IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 October 2024, accepted 12 November 2024, date of publication 20 November 2024,
date of current version 29 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3502920

RESEARCH ARTICLE

Robust Supervisory Control for Automated Manufacturing Systems With Minimal Decentralized Switch-Buffer Controllers

UMAR SULEIMAN ABUBAKAR[1], AHMAD BALA ALHASSAN[2], (Member, IEEE), AND GRIDSADA PHANOMCHOENG[1,3]

[1]Department of Mechanical Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
[2]Department of Robotics and Mechatronics, Nazarbayev University, 010000 Astana, Kazakhstan
[3]Human-Robot Collaboration and Systems Integration Research Unit, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand

Corresponding author: Gridsada Phanomchoeng (gridsada.phanomchoeng@gmail.com)

This research was supported in part by the Ratchadapisek Somphot Fund for Postdoctoral Fellowship through the Graduate School, Chulalongkorn University, Bangkok, Thailand; and funded by Thailand Science Research and Innovation Fund Chulalongkorn University (IND_FF_68_007_2100_001).

ABSTRACT Automated manufacturing systems require effective supervisory control mechanisms to ensure their efficient operation and prevent fatal errors. Majority of the deadlock supervisory control policies for automated manufacturing systems are formulated based on the presupposition that the systems' resources are all reliable. However, in actuality systems' resources experience failure. Deadlocks and blockages could still occur in a system with failure-prone resources to which a supervisory controller is deployed if a failure-prone resource in the system fails. Extra buffers (redundant buffers) are usually used on production floors to store work-in-process parts in order to prevent deadlocks or blockages caused by the failure of failure-prone resources. However, additional buffers signify extra costs to the enterprise. To this end, this paper presents a robust supervisory control framework for automated manufacturing systems that utilizes minimum amount of decentralized buffer units and switch controllers, viz., switch-buffer controllers. The policy combines control places computed to prevent deadlocks in the system and switch-buffer controllers to prevent deadlocks/blockages in the event of failure-prone resource failure. Both the plant and the proposed supervisory controller are modeled using Petri nets. Examples are used to validate and demonstrate the effectiveness of the proposed approach. Furthermore, the results, compared with some existing techniques, indicate that the proposed technique has more advantages over them.

INDEX TERMS Robust supervisory control, automated manufacturing system, deadlock, Petri net, resource failure.

I. INTRODUCTION

Efficient operation of modern manufacturing industry heavily relies on automated manufacturing systems (AMSs) that produce high-quality products with minimal human intervention. The effectiveness of these systems is highly dependent on their supervisory control ability to ensure their safe and efficient operations. The literature abounds with deadlock supervisory control policies, such as [1], [2], [3], and [4], without considering possible failure of systems' components. However, AMSs are susceptible to various uncertainties such as unexpected faults and disturbances that may lead to catastrophic failures. Therefore, developing robust supervisory control mechanisms that can mitigate the impact of these uncertainties and ensure that the systems operate reliably is imperative.

In recent years, the emerging supervisory control policies reflect the growing concern over deadlocks and blockages that may occur as a result of failure of systems' resources. Majority of the existing studies use automaton or Petri net to model and design the supervisory control policies. As a result of Automata's structural characteristic limitations, they can

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

only be utilized to model simple systems. By contrast, Petri nets can represent the majority of the behaviors of discrete event systems, including concurrency, synchronization and conflict. Thus, Petri nets are more potent modeling tools than automata and more effective in modeling, identifying and resolving deadlock issues. Liu et al. [5] propose a robust supervisory controller using Petri nets. The policy requires addition of failure and recovery subnets to each operation place that is a holder of a failure-prone resource. Monitors are connected to recovery subnets by normal and inhibitor arcs, ensuring that no empty siphon is present at any moment. However, there is a wait-for-repair state whenever a resource fails to function, which implies that no operation can be carried out until the failed resource is repaired. Wu et al. [6] improves the robust supervisory controller in [5] by eliminating the wait-for-repair states and inhibitor arcs, which significantly reduces the structural complexity of the controller.

The work [7] utilizes the powerful modeling capabilities of Petri nets to model AMSs with multi-type and multi-quantity resource acquisition and proposes a robust supervisory controller. First, to ensure the system remains live, strict minimal siphons are managed by adding control places to prevent them from becoming under-marked. Second, in order to address blocking problems that arise from resource failures, a set of shared resource constraints are constructed as inequalities based on the minimal resource requirements of processes and the available capacity of shared resources. Robust monitors are added to restrict tokens distribution in failure-prone resource's neighboring places. Based on a reachability graph technique, [8] introduces a robust optimization Petri net controller for AMSs with failure-prone resources. The strategy divides reachability graph of an AMS into forbidden markings and robust legal markings. An invariant-based controller is computed that guarantees no forbidden markings are reachable while robust legal ones are reachable. Furthermore, the idea of improved recovery subnets is introduced for the first time. This notion of improved recovery subnets represents a resource failure event in which, upon failure, the failed resource is taken away for repair and the work-in-process part is subsequently returned to the system for processing. Another method that develops modified recovery subnets is [9], albeit it is an adaptive supervisory control policy and it is a siphon control-based technique.

A number of robust supervisory control policies propose different ideas of utilizing buffer slots of AMSs' workstations to achieve systems' robustness. Chew and Lawley [10] present a robust supervisory control strategy for AMSs with multiple failure-prone resources by combining two policies. The first policy is a combination of a neighborhood policy, which permits system states that comply with the run-time requirements of the neighborhood constraints and a modified banker's algorithm. The second policy combines the neighborhood policy with a single-step look ahead policy. In both policies a part type can only require one failure-prone resource in its processing route. The study [11] proposes the idea of single-route neighborhood policy in conjunction with a siphon-based policy in the framework of Petri nets. The single-route neighborhood policy is a modification of the neighborhood policy in which neighborhoods of every failure-dependent resource is computed according the processing routes of part types. The single-route neighborhood policy could be more permissive in behavior than the neighborhood policy in [10].

Yue et al. [12] tackle the deadlock and blockage problems in AMSs with multiple failure-prone resources. The study establishes a robust control policy that enables the system to continue handling all part types without human involvement as long as no failed resource is needed. The policy employs a set of remaining resource capacity constraints and a modified banker's algorithm. A directed graph for the residual resource capacity is constructed by treating each failure-dependent resource as a vertex. Through the computation of a set of strongly connected components within the directed graph, a set of inequality constraints are generated that guarantee only a single resource within a strongly connected component shall have all its buffer spaces completely occupied. Based on the shared resource capacity, [13] establishes a supervisory control policy for AMSs with failure-prone resources that distributes all parts that require failed resources on their remaining routes among the shared resources' buffer spaces. However, it requires constructing inequality constraints, which increases exponentially with the size of the system.

Wang et al. [14] categorize the robust control policies into two types. The first type, referred to as the ''absorbing'' type, involves assimilating all the parts that require failed resources in their remaining routes into the buffer slots of failure-dependent resources. The second type, known as the ''distributing'' type, involves distributing the parts that require failed resources in their remaining routes among the buffer slots of shared resources. The research in [15] centers around addressing the issue of resource failure in AMSs, and proposes a solution through the control of buffer space allocation without utilizing central buffers. The study's findings indicate that implementing the buffer space allocation control method can yield notable improvements in the throughput of AMSs. All these policies so far, utilize the buffers of the AMSs' workstation without extra (redundant) buffers.

Chew et al. [16] use automaton to design a robust supervisory controller for AMSs with multiple failure-prone resources in which part types require more than one failure-prone resource in their processing routes. The policy utilizes the same policies as the ones in [10] but with additional central buffer. Either of the policies divides the processing paths into subroutes and uses the central buffers to temporarily store part types in case a failure-prone resource fails. Central buffer constraints are formulated in order to determine the capacity of the central buffer. The approach makes an assumption that if a workstation's server fails, it can still allocate buffer spaces up to its capacity. However,

any waiting parts cannot be processed, resulting in delays along their routes until the server is repaired. In [17] an adaptive supervisory control policy in the framework of Petri net is proposed. The method employs additional buffers in combination with switch controllers and observers that observe resources' failures. If a resource fails, the switch controllers are activated in order to remark siphons that may become unmarked through the buffer places. However, the work fails to specify the capacity of the extra buffers required for the design of the controller.

Utilization of buffers is one of the techniques used to provide resilient operations in AMSs. Buffers are extra resources assigned to different portions of an AMS that can be utilized to alleviate the consequences of unexpected factors that may interfere with the smooth operation of production lines. It is feasible to absorb work-in-process parts that require a failed resource in order to ensure continued operation of other processes in the event of partial system failure by distributing buffer spaces among many components of the system. This improves the AMS's overall reliability and prevents deadlocks, blockages or complete system breakdown [18]. In most of the existing supervisory control techniques, if extra buffers are used, the buffers are usually centralized. Therefore, allocation of buffer spaces and assimilations of work-in-process part types into the centralized buffer spaces require additional hardware such as robots, automated guided vehicles, etc.

Leveraging the superior characteristics of Petri nets, this paper explores a novel way by designing a robust supervisory controller utilizing minimum amount of decentralized buffer units to achieve systems operational robustness with respect to resource failures. Additionally, the proposed policy can enhance system's robustness to resource failure by providing redundancy and allowing processes in the system that do not require the failed resource to continue without interruption. The utilization of decentralized buffers can significantly increase the permissive behavior of an AMS that is prone to resource failures and reduce cost of implementing the controllers. Furthermore, decentralizing the buffers by situating them close to failure-prone resources can ease relocation of part types from the failure-prone resources' buffer spaces and vise versa in the event of the failure-prone resources failure.

The amount of extra buffer units required for the design of the proposed controller is minimized by solving a linear integer programming problem in order to find the maximum marking of a set of operation stages that require a failure-prone resource. The supervisory control policy employs switch-buffer controllers that change system's operational modes. When there is no resource failure the system operates in a normal mode. In this mode, monitors designed to prevent siphons from emptying keep the system live and prevent the system from entering deadlock states. If a failure-prone resource fails, the system enters a degraded mode. In this mode, the switch-buffer controllers are activated in order to move some of the part types requiring failed resources into the decentralized buffer spaces. This allows other processes not requiring the failed resource to proceed without blockage. If the failed resource's operation is recovered, the system returns to its normal operational mode. To the best of our knowledge, there is no existing policy that, in the design of the supervisory controller with extra buffers using Petri nets, attempts to minimize the extra buffers required to absorb part types requiring a failed resource, and also decentralize the extra buffers. Though the proposed policy is not optimal in terms of permissive behavior since it is based on siphons, it is has less computational complexity than the reachability graph-based approaches that are known to have better permissive behaviors.

The remainder of the paper is arranged as follows. Section II is a preliminary section that recaps important definitions, theorems, propositions and properties of Petri nets that are crucial to understanding the work. Section III delineates AMSs with unreliable resources and their characteristics and introduces failure-prone resource's server failure and recovery subnets. The design of robust supervisory controllers is presented in Section IV. Section V discusses and compares this work with previous studies. Section VI concludes the study.

II. PRELIMINARIES

A. BASICS OF PETRI NETS

Petri nets, commonly abbreviated as PNs, are widely utilized for modelling the flow of products in automated manufacturing systems (AMS). As a mathematical tool, PNs possess interesting and beneficial properties. When applied to a modeled manufacturing system, these properties enable one to determine whether the system has functional properties or not. For a comprehensive understanding of PN theory and its applications, interested readers can refer to [19]. In what follows, we provide a concise review of specific definitions that are relevant to our research.

