Circulatory Load-sharing Redundancy for Resource Single Point of Failure in Automated Manufacturing Systems Using Petri Nets

Ebrahim A. Alzalab[1,2] , Adeeb A. Ahmed [3], Haithm M. Al-Gunid[4], Sadiq Ghalib[5]
[1] *Department of Information Technology, Faculty of Computer, Al-Razi University, Sana'a, Yemen*
[2] *Department of Mechatronics Engineering, AL-Rasheed Smart University, Sana'a, Yemen*
[3] *School of Electro-Mechanical Engineering, Xidian University, Xi'an, 710071, Shaanxi, China*
[4] *Department of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei 230026, China, and with the ICT Lab, Griffith University, Gold Coast, QLD 4222, Australia*
[5] *Department of Information Technology at Arab Academics University, Sana'a, Yemen*

***Abstract*—Automated manufacturing systems (AMS) are exposed to risks from failure of resources, which result in the whole system stopping working and losing considerable production. This paper presents a new circular load sharing redundancy recovery (C-LSRR) technique to increase resilience from failure of one resource, without using any extra backup resources and hierarchy. According to the proposed technique, the unreliable resources will be positioned in a circle such that each resource serves as a backup for the nearest neighbor. If one of the resources fails, all associated tasks will be transferred to the next resource in the circle and, after the restoration, they will automatically return to the faulty resource. The suggested technique is formalized with the use of petri net, extended with a circular switching subnet free from inhibitors. Its effectiveness is demonstrated through experiments carried out on a typical S^4R model. Experiments have shown that the system without redundancy stops working, while the suggested circular redundancy technique results in high recovery efficiency without loss of operational efficiency. The problem of extension to more than one failure is not considered.**

Automated manufacturing system, circular redundancy, load balancing, Petri net model, fault tolerance, individual component failure, deadlock avoidance.

I. Introduction

The automated manufacturing systems (AMSs) are fundamental elements of today's manufacturing processes, offering high speed and versatility. Still, the AMSs are highly susceptible to failures, which may occur at various stages of the process, e.g., due to malfunctioning robots, machines, or faulty sensors [1], [2]. Failures of this kind can cause disruption of the entire process, leading to considerable financial damage, production downtime, and potential danger [3], [4]. Thus, there is a need for fault-tolerant techniques and repair procedures [5], [6].

Control problems of DESs have been extensively researched by automata and Petri nets [7], [8]. Among those tools, Petri nets can effectively model parallelism, shared resource access, synchronization, and deadlock analysis of AMSs [9]–[11]. There exist several deadlock control strategies which have been designed for AMSs whose resources are reliable, like siphon-based strategies [12], reachability graph-based strategies [13], and theory of regions [14]. Unreliable resources complicate matters since even deadlock prevention does not necessarily assure that the process is maintained after failure of a resource [15], [16].

In order to solve this problem, control strategies for supervisory control systems that can accommodate faulty resources have been put forth. Supervisory control approaches employing central buffers for managing several faulty resources have been presented by Chew et al. [17]. Liu et al. [18] have employed recovery subnets within control places following a divide and conquer approach. A modification of the Banker's algorithm has been used by Yue et al. [19] for deadlock avoidance in faulty resource environments. Wang et al. [20] have proposed three supervisors that employ redundancy for deadlock-free execution in the event of faulty resources. Recently, Alzalab et al. [21] have introduced the concept of load-sharing redundant resource (LSRR), which involves a backup resource sharing the work with the unreliable target resource and taking full charge after target failure. This methodology is only applicable for one unreliable resource at a time and uses a dedicated backup resource, leading to higher hardware expenses. Fault recovery and repair mechanisms in discrete event systems have been explored using Petri nets by us [21], [22]. However, those approaches have depended on dedicated backup resources or hierarchical backup approaches, adding control and priority inversion issues.

In real life, an AMS may involve more than one unreliable resource. In such a case, extending the LSRR methodology to handle more than one unreliable resources will involve provision of one additional backup for each individual unreliable resource. This will entail considerable costs on both hardware and logic design aspects. Hierarchical redundancy systems [23], [24], on the other hand, are complex systems with layered switching mechanisms which are hard to analyze. To solve this problem, the paper proposes a new methodology known as *circular load sharing redundancy* (C-LSRR). C-LSRR involves handling of more than one unreliable resource

Corresponding author: alzalabibrahim01@gmail.com

using a single failure assumption without any layer of hierarchy or dedicated resources for backing up any resource. In C-LSRR, the unreliable resources are arranged in circular manner, where each one of the unreliable resources becomes the backup for the neighboring unreliable resource. On failure of an unreliable resource, the subsequent neighbor performs all the functions of the failed resource. On restoration of the failed resource, the functions are transferred back to the failed resource automatically.

The suggested technique is represented through Petri nets with a circular switch sub-net without inhibitor arcs, thus saving computation resources. The supervision controller is designed based on the TGAL method [25] applied to the C-LSRR net. In particular, we use the TGAL algorithm for the S^4R net containing 50 parts in order to find monitors for normal and degraded conditions, and prove the boundedness and liveness preservation property in case of one failure (see Theorems 1–3 in Section IV-F). The efficiency of the algorithm has been tested through numerous simulations performed for the S^4R net containing 50 parts with one failure for robot R_1. Simulation results demonstrate that the system without redundancy stops working entirely (recovery 0%), while our technique guarantees 90.5% of recovery rate. Additionally, it should be noted that our redundancy solution does not affect the performance of the system during its normal functioning with the throughput decreasing from 9.3 to 7.7 parts. Key contributions of this work include the following: (i) a circular redundancy mechanism using multiple unreliable resources (single failure case scenario), which does not need any additional backup resources or any hierarchy, (ii) a Petri net model with a circular switch subnet to cope with failures and repairs without using any inhibitor arcs, (iii) formal definitions and sketch proofs of liveness, boundedness, and deadlock avoidance with a single resource failure, (iv) implementation of TGAL on C-LSRR net, and (v) verification by simulation experiments based on an S^4R net with 50 initial parts resulting in 90.5% recovery.

