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

TR #02-024

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 k 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 340/A 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

*Baskar Sridharan's work was supported in part by SERC.
†Aditya Mathur's work was supported in part by SERC and NSF.
‡Kai-Yuan Cai's work was supported by the National Natural Science Foundation of China and the "863" programme of China.

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,i=1 ... 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, Σ 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 $SD : Q \rightarrow string$ is the *semantic description* function. Q_s and Q_u are non-empty sets of states such that $\forall s \in Q_u$, there is a $\tau \in \Sigma$ such that $\delta(s, \tau) \in Q_s$. 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 any unsafe situation. For the purpose of this paper, we assume that Ω gives the maximum time duration required to complete a transition.

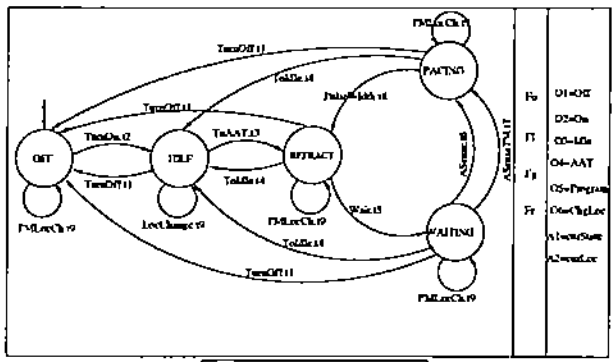
- $F = \{f_0, f_1, f_2, \ldots, f_b\}$ is a set of mapping functions. $f_o : O \rightarrow \Sigma$ must be defined by all devices. f_o maps the functionality of the device to an event in the finite state machine.

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

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

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



The mapping function f_o	
service	event
Off	TurnOff
On	TurnOn
Idle	ToIdle
AAT	ToAAT
ChgLoc	PMLocCh

Figure 1. A DDM for a cardiac Pacemaker.

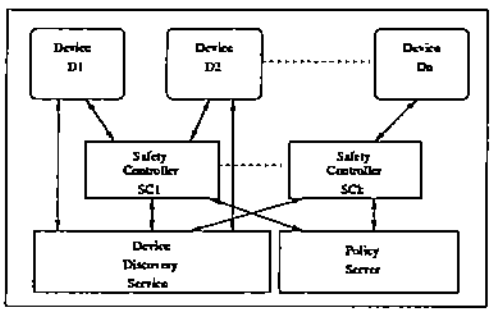


Figure 2. An enforcement architecture with *distributed* safety controllers.

3. Safety policies

A safety policy P is a 3-tuple given by $P=(I, O, r)$ where I is a boolean expression in the attributes and states of the devices in the ConnectedSpace, O is an ordered list of labels that identify a device, and $r \in \mathbb{N}$. The list O consists of the labels $C_i, i = 1 \ldots p$, of those devices whose states or attributes occur in I. A label $C_i, i = 1 \ldots p$, is associated with device D_i. O is represented as $<C_1, C_2, \ldots, C_p>$ where $\text{SCR}(P, D_i) < \text{SCR}(P, D_{i+1}), i = 1 \ldots p-1$. SCR: $P \times CS \rightarrow \mathbb{N}$ defines the rank of a device w.r.t a policy P. $\text{SCR}(P, D_i)$ is the *safety criticality rank* of D_i w.r.t P. The boolean expression I is formed using the attributes and the Q_u states of one or more devices.

A safety policy P is considered *violated* when I remains *false* for a duration greater than r. r, specified in appropriate time units, is the *relaxation duration* for P. It is possible that ConnectedSpaces are deployed in environments 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 health care 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 = $\neg(curState_{PM}$ = PACING $\wedge\ curState_{MRI}$ = IMAGING)<PM,MRI>(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 $pp = \{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_k\}$, $2 \leq k \leq g$, where $\widehat{G}_i = (\widehat{V}_i, \widehat{E}_i)$, $\widehat{V}_i \subseteq SP$ and $\widehat{E}_i \subseteq CS$, $i = 1 \ldots k$. The following properties are valid for the partitions produced by PPS.