A Petri net system is represented by the five-tuple $N = (P, T, F, W, M_0)$, where $P = \{p_1, p_2, p_3, \ldots, p_m\}$ is a finite set of places ($m \geq 0$) and $T = \{t_1, t_2, t_3, \ldots, t_n\}$ is a finite set of transitions ($n \geq 0$) with $P \cup T \neq \emptyset$ and $P \cap T \neq \emptyset$. A PN is actually a bipartite-directed graph. From the standpoint of a graph, the elements of $P \cup T$ are regarded as nodes. The set of all directed arcs between the nodes is denoted by $F \subseteq (T \times P) \cup (P \times T)$, where the output function $T \times P \to \mathbb{N}$ determines the set of directed arcs from T to P, and the input function is $P \times T \to \mathbb{N}$, which determines the set of directed arcs from P to T, where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is a set of non-negative integers. By definition, there is never a directed arc connecting any two items of the same type, such as a place and a transition. The weight function $W : F \to \mathbb{N}$ is a mapping and the initial marking $M_0 : P \to \mathbb{N}$. ${}^\bullet p$ (resp., $p^\bullet$) represents the set of input (resp., output) transitions of a place p. Likewise, ${}^\bullet t$ (resp., $t^\bullet$) represents the set of input (resp., output) places of a transition t. $G = (P, T, F, W)$ is a PN structure with no initial marking specified. A net structure G with a specified initial marking M_0 is represented

by (G, M_0), which is a PN system, as described at the outset, i.e., $N = (G, M_0)$. If its graph is interconnected, a Petri net is considered to be interconnected.

A transition t is considered enabled or firable when all input places $p \in {}^\bullet t$ have a minimum of $W(p,t)$ tokens, where $W(p,t)$ represents the weight of the arc from p to t. If a transition is enabled, it can be fired. Upon firing an enabled transition t, $W(p,t)$ tokens are removed from each input place $p \in {}^\bullet t$, and $W(t,p)$ tokens are added to each output place $p \in {}^\bullet t$, where $W(t,p)$ represents the weight of the arc from t to p. This entire process can be represented by $M[t\rangle M'$. The modeled system is currently in its current state, represented by the marking M, which displays the number of tokens in each place. The firing of a sequence of transitions $\sigma = t_0 t_1 t_2 \ldots t_k \in T^*$ from a marking M results in the marking M', denoted by $M[\sigma\rangle M'$, where T^* is the Kleene closure of set T. The set of all reachable markings of a net system with the initial marking M_0 is denoted by $RM(G, M_0)$, i.e., $RM(G, M_0) = \{M \in \mathbb{N}^{|P|} \mid \exists \sigma \in T^* : M_0[\sigma\rangle M\}$.

If there is a place p and a transition t such that $p \in t^\bullet$ and $p \in {}^\bullet t$ hold, then the combination of p and t is referred to as a self-loop. A Petri net is considered *pure* if it has no self-loops. A Petri net is considered *ordinary* if the weight of every arc is 1. A net (G, M_0) is referred to be simply bounded or k-bounded if, at any reachable marking $M \in RM(G, M_0)$, the number of tokens at any place p is not greater than a finite number k, i.e., $M(p) \leq k$, where $k \in \mathbb{N}$ is a non-negative integer. If the number of tokens in a place p at any marking is no greater than k, the place p is referred to as k-bounded. If a net system satisfies the condition of being 1-bounded, then it is characterized as safe. A place that satisfies the condition of being 1-bounded is defined as a safe place. At the initial marking M_0, a transition t is considered live if there is a sequence of transitions σ that can be fired from M_0 to reach a marking $M_0[\sigma\rangle M$ such that t is enabled at M. If M_0 has all transitions as live, then (G, M_0) net system is considered live. In case there exists a marking $M \in RM(G, M_0)$ at which no transition is enabled, then a deadlock is present in the net system (G, M_0). Such a marking is referred to as a dead marking.

Let $\sigma \in T^*$ be a sequence of transitions of a Petri net. The number of occurrences of a transition t_i in σ, without regard to their positions, is represented by $\#\sigma(t_i)$. The parikh vector of σ is a column vector denoted by $\vec{\sigma} = [\#\sigma(t_1), \#\sigma(t_2), \ldots, \#\sigma(t_m)]^T$, where $m = |T|$. A P-vector is a column vector $I : P \longrightarrow \mathbb{Z}$, indexed by P, where $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. A P-vector I is a place invariant if $I \neq 0$ and $I^T[N] = 0^T$. A T-vector is a column vector $J : T \longrightarrow \mathbb{Z}$ indexed by T. A T-vector J is a transition invariant if $J \neq 0$ and $[N]J = 0$. The support of a place (transition) invariant $I(J)$ is denoted by $\|I\| = \{p|I(p) \neq 0\}$ ($\|J\| = t|J(t) \neq 0$). A P-invariant I is said to be a P-semiflow if for all $p \in P, I(p) \geq 0$. It is called a minimal P-invariant if $\|I\|$ is not a proper superset of the support of any other one and its components are mutually prime.

A siphon in a nonempty set $S \subseteq P$ is defined as follows: S is a siphon (trap) if the following condition holds: ${}^\bullet S \subseteq S^\bullet$. $S \subseteq P$ is a trap if $S^\bullet \subseteq {}^\bullet S$. A siphon (trap) is said to be minimal if there is no siphon (trap) contained in it as a proper subset. A minimal siphon is said to be strict minimal if it does not contain a marked trap. A strict minimal siphon is abbreviated as SMS. The solution for SMS in a Petri net is theoretically well-developed [20]. The approach involves first identifying siphons, minimal siphons, and P-invariants in a Petri net, and then eliminating the minimal siphons that contain the support of a P-invariant.

Let (N, M_0) be ordinary, I be a P-invariant, and $S \subseteq P$ be a siphon of N. S is said to be controlled by P-invariant I at M_0if $I^T M_0 > 0$ and $I(p) \leq 0$ for all $p \in P \setminus S$ hold, or equivalently, $I^T M_0 > 0$ and $\{p \in P|I(p) > 0\} \subseteq S$. Such a siphon is called invariant-controlled. An invariant-controlled S cannot be emptied.

B. S^{3}PR *MODELS*

The AMS model that is used in this paper can be modelled using systems of simple sequential processes with resources (S^{3}PR) sub-class of PN model. Therefore, it is vital for better understanding of the paper to provide some definitions and properties of S^{3}PR that are relevant to the work.

Definition 1: A PN $N = (\{p^0\} \cup P_A, T, F)$ is considered a simple sequential process (S^{2}P) if it satisfies the following conditions: 1) The set of activity places $P_A \neq \emptyset$; 2) the idle place $p^0 \notin P_A$; 3) place p^0 is present in every circuit of N; and 4) N is a strongly connected state machine; ◇

Definition 2: A PN $N = (P_A \cup \{p^0\} \cup P_R, T, F)$ fulfilling the following requirements is referred to as an (S^{2}P) with resources (S^{2}PR): 1) $P_R \neq \emptyset$; $(\{p^0\} \cup P_A) \cap P_R = \emptyset$; 2) $\forall p \in P_A, \forall t \in {}^\bullet p, \forall t' \in p^\bullet, \exists r_p \in P_R, {}^\bullet t \cap P_R = t'^\bullet \cap P_R = \{r_p\}$; 3) the subnet generated from $X = \{p^0\} \cup P_A \cup T$ is an S^{2}P; 4) $\forall r \in P_R, {}^{\bullet\bullet} r \cap P_A = r^{\bullet\bullet} \cap P_A \neq \emptyset$; $\forall r \in P_R, {}^\bullet r \cap r^\bullet = \emptyset$; 5) ${}^{\bullet\bullet}(p^0) \cap P_R = (p^0)^{\bullet\bullet} \cap P_R = \emptyset$. ◇

Definition 3: If an S^{2}PR N$=(\{p^0\} \cup P_A \cup P_R, T, F)$ is given, an initial marking M_0 is considered acceptable for N if and only if the following criteria are satisfied: 1) $M_0(p) = 0, \forall p \in P_A$; 2) $M_0(p^0) \geq 1$; 3) $M_0(r) \geq 1, \forall r \in P_R$; ◇

Definition 4: A recursive formulation for an S^{3}PR, which represents a system of S^{3}PRs, can be expressed as follows: 1) An S^{2}PR is an S^{3}PR; 2) Let $N_1 = (P_{R_1} \cup P_{A_1} \cup \{p_1^0\}, T_1, F_1)$ and $N_2 = (P_{R_2} \cup P_{A_2} \cup \{p_2^0\}, T_2, F_2)$ be two S^{3}PR, satisfying $(P_{A_1} \cup \{p_1^0\}) \cap (P_{A_2} \cup \{p_2^0\}) = \emptyset, P_{R_1} \cap P_{R_2} = P_C \neq \emptyset$, and $T_1 \cap T_2 \neq \emptyset$. The Petri net $N = (P_R \cup P_A \cup P^0, T, F)$ composed by N_1 and N_2 through P_C, denoted by $N = N_1 \circ N_2$, is still an S^{3}PR, defined as $P_R = P_{R_1} \cup P_{R_2}$, $P_A = P_{A_1} \cup P_{A_2}$, $P^0 = \{p_1^0\} \cup \{p_2^0\}$, $T = T_1 \cup T_2$, and $F = F_1 \cup F_2$. ◇

Definition 5: Let N be an S^{3}PR. (N, M_0) is called an acceptably marked S^{3}PR if one of the following conditions is satisfied: 1) (N, M_0) is an acceptably marked S^{3}PR. 2) $N = N_1 \circ N_2$, where $(N_i, M_{0_i})\,(i = 1, 2)$ is an

acceptably marked S^{3}PR. Moreover, for all $i \in \{1, 2\}$ and $p \in P_{A_i} \cup \{p_i^0\}, M_0(p) = M_{0_i}(p)$; for all $i \in \{1, 2\}$ and $r \in P_{R_i} \backslash P_C, M_0(r) = M_{0_i}(r)$; for all $r \in P_C, M_0(r) = \max\{M_{0_1}(r), M_{0_2}(r)\}$. ◇

C. ELEMENTARY AND DEPENDENT SIPHONS

In this section, we establish the notions of elementary and dependent siphons within a PN.

Definition 6 [21]: Suppose that $S \subseteq P$ is a subset of the places of N. A P-vector λ_S is said to be the characteristic P-vector of S if $\forall p \in S : \lambda_S(p) = 1$; otherwise $\lambda_S(p) = 0$. ◇

Definition 7 [21]: Suppose that $S \subseteq P$ is a subset of places within N, and let η_S denote the characteristic P-vector of S. The characteristic T-vector of S, denoted by η_S, is defined as $\eta_S^T = \lambda_S^T \bullet [N]$. ◇

Property 1 [21]: The T-vector that characterizes the support of a P-invariant is 0 when the highest absolute value of any element within the P-invariant is a unit. ◇

Property 2: Suppose that S is a subset of a net $N = (P, T, F)$, and let η_S denote the characteristic T-vector of S. The sets of transitions that increase, maintain, and decrease the number of tokens in S can be represented as $\{t \in T \mid \eta_S(t) > 0\}$, $\{t \in T \mid \eta_S(t) = 0\}$, and $\{t \in T \mid \eta_S(t) < 0\}$, respectively. ◇

In what follows Π represents the set of SMS of a net N, while η_{S_i} refers to the characteristic T-vector of siphon S_i, where $i \in \mathbb{N}$.