Organization of the rest of the paper is as follows: In Section 2 we introduce motivations and preliminaries on Petri nets. In Section 3, the problem formulation is given. In Section 4, the circular redundancy modeling approach based on load-sharing is described. In Section 5, design of the supervisory controller using TGAL is explained. Experiment results are presented in Section 6.

II. Motivation and Preliminaries

As an example of the limitations of conventional methods of load sharing redundancy and to justify the concept of circular redundancy, we consider the robotic manufacturing cell illustrated in Fig. 1. This cell consists of various unreliable robots that are used to load and unload the system. In the single LSRR scheme presented in [21], providing a backup robot to each unreliable resource will be too costly from the perspective of hardware installation. In case of hierarchical backup schemes [23], [24], switching problems arise, along with priority inversion and deadlock. However, according to our proposed circular scheme, resources are organized in a circle, such that each resource is responsible for the resource adjacent to it. When the system operates normally, resources do their own jobs, but in case of the failure of any resource, the neighboring resource in the circle takes over its job. Our proposed approach reduces the complexity involved in the hierarchy, and there is no need for additional backup hardware.

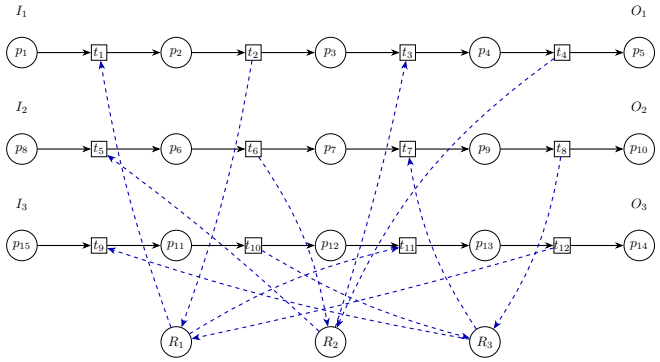


Fig. 1: A robotic manufacturing cell having more than one unreliable resources.

A Petri net is a basic mathematical tool to represent and analyze Discrete Event Systems (DESs) [26], [27]. A Petri net is denoted as $N = (P, T, F, W)$, where P denotes the finite set of places, T the finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ the set of directed arcs, and $W : F \to \mathbb{N}^+$ the weight function, where 1 is the default value of weight. The finite sets P and T do not have any common element. The mapping $M : P \to \mathbb{N}$ assigns a non-negative integer number of tokens to places; the Petri net system (N, M_0) has an initial marking M_0. An enabled transition t at marking M satisfies the condition that for every input place $p \in {}^\bullet t$, $M(p) \geq W(p, t)$, where ${}^\bullet t = \{p \mid (p, t) \in F\}$. Marking M' resulting from firing the enabled transition t is expressed by $M'(p) = M(p) - W(p, t) + W(t, p)$ for all $p \in P$.

Reachability, liveness, and boundedness are three major behavioral properties [28], [29]. The set of reachable markings $R(N, M_0)$ consists of all markings which can be obtained from M_0 through the firing of some set of transitions. The system is said to be live if, for each transition t and each marking M belonging to $R(N, M_0)$, there is a sequence of transitions fired after M which will make t enabled. Liveness implies that the system is deadlock-free and will eventually complete all processes. The system is said to be k-bounded if, for every place p and every marking M, we have $M(p) \leq k$.

The known class of Petri nets referred to as S^3PR (System of Simple Sequential Processes with Resources) [30], [31] is used in the modeling of automated manufacturing systems. In an S^3PR net, there are process places P_A (activities), resource places P_R (machines, robots, buffers), and idle places P^0 (entry/exit of parts). The unreliability of S^3PR leads to what is called the US^3PR. As such, an US^3PR is an extension of S^3PR where one or more resources have been made failure-prone, together with the inclusion of additional places and transitions to represent failures and repairs [21], [22]. For any resource $r \in P_R$, the holder set is $H(r) = {}^\bullet r \cap P_A$, the set of activity places that require r.

According to [22], the requirement for load-sharing redundant resource r_s for a faulty resource r_u is $|H(r_u)| > 1$, it has an initial common subset of $H(r_u)$, and on failure of r_u, r_s is able to carry out the functions of r_u by switching of tokens through a switch subnet. The drawback with this solution is that it requires redundant resources.

For addressing the above problem, the concept of Circular Load Sharing Redundancy Resource (C-LSRR) is proposed for the case where only single resource failures occur. Consider that $R = \{r_1, r_2, \ldots, r_n\}$ consists of $n \geq 2$ unreliable resources and they are arranged in a circular fashion as $(r_1, r_2, \ldots, r_n, r_1)$. Define that r_{i+1} (where $r_{n+1} = r_1$) serves as the circular backup for r_i. Normally, each resource r_i works on its own workload $H(r_i)$; furthermore, r_{i+1} also handles a subset of $H(r_i)$ to ensure smooth transition in case r_i fails. On the occurrence of failure of r_i, all the tokens of $H(r_i)$ are directed to r_{i+1}. There is no need of additional backup resources or hierarchy; only single failures are considered.