`Prop1`. $\widehat{V}_i \cap \widehat{V}_j = \emptyset, i \neq j, i, j = 1 \ldots k$.

`Prop2`. $\forall e \in \widehat{E}_i \cap \widehat{E}_j$, $e \notin \widehat{E}_i \cap \widehat{E}_p$ and $e \notin \widehat{E}_q \cap \widehat{E}_j$, where $\widehat{G}_p, \widehat{G}_q \in pp$.

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 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, $\widehat{V}_i \cap \widehat{V}_j = \emptyset$, $i \neq j$ is satisfied for the sets $\widehat{V}_i, i, j = 1 \ldots k$ produced by Step 6 of `Proc3`. Hence, `Prop1` is true. Also, `SPLIT` produces partitions such that $E_i \cap E_j = \emptyset$, $i \neq j$, $i, j = 1 \ldots k$, and $E - \{E_1, E_2, \ldots, E_k\} = S$. Hence, $\widehat{E}_i \cap \widehat{E}_j = \emptyset$, $i \neq j$, $i, j = 1 \ldots k$ 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 $\widehat{E}_g$ and $\widehat{E}_h$. Hence, `Prop2` is true.

PROCEDURE `Proc2`.

INPUT: A policy partition matrix $PPM_{i,j}$, $i = 1 \ldots n$, $j = 1 \ldots m$.
OUTPUT: (a) An edge-weighted, undirected graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n, d_1, \ldots, d_b\}$, $b \leq m$, and $E = \{e_1, e_2, \ldots, e_r\}$, $r \geq m$ are the set of vertices and edges, respectively. (b) A weight function $w : E \rightarrow \mathbb{R}^+$.

1. initialize $N_j = \{\}$, $j = 1 \ldots m$, $V = \emptyset$, $E = \emptyset$.
2. for $i = 1 \ldots n$ do
   create vertex v_i and add to V.
   for $j = 1 \ldots m$ do
   if $\sum_{i=1}^{n} \text{PPM}_{i,j} = 1$ or $\sum_{i=1}^{n} \text{PPM}_{i,j} > 2$ then
   create vertex d_j and add it to V.
3. for $j = 1 \ldots m$ do
   for $i = 1 \ldots n$ do
   if $\text{PPM}_{i,j} = 1$ then
   add v_i to N_j.
   if $\sum_{i=1}^{n} \text{PPM}_{i,j} = 2$ then
   let $e = (v_i, v_q)$ be an edge labelled $D_{j,p,q}$ such that $\text{PPM}_{i,q} = 1$, $i \neq q$.
   if $e \notin E$ then add e to E.
   set $w(e) = 1$.
   else
   add edge $e = (v_i, d_j)$ to E.
   set $w(e) = \infty$. (continued in Figure 4)

Figure 3. The procedure `Proc2` to express a policy partition matrix as an edge-weighted, undirected graph G.

5.1. An example

Consider a ConnectedSpace `CSPACE` = $\{D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}\}$ where D_i, i=1...10, is the DDM for device D_i. Let `polset` = $\{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$ where P_i=(I_i, O_i, r_i), i=1...8, be the set of safety policies. Let $I_1 = \neg(curState_1 = q_{1,1} \wedge curState_2 = q_{2,1})$, $I_2 = \neg(curState_2 = q_{2,2} \wedge curState_3 = q_{3,1})$, $I_3 = \neg(curState_2 = q_{2,1} \vee curState_3 = q_{3,0} \vee curState_8 = q_{8,4})$, $I_4 = \neg(curState_3 = q_{3,0} \wedge (curState_5 = q_{5,3} \vee curState_3 = q_{3,0} \vee curState_8 = q_{8,4})$, $I_5 = \neg(curState_4 = q_{4,6})$, $I_6 = \neg(curState_4$

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 $\sum_{i=1}^{n} \text{PPM}_{i,j} \leq 2$ **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 $\sum_{i=1}^{n} \text{PPM}_{i,j} > 2$ and $ts_j = 0$ **then**
       for $z = 1 \ldots |N_j|$ **do**
         let $N_j = U_1 \cup U_2$ 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 degree(v) $\leq$degree(u), $\forall u \in N_j, u \neq v$.
       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 $V = \{v_1, v_2, \ldots, v_n, d_1, d2, \ldots, d_b\}$, $0 \leq b \leq m$, 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 $pp = \{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_k\}$, where $k \leq g$, $\widehat{G}_i = (\widehat{V}_i, \widehat{E}_i)$, $\widehat{V}_i \subseteq SP$, and $\widehat{E} \subseteq CS$, $i = 1 \ldots k$.