Definition 8: $\forall S_0 \in \Pi$, if $\nexists S_1, S_2, \ldots, S_n \in \Pi(\forall i \in \{1, 2, \ldots, n\}, S_0 \neq S_i)$ such that $\eta_{S_0} = a_1\eta_{S_1} + a_2\eta_{S_2} + \cdots + a_n\eta_{S_n}$ holds, where $a_1, a_2, \ldots a_n \in \mathbb{N}\backslash\{0\}$, then S_0 is referred to as an elementary siphon of N. ◇

We use Π_E to denote the set of elementary siphons in a net N.

Definition 9: Let $S_0 \in \Pi\backslash\Pi_E$ be a siphon in a net N and $S_1, S_2, \ldots, S_n \in \Pi_E$ be its elementary siphons. S_0 is called a strict redundant siphon w.r.t. $S_1, S_2, \ldots$, and S_n if $\eta_{S_0} = a_1\,\eta_{S_1} + a_2\eta_{S_2} + \cdots + a_n\eta_{S_n}$ holds, where $a_1, a_2, \ldots a_n \in \mathbb{N}\backslash\{0\}$. ◇

III. S^{3}PR WITH UNRELIABLE RESOURCE

In this section, a PN model of an AMS with a failure-prone resource is presented. This PN model will be used as a case study to develop the idea of the robust supervisory control policy proposed in this study. Fig. 1 visualizes a PN model of a single-unit resource allocation system (SU-RAS), which is modeled as an S^{3}PR subclass of PN model [20]. We consider every resource of a system as workstation, consisting of server or processor to process different types of parts. Each workstation has buffer slots to hold part types that will be processed by the workstation's server or processor. The initial marking of every resource place in a PN model represents the resource's or workstation's buffer slots capacity (number of buffer slots). Therefore, a net system represents the allocation and de-allocation of buffer slots of an AMS and the movement of part types within the buffer slots. This type of PN model is called a buffer net [22]. The S^{3}PRs considered in this study are buffer nets. The buffer PN model in Fig. 1 has six resources whose set is denoted by $P_R = \{r_1, r_2 \ldots, r_6\}$. It has one failure-prone resource r_3, while the remaining resources are all considered to be reliable. The buffer capacity of a workstation or resource is donated by B_i, where $i = \{1, 2, \ldots, n\}$, n is the number of resources or workstations in the system and $B_i = M_0(r_i)$. As indicated in Fig. 1, $B_1 = B_3 = B_4 = 2$ and $B_2 = B_5 = B_6 = 1$. Failure of a resource is assumed to be the failure of the server and it does not affect the buffer slots of the resource. Thus, the buffer spaces of a workstation whose server fails can still be used to store work-in-process parts until the server is restored.

P^0 denotes the set of idle places in a PN model. P^0 represents the set of part types processed in an AMS. A part type $p_j \in P^0$ has a set of b-ordered processing stages, denoted as $P_j = \{p_{j1}, p_{j2}, \ldots, p_{jb}\}$. In a PN model, each part type stage is represented as an operation place, also known as an activity place, where p_{jk} represents the k-th processing stage of part type p_j. We denote by $P_A = \cup_{j=1}^{m} P_j$ the set of all operation places in a PN model, where $m = |P^0|$ denotes the number of part types in an AMS [23].

Let $R(p_{jk}) = r_i$ represent the fact that r_i supports part type p_j at its k-th processing stage. The set $H(r_i) = \{p_{jk} \in P_A | R(p_{jk}) = r_i\}$ denotes the set of operation places that are holders of resource r_i, i.e., the set of operation stages supported by r_i. Let $T_j = \{t_{j1}, t_{j2}, \ldots, t_{j(b+1)}\}$ be the set of transitions of processing stages of p_j, where t_{jk} represents the transition of part type p_j from operation stage $p_{j(k-1)}$ to operation stage p_{jk}. We donate $\mathfrak{R}_j$ to be the processing route of the part type p_j.

Let $(N, M_0) = (P_A \cup P^0 \cup P_R, T, F, M_0)$ be a marked S^{3}PR PN model of an unreliable AMS. Let the set of resources of (N, M_0) be $P_R = P_R^r \cup P_R^u$ with $P_R^r \cap P_R^u = \emptyset$, where P_R^r is the set of reliable resources, and P_R^u is the set of unreliable resources.

Definition 10: Let $(N, M_0) = (P_A \cup P^0 \cup P_R, T, F, M_0)$ be a marked S^{3}PR and $r_u \in P_R^u$ be an unreliable resource. The server failure and recovery subnet of an unreliable resource is a PN $(N_{ru}, M_{ru_0}) = (\{p_{s_u}, p'_{s_u}\}, \{t_{f_u}, t_{r_u}, t_{jk} \in {}^\bullet r_u\}, F_{r_u}, M_{ru_0})$, where t_{f_u} denotes the occurrence of failure of the server of r_u, t_{r_u} denotes the completion of repair and recovery of a failed server, and $F_{r_u} = \{(p_{s_u}, t_{f_u}), (t_{f_u}, p'_{s_u}), (p'_{s_u}, t_{r_u}), (t_{r_u}, p_{s_u}), (p_{s_u}, t_{jk} \in {}^\bullet r_u), (t_{jk} \in {}^\bullet r_u, p_{s_u})\}$. By $M_{ru}(p_{s_u}) = 1$ signifies that the server of r_u is available and $M_{ru}(p'_{sv_u}) = 1$ indicates that r_u's server is down. ◇

(N_{ru}, M_{ru_0}) is a marked server failure and recovery subnet of a failure-prone resource in a PN model. At the initial marking of the unreliable PN model, $M_{ru_0}(p_{s_u}) = 1$ and $M_{ru_0}(p'_{s_u}) = 0$. This means that the server of a failure-prone resource is available at the initial marking, indicating that failure does not occur at the initial marking. Fig. 2 depicts the server failure and recovery subnet.

Addition of server failure and recovery subnet to an S^{3}PR net model of an AMS generates an unreliable S^{3}PR buffer net model denoted by US^{3}PR, which is defined as $(N_u, M_{u_0}) = (N, M_0) \| (N_{ru}, M_{ru_0})$. This is called a parallel composition

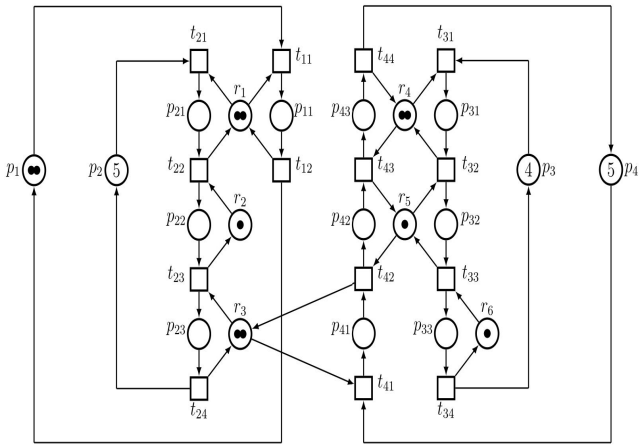

FIGURE 1. Petri net model.

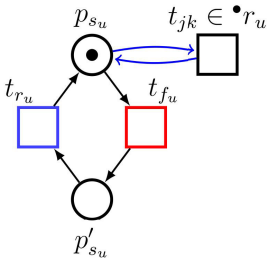

FIGURE 2. Server failure and recovery subnet.

of two net systems, where the symbol $\|$ denotes parallel composition of two PNs having no common place.

The self-loop $(p_{s_u}, t_{jk} \in {}^\bullet r_u)$, $(t_{jk} \in {}^\bullet r_u, p_{s_u})$ in Fig. 2 enables and disables $t_{jk} \in {}^\bullet r_u$ in the following manner. If the server of a failure-prone resource r_u in a US^{3}PR fails while processing a part type p_{jk}, the transition t_{jk+1} (with $t_{jk+1} \in {}^\bullet r_u$) becomes disabled since $M_u(p_{s_u}) = 0$. This implies that the part type p_{jk} will remain in the buffer space of r_u until the server is recovered, i.e., $M_u(p_{s_u}) = 1$, t_{jk+1} becomes enabled, and r_u resumes processing p_{jk}. After the processing of p_{jk} by r_u is completed, p_{jk}, if allocated enough resources, moves to the next workstation in its processing route (for the next stage p_{jk+1} of p_{jk}) [23].

Example 1: Fig. 3 shows the unreliable buffer net model US^{3}PR of the AMS of Fig. 1 generated as a result of the composition of PN model of Fig. 1 and server failure and recovery subnet of the failure-prone resource r_3. The reachability graph of the PN model shown in Fig. 1 has 2448 reachable markings (states), whose live zone contains 2,304 good states and whose dead zone contains 144 bad states, 1 of which is a deadlock state. The reachability graph of the PN model shown in Fig. 3 with added failure and recovery subnet, has 4,896 markings with 4,608 reachable states, whose live zone contains 4,608 good states and whose dead zone contains 288 bad states. The dead zone has no deadlock state. The net system has a set of resources $P_R = \{r_1, r_2, \ldots, r_6\}$ with a set of reliable resources $P_R^r = \{r_1, r_2, r_4, r_5, r_6\}$ and a set of unreliable resource $P_R^u = \{r_3\}$. If $M(p_{s_3}) = 1$ and $M(p'_{s_3}) = 0$, it indicates that the server of r_3 is available. If $M(p_{s_3}) = 0$ and $M(p'_{s_3}) = 1$, the server of r_3 is down, t_{f_3} represents the occurrence of failure of the server of r_3, and t_{r_3} represents the recovery of the failed server of r_3. ◇

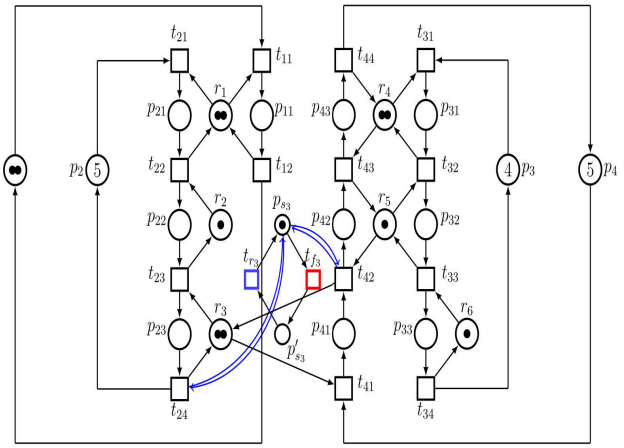

FIGURE 3. PN model with added server failure and recovery subnet.

IV. ROBUST SUPERVISORY CONTROL

This section presents the robust supervisory control for a US^{3}PR PN model of an AMS with failure-prone resources. The robust supervisory control can be applied to a US^{3}PR model with multiple unreliable resources in which each part type may require more than one unreliable resource in its processing stages. The proposed control policy is composed of control places computed to prevent elementary siphons from emptying and decentralized switch-buffer controllers. The decentralized switch-buffer controllers are activated only when a server of an unreliable resource fails in order to relocate part types released into the system that require the failed resource into the buffer spaces of the decentralized buffers. After the recovery of the failed resource, the decentralized switch-buffer controllers are deactivated and the normal operational mode of the system is restored. The following discussion and definitions are vital to understanding the design of the proposed robust supervisory control.