To verify, we apply the S^4R (System of Sequential Processes with Shared Resources) model [32] which is an extension of the S^3PR with increased complexity of resource interaction. This model consists of 23 places and 18 transitions, depicting several sequences of productions, shared machines, and four faulty robots placed in the sequence $R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_4 \rightarrow R_1$.

In order to create the liveness-enforcing supervisors, the TGAL approach [25] is used. TGAL repeatedly introduces a global sink/source place for restricting the number of workpieces, constructs the reachability graph, detects bad markings (deadlock markings), derives place invariants (monitors) to prevent those bad markings, and introduces the resulting monitors to the net. This procedure is repeated until the system becomes live. TGAL guarantees maximal permissiveness and does not incur computational complexity related to the integer linear programming problem. In the current paper, TGAL is applied to the C-LSRR net, and monitor tables are given in Section 5.

III. Problem Formulation

An example of a considered system is a manufacturing system that can be described using a Petri net from the class of S^3PR (System of Simple Sequential Processes with Resources) [30], [31] extended to account for unreliability of the resources. A marked S^3PR net is defined as a Petri net $(N, M_0) = (P_A \cup P^0 \cup P_R, T, F, M_0)$, where P_A is the set of activity places (processes), P^0 is the set of idle places (entry and exit of parts), P_R is the set of resource places (machines, robots, buffers), T is the set of transitions (start and completion of processes), F is the set of arcs, and M_0 is the initial marking. In an *unreliable* S^3PR (US^3PR), besides these components, an extension is done to describe failure and repair of a subset of unreliable resources $R_u \subseteq P_R$ [21], [22]. The *holder set* of a resource $r \in P_R$ is denoted as $H(r) = \{p \in P_A \mid (p, r) \in F \text{ or } (r, p) \in F\} = {}^\bullet r \cap P_A$. representing all operations that consume or produce tokens from r.

An unreliable resource $r_u \in R_u$ may be either working or failed. Transition from working state to failed state (or vice versa) can be represented by two types of events: t_{fail,r_u} and t_{rep,r_u}. Occurrence of both these events is nondeterministic.

Considering a US^3PR net N having a set of unreliable resources $R_u = \{r_1, r_2, \ldots, r_n\}, n \geq 2$, we propose a circular load sharing mechanism which doesn't require any additional backup resources and hierarchies. Whenever a resource r_i becomes failed, all the tasks which were assigned to it should automatically be reassigned to a backup circular resource r_{i+1}. On restoration of the failed resource, all the tasks should be restored too. Moreover, controlled system should remain live and deadlock-free both in normal mode and in any failure situation.

The definition of circular backup order states that it is a cycle $(r_1, r_2, \ldots, r_n, r_1)$ in which each element r_i backs up r_{i+1}, such that $r_{n+1} = r_1$. In light of this, our definition for circular load-sharing redundancy (C-LSRR) will be that it is the conversion of the initial US^3PR network (N_u, M_{u0}) to (N_c, M_{c0}) through the following procedure. Given any $r_i \in R_u$, one introduces a new failure place $fail_i$. One introduces a duplicated transition $t^{(i)}$ for each initial transition t that has used r_i and uses r_{i+1} rather than r_i. This new transition can only fire if $fail_i$ is occupied by a token, which implies that all the original transitions are blocked. Repair causes the token to go back to r_i, which then blocks the duplicated transitions and activates the original ones.

All throughout this paper, we consider the following assumptions: there is a maximum of one failed resource per point in time (multiple failures are deferred to later work), all failed resources are repairable in a finite period of time, each activity place has only one associated resource, and the initial US^3PR net is bounded. With the above assumptions, we show that the resulting redundant net (N_c, M_{c0}) is live and bounded, the system retains a considerable amount of its throughput capacity even in the presence of faults relative to the non-redundant approach (which ceases to function entirely), and the complexity overhead is negligible.

To better understand the concept of C-LSRR, let us take an example with two non-reliable resources, R_1 and R_2. The circular order for this case would be $R_1 \rightarrow R_2 \rightarrow R_1$, meaning R_2 serves as a backup for R_1 and vice versa, as illustrated in Figure 2. If R_1 goes down, then all the jobs assigned to R_1 will move to R_2. In a similar way, when R_2 gets repaired, the jobs go back to R_1. This example involving two resources provides the basis for extension to n resources.

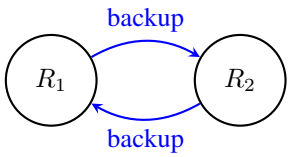


Fig. 2: Backup relationship for circular arrangement of two resources: mutual backup is performed among the two resources.

AMS Modeling with Circular Load-Sharing Redundancy:

IV. AMS Modeling with Circular Load-Sharing Redundancy

In this section, we discuss an approach for modeling AMSs

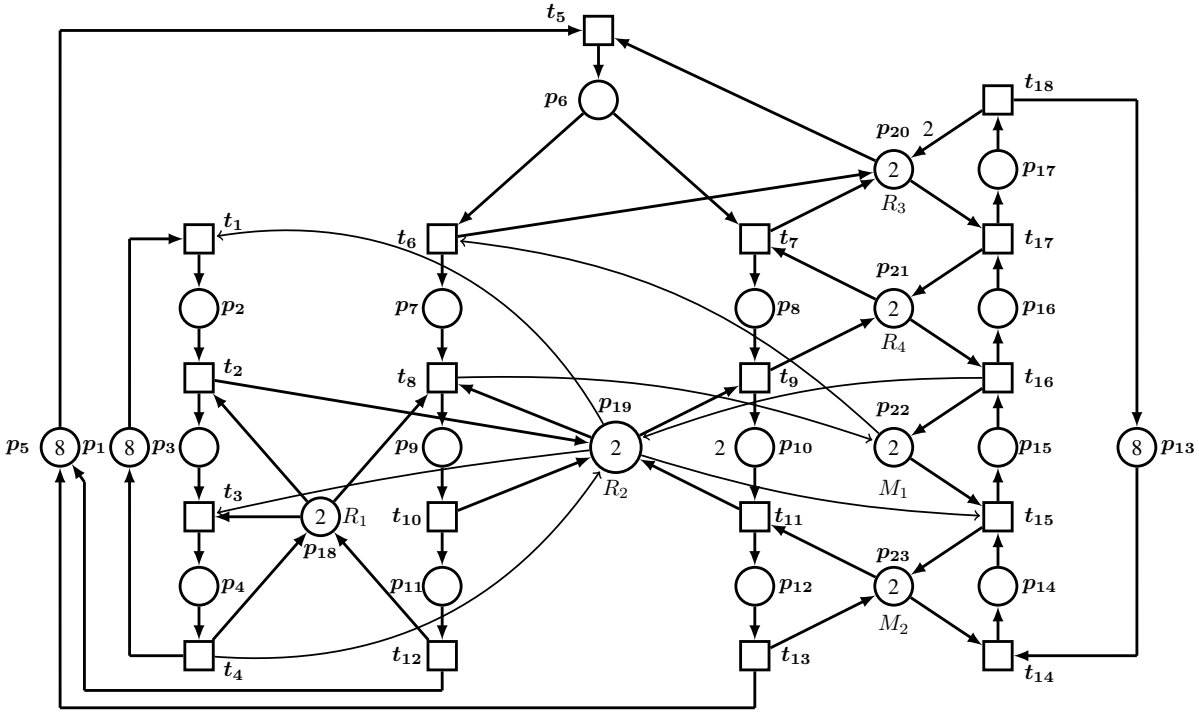


Fig. 3: Base S^4R Petri net model of AMS.

with circular load-sharing redundancy (C-LSRR) using the Petri net model. The proposed modeling technique is an extension of US^3PR model [21], [22], wherein a special type of subnet, namely, circular switch subnet, is introduced to handle failure and recovery of the unreliable resources.

A. Base Model and Circular Backup Strategy

The base model chosen is the popular S^4R benchmark model [32], shown in Figure 3: It is composed of 23 places and 18 transitions and represents three sequences in manufacturing operations that use four robots (R_1, R_2, R_3, R_4) and two machines (M_1, M_2). Initially, eight tokens are placed in each idle place (p_1, p_5, p_{13}), while each resource place (p_{18} to p_{23}) gets two tokens. The Pre and Post incidence matrices are listed in [32].

The circular backup arrangement for the S^4R system illustrated in Figure 3 is described below:

$$R_1 \longrightarrow R_2 \longrightarrow R_3 \longrightarrow R_4 \longrightarrow R_1.$$

In this manner, each robot acts as a backup for the previous one: Robot R_2 is the backup for Robot R_1, Robot R_3 for Robot R_2, Robot R_4 for Robot R_3, and lastly, Robot R_1 is the backup for Robot R_4. There is no precedence order involved in this circular redundancy scheme. Thus, all the robots act as primary resources and backups at the same time, making the C-LSRR framework unique.

B. Failure/Repair Model and Redundant Transitions

For any one of the unreliable robot $r_i \in R_1, R_2, R_3, R_4$, we define a local failure/repair subnet. The machines M_1 and M_2 are considered to be reliable and thus do not require failure/recovery subnets. Comprising a failure place $fail_{r_i}$ and two transitions: a failure transition t_{fail,r_i} and a repair transition t_{rep,r_i}. The initial marking consists of one token at r_i (working resource) and zero tokens at $fail_{r_i}$. Upon firing of the former transition, the token is moved to $fail_{r_i}$, thus modeling failure. Similarly, firing of the latter restores the token back, signifying repair. This approach does not employ inhibitor arcs. Mathematically:

$$\begin{aligned} {}^\bullet t_{fail,r_i} &= \{r_i\}, & t_{fail,r_i}^\bullet &= \{fail_{r_i}\}, \\ {}^\bullet t_{rep,r_i} &= \{fail_{r_i}\}, & t_{rep,r_i}^\bullet &= \{r_i\}. \end{aligned}$$

To make the task redirection work in case of failure, the system is designed by duplicating the transition t for each transition t that uses an unreliable resource. The new transition $t^{(i)}$ utilizes another, alternative resource r_{i+1} (where $r_5 = r_1$) instead of the previous one, but can be activated only if $fail_{r_i}$ occurs. In formal notation:

$${}^\bullet t^{(i)} = ({}^\bullet t \backslash \{r_i\}) \cup \{r_{i+1}, fail_{r_i}\}, \qquad t^{(i)\bullet} = (t^\bullet \backslash \{r_i\}) \cup \{r_{i+1}\}.$$

All the original transitions remain as is; but once there is a failure, the token exits from r_i and goes to $fail_{r_i}$, disabling the original transition and activating the duplicate transition automatically.