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 $G' = (V, E_1)$.
3. if $g > k_{max}$ **then**
     set $k = k_{max}$, $pp = cc$ and proceed to Step 6.
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 $S = \{e_1, e_2, \ldots, e_s\}$ 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 $\widehat{V}_i = \emptyset$ and $\widehat{E}_i = \emptyset$.
     for each $v_r \in V_i$ **do**
       add the corresponding $P_r \in SP$ to $\widehat{V}_i$.
     for each edge $e = (v_p, v_q) \in E_i$ labelled $D_{j,p,q}$ **do**
       add the corresponding $D_j \in CS$ to $\widehat{E}_i$.
7. for each edge $e_i = (v_p, d_j) \in S$, $i = 1 \ldots s$ **do**
     add the corresponding $D_j \in CS$ to $\widehat{E}_w$ and $\widehat{E}_h$ where $v_p \in \widehat{E}_w$ and $v_q \in \widehat{E}_h$, $p \neq q$.
8. set $pp = \{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_k\}$ where $\widehat{G}_i = (\widehat{V}_i, \widehat{E}_i)$, $i = 1 \ldots k$.

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

$= q_{4,1} \wedge curState_6 = q_{6,0})$, $I_7 = \neg(curState_5 = q_{5,5} \vee curState_6 = q_{6,3})$, and $I_8 = \neg(curState_7 = q_{7,3} \vee curState_9 = q_{9,2})$. $q_{j,k} \in Q_j$ is a state of device D_j. Table 1 shows the PPM for CSPACE. Since D_{10} does not occur in any policy, the corresponding column C_{10} is removed from the PPM.

Step 2 of Proc2 creates the vertex set V=$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, d_1$=$v_9, d_2$=$v_{10}, d_4$=$v_{11}, d_7$=$v_{12}, d_9$=$v_{13}\}$ of G. Step 3 computes N_1=$\{v_1\}$, N_2=$\{v_1, v_2, v_3\}$, N_3=$\{v_2, v_3\}$, N_4=$\{v_4, v_5, v_6\}$, N_5=$\{v_4, v_7\}$, N_6=$\{v_6, v_7\}$, N_7=$\{v_8\}$, N_8=$\{v_3, v_4\}$, and N_9=$\{v_8\}$. In Step 3, the edges are created and added to E. Since $|N_1| = 1$, an edge $D_{1,1,9}$ is added between vertices v_1 and v_9. Since $|N_2| = 3$, edges $D_{2,1,10}$, $D_{2,1,10}$, and $D_{2,3,10}$ are added between the pairs of vertices (v_2, v_{10}), (v_1, v_{10}), and (v_3, v_{10}), respectively. Step 3 produces $E = \{D_{1,1,9}, D_{2,1,10}, D_{2,2,10}, D_{2,3,10}, D_{3,2,3}, D_{4,4,11}, D_{4,1,11}, D_{4,6,11}, D_{5,4,7}, D_{6,6,7}, D_{7,12,8}, D_{8,3,4}, D_{9,13,8}\}$. Figure 6 shows the undirected graph G and the weight function w for CSPACE.

Let the suggested number of partitions be $g = 5$. Proc3 computes $k_{max} = 6$ by executing SPLIT for $k = g = 5$. Proc3 computes $p = \{G_1, G_2, G_3, G_4, G_5\}$ where $G_1 = (\{v_1, v_9\}, \{D_{1,1,9}, D_{2,1,10}\})$, $G_2 = (\{v_2, v_3, v_{10}\}, \{D_{2,1,0}, D_{2,2,10}, D_{2,3,10}, D_{3,2,3}, D_{8,3,4}\})$, $G_3 = (\{v_4, v_6, v_7, v_{11}\}, \{D_{4,4,11}, D_{4,6,11}, D_{4,5,11}, D_{5,4,7}, D_{6,6,7}, D_{8,3,4}\})$, $G_4 = (\{v_5\}, \{D_{4,5,11}\})$, and $G_5 = (\{v_8, v_{12}, v_{13}\}, \{D_{9,13,8}, D_{7,12,8}\})$. The set of partitions $pp = \{\widehat{G}_1, \widehat{G}_2, \widehat{G}_3, \widehat{G}_4, \widehat{G}_5\}$ produced by Step 8 is shown in Table 2.