Theorem 1 [24]: Let (N, M_0) with $N = (P, T, F)$ be a net system and S be a strongly dependent siphon of N. Let $S_1, S_2, \ldots,$ and S_n be elementary siphons of S with $\eta_S = a_1\eta_{S_1} + a_2\eta_{S_2} + \ldots + a_n\eta_{S_n}$. S is invariant-controlled if

1) For all $i \in \{1, 2, \ldots, n\}$, I_i is a P-invariant of N, $\|I_i\|^+ = S_i$, and for all $p \in S_i$, $I(p) = 1$;
2) $M_0(S) > \sum_{i=1}^{n} \sum_{p \in \|I_i\|^-} (a_i |I_i(p)| M_0(p))$. ◇

Proposition 1: Let $N_u = (P_A \cup P^0 \cup P_R, T, F)$ be a marked US^{3}PR and S be an SMS of N_u. Add monitor v_S to N_u and the augmented net is denoted by (N_v, M_{v_0}), where $p \in P_A \cup P^0 \cup P_R, M_{v_0}(p) = M_{u_0}(p), M_{v_0}(v_S) = M_{u_0}(S) - 1$. v_S is added such that $I = p_x + \ldots + p_y + p_\alpha + \ldots + p_\beta + v_S$ is a P-

invariant of N_1, where $\{p_x, \ldots, p_y\} = [S]$, $\{p_\alpha, \ldots, p_\beta\} = \mathfrak{B}_S = \cup_{i=1}^{n} \mathfrak{B}_S^i$, where $\mathfrak{B}_S^i = \{p \in P_i | p <_N p', p' \in [S] \cap P_i, \nexists p'' \in SP(p', p_i^0)\}$, and $\mathfrak{B}_S \cap [S] = \emptyset$. Then S is invariant-controlled. ◇

Property 3 [24]: $v_S + \sum_{p \in B_S} p + \sum_{p \in [S]} p$ is a P-invariant of N. ◇

Theorem 2 [24]: S is controlled if $M_{v_0}(S) > \sum_{i=1}^{n} a_i M_{u_0}(S_i) - \sum_{i=1}^{n} a_i$, where $a_i \geq 0$. ◇

A control place is usually designed to limit the number of tokens in operation places in a PN model of an AMS to prevent the system from reaching some undesired states. This is achieved by restricting the flow of tokens from related resources. Let $\mathcal{P}(v_S) = \sum_{p \in B_S} p + \sum_{p \in [S]} p$. If an operation place $p_{jk} \in \mathcal{P}(v_S)$, it is said to be associated with control place v_S. Similarly, if $\exists p_{jk} \in H(r_i)$ such that $v_S^\bullet <_N p_{jk}, p_{jk} <_N {}^\bullet v_S$, resource r_i is said to be associated with control place v_S.

Let $\mathcal{P}(r_u) = \{p_{jk} | \exists c \geq 0, p_{j(k+c)} = H(r_u)\}$ be the set of operation stages that require a failure-prone resource r_u now or later in their processing stages. Let $\mathcal{F}(r_u) = \{r_i \in P_R \mid \forall p_{jk} \in H(r_i), \exists c \geq 0, p_{j(k+c)} \in H(r_u)\}$ be a set of resources that process only part types that require a failure-prone resource r_u without any intervening failure-prone resource or another resource that processes both part types $p_{vw} \in \mathcal{P}(r_u)$ and $p_{xy} \notin \mathcal{P}(r_u)$, including the failure-prone resource r_u, where $p_{vw}, p_{xy} \in P_A$.

Example 2: In the PN model of Fig. 2, r_3 is the only failure-prone resource. The part type stage p_{22} of p_2, which is the only part type processed by the reliable resource r_2 will later require r_3. Therefore, the set $\mathcal{F}(r_3) = \{r_2, r_3\}$ and $\mathcal{P}(r_3) = \{p_{21}, p_{22}, p_{23}, p_{41}\}$. ◇

Example 3: The PN model of Fig. 3 has one SMS $S_1 = \{p_{32}, p_{43}, r_4, r_5\}$ and its corresponding complementary set is $[S_1] = \{p_{31}, p_{42}\}$. Based on Proposition 1, $M_{v_0}(v_{S_1}) = 3 - 1 = 2$. The corresponding sets of input and output arcs of the control place are ${}^\bullet v_{S_1} = \{t_{32}, t_{43}\}$ and $v_{S_1}^\bullet = \{t_{31}, t_{41}\}$. The resultant net system with added control places (N_v, M_{v_0}) is depicted in Fig. 4. The resultant net system is live as long as r_3 does not fail, i.e., the system does not reach markings at which $M_v(p_{s_3}) = 0$ and $M_v(p'_{s_3}) = 1$. For instance, if the system reaches a marking $M' = p'_{s_3} + 2p_1 + 2p_2 + 3p_3 + 2p_4 + 2p_{21} + p_{22} + p_{33} + 2p_{41} + p_{43} + r_4 + r_5$, the server of r_3 fails and the system enters a deadlock state. At this marking, the server of r_3 fails while it is processing two part type stages p_{41}'s of part type p_4. Since the processing of the p_{41}'s is not completed by r_3 before its failure, the two p_{41}'s must stay in the buffer spaces of r_2 pending the repair and recovery of the r_3's server. Under this situation, the part type p_1, which is not supported by the failed resource r_3, is blocked from acquiring the resource r_1 by the two part type stages p_{21}'s currently occupying the two buffer spaces of r_1. As a result, the part type stage of p_1, p_{11}, cannot be processed by r_1 as all the buffer spaces of r_1 are occupied by p_{21}'s. ◇

The occurrence of deadlock/blockage in the controlled unreliable net system, as typifies in Example 3, is the consequence of the failure of the failure-prone resource, r_3, in the system. The deadlock/blockage problem could be avoided if the unreliable nature of the failure-prone resource was taken into account in the design of the supervisory controller. A robust supervisory control structure is expected to guarantee that part types not requiring a failure-prone resource can be processed if a failure-prone resource fails. In Example 3, only two parts, p_1 and p_4 require failure-prone resource r_3 in their processing routes. However, if r_3 fails, the system becomes completely blocked. It can be observed that the blockage of p_1 from being processed due to the failure of r_3 is caused by part type stages p_{41}'s requiring r_3 occupying all the buffer spaces. Conversely, the blockage of p_3 is as a result of v_{S_1} becoming unmarked, i.e., $M_v(v_{S_1}) = 0$ at the marking M', which is caused by the failure of r_3.

Example 3 demonstrates that deadlock supervisory structures designed without considering the unreliable nature of failure-prone resources may not be effective in the event of the failure of the failure-prone resources. Hence, underscores the urgency of situating the reality of resource failure by supervisory control practitioners in the design of supervisory controllers.

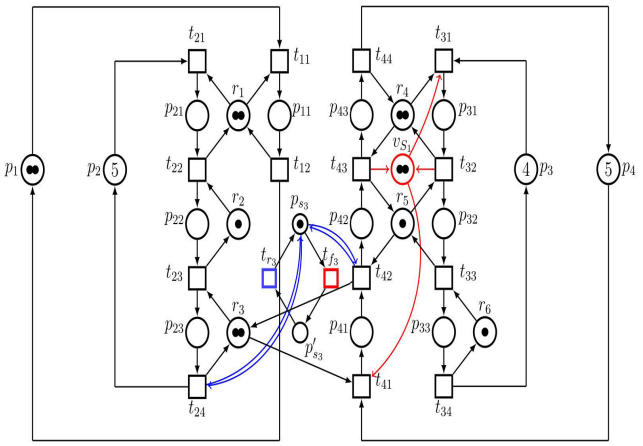

FIGURE 4. PN model with added control place v_{S_1}.

A. DESIGN OF THE SWITCH-BUFFER CONTROLLER

The switch-buffer controller is composed of two parts: (1) buffer subnet that provides the buffer spaces in order to store part type stages temporarily in the event of a resource failure, and (2) switch subnet. If a resource fails, the switch subnet is immediately activated in order to stop any additional release of the part types requiring the failed resource into the system. This is necessary owing to the fact that if new part types requiring the failed resource are released into the system, they will occupy the buffer spaces reserved for staging part types not requiring the failed resource. Thus, blocking them yet again from being processed.

Definition 11: A US^{3}PR model of SU-RAS is said to be in its normal operation mode if there is no resource failure, and it is said to be in its degraded mode if there is a resource failure and recovery. ◇

Definition 12: The supervisory control of a US^{3}PR model of a SU-RAS is said to be *robust* if it guarantees the

deadlock/blockage-free operation of the system in both its normal and degraded modes. ◇

It has been established in the previous section that failure of a failure-prone resource could block other processes that do not require the failed resource. Conventionally, part types requiring a failed resource that block other processes are transferred to extra buffers to be stored temporarily pending the repair and recovery of the failed resource. However, using extra buffers to prevent deadlocks and blockages can increase the cost of implementing a supervisory control structure. Therefore, minimizing the number of extra buffers used can reduce the cost of implementing a system's supervisory controller. To this end, it is imperative, after designing monitors to prevent SMS from becoming unmarked in a system, to check the possibility of the occurrence of either or both of the scenarios depicted in Example 3. First, we check the possibility of the failure of a failure-prone resource causing part types requiring the failed resource to block other processes that do not require the failed resource in the system. If this situation could occur, we determine the minimum number of buffer units required for a switch-buffer controller to accommodate part type stages requiring the failed resource in order to keep other processes in the system running without interruption, otherwise no switch-buffer controller should be added. Second, we investigate the possibility of the failure of a failure-prone resource causing some monitors in the system to become permanently unmarked, thereby interrupting other processes that do not require the failed resource. If this possibility exists, we design a buffer-switch controller to remark the monitors in order to ensure uninterruptedness of the other processes. The following are important to understanding how these two possibilities can be checked in a US^{3}PR PN model.

To determine the minimum number of buffers required in the design of a buffer-switch controller that should be added to a failure-prone resource r_u, we need to first ascertain the maximum marking of $\mathcal{P}(r_u)$ at any marking $M \in R(N_v, M_{v_0})$ denoted by $M_v(\mathcal{P}(r_u))_{max}$. $M_v(\mathcal{P}(r_u))_{max}$ can be found by solving the linear programming problem in (1).

$$\begin{aligned} max \quad & M_v(\mathcal{P}(r_u)) \\ s.t. \quad & M = M_{v_0} + [N_v]Y \\ & M_v \geq 0, Y \geq 0 \end{aligned} \tag{1}$$

If r_u supports more than one processing route, i.e., more than one part type, we find the maximum marking of the set of part type stages in each route denoted by $M_v(\mathcal{P}(r_u))_{max_j}$. $M_v(\mathcal{P}(r_u))_{max_j}$ can be obtained by solving the linear programming problem in (2).

$$\begin{aligned} max \quad & M_v(\mathcal{P}_j(r_u)) \\ s.t. \quad & M = M_{v_0} + [N_v]Y \\ & M_v \geq 0, Y \geq 0, \end{aligned} \tag{2}$$

where $\mathcal{P}_j(r_u)$ denotes the set of operation stages of part type p_j with processing route $\mathfrak{R}_j$ that require a failure-prone resource r_u now or later in their processing stages. Definition 13 describes the subnet of the proposed switch-buffer controller.