C. Complete Petri Net Model and Properties

The complete C-LSRR net model (N_c, M_{c0}) is created using composition of the base net US^3PR along with all the local failure/recovery subnets and duplicate transitions:

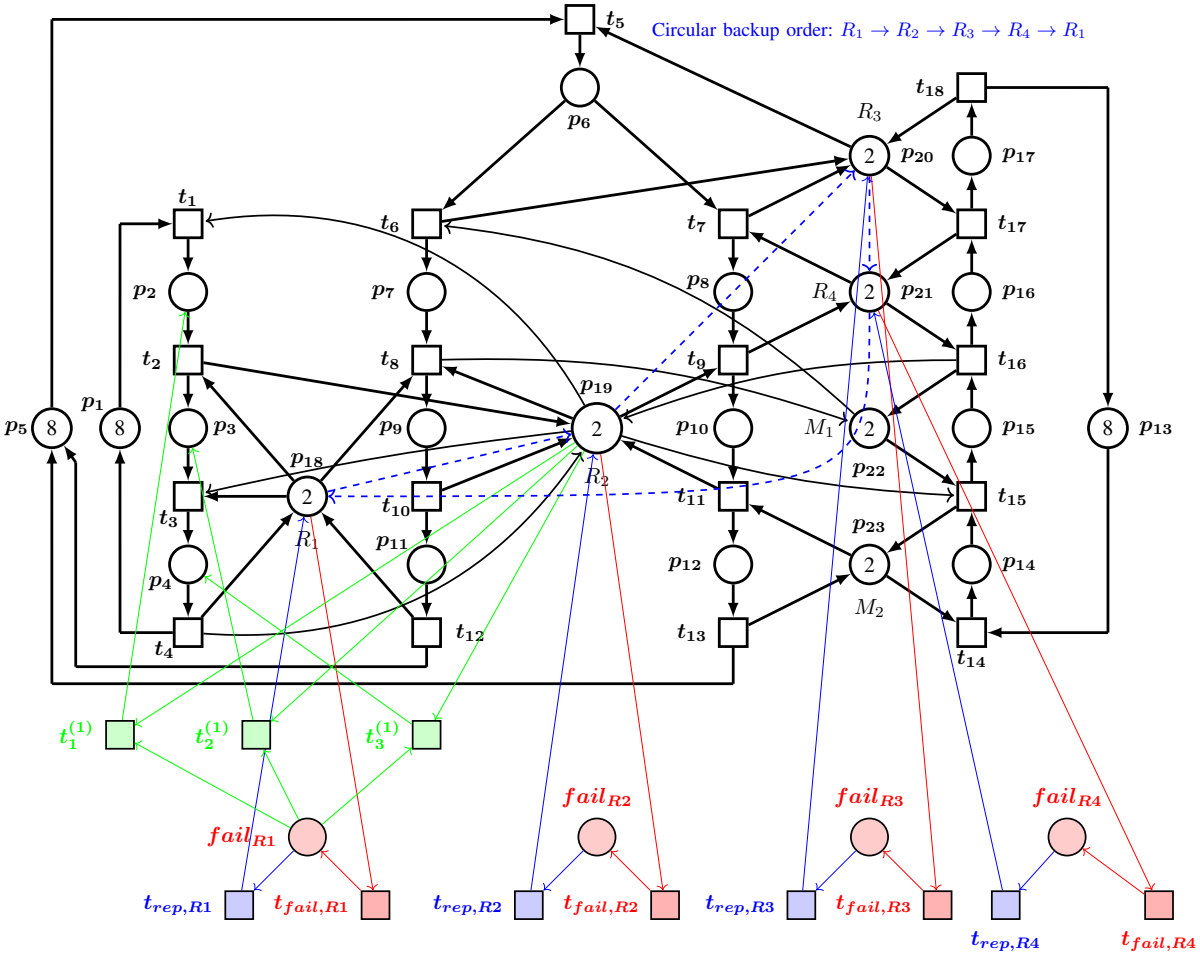


Fig. 4: Complete S^4R model with circular load-sharing redundancy (C-LSRR).

$$N_c = N_u \otimes \left(\bigotimes_{i=1}^{4} N_{f_i} \right) \otimes \left(\bigotimes_{i=1}^{4} \bigotimes_{t \in T_i} \{t^{(i)}\} \right),$$

where $\otimes$ represents the composition (union of places and transitions having common elements), N_{f_i} represents the local failure/recovery subnet for r_i, and T_i represents the set of transitions that consume r_i. The initial marking M_{c0} is identical to M_{u0} with respect to the original places, while all $fail_{r_i}$ sets are empty at the beginning. The resulting net is depicted in Fig. 4. The original net is extended with failure places (in red), failure/repair transitions (in red/blue), and duplicated transitions (in green, only partly illustrated for simplicity). The blue dashed arrow indicates the circular backup order $R_1 \to R_2 \to R_3 \to R_4 \to R_1$.

Properties of the suggested construction of C-LSRR nets can be described through several theorems.

Theorem 1 (Boundedness). If the initial US^3PR net (N_u, M_{u0}) is bounded, then C-LSRR net (N_c, M_{c0}) is bounded too. *Proof.* The addition to the net consists of a limited number of new places (4 failure places) and transitions (4 failure transitions, 4 repair transitions, and $\sum_i |T_i|$ duplicate transitions). Failure places have not more than one token, and duplicate transitions just transfer the tokens between places of the bounded net.

Theorem 2 (Liveness in the normal case). The C-LSRR net will be live under normal operations if the original net is live, since the duplicated transitions have been disabled and behave similarly to the original net.

Theorem 3 (Liveness in the event of single failure). If there is single failure of a particular resource r_i, such that the backup resource r_{i+1} can accommodate additional workload, then the C-LSRR net will still be live. *Informal proof.* Since the duplicated transitions $t^{(i)}$ divert tasks that need to be processed by r_i to r_{i+1}, the resulting net will behave equivalently to a US^3PR net where r_i and r_{i+1} are replaced with a single resource having a total capacity. This net will be live as per the condition of original net being live and the preservation of siphon control properties [31]. In the case of S^4R, it is ensured that the capacity condition holds because total workload will never exceed the sum of capacities of both resources.

