE.

### Table 4. The assignment of devices and policies to safety controllers, for CSPACE.

<table>
<thead>
<tr>
<th>Device</th>
<th>Safety Controller</th>
<th>Policy</th>
<th>Safety Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>$SC_1$</td>
<td>$P_1$</td>
<td>$SC_1$</td>
</tr>
<tr>
<td>$D_2$</td>
<td>$SC_1, SC_2$</td>
<td>$P_2$</td>
<td>$SC_2$</td>
</tr>
<tr>
<td>$D_3$</td>
<td>$SC_3, SC_4$</td>
<td>$P_3$</td>
<td>$SC_3$</td>
</tr>
<tr>
<td>$D_4$</td>
<td>$SC_5$</td>
<td>$P_4$</td>
<td>$SC_3$</td>
</tr>
<tr>
<td>$D_5$</td>
<td>$SC_5$</td>
<td>$P_5$</td>
<td>$SC_4$</td>
</tr>
<tr>
<td>$D_6$</td>
<td>$SC_5$</td>
<td>$P_6$</td>
<td>$SC_5$</td>
</tr>
<tr>
<td>$D_7$</td>
<td>$SC_5$</td>
<td>$P_7$</td>
<td>$SC_6$</td>
</tr>
<tr>
<td>$D_8$</td>
<td>$SC_5$</td>
<td>$P_8$</td>
<td>$SC_6$</td>
</tr>
<tr>
<td>$D_9$</td>
<td>$SC_5$</td>
<td>$P_9$</td>
<td>$SC_6$</td>
</tr>
<tr>
<td>$D_{10}$</td>
<td>$SC_5$</td>
<td>$P_{10}$</td>
<td>$SC_6$</td>
</tr>
</tbody>
</table>

Let $D_7 \in \text{CSPACE}$ and $D_9 \in \text{CSPACE}$ be PM and MRI, respectively. The DDMs for PM and MRI are shown in Figure 1 and [16], respectively. Let $P_8 \in \text{polset}$ be $P_8 = (\text{curState}_{PM} = \text{PACING} \land \text{curState}_{MRI} = \text{IMAGING} < \text{PM, MRI}>)(25)$. Hence, $q_{7,3} = \text{PACING}$ and $q_{9,2} = \text{IMAGING}$. The safety controller $SC_5$ may be synthesized by using SynthProc on input $G_5$ of CSPACE [16]. The states, the state transition function, and the control table for $SC_5$ are described in [16]. If MRI is currently in state IMAGING, and ASense is requested, $SC_5$ will enable (allow) ASense and then force the transition Stop on MRI. When PM is in state PACING, and the transition Start is requested, it will be disabled (denied). Hence, all transitions of a higher ranked device, such as PM in the example, will be enabled while adhering to the safety policies.

8. Summary and further work

ConnectedSpaces and Digital Device Manuals are described in this paper. A policy-based partitioning scheme is proposed for the synthesis of distributed safety controllers. The results of an experimental evaluation of the scheme is presented. A procedure for synthesizing the safety controllers, using the theory of supervisory control, is described.

Two limitations of PPS need to be addressed. (1) For safety policies with density $> 0.5$, the bridge-based heuristic may only produce a maximum of 2 partitions. To address this limitation, alternate heuristics need to be investigated. (2) In addition to the boolean expression $I$ of a safety policy, the safety criticality ranks of the devices may also need to be incorporated into the graph model. Addressing this limitation may be useful in environments where a subset of the safety controllers need to be made more fault-tolerant than the others.

The constructive approach for controller synthesis described in this paper is suitable in environments where the controllers are deployed external to the devices. Situations might arise where it may not be possible to deploy external safety controllers. A ConnectedSpace in a battle field or an open soccer field are examples of such situations. These situations may require the use of embedded safety controllers. The possible use of PPS for the synthesis of embedded controllers needs to be investigated. In these situations, it is also possible that the number and types of devices, and the safety policies are modified frequently. A long execution duration of PPS may not be acceptable in such cases. To address
Figure 9. The effect of density on the average degree per vertex $d_{avg}$ of $G$.

Figure 10. The effect of density on the total number of edges $|E|$ of $G$.

this issue, the use of incremental partitioning schemes [11] needs to be investigated.

References