Definition 13: Let (N_s, M_{s_0}) be the switch subnet and (N_b, M_{b_0}) be the buffer subnet for a failure-prone resource r_u with $N_s = (p_{s_u}, t_{j1}, F_s)$ and $N_b = (\{p_{b_{jk}}, p'_{jk}, p_{f_{jk}}, p_{su}, p'_{su}, r_u, v_s\}, \{t'_{jk}, t'_{j(k+1)}\}, F_b)$, where p'_{jk} represents the operation stage p_{jk} at the decentralized buffer slot, t'_{jk} and $t'_{j(k+1)}$ are the pre and post transitions of p'_{jk}, respectively. r_u is the unreliable resource. The marking of $p_{b_{jk}}$ represents the capacity (or units) of the buffer, $p_{f_{jk}} \in H(r_u)$, $t_{j1} \in p_j^\bullet$, and v_S is the control place that r_u, p_{jk} are associated with, i.e., $p_{jk} \in \mathcal{P}(v_S)$, p_{s_u}, if marked, denotes the operation of the server of r_u, and p'_{s_u}, if marked, represents the failure of the r_u's server. The transition flow of the switch subnet is $F_s = \{(p_{su}, t_{j1}), (t_{j1}, p_{su})\}$, and that of the buffer subnet is $F_b = \{(t'_{jk}, p'_{jk}), (p'_{jk}, t'_{j(k+1)}), (t'_{j(k+1)}, p_{b_{jk}}), (p_{b_{jk}}, t'_{jk}), (r_u, t'_{j(k+1)}), (t'_{jk}, r_u), (v_S, t'_{j(k+1)}), (t'_{jk}, v_S), (p_{f_{jk}}, t'_{jk}), (t'_{j(k+1)}, p_{f_{jk}}), (t'_{jk}, p'_{su}), (p'_{su}, t'_{jk}), (t'_{j(k+1)}, p_{su}), (p_{su}, t'_{j(k+1)})\}$. ◇

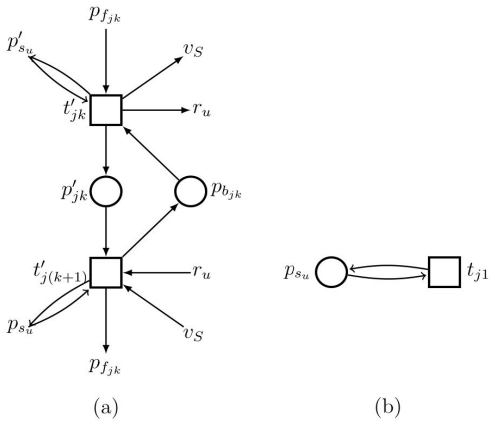

FIGURE 5. Switch-buffer controller subnets: (a) Buffer subnet (b) switch subnet.

Fig. 5 depicts the switch-buffer controller subnets. For a US^{3}PR with added control places to prevent siphons from emptying (N_v, M_{v_0}), if a buffer-switch controller is designed and added to the system to prevent deadlocks/blockages in the event of failure-prone resource failure, the final controlled net system, which is the composition of (N_v, M_{v_0}), switch subnet (N_s, M_{s_0}) and buffer subnet (N_b, M_{b_0}), is denoted by (N_C, M_{C_0}).

Given a US^{3}PR with a failure-prone resources, we compute the elementary siphons in the system and add monitors in the form of places to prevent the siphons from becoming unmarked. For any failure-prone resource r_u in the system, we check if there exist r_i and $p_{vw}, p_{xy} \in P_A$ such that $p_{vw} \in \mathcal{P}(r_u)$, $p_{xy} \notin \mathcal{P}(r_u)$ and $p_{vw}, p_{xy} \in H(r_i)$. If this holds true, then we compute $M_v(\mathcal{P}(r_u)_{max_v}$. If $M_v(\mathcal{P}(r_u))_{max_v} > M_{v_0}(\mathcal{F}(r_u))$, it indicates that, in the event of the failure of r_u, p_{vw} could block p_{xy} from being processed by r_i. As a result, we add a switch-buffer controller according to Definition 13. The initial marking of $p_{b_{jk}}$, $M_{b_0}(p_{b_{jk}})$, is determined by (3).

$$M_{b_0}(p_{b_{jk}}) = M_v(\mathcal{P}(r_u))_{max_j} - M_{v_0}(\mathcal{F}(r_u)) \tag{3}$$

$M_{b_0}(p_{b_{jk}})$ represents the minimum capacity or units of buffers required to be added in order to avoid blockage in a system. Here, we utilize two types of buffer spaces to store the part type stages that require a failure-prone resource r_u in the event of its failure. The first type is the buffer spaces of the resources in $\mathcal{F}(r_u)$. If the capacity of $\mathcal{F}(r_u)$'s buffer spaces cannot hold all the part type stages $p_{vw} \in \mathcal{P}(r_u)$, then the second type of buffer spaces, which is the additional or extra buffer spaces of the switch-buffer controllers, can be utilized. This leads to the following result.

Proposition 2: Let (N_v, M_{v_0}) be a marked US^{3}PR net system with added control places to prevent siphons from being emptied. Let r_u be a failure-prone resource, v_S be a control place added to the system and let there exist r_i and $p_{vw}, p_{xy} \in P_A$ such that $p_{vw} \in \mathcal{P}(r_u)$, $p_{xy} \notin \mathcal{P}(r_u)$ and $p_{vw}, p_{xy} \in H(r_i)$. If $M_v(\mathcal{P}(r_u))_{max_v} > M_{v_0}(\mathcal{F}(r_u))$, a switch-buffer controller added for r_u with $M_{b_0}(p_{b_{jk}}) = M_v(\mathcal{P}(r_u))_{max_j} - M_{v_0}(\mathcal{F}(r_u))$ ensures that $p_{vw} \in \mathcal{P}(r_u)$ does not block $p_{xy} \notin \mathcal{P}(r_u)$ from being processed by r_i. ◇

Similarly, if there exist v_S and $p_{uv}, p_{xy} \in \mathcal{P}(v_S)$ such that $p_{vw} \in \mathcal{P}(r_u)$ and $p_{xy} \notin \mathcal{P}(r_u)$. We then check if $M_v(\mathcal{P}(r_u) \cap \mathcal{P}(v_S))_{max} = M_{v_0}(v_S)$. If this holds true, it signifies that v_S could become unmarked if r_u fails. Then we design a switch-buffer controller based on 13 and the initial marking of $p_{b_{jk}}$ $M_{b_0}(p_{b_{jk}})$ is determined by (4).

$$0 < M_{b_0}(p_{b_{jk}}) \leq M_{v_0}(v_S) \tag{4}$$

For an US^{3}PR, if $M_v(v_S) > 0$ at any marking $M \in R(N_v, M_{v0})$, v_S never becomes empty. Therefore, deadlock/blockage cannot occur. To use a switch-buffer controller as a mechanism to remark v_S, the initial marking of $p_{b_{jk}}$ must be within the range specified in (4). This analysis leads to the following proposition.

Proposition 3: Let (N_v, M_{v_0}) be a marked US^{3}PR net system with added control places to prevent siphons from being emptied. Let r_u be a failure-prone resource, v_S be a control place added to the system and let there exist $p_{uv}, p_{xy} \in \mathcal{P}(v_S)$ such that $p_{vw} \in \mathcal{P}(r_u)$ and $p_{xy} \notin \mathcal{P}(r_u)$. If $M_v(\mathcal{P}(r_u) \cap \mathcal{P}(v_S))_{max} = M_{v_0}(v_S)$ holds, a switch-buffer controller added for r_u with $0 < M_{b_0}(p_{b_{jk}}) \leq M_{v_0}(v_S)$ can prevent v_S from becoming permanently unmarked in the event of r_u failure by ensuring that $M_{v_0}(v_S) > 0$. ◇

Remark 1: Setting $M_{b_0}(p_{b_{jk}}) = 1$ allows only one part type stage $p_{xy} \notin \mathcal{P}(r_u)$ to be processed at a time if r_u fails. Likewise, setting $M_0(p_{b_{jk}}) > 1$, as long as $M_{b_0}(p_{b_{jk}}) \leq M_{v_0}(v_S)$, will correspondingly increase the number of part type stages $p_{xy} \notin \mathcal{P}(r_u)$ that will be processed at a time if r_u fails. However, setting $M_{b_0}(p_{b_{jk}}) > 1$ requires increase in the capacity of the additional buffers (number of buffer slots) for the switch-buffer controller, which in turn increases the cost of implementing the switch-buffer controller. Therefore, increasing the number of part type stages $p_{xy} \notin \mathcal{P}(r_u)$ to be processed simultaneously when r_u fails comes at a cost. As a result, it is important to consider, in the course of implementing the supervisory controller, the trade-off between having a significant number of part types $p_{xy} \notin \mathcal{P}(r_u)$ being processed simultaneously in the event of r_u failure and the cost of the additional buffer units. ◇

In this study's method of designing switch-buffer controllers for a US^{3}R to deal with failure-prone resource failure and its concomitant deadlock/blockage could lead to computing identical switch-buffer controllers. Such a situation makes some switch-buffer controllers redundant. For instance, suppose we first compute a switch-buffer controller for a failure-prone resource r_u with a set of operation stages $\mathcal{P}(r_u)$ as result of $M_v(\mathcal{P}(r_u))_{max_j} > M_{v_0}(\mathcal{F}(r_u))$. If there exist $p_{uv}, p_{xy} \in \mathcal{P}(v_S)$ such that $p_{vw} \in \mathcal{P}(r_u)$, $p_{xy} \notin \mathcal{P}(r_u)$ and $\mathcal{P}(r_u) \cap \mathcal{P}(v_S) = \mathcal{P}(r_u)$, then the switch-buffer controller computed due to $M_v(\mathcal{P}(r_u) \cap \mathcal{P}(v_S))_{max} = M_{v_0}(v_S)$ will be identical to the one designed as a result of $M_v(\mathcal{P}(r_u))_{max_j} > M_{v_0}(\mathcal{F}(r_u))$. Thus, the second switch-buffer controller computed (due to $M_v(\mathcal{P}(r_u) \cap \mathcal{P}(v_S))_{max} = M_{v_0}(v_S)$) becomes redundant since both controllers will have the same structure and connected to the same place $p_{f_{jk}} \in H(r_u)$. The following Proposition immediately follows.

Proposition 4: Let (N_v, M_{v_0}) be a marked US^{3}PR net system with added control places to prevent siphons from being emptied. Let r_u be a failure-prone resource and v_S be a control place added to the system. Suppose that a switch-buffer controller is computed for r_u with a set of operation stages $\mathcal{P}(r_u)$ as result of $M_v(\mathcal{P}(r_u))_{max_j} > M_{v_0}(\mathcal{F}(r_u))$. If there exist $p_{uv}, p_{xy} \in \mathcal{P}(v_S)$ such that $p_{vw} \in \mathcal{P}(r_u)$, $p_{xy} \notin \mathcal{P}(r_u)$ and $\mathcal{P}(r_u) \cap \mathcal{P}(v_S) = \mathcal{P}(r_u)$, then the switch-buffer controller computed due to $M_v(\mathcal{P}(r_u) \cap \mathcal{P}(v_S))_{max} = M_{v_0}(v_S)$ is redundant. ◇

Algorithm 1 describes the proposed robust supervisory control policy of this study.

Theorem 3: Algorithm 1 outputs a robust supervisory control to failures of failure-prone resources of a US^{3}PR model of a single-unit resource allocation system.