A few of the important choices made in the design include the following. First, since there are duplicate transitions and failures used, no inhibitors will be required for analysis. Second, the cyclic order in which backups are done results in

a symmetric structure without any hierarchy involved. Finally, the complexity added to the model due to our choice of design is only $O(n)$.

V. Design of Supervisory Controller for C-LSRR Nets

This section presents a supervisory control technique to be used in the C-LSRR net (N_c, M_{c0}) presented in Section 4. This controller should guarantee liveness in terms of absence of deadlock, not only during normal behavior when all resources function but also during the degraded behavior when one resource fails. For this purpose, the TGAL technique [25] is employed.

A. TGAL Algorithm and its Modification for C-LSRR Nets

The Think Globally, Act Locally (TGAL) algorithm [25] is a recursive procedure aimed at synthesizing liveness enforcing supervisors (LES) for asynchronous machine systems (AMSs) described using Petri nets. In contrast to siphon-based algorithms [12], [31], which use structural properties, TGAL uses reachability to construct a set of monitors such that only those deadlock states will be prevented that make the system deadlocked while preserving maximum permissiveness. The main idea of TGAL is based on iterative adding a sink/source place, finding reachability graph, recognizing bad markings (deadlock states), constructing place invariants (monitors) that forbid these bad markings, and removing the redundant monitors.

The use of TGAL on C-LSRR nets faces two major difficulties: mode dependency (one needs to analyze each failure mode separately) and state space explosion, caused by the larger number of places and transitions in the resulting nets. As a solution to these issues, we suggest the following three-step synthesis algorithm.

The first step involves constructing a supervisor for the nominal (non-faulty) mode (where all resources are alive). We refer to N_{nom} as the subnet of N_c where all resources work (there are no tokens in $fail_i$). The only difference between N_{nom} and the initial US^3PR net is that its duplicate transitions are disabled (they need tokens from $fail_i$), hence the supervisor of the S^4R system consists of four monitors. Here, we do not provide the detailed monitor tables since the problem is discussed in the previous sections, but they can be provided upon request.

At Stage 2, a supervisor is designed for each degraded mode due to a single failure. A degraded mode net N^i_{deg} will be constructed for each resource $r_i \in \{R_1, R_2, R_3, R_4\}$, whereby resource r_i has experienced a failure. In such case, the token at r_i is moved to $fail_{r_i}$, thereby permitting the duplicate transitions $t^{(i)}$, which direct tasks from the failing resource r_i to the back-up resource r_{i+1}. TGAL may be applied to each N^i_{deg} through simplification by collapsing the failed resource and its back-up into a single resource, $\hat{r}$. This procedure is justified due to the fact that the duplicated transitions allow pooling of the two resources' capacities. The simplified degraded mode net is an S^3PR-like net, thus making it easier to apply TGAL. At S^4R, each failure results in three monitors. All monitors resulting from degraded modes are available upon request from the authors.

Stage 3 involves the consolidation of all the supervisors into a single controller by removing redundant monitors. In Stage 3, first of all, we obtain the monitors for the nominal controller (C_{nom}) and those for each degraded mode (C^i_{deg}). Then, we consolidate them together into a single supervisor. A monitor c that has an invariant of I_c and a bound of b_c is said to be redundant if there is another monitor c', where $I_c \subseteq I_{c'}$ and $b_c \geq b_{c'}$; all such monitors will be eliminated [25]. Under the S^4R framework, this step brings the number of total monitors down from around 16 monitors to just 8 monitors.

B. Combining and Complexity

The final controlled net N^c_c will be the result of combining the monitored nets with the C-LSRR net N_c. The transitions $t_{fail,i}$ and $t_{rep,i}$ will not be affected by combining because their behavior does not contradict monitor invariants. In any monitor created based on either nominal or faulty TGAL, the firing of failure and repair transitions does not break any monitor invariant. This is because firing of failure transitions transfers one token from place r_i to place $fail_{r_i}$, leaving other invariant quantities unaffected, while firing of repair transitions moves token from place $fail_{r_i}$ to place r_i.

Computational cost of synthesizing the monitors mainly consists of calculating reachability; it is equal to $O(|R|*2^{|P_A|})$ for the nominal scenario and $O(4*|R|*2^{|P_A|})$ for the other four possible scenarios. The redundancy elimination step involves comparing all $O(k^2)$ pairs of monitors where k is the total number of monitors synthesized. This is negligible compared to reachability. Using the S^4R model considered in this paper (23 places, 18 transitions), our approach took less than 5 seconds to synthesize the monitors.

Next, we present experimental results for the S^4R model.

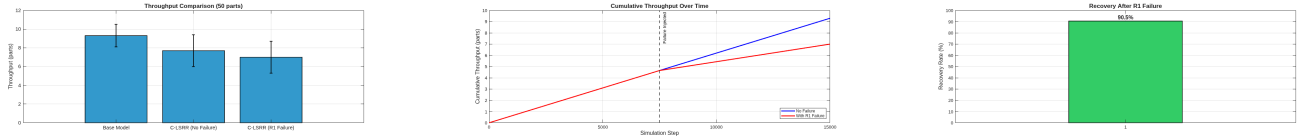
VI. Experimental Results

We now test our circular load sharing redundancy (C-LSRR) scheme via comprehensive simulation experiments using the S^4R architecture discussed in Section 4. This study focuses on testing our redundancy scheme when there is a failure in one of the resources.