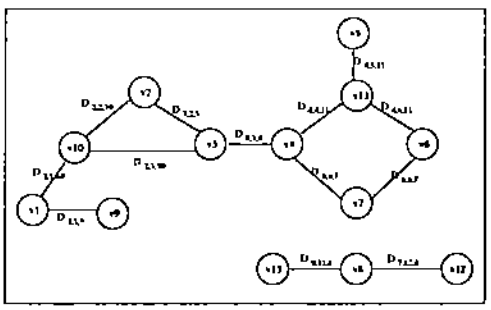
6. Experimental evaluation of PPS

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

Table 1. The policy partition matrix for CSPACE.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
R_1	1	1	0	0	0	0	0	0	0
R_2	0	1	1	0	0	0	0	0	0
R_3	0	1	1	0	0	0	0	1	0
R_4	0	0	0	1	1	0	0	1	0
R_5	0	0	0	1	0	0	0	0	0
R_6	0	0	0	1	0	1	0	0	0
R_7	0	0	0	0	1	1	0	0	0
R_8	0	0	0	0	0	0	1	0	1



$B \subseteq E$	$w(b), b \in B$
$\{D_{1,1,9}, D_{2,1,10}, D_{2,2,10}, D_{2,3,10}, D_{4,4,11}, D_{4,5,11}, D_{4,6,11}, D_{7,12,8}, D_{9,13,8}\}$	∞
$\{D_{2,1,10}, D_{3,2,3}, D_{4,5,11} D_{5,4,7}, D_{6,6,7}, D_{8,3,4}\}$	1

Figure 6. The undirected graph $G = (V, E)$ and the weight function w for CSPACE.

the corresponding PPM. Specifically, density is calculated as $d = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \text{PPM}_{i,j}}{n*m}$. 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 `Proc1`, `Proc2`, and `Proc3` are denoted by $T_{\texttt{Proc1}}$, $T_{\texttt{Proc2}}$, and $T_{\texttt{Proc3}}$, respectively. The CPU usage for the combined execution of `Proc1`, `Proc2`, and `Proc3`, 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) $T_{\texttt{Proc1}}$, $T_{\texttt{Proc2}}$, and $T_{\texttt{Proc3}}$, (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 deg_{avg} of G.

Table 2. The partitions for CSPACE.

Partitions
$\widehat{G}_1 = (\{P_1\}, \{D_1, D_2\})$
$\widehat{G}_2 = (\{P_2, P_3\}, \{D_2, D_3, D_8\})$
$\widehat{G}_3 = (\{P_4, P_6, P_7\}, \{D_4, D_5, D_6, D_8\})$
$\widehat{G}_4 = (\{P_5\}, \{D_4\})$
$\widehat{G}_5 = (\{P_8\}, \{D_7, D_9\})$

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}, $T_{\texttt{Proc1}}$, $T_{\texttt{Proc2}}$, $T_{\texttt{Proc3}}$, $|E|$, $|V|$, and deg_{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 $T_{\texttt{Proc2}}$, $T_{\texttt{Proc3}}$, 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 `Proc3` increases with increase in the size of the ConnectedSpace. Also, the running time of PPS is dominated by `Proc3`.

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 deg_{avg} with increasing d. Figure 9 and Figure 10 show the impact of d on deg_{avg} and $|E|$, respectively. As may be observed from the figures, both d_{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 $T_{\texttt{Proc2}}$, $T_{\texttt{Proc3}}$, and T_{PPS} for $d = 0.4$ are also shown.