Proof: Algorithm 1 is executed in two stages of four steps. The first stage covers Steps 1 and 2, which is the design of monitors to prevent siphons from becoming unmarked. Given a failure-prone US^{3}PR, without resource failure, it is has been established that executing Steps 1 and 2 of the algorithm can guarantee deadlock-free operation in the normal operational mode of the system. This follows immediately from Proposition 1, Property 3 and Theorems 2. The second stage is executed in Step 3, which involves designing switch-buffer controllers to prevent deadlock/blockage as a result of the failure of a failure-prone resources based on Propositions 2 and 3, Definition 13, Eq. (3) and (4). Step 4 is the addition of the computed switch-buffer controllers and outputting the resultant controlled net system (N_C, M_{C_0}). Since Algorithm 1 can guarantee deadlock-free operation in the system without resource failure, and also can guarantee deadlock/blockage-free operation in the event of a failure-prone resource failure, we can conclude that the algorithm is robust to failures of failure-prone resources of a US^{3}PR model of a single-unit resource allocation system. ◇

Example 4: In the PN model in Fig 4, $\mathcal{F}(r_3) = \{r_2, r_3\}$ and $\mathcal{P}(r_3) = \{p_{21}, p_{22}, p_{23}, p_{41}\}$. The failure-prone resource

Algorithm 1 Robust Supervisory Control for US^{3}PR (N_{u}, M_{u_0})

Input: US^{3}PR (N_{u}, M_{u_0})
Output: Robust controlled system (N_{C}, M_{C_0})
Begin{
Step 1: Find the sets of elementary siphons Π_E and
dependent siphons Π_D
Step 2: Compute monitors v_S for the elementary siphons
Step 3: Find r_u and compute the set P_R^u.
for every r_u **do**
  if there exist r_i and $p_{vw}, p_{xy} \in P_A$ s.t $p_{vw} \in \mathcal{P}(r_u)$, $p_{xy} \notin \mathcal{P}(r_u)$ and $p_{vw}, p_{xy} \in H(r_i)$ **then**
    Compute $M_v(\mathcal{P}(r_u))_{max_v}$ based on (2)
    if $M_v(\mathcal{P}(r_u))_{max_v} > M_{v_0}(\mathcal{F}(r_u))$ **then**
      Design a switch-buffer controller according
      to Definition 13 and set $M_{b_0}(p_{b_{jk}})$ based on (3)
    end if
    if there exist v_S and $p_{uv}, p_{xy} \in \mathcal{P}(v_S)$ s.t $p_{vw} \in \mathcal{P}(r_u)$ and $p_{xy} \notin \mathcal{P}(r_u)$ **then**
      if $M_v(\mathcal{P}(r_u) \cap \mathcal{P}(v_S))_{max} = M_{v_0}(v_S)$ and the computed controller is not redundant based on Proposition 4 **then**
        Design a switch-buffer controller accor-
        ding to Definition 13 and set $M_{b_0}(p_{b_{jk}})$
        based on (4)
      else
        Exit
      end if
    end if
  else
    if there exist v_S and $p_{uv}, p_{xy} \in \mathcal{P}(v_S)$ s.t $p_{vw} \in \mathcal{P}(r_u)$ and $p_{xy} \notin \mathcal{P}(r_u)$ **then**
      if $M_v(\mathcal{P}(r_u) \cap \mathcal{P}(v_S))_{max} = M_{v_0}(v_S)$ and the computed controller is not redundant based on Proposition 4 **then**
        Design a switch-buffer controller accor-
        ding to Definition 13 and set $M_{b_0}(p_{b_{jk}})$
        based on (4)
      else
        Exit
      end if
    else
      Exit
    end if
  end if
end for
Step 4: Add the buffer-switch controllers designed to
(N_v, M_{v_0})
Step 5: **Output** (N_{C}, M_{C_0})

}End of the algorithm

r_3 supports two routes $\mathfrak{R}_2$ and $\mathfrak{R}_4$ with $\mathcal{P}_2(r_3) = \{p_{21}, p_{22}, p_{23}\}$, $\mathcal{P}_4(r_3) = \{p_{41}\}$, $\mathcal{F}_2(r_3) = \{r_2, r_3\}$ and $\mathcal{F}_4(r_3) = \{r_3\}$. We need to investigate if the failure of r_3 could interrupt the processing of p_1 and p_3, which do not require r_3 since we have $p_{11}, p_{21} \in H(r_1)$, $p_{21} \in \mathcal{P}(r_3)$ and $p_{11} \notin \mathcal{P}(r_3)$. Solving the linear programming problem in (2), $M_v(\mathcal{P}(r_3))_{max_2} = 5$ and the marking of the places in $\mathcal{P}_2(r_3)$ is $2p_{21} + p_{22} + 2p_{23}$. $M_{v_0}(\mathcal{F}_2(r_2)) = 3$ and $M_v(\mathcal{P}(r_3))_{max_2} > M_{v_0}(\mathcal{F}_2(r_3))$. Therefore, we design a switch-buffer controller for r_3 according Definition 13. The initial marking of $p_{b_{23}}$, $M_{v_0}(p_{b_{23}}) = M_v(\mathcal{P}(r_3))_{max_2} - M_{v_0}(\mathcal{F}_2(r_3)) = 5 - 3 = 2$. Likewise, on $\mathfrak{R}_4$, $M_v(\mathcal{P}(r_3))_{max_4} = 2$ and $M_{v_0}(\mathcal{F}_4(r_3)) = M_{v_0}(r_3) = 2$. Therefore, $M_v(\mathcal{P}(r_3))_{max_4} = M_{v_0}(\mathcal{F}_4(r_3))$, and no switch-buffer controller is required.

There is only one control place v_{S_1} in the system. We need to verify if the failure of r_3 could make v_{S_1} unmarked since we have $p_{31}, p_{41}, p_{42} \in \mathcal{P}(v_{S_1})$, $p_{41} \in (\mathcal{P}(r_3) \cap \mathcal{P}(v_{S_1}))$ and $p_{31}, p_{42} \notin \mathcal{P}(r_3)$. Therefore, we check whether v_{S_1} may become unmarked when r_3 fails. This requires us to check the maximum possible marking of the set $\mathcal{P}(r_3) \cap \mathcal{P}(v_{S_1})$, i.e., $M_v(p_{41})_{max}$. By solving the linear integer programming we find $M_v(p_{41})_{max} = 2$. Since $M_{v_0}(v_{S_1}) = 2$, then $M_v(p_{41})_{max} = M_{v_0}(v_{S_1})$. As a result, v_{S_1} could become unmarked if r_4 fails. Thus, we design a switch-buffer controller based on Definition 13 with $M_{b_0}(p_{b_{41}}) = 1$ according to (4). Fig. 6 shows the switch-buffer controllers designed for the system in Fig 4. The resultant net achieves, without resource failure, 1,848 reachable states, which is 80.8% of the 2,304 legal states. ◇

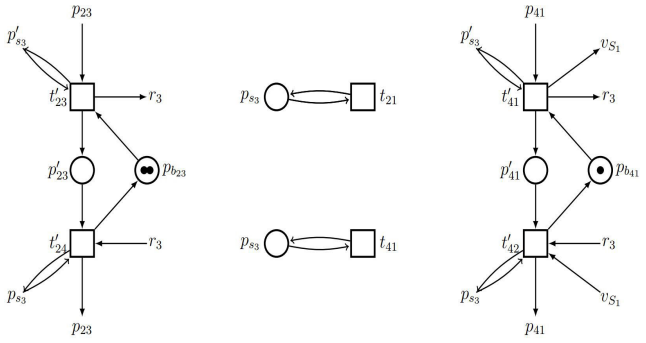

FIGURE 6. Switch-buffer controllers for the unreliable PN model in Fig 4.

B. ROBUST SUPERVISORY CONTROL FOR US^{3}PR WITH MULTIPLE FAILURE-PRONE RESOURCES

In this section, we present a PN model with multiple failure-prone resources in which a part type may require multiple failure-prone resources in its processing route. The proposed robust supervisory control policy is applied to the PN model in order to demonstrate its applicability on such a net system. We use the AMS in Fig. 7, which is first reported in [23]. The uncontrolled net system has, in its normal operating mode (i.e., without resource failure), 233,496 reachable markings of which 199,584 are live markings and 33,912 dead markings.

Example 5: Consider the unreliable US^{3}PR PN model in Fig. 7. The net system has nine resources P_R =

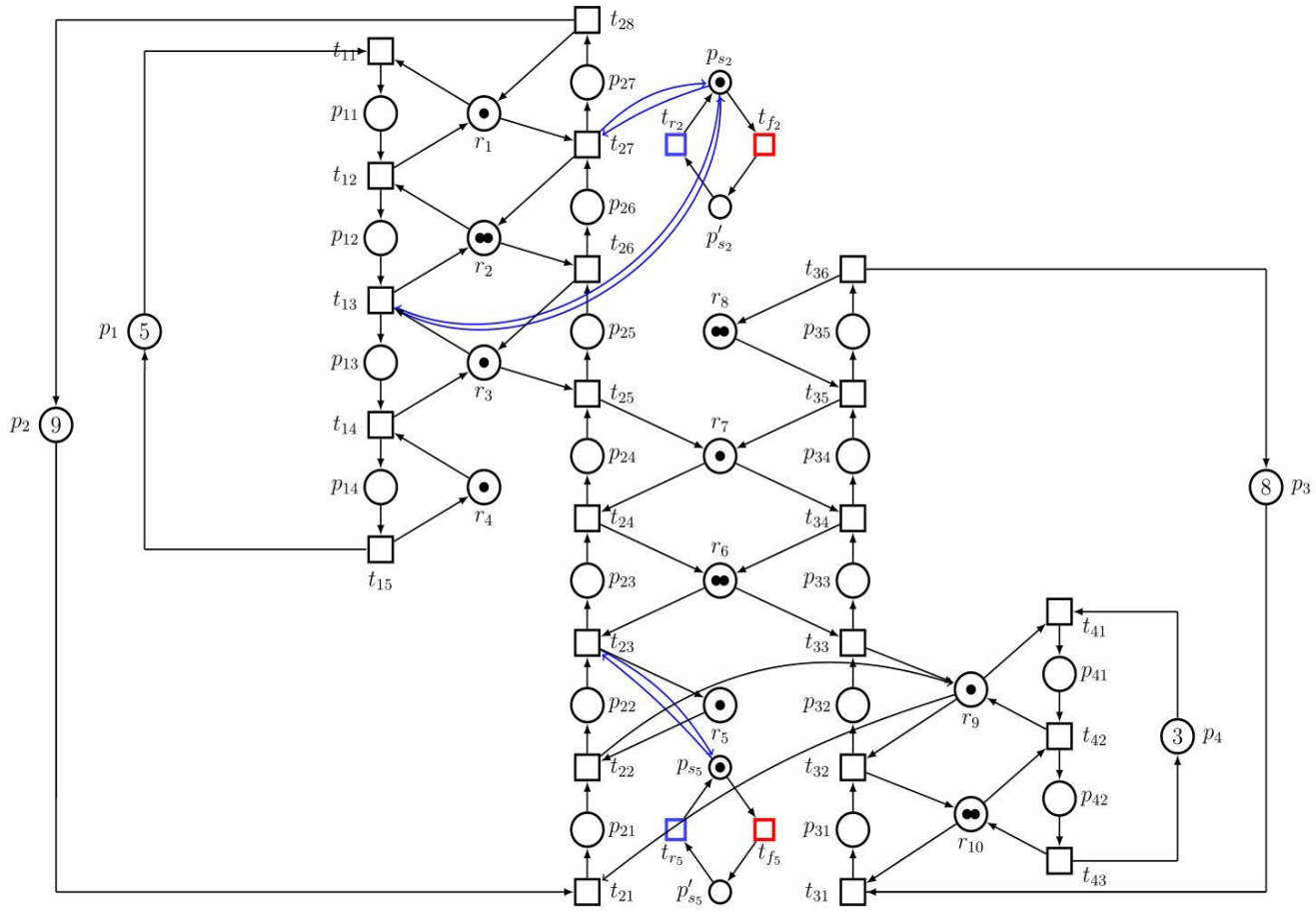

FIGURE 7. PN model with multiple failure-prone resources [23].

