Deadlock Handling For Distributed Workflows

Zbigniew A. Banaszak¹ &
Mowafak H. Abdul-Hussin²

Received on: 7/1/2001
Accepted on: 13/5/2002

Abstract
This paper presents a distributed deadlock avoidance control for Flexible Manufacturing Systems (FMSs). The proposed approach differs from previous works on modelling and control of such systems so that it constructs feasible deadlock avoidance rules by exploiting the information about resource requirements of each particular operation of a manufacturing process and/or the repetitive character of the material and data process flow. The policies proposed are less conservative than existing deadlock avoidance algorithms for computer operating systems because they are based only on resource requirements in the current production route for each job rather than the entire set of resource requirements.

Index Terms: Petri net models, Workflows control, deadlock handling, Flexible Manufacturing System, Distributed control, self-synchronisation

1- Institute of Informatics and Management, Technical University of Zielona Gora, Ul. Podgorna 50, 65-246 Zielona Gora, POLAND.
2- Dept. of Computer & Software Engineering, University of Technology, Tell-Mohammed, Baghdad 12906, IRAQ.
1. INTRODUCTION

Integration of different material, energetic and information processes enables to treat a Flexible Manufacturing System (FMS) as a logistic system. A typical logistic system consists of the flow of goods and services, and the monitoring and control of these flows [11]. Among its activities, the transportation, inventory management, order processing, warehousing, distribution and production are the most characteristic. Coordination of these processes and activities is the main objective of the management of the logistic systems. In other words, the goal is to achieve a well-synchronized behavior of the dynamically interacting components, where the right quantity of the right material is provided in the right place, and at the right time [11]. Among many qualitative properties (such as bottlenecks and starvation) that determine the quality of system behavior the processes blocking (deadlocks) play the most important role.

Unlike the case of qualitative properties, the performance evaluation of the quantitative characteristics is mainly based on the computer simulation techniques. The results of the recent investigations suggest the possibility of the implementation of the two alternative techniques employing the max-algebra [4] and Timed Event-Graph [10] formalism, respectively. These techniques allow to calculate the dynamic parameters of the manufacturing systems under different assumptions, regarding of the initial state and the types of jobs flow.

A main objective of modern control systems is to increase a FMS productivity and reliability [2]. Such a requirement can be satisfied, however, by distributed control systems where decisions are undertaken by local, autonomous controllers the linked each other through a computer network. This approach, in turn, imposes needs for some new distributed control oriented policies, in particular distributed deadlock avoidance procedures. So far developed procedures are aimed at computer/communication networks [1,2], [7], [13], [14] and do not take into account a FMS specifics, e.g., data regarding resources allocation requirements. From another side, however, known methods provide only centralized solutions of the FMS deadlock avoidance problem [1,2,3,5,6,9,12,15].

We shall focus our attention on the deadlock prevention/avoidance problem. The goal of these approaches (prevention and avoidance) to the deadlock problem is to add to the system a control policy preserving the system from deadlock situations. But the way both approaches deal with the problem is different. The deadlock prevention approach establishes the control policy in a static way, so that, once established, we are sure that the system cannot reach undesirable deadlock situations. In [6,7,8,9] different approaches of this kind may be found. The deadlock avoidance approach is different: at each system state, the control policy determines (online) which system evolutions, among the set of feasible ones, are correct. In [3,8,9], solutions of this kind have been adopted.

In our approach we have adopted Petri nets as a tool for modeling of the dynamic behavior of the system. This tool has also been adopted in several
papers related to the study of deadlock problems in FMS environments [1,2,3,4,8,9,15]. For a general class of Petri net models, in [1,12] both prevention and avoidance control policies are proposed. The first one is based on the net reachability graph, while the second one is based on a look-ahead procedure that searches for deadlock situations by simulating the system evolution for a preestablished number of steps. Due to the fact that the avoidance policy does not assure that deadlocks are not reachable, they propose to combine this policy with a deadlock recovery system. In [3] a deadlock avoidance algorithm is proposed for a class of Petri net models formed by a set of sequential processes (without alternatives in its execution) that use a resource in each state. The algorithm controls the input of new tokens in a model "zone", assuring that system evolutions are always possible.

For the same class of models, Jeng, and DiCesare propose in [9] a different deadlock avoidance control policy based on the concept of Minimal Resource Requirement (minimal number of resources assuring the existence of a system evolution that allows to complete all the jobs in the system).

The phenomenon of deadlocks has been extensively studied in the context of computer operating systems and data communication systems [2]. However, the objective of the investigations is still pursued [3], [15]. In general, the methods of deadlock avoidance implemented in computer operating systems assume nothing about the order in which resources are requested or released, and differ in ways of testing algorithms with different computational complexity. An alternative approach takes advantage of the information about the resource use order, which is available in the manufacturing processes. This information allows to develop less restrictive algorithm than existing ones.

Because of the NP-complete character of a deadlock avoidance problem, the computational complexity of an algorithm of checking if a given deadlock avoidance condition is satisfied or not, must have the polynomial character. Therefore, the considered problem can be seen as a task of the development of the satisfactory conditions, implementation, of which will guarantee the deadlock-free execution of the resultant models.