A. Simulation Environment and Results

The simulations were carried out on a personal computer (Intel Core i7, 16GB RAM) with the help of MATLAB R2022b. In order to implement Petri nets in matrix form, incidence matrices were used. Table I presents the simulation parameters. Simulation of three cases was performed: (i) S^4R model without redundancy, (ii) C-LSRR net with all resources functioning properly, and (iii) C-LSRR net with fault in robot R_1 occurring at step 7,500.

Simulation results are shown in Table II and Figure 5. The baseline model (without any redundancy) generates an average number of 9.3 units without failure (the standard deviation of 1.2). On the other hand, the number of generated units will be 0 when R_1 experiences failure. For the baseline model,



(a) Throughput comparison. (b) Cumulative throughput over time. (c) Recovery rate (90.5%).

Fig. 5: Simulation output for the S^4R system with 50 components: (a) throughputs comparison, (b) cumulative throughputs with indication of the failure after 7,500 steps, (c) recovery rate 90.5%.

TABLE I: Simulation parameters

Parameter	Value
Initial number of parts	50
Simulation steps per run	15,000
Failure injection step	7,500 (midpoint)
Number of simulations per scenario	30
Number of unreliable resources	4 (R_1, R_2, R_3, R_4)
Circular backup order	$R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_4 \rightarrow R_1$

the proposed C-LSRR model generates an average throughput of 7.7 units with a standard deviation of 1.7, slightly lower than that of the baseline because of the overhead caused by the introduction of the redundancy. In case a failure of R_1 is introduced, the C-LSRR system is able to continue functioning with an average throughput of 7.0 units and a standard deviation of 1.7, giving a recovery rate of 90.5%.

B. Overhead, Comparison, and Discussion

Overhead is presented in Table III. It is small (only 17% more places, 111% more transitions, and 40% simulation time) and reasonable for practical usage.

TABLE III: Computational overhead of C-LSRR

Metric	Base Model	C-LSRR Model
Number of places	23	27 (+17%)
Number of transitions	18	38 (+111%)
Number of monitors	0	8
Simulation time (seconds)	3.2	4.5 (+40%)

A comparison between the proposed C-LSRR algorithm and non-redundant as well as the dedicated LSRR algorithm discussed in the literature is provided in Table IV. Although actual values for throughput may vary depending on the size of the model used, the value for recovery rate serves as the important performance indicator in this scenario. The proposed solution provides 90.5% recovery without using dedicated backup nodes; it outperforms non-redundant solutions (0%) and is also competitive against the dedicated LSRR solution (77.8% in [22]).

Based on the experimental outcomes, it can be concluded that C-LSRR provides fault tolerance capabilities without special hardware and performance degradation during the normal operation mode. The decrease in throughput from 9.3 to 7.7 parts (17%) is an acceptable loss considering the recovery level of 90.5%. Among the limitations of the approach are the assumption of only one possible failure at any given moment of time, random transition selection (which means deterministic transitions would likely increase the throughput), and limited size of the model.

VII. Conclusions

The approach discussed here involves C-LSRR that helps improve the fault tolerance of automatic manufacturing systems without using any extra backup components or hierarchies. It involves placing unreliable components in a circular fashion, whereby one component serves as the backup for another component adjacent to it. When one component fails, jobs are re-assigned to the next component in the circular fashion and when the component is repaired, jobs are automatically reassigned back. The method was modeled using Petri Nets augmented with a circular switch subnet with no inhibitor arcs, followed by synthesis of supervisory controller using the TGAL method. The validity of the proposed technique was shown by simulating the S^4R system with 50 initial number of parts when one of the robots R_1 fails. From the above data, we can observe that in the absence of redundancy, the system fails to work at all (0% of recovery). On the other hand, in the case of the use of C-LSRR system, the recovery rate is 90.5%. What is more, the system shows no decrease in throughput during normal operation, which only decreases from 9.3 to 7.7 parts. Thus, the small decrease of throughput by almost 17% during normal operation can be justified by the achievement of the 90.5% recovery rate without additional equipment. Future works will include developing this approach for dealing with multiple failures simultaneously, introducing deterministic approaches to increase throughput efficiency, testing with more complex models, and introducing prediction approaches based on machine learning methods.

Disclaimers

Funding: No financial support was obtained for the preparation of this article. The authors wish to state that no funds, grants, or support of any kind have been received for the writing of this paper.

Trial registration: This paper is not a clinical trial and has no involvement of any humans or animals. Hence, trial registration is not relevant.

Ethics Approval/Informed Consent: Not required as this study involves no human or animal subjects and there is no requirement for ethical approval. The consent to publish is not applicable since the manuscript does not contain any personal data of an individual person.

References

[1] N. Viswanadham and T. Johnson, "Fault detection and diagnosis of automated manufacturing systems," *IFAC Proceedings Volumes*, vol. 21, no. 1, pp. 95–102, 1988.

TABLE II: Simulation results for the S^4R model with 50 parts

Scenario	Throughput (parts)	Std Dev	Recovery Rate
Base model (no redundancy)	9.3	1.2	–
Base model (with R_1 failure)	0.0	0.0	0%
C-LSRR (no failure)	7.7	1.7	–
C-LSRR (with R_1 failure)	7.0	1.7	**90.5%**

TABLE IV: Comparison with existing approaches

Approach	Throughput (no failure)	Throughput (with failure)	Recovery Rate
No redundancy (base model)	9.3	0.0	0%
Wu et al. [33]	122	0	0%
Zhang et al. [34]	149	0	0%
Alzalab et al. (LSRR) [22]	198	154	77.8%
Proposed C-LSRR	**7.7**	**7.0**	**90.5%**

[2] G. Zhu, Z. Li, and N. Wu, "Model-based fault identification of discrete event systems using partially observed petri nets," *Automatica*, vol. 96, pp. 201–212, 2018.