$\{r_1, r_2, \ldots, r_9\}$ with two failure-prone resources whose set $P_R^u = \{r_2, r_5\}$ and the set of reliable resources $P_R^u = \{r_1, r_3, r_4, r_6, \ldots, r_9\}$. The AMS has four SMSs $S_1 = \{p_{32}, p_{42}, r_9, r_{10}\}$, $S_2 = \{p_{12}, p_{27}, r_1, r_2\}$, $S_3 = \{p_{13}, p_{26}, r_2, r_3\}$, $S_4 = \{p_{13}, p_{27}, r_1, r_2, r_3\}$ The complementary sets of places of S_1, S_2, S_3 and S_4 are $[S_1] = \{p_{31}, p_{41}\}$, $[S_2] = \{p_{11}, p_{26}\}$, $[S_3] = \{p_{12}, p_{25}\}$ and $[S_4] = \{p_{11}, p_{12}, p_{25}, p_{26}\}$, respectively. We use the steps laid down in Algorithm 1 to design the robust supervisory control for the AMS. To design monitors to prevent siphons from becoming unmarked we execute the first two steps of the algorithm as follows.

According to Property 1, $\eta_1 = -t_{31} + t_{32} - t_{41} + t_{42}$, $\eta_2 = -t_{11} + t_{12} - t_{26} + t_{27}$, $\eta_3 = -t_{12} + t_{13} - t_{25} + t_{26}$, and $\eta_4 = -t_{11} + t_{13} - t_{25} + t_{27}$. It is verified that $\eta_4 = \eta_2 + \eta_3$. Therefore, based on Property 2, S_4 is a strongly dependent siphon and S_1, S_2 and S_3 are elementary siphons. As a result, $\Pi_E = \{S_1, S_2, S_3\}$, $\Pi_D = \{S_4\}$, and $M_{v_0}(v_{S_1}) = M_{v_0}(v_{S_2}) = M_{v_0}(v_{S_3}) = 2$ based on Proposition 1. The corresponding sets of input and output arcs of the control places are ${}^\bullet v_{S_1} = \{t_{32}, t_{42}\}$, ${}^\bullet v_{S_2} = \{t_{12}, t_{27}\}$, ${}^\bullet v_{S_3} = \{t_{13}, t_{26}\}$, $v_{S_1}^\bullet = \{t_{31}, t_{41}\}$, $v_{S_2}^\bullet = \{t_{11}, t_{21}\}$, and $v_{S_3}^\bullet = \{t_{11}, t_{21}\}$. Fig 8(a) visualizes the control places v_{S_1}, v_{S_2} and v_{S_3}.

Now, we design switch-buffer controllers for the failure-prone resource, r_2 and r_5 in the system. It can be verified that $\mathcal{P}(r_2) = \{p_{11}, p_{12}, p_{21}, \ldots, p_{26}\}$, $\mathcal{F}(r_2) = \{r_2\}$, $\mathcal{P}(r_5) = \{p_{21}, p_{22}\}$ and $\mathcal{F}(r_5) = \{r_5\}$. If we consider the failure-prone resource r_2, it supports two processing routes $\mathfrak{R}_1$ and $\mathfrak{R}_2$. Thus, we have $\mathcal{P}_1(r_2) = \{p_{11}, p_{12}\}$, $\mathcal{P}_2(r_2) = \{p_{21}, \ldots, p_{26}\}$, and $\mathcal{F}_1(r_2) = \mathcal{F}_2(r_2) = \{r_2\}$. By solving the linear programming problem in (2), $M_v(\mathcal{P}(r_2))_{max_1} = 2$. This implies that $M_v(\mathcal{P}(r_2))_{max_1} = M_{v_0}(\mathcal{F}_1(r_2))$ since $M_{v_0}(\mathcal{F}_1(r_2)) = 2$. Therefore, according to Proposition 2 no switch-buffer subnet is required as all the part type stages on $\mathfrak{R}_1$ can be stored in the buffer spaces of $\mathcal{F}_1(r_2)$ if r_2 fails. Following the same procedure we find that $M_v(\mathcal{P}(r_2))_{max_2} = 2$, and $M_v(\mathcal{P}(r_2))_{max_2} = M_{v_0}(\mathcal{F}_2(r_2))$. Hence, no switch-buffer controller is required for r_2 on $\mathfrak{R}_2$.

The failure-prone resource r_5 supports only the processing route $\mathfrak{R}_2$ of part type p_2 with $\mathcal{F}_2(r_5) = \mathcal{F}(r_5) = \{r_5\}$. Therefore, solving the linear programming problem in (2) yields $M_v(\mathcal{P}(r_5))_{max_2} = 2$ and $M_{v_0}(\mathcal{F}(r_5)) = 1$. Thus, $M_v(\mathcal{P}(r_5))_{max_2} > M_{v_0}(\mathcal{F}(r_5))$. Therefore, a switch-buffer controller is required for r_5 on $\mathfrak{R}_2$ with $M_{b_0}(p_{b_{22}}) = M_v(\mathcal{P}(r_5))_{max_2} - M_{v_0}(\mathcal{F}(r_5)) = 1$.

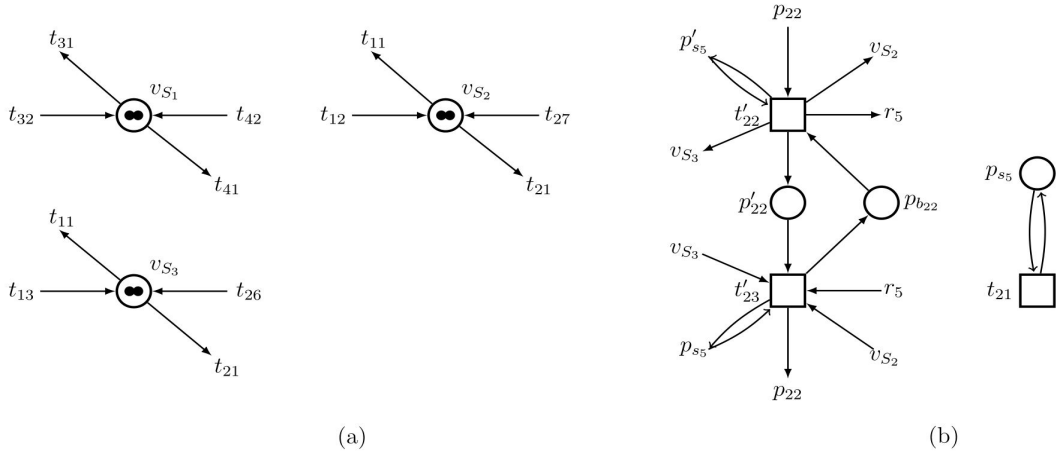

FIGURE 8. Controllers designed for the PN model in Fig 7: (a) Monitor part of the controller, (b) Switch-buffer controller.

Next, we check the possibility of control places becoming permanently unmarked in the AMS as a result of the failure of a failure-prone resource. We have $\mathcal{P}(r_2) \cap \mathcal{P}(v_{S_1}) = \emptyset$, and $\mathcal{P}(r_5) \cap \mathcal{P}(v_{S_1}) = \emptyset$. Therefore, failure of r_2 or r_5 does not make v_{S_1} permanently unmarked. Thus, no switch-buffer controllers are needed. Similarly, $\mathcal{P}(r_2) \cap \mathcal{P}(v_{S_2}) = \{p_{11}, p_{21}, \ldots, p_{26}\}$ and $\mathcal{P}(r_2) \cap \mathcal{P}(v_{S_3}) = \{p_{11}, p_{12}, p_{21}, \ldots, p_{25}\}$. It is obvious that $\mathcal{P}(v_{S_2}), \mathcal{P}(v_{S_3}) \subseteq \mathcal{P}(r_2)$. Therefore, there is no operation stage $p_{jk} \in \mathcal{P}(v_{S_2})$ or $p_{jk} \in \mathcal{P}(v_{S_3})$ that does not belong to $\mathcal{P}(r_2)$. Consequently, no switch-buffer controller is required for remarking v_{S_2} or v_{S_3} if r_2 fails.

Since r_5 supports only $\mathfrak{R}_2$ and there are operation stages on $\mathfrak{R}_2$ that belong to $\mathcal{P}(v_{S_2})$ and $\mathcal{P}(v_{S_3})$, we need to verify whether the failure of r_5 can make v_{S_2} or v_{S_3} to become unmarked or not. Similarly, $\mathcal{P}(r_5) \cap \mathcal{P}(v_{S_2}) = \{p_{21}, p_{22}\}$ and $\mathcal{P}(r_5) \cap \mathcal{P}(v_{S_3}) = \{p_{21}, p_{22}\}$. Obviously, $\mathcal{P}(r_5) \cap \mathcal{P}(v_{S_2}) = \mathcal{P}(r_5) \cap \mathcal{P}(v_{S_3}) = \mathcal{P}(r_5)$. Therefore, these two computed switch-buffer controllers are redundant since we have already computed the same switch-buffer controller due to $M_v(\mathcal{P}(r_5))_{max_2} > M_{v_0}(\mathcal{F}(r_5))$. Hence, no additional switch-buffer controller is needed according to Proposition 4. The switch-buffer controllers designed for the system is depicted in Fig 8(b). The controlled net system is live. ◇

The last part of Example 5 demonstrates further how Proposition 4 can be used to eliminate the possibility of having redundant switch-buffer controllers. We can observe this in the case where we have already found, by solving the linear programming problem in (2), $M_v(\mathcal{P}(r_5))_{max_2} = 2$, $M_{v_0}(v_{S_2}) = 2$ and $M_0(v_{S_3}) = 2$. If we add switch-buffer controllers as a result of $M_v(\mathcal{P}(r_5))_{max_2} = M_{v_0}(v_{S_2})$ and $M_v(\mathcal{P}(r_5))_{max_2} = M_{v_0}(v_{S_3})$ with capacity $M_{v_0}(p_{b_{22}}) = 1$, the controllers will have the same structure and they will be connected at p_{22}, which is the same point the first switch-buffer controller designed earlier for $M_v(\mathcal{P}(r_5))_{max_2} > M_{v_0}(\mathcal{F}(r_5))$ is connected. Therefore, adding two more controllers will be a duplicate and create redundant controllers. Thus, one controller is enough to remark v_{S_2} and v_{S_3} to keep part type p_1 on $\mathfrak{R}_1$ continuously processed and relocate p_{21} to the additional buffer in order to allow p_{32} and p_{41} to be processed by r_9 in the event of r_5 failure.

V. RESULTS AND COMPARISON WITH PREVIOUS WORKS

The studies that consider the same type of SU-RAS as this work are found in [10], [12], [13], [14], [16], [25], [26], [27], and [29]. However, in most of them the systems modeling and supervisory control policies proposed in these studies are in the framework of automata, and are implemented online. Therefore, these policies are deadlock avoidance as opposed to the deadlock prevention proposed in this paper. So far as we know, the majority of the deadlock avoidance policies are based on automata. However, few studies [22], [28], [29] utilize PN to model AMSs and design robust deadlock avoidance policies for AMSs.

We compare the policy of this paper and those in [11], [13], [23], and [29]. All these policies are based on Petri net framework and are deadlock prevention policies except [13], which is a deadlock avoidance policy and is in the automata framework. However, the constraints formulated to achieve robust control in [13] can also be implemented using Petri nets. For the purpose of comparison, we implement the policies in [13] as deadlock prevention policies, like in this paper and [11], [23], [29], using PNs in which the original unreliable PN model of a plant and controllers are unified as one PN model (resultant controlled PN model). The technique in [11] proposes an idea of single-route neighborhood policy (a modified neighborhood policy) in conjunction with a siphon based method to achieve robust control in an AMS. The method in [23] uses three types of buffer spaces scilicet, buffer spaces of failure-dependent resources, those of sheared resources and buffer spaces borrowed from other workstations if there are available ones. If a failure-prone resource fails, part types requiring the failed resources are advanced and redistributed into these three types of buffer spaces in order to allow other processes to continue. The PN model in Fig. 3 is used for the comparison. In this case, r_2 and

TABLE 1. Comparison between the policy in this paper and others as implemented in the PN model of Fig. 3.