						CPU usage (secs)		
	n	m	k	$\|V\|$	$\|E\|$	$T_{\texttt{Proc2}}$	$T_{\texttt{Proc3}}$	T_{PPS}
small	5	20	3	18	33	0.0	0.01	0.01
medium	25	100	6	125	1000	0.01	0.04	0.05
large	125	700	10	825	35000	0.2	2.21	2.44
huge	250	2000	20	2250	200000	1.33	18.15	19.6

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 $pp = \{\widehat{G}_1, \widehat{G}_2, \ldots, \widehat{G}_k\}$, a controller is assigned to each partition. The set of k controllers may then be obtained by executing `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-

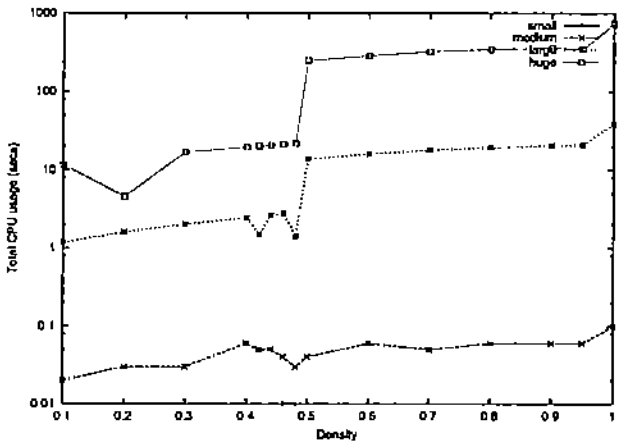


Figure 7. The effect of density on the total CPU usage T_{PPS} for executing PPS.

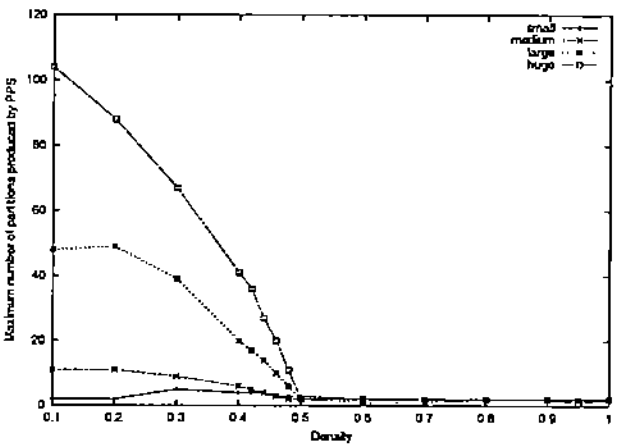


Figure 8. The effect of density on the maximum number of partitions k_{max} produced by PPS.

trollers. 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.

Device	Safety Controller
D_1	SC_1
D_2	SC_1, SC_2
D_3	SC_2
D_4	SC_3, SC_4
D_5	SC_3
D_6	SC_3
D_7	SC_5
D_8	SC_2, SC_3
D_9	SC_5
D_{10}	-

Policy	Safety Controller
P_1	SC_1
P_2	SC_2
P_3	SC_3
P_4	SC_3
P_5	SC_4
P_6	SC_3
P_7	SC_3
P_8	SC_5

Let $D_7 \in$ CSPACE and $D_9 \in$ CSPACE be PM and MRI, respectively. The DDMs for PM and MRI are shown in Figure 1 and [16], respectively. Let $P_8 \in$ polset be P_8=¬($curState_{PM}$=PACING ∧ $curState_{MRI}$=IMAGING) <PM,MRI>(25). Hence, $q_{7,3}$=PACING and $q_{9,2}$=IMAGING. The safety controller SC_5 may be synthesized by using SynthProc on input $\widehat{G}_5$ 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

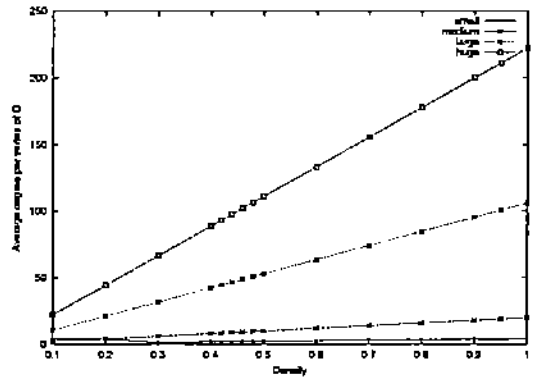


Figure 9. The effect of density on the average degree per vertex d_{avg} of G.