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[4] E. Alzalab, A. El-Sherbeeny, M. El-Meligy, and H. Rauf, "Trust-based petri net model for fault detection and treatment in automated manufacturing systems," *IEEE Access*, vol. 9, pp. 157 997–158 009, 2021.

[5] J. Luo, Z. Liu, S. Wang, and K. Xing, "Robust deadlock avoidance policy for automated manufacturing system with multiple unreliable resources," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 812–821, 2020.

[6] D. Xiang, G. Liu, C. Yan, and C. Jiang, "Detecting data-flow errors based on petri nets with data operations," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 251–260, 2017.

[7] F. Cabral, M. Moreira, O. Diene, and J. Basilio, "A petri net diagnoser for discrete event systems modeled by finite state automata," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 59–71, 2014.

[8] A. Bonafin, F. Cabral, and M. Moreira, "An effective approach for fault diagnosis of discrete-event systems modeled as safe labeled petri nets," *Control Engineering Practice*, vol. 123, p. 105168, 2022.

[9] Z. Li, S. Zhu, and M. Zhou, "A divide-and-conquer strategy to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 39, no. 2, pp. 156–169, 2009.

[10] Y. Chen, Z. Li, and A. Al-Ahmari, "Nonpure petri net supervisors for optimal deadlock control of flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 2, pp. 252–265, 2012.

[11] A. Wang, Z. Li, J. Jia, and M. Zhou, "An effective algorithm to find elementary siphons in a class of petri nets," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 4, pp. 912–923, 2009.

[12] M. Uzam and M. Zhou, "An iterative synthesis approach to petri net-based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 3, pp. 362–371, 2007.

[13] G. Liu, P. Li, Z. Li, and N. Wu, "Robust deadlock control for automated manufacturing systems with unreliable resources based on petri net reachability graphs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 7, pp. 1371–1385, 2018.

[14] Y. Chen and Z. Li, "Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems," *Automatica*, vol. 47, no. 5, pp. 1028–1034, 2011.

[15] S. Chew and M. Lawley, "Robust supervisory control for production systems with multiple resource failures," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 3, pp. 309–323, 2006.

[16] H. Yue, K. Xing, H. Hu, W. Wu, and H. Su, "Supervisory control of deadlock-prone production systems with routing flexibility and unreliable resources," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 10, pp. 3528–3540, 2019.

[17] S. Chew, S. Wang, and M. Lawley, "Robust supervisory control for product routings with multiple unreliable resources," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 1, pp. 195–200, 2008.

[18] G. Liu, Z. Li, K. Barkaoui, and A. Al-Ahmari, "Robustness of deadlock control for a class of petri nets with unreliable resources," *Information Sciences*, vol. 235, pp. 259–279, 2013.

[19] H. Yue, K. Xing, and Z. Hu, "Robust supervisory control policy for avoiding deadlock in automated manufacturing systems with unreliable resources," *International Journal of Production Research*, vol. 52, no. 6, pp. 1573–1591, 2014.

[20] F. Wang, K. Xing, M. Zhou, X. Xu, and L. Han, "A robust deadlock prevention control for automated manufacturing systems with unreliable resources," *Information Sciences*, vol. 345, pp. 243–256, 2016.

[21] E. Alzalab, Z. Yu, N. Wu, and H. Kaid, "Fault-recovery and repair modeling of discrete event systems using petri nets," *IEEE Access*, vol. 8, pp. 170 237–170 247, 2020.

[22] E. Alzalab, U. Abubakar, H. E, Z. Li, M. El-Meligy, and A. El-Sherbeeny, "Modeling of fault recovery and repair for automated manufacturing cells with load-sharing redundant elements using petri nets," *Processes*, vol. 11, no. 5, p. 1501, 2023.

[23] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable manufacturing systems," *CIRP Annals*, vol. 48, no. 2, pp. 527–540, 1999.

[24] S. Mortensen and O. Madsen, "Operational classification and method for reconfiguration and recommissioning of changeable manufacturing systems on system level," *Procedia Manufacturing*, vol. 28, pp. 90–95, 2019.

[25] M. Uzam, Z. Li, and U. Abubakar, "Think globally act locally approach for the synthesis of a liveness-enforcing supervisor of fmss based on petri nets," *International Journal of Production Research*, vol. 54, no. 15, pp. 4634–4657, 2016.

[26] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[27] J. L. Peterson, "Petri nets," *ACM Computing Surveys*, vol. 9, no. 3, pp. 223–252, 1977.

[28] A. Giua and M. Silva, "Petri nets and automatic control: A historical perspective," *Annual Reviews in Control*, vol. 45, pp. 223–239, 2018.

[29] M. Zhou and K. Venkatesh, "Modeling, simulation, and control of flexible manufacturing systems: A petri net approach," *World Scientific*, 1998.

[30] J. Ezpeleta, J. M. Colom, and J. Martinez, "A petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 173–184, 1995.

[31] Z. Li and M. Zhou, "Elementary siphons of petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 34, no. 1, pp. 38–51, 2004.

[32] H. Hu, M. Zhou, and Z. Li, "Supervisor optimization for deadlock resolution in automated manufacturing systems with petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 4, pp. 794–804, 2011.

[33] Y. Wu, K. Xing, J. Luo, and Y. Feng, "Robust deadlock control for automated manufacturing systems with an unreliable resource," *Information Sciences*, vol. 346, pp. 17–28, 2016.

[34] Z. Zhang, G. Liu, K. Barkaoui, and Z. Li, "Adaptive deadlock control for a class of petri nets with unreliable resources," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 5, pp. 3113–3125, 2021.