Supervisor	[13]	[29]	[11]	[23]	This paper
No. reachable markings without resource failure	730	1,536	1,540	1,816	1,848
Percentage of reachable legal markings without resource failure	31.68%	66.67%	66.80%	78.88%	80.20%
Can be used for AMSs with multiple unreliable resources	Yes	No	Yes	Yes	Yes
Can handle part types requiring more than one failure-prone resource in their processing routes	No	No	No	No	Yes
Can handle failure of multiple resources at the same time	No	No	Yes	Yes	Yes
Require extra buffers	No	No	No	No	Yes
Complexity	Exponential	Exponential	Exponential	Exponential	Exponential

r_3 are the failure-dependent resources of the failure-prone resource r_3, and r_1 is the shared-resource and also a buffer slot borrower. Resource r_4 is used as the buffer-slot lender. Therefore, their buffer spaces can be used to store part stages p_{21}, p_{22} and p_{23} of p_2. Details on these can be found in [23].

Table 1 sums up the comparison. All the five policies can be applied to the same SU-RAS. We compare the policies in terms of their capacity to handle failure of multiple failure-prone resources, and to handle failure of multiple resources at the same time. However, the policy in [11], [13], [23], and [29] cannot be applied to systems where part types may require multiple failure-prone resources in their processing routes, while the policy in this paper can be applied to such systems. Additionally, the policy in this paper is more permissive than those in [11], [13], [23], and [29] due to utilization of additional buffer units. The supervisory control policy in this paper, without resource failure, achieves 1,848 of the reachable states, which is 80.20% of the reachable legal states of the net system. The technique in [13] allows 730 states to be reached, which is 31.68% of the total legal states. The policy in [29] achieves 1,536 of the legal states, which is 66.67% of the legal states. The policy in [11] achieves 1,664, which is 66.80% of the legal reachable states, while [23] achieves 1,816 states, which is 78.80% of the legal reachable states. Based on the comparison in Table 1, the proposed technique has more advantages over those in [11], [13], [23], and [29]. The only advantage these four techniques have over the technique in this paper is the requirement of extra buffers by the technique in this paper.

VI. CONCLUSION

This paper introduces a novel supervisory control framework for AMSs that utilizes decentralized buffers and switch to form a switch-buffer controller. The proposed approach integrates resource failure detection and diagnosis techniques to identify and isolate resource failures from affecting other parts that do not require the failed resource. By providing switch-buffer controllers, the decentralized switch-buffer mechanism improves system robustness, allowing for continued operation in the event of a resource failure. To validate the efficacy of the proposed approach, AMSs are used as case studies to demonstrate its application. Results indicate that the proposed approach effectively enhances the reliability and efficiency of AMSs. Future work will focus on optimizing the proposed concept in order to enhance the acceptability of the concepts. The scalability of the proposed approach to more complex manufacturing systems, such as more generalized sub-classes of PN models, will also be investigated.

REFERENCES

[1] F. Basile, P. Chiacchio, and A. Giua, "An optimization approach to Petri net monitor design," *IEEE Trans. Autom. Control*, vol. 52, no. 2, pp. 306–311, Feb. 2007.
[2] Z. Li, G. Liu, H.-M. Hanisch, and M. Zhou, "Deadlock prevention based on structure reuse of Petri net supervisors for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 1, pp. 178–191, Jan. 2012.
[3] Y. Chen, Z. Li, and M. Zhou, "Behaviorally optimal and structurally simple liveness-enforcing supervisors of flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 3, pp. 615–629, May 2012.
[4] K. Barkaoui and J. F. Pradat-Peyre, *On Liveness and Controlled Siphons in Petri Nets* (Lecture Notes in Computer Science), vol. 1091. Berlin, Germany: Springer, 1996, pp. 57–72.
[5] G. Y. Liu, Z. W. Li, K. Barkaoui, and A. M. Al-Ahmari, "Robustness of deadlock control for a class of Petri nets with unreliable resources," *Inf. Sci.*, vol. 235, pp. 259–279, Jun. 2013.
[6] Y. Wu, K. Xing, J. Luo, and Y. Feng, "Robust deadlock control for automated manufacturing systems with an unreliable resource," *Inf. Sci.*, vols. 346–347, pp. 17–28, Jun. 2016.
[7] N. Du and H. Hu, "A robust prevention method for automated manufacturing systems with unreliable resources using Petri nets," *IEEE Access*, vol. 6, pp. 78598–78608, 2018.
[8] Z. L. Zhang, G. Y. Liu, and Y. Sun, "Design of robust optimization Petri nets controller for automated manufacturing systems with unreliable resources," in *Proc. 18th Int. Conf. Autom. Sci. Eng.* Mexico City, Mexico: IEEE, Aug. 2022, pp. 640–645.
[9] Z. Zhang, G. Liu, and Z. Li, "Adaptive supervisory control of automated manufacturing systems with unreliable resources based on smart switch controllers," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 4, pp. 5445–5456, Oct. 2024.
[10] S. Foh Chew and M. A. Lawley, "Robust supervisory control for production systems with multiple resource failures," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 3, pp. 309–323, Jul. 2006.
[11] U. S. Abubakar, G. Liu, and M. Uzam, "Petri net-based robust supervisory control of automated manufacturing systems with multiple unreliable resources," *IEEE Access*, vol. 9, pp. 100264–100278, 2021.
[12] H. Yue, K. Xing, and Z. Hu, "Robust supervisory control policy for avoiding deadlock in automated manufacturing systems with unreliable resources," *Int. J. Prod. Res.*, vol. 52, no. 6, pp. 1573–1591, Mar. 2014.
[13] S. Wang, S. Foh Chew, and M. A. Lawley, "Using shared-resource capacity for robust control of failure-prone manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 38, no. 3, pp. 605–627, May 2008.
[14] S. Wang, S. F. Chew, and M. Lawley, "Guidelines for implementing robust supervisors in flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 47, no. 23, pp. 6499–6524, Dec. 2009.
[15] H. Yue, K. Xing, H. Hu, W. Wu, and H. Su, "Resource failure and buffer space allocation control for automated manufacturing systems," *Inf. Sci.*, vol. 450, pp. 392–408, Jun. 2018.

[16] S. Foh Chew, S. Wang, and M. A. Lawley, "Robust supervisory control for product routings with multiple unreliable resources," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 195–200, Jan. 2009.
[17] Z. Zhang, G. Liu, K. Barkaoui, and Z. Li, "Adaptive deadlock control for a class of Petri nets with unreliable resources," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 5, pp. 3113–3125, May 2022.
[18] J. Huang, Q. Chang, and J. Arinez, "Modeling and dynamic assignment of the adaptive buffer spaces in serial production lines," *J. Manuf. Sci. Eng.*, vol. 143, no. 3, Mar. 2021, Art. no. 031005.
[19] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[20] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995.
[21] Z. W. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. London, U.K.: Springer, 2009.
[22] J. Luo, K. Xing, and Y. Wu, "Robust supervisory control policy for automated manufacturing systems with a single unreliable resource," *Trans. Inst. Meas. Control*, vol. 39, no. 6, pp. 793–806, Jun. 2017.
[23] U. S. Abubakar and G. Liu, "Adaptive supervisory control for automated manufacturing systems using borrowed-buffer slots," *Inf. Sci.*, vol. 667, May 2024, Art. no. 120460.
[24] Z. Li and M. Zhou, "Two-stage method for synthesizing liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Trans. Ind. Informat.*, vol. 2, no. 4, pp. 313–325, Nov. 2006.
[25] S. F. Chew, S. Wang, and M. A. Lawley, "Resource failure and blockage control for production systems," *Int. J. Comput. Integr. Manuf.*, vol. 24, no. 3, pp. 229–241, Mar. 2011.
[26] H. Yue and K. Xing, "Robust supervisory control for avoiding deadlocks in automated manufacturing systems with one specified unreliable resource," *Trans. Inst. Meas. Control*, vol. 36, no. 4, pp. 435–444, Jun. 2014.
[27] H. Yue, K. Xing, H. Hu, W. Wu, and H. Su, "Robust supervision using shared-buffers in automated manufacturing systems with unreliable resources," *Comput. Ind. Eng.*, vol. 83, pp. 139–150, May 2015.
[28] X. Wang and H. Hu, "A robust control approach to automated manufacturing systems allowing multitype and multiquantity of resources with Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3499–3514, Oct. 2020.
[29] F. Wang, K.-Y. Xing, M.-C. Zhou, X.-P. Xu, and L.-B. Han, "A robust deadlock prevention control for automated manufacturing systems with unreliable resources," *Inf. Sci.*, vol. 345, pp. 243–256, Jun. 2016.

UMAR SULEIMAN ABUBAKAR received the B.Eng. degree in electrical engineering from Bayero University, Kano, Nigeria, in 2008, the M.Sc. degree in electrical and computer engineering from Meliksah University, Kayseri, Türkiye, in 2014, and the Ph.D. degree in control theory and control engineering from Xidian University, Xi'an, China, in 2022. He is currently a Postdoctoral Research Fellow with the Department of Mechanical Engineering, Chulalongkorn University, Thailand. His research interests include Petri net theory and applications, robust/adaptive supervisory control of automated manufacturing systems, and application statistical learning in autonomous multi-agent systems.

AHMAD BALA ALHASSAN (Member, IEEE) received the B.Eng. degree in electrical engineering from Bayero University, Kano, Nigeria, in 2011, the M.Eng. degree in mechatronics and automatic control from the University of Technology Malaysia (UTM), in 2016, and the Ph.D. degree in mechanical engineering from Xi'an Jiaotong University, China, in 2022. He was with the Department of Mechanical Engineering, Chulalongkorn University, Thailand, as a Postdoctoral Researcher, in 2023. He is currently with the Department of Robotics and Mechatronics, Nazarbayev University, Kazakhstan. His current research interests include modeling, simulation, and control of mechatronic systems. He has authored or co-authored many research articles and international conference papers on dynamic analysis and control of crane systems, rehabilitation robots, elderly-assistant robots, power transmission line inspection robots, and wind energy conversion systems. He has been a member of IEEE, including the IEEE Young Professionals, the IEEE Control Systems Society, and the IEEE Industrial Electronics Society. He was a keynote speaker at the 5th International Conference on Intelligent Science and Technology (ICIST 2023), Lanzhou, China. He has also served as a Reviewer for many refereed journals, including IEEE Access, *Journal of Mechanical Science and Technology*, and *Robotics and Autonomous Systems*.

GRIDSADA PHANOMCHOENG received the B.S. degree from Chulalongkorn University, Bangkok, Thailand, in 2002, and the M.S. degree in aerospace engineering and mechanics and the Ph.D. degree in control science and dynamical systems from the University of Minnesota, Twin Cities, Minneapolis, MN, USA, in 2007 and 2011, respectively. In 2012, he was a Postdoctoral Researcher with the Dr. Rajamani's Mechanical Engineering Laboratory, University of Minnesota, Twin Cities. Currently, he is a Faculty Member of mechanical engineering with Chulalongkorn University. His research interests include advanced control system design, observer design for nonlinear systems, system identification, applications to automotive systems, energy harvesting, and machine vision.