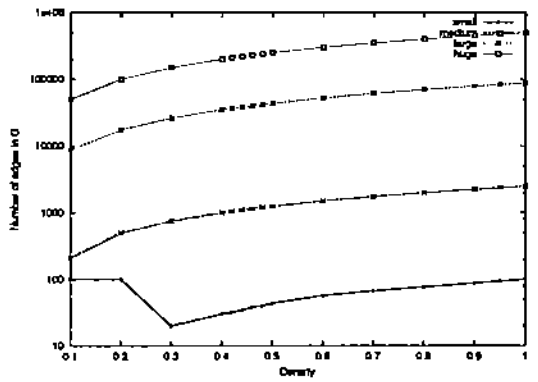


Figure 10. The effect of density on the total number of edges $|E|$ of G.

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

References

[1] http://www.users.on.net/vision/misc/pacemaker-death.html.

[2] NASA. *Air Safety Reporting System*, http://www.ire.org/datalibrary/databases/Asrs/.

[3] Portable electronic devices: Do they really pose a safety hazard on aircraft? *Aviation Subcommittee Hearing*, http://www.house.gov/transportation/aviation/hearing/07-20-00/07-20-00memo.html.

[4] B. A. Brandin and W. M. Wonham. Supervisory control of timed discrete-event systems. In *Proceedings of the 31st IEEE Conference on Decision and Control*, pages 3357–3362, Tucson, Arizona, USA, December 1992.

[5] ECRI. Medical device safety reports. http://www.mdsr.ecri.org/.

[6] N. G. Leveson. *Safeware:System Safety and Computers*. Addison-Wesley Publishing Company, 1995.

[7] N. G. Leveson and J. L. Stolzy. Safety analysis using petri nets. *IEEE Transactions on Software Engineering*, 13(3):386–397, March 1987.

[8] K. Mehlhorn and S. N. Naher. LEDA, a library of efficient data types and algorithms. In *Proceedings of the 14th International Symposium on Mathematical Foundations of Computer Science*, volume 379 of *Lecture Notes in Computer Science*, pages 88–106. Springer-Verlag, 1989.

[9] J. S. Ostroff. Synthesis of controllers for real-time discrete event systems. In *Proceedings of the IEEE 28th Conference on Decision and Control*, pages 138–144, 1989.

[10] J. S. Ostroff. A logic for real-time discrete event processes. *IEEE Control Systems Magazine*, 10(4):95–102, June 1990.

[11] C.-W. Ou and S. Ranka. Parallel incremental graph partitioning. *IEEE Transactions on Parallel and Distributed Systems*, 8(8):884–896, August 1997.

[12] P. J. Ramadge and W. M. Wonham. On a supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, May 1987.

[13] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, January 1987.

[14] H. Saran and V. V. Vazirani. Finding k-cuts within twice the optimal. *SIAM Journal on Computing*, 24(1):101–108, February 1995.

[15] B. Sridharan, A. P. Mathur, and K. Y. Cai. Synthesis of a safety controller for ConnectedSpaces using supervisory control. Technical Report SERC-TR-178-P, Software Engineering Research Center, Purdue University, West Lafayette, IN, USA, September 2002.

[16] B. Sridharan, A. P. Mathur, and K. Y. Cai. Synthesizing distributed safety controllers for ConnectedSpaces. Technical Report 02-024, Department of Computer Sciences, Purdue University, West Lafayette, IN, USA, September 2002.

[17] B. Sridharan, A. P. Mathur, and S. G. Ungar. Digital Device Manuals for the management of ConnectedSpaces. *IEEE Communications Magazine*, 40(8):78–85, August 2002.

[18] J. L. Tri, D. L. Hayes, T. Smith, and R. P. Severson. Cellular phone interference with external cardiopulmonary monitoring devices. *Mayo Clinic Proceedings*, 76(1):11–15, January 2001.