This paper presents a new approach for distributed control of Flexible Manufacturing Systems (FMSs). The proposed approach differs from previous works on modeling and control of such systems [1,3] so that it constructs feasible deadlock avoidance rules by exploiting the information about resource requirements of each particular operation of a manufacturing process and the repetitive character of the material and data process flow. The following distributed control deadlock avoidance policies are: synchronization zones, a Request the Allocation Graph, and a capacity / protocols allocation. The presented policies are distributed oriented (the control is distributed between zones, segments, and locally acting dispatching rules, respectively). However, they do not contain communication protocols (message exchange rules) that allow one to implement the relevant procedures in a network of autonomous controllers. This is still an open problem and should further be investigated.
The following section provides introduction to the concept of distributed control in modern manufacturing systems. A problem statement follows an illustrative example. Section 3, introduces the formalism of Place/Transition nets theory in the extent that is applied further. Section 4, presents the modeling framework employed. In Section 5, the deadlock avoidance conditions providing the formal background for particular control policies are presented. Finally, Section 6 concludes the paper.

2. METHODS OF DEADLOCK HANDLING

Illustrative example. Consider the Flexible Machining Cell (FMC) shown in Fig.1. The FMC consists of machine tools M1 and M2, industrial robots R1 and R2, a gripper changing station equipped with grippers G1 and G2, input storage IB1 and IB2, and output storage OB1 and OB2. A machining cycle is initiated when a raw part is available at the input storage IB1 (IB2).

The robots R (R2) equipped with gripper G1 (G2) picks up a raw part from IB1 (IB2) and loads it on the machine tool M1 (M2) to carry out the machining operation. The machined part is unloaded from the machine M1 (M2) by the robots R (R1) equipped with gripper G2 (G1) and placed in the output storage OB1 (OB2).

Assume the capacity of each resource is equal to one, and the processes specification are given as follows:

\[ P_{R_1} = (O_{1,1}, [R_1, G_1]), \]
\[ (O_{1,2}, [M_1]), (O_{1,3}, [R_2, G_2]) \]

\[ P_{R_2} = (O_{2,1}, [R_2, G_2]), \ldots (I) \]
\[ (O_{2,2}, [M_2]), (O_{2,3}, [R_1, G_1]) \]

where, \((O_{im}, \{RE_i\ j \in J_{im}\})\) means the u-th operation in the i-th production route completion of which requires the subset \(\{RE_j \mid j \in J_{im}\}\) of FMC resources, RC (OT) is the set of Resources Capacity (Operation Times).

As one can see the asynchronous execution of processes can result in their mutual blocking. A possible deadlock situation corresponds to the marking of Petri net Fig. 2(a):

\[ M = (1,1,0,1,1,0,1,1,1,1,1,1) \]

Moreover, from the analysis of the reachability graph, it follows, that either the transition \(t_1\) at the marking \(M = (0,1,0,1,1,0,0,0,1,1,1,1)\) or the transition \(t_2\) at the marking \(M'' = (1,1,0,1,0,1,1,1,0,1,0,1,1,1,1)\) has to be inhibited from in order to avoid the deadlock marking \(M\) occurrence.

Deadlocks handling. A state of the mutual processes deadlock denotes the state of the distribution of non-preemptive resources in the system, where the completion of some processes is impossible because they request non-preemptive resources allocated to other processes (tasks). In general case in order to prevent a deadlock occurrence at least one of the following conditions must not hold:

- mutual exclusion: at any given moment only one process can use a given resource,
- non-preemption (sequencing without disposing): a resource can be released only by the process which first took possession of it,
- hold while waiting (resource stoppage): while waiting for release the resources to be released by other processes a process does not release
the resources allocated to it which is still needs.

- circular wait: there exists a closed, cyclic chain of processes waiting for the release of resources possessed by other processes.

The existing methods of deadlock handling can be classified according to the following approaches:

- deadlock detection and recovery: allows deadlock detection, i.e., determines which processes and resources cause blocking, removes it by the successive reallocation of resource units from the blocked processes, and checks whether that helps to recover the blocking.

- deadlock prevention: uses protocols specifying the ways of requesting resources which make it impossible to satisfy one of the conditions necessary for blocking.

- deadlock avoidance: based on additional a priori information, describes the way of using the resources by each process and allows the selection of resource requests which ensure the system transition from one safe state to another. The methods of deadlock avoidance ensure the highest level of system resource utilization.

3. BASIC PETRI NET DEFINITIONS

A CLASS OF PETRI NET MODELS FOR FMS

For brevity, it is assumed that the reader is familiar with basic Petri net notation, transition firing rules, and basic properties of Petri net models (see [1, 3, 11,13] for a nice recent survey). The purpose of this section is to make some notations precise since they will be extensively used in the sequel.

A class of simple, and safe Place/Transition nets is considered.

**DEFINITION 1**

A Petri net (PN) is a five-tuple, $G = (P,T,I,O,M_0)$, where $P$ is a finite set of places, $T$ is a finite set of transitions, $I : P \times T \rightarrow \{0,1\}$ is the input function that specifies input places of transitions, $O : P \times T \rightarrow \{0,1\}$ is the output function that specifies output places of transitions, and $M_0 : P \rightarrow N$ is the initial marking of the PN, where $N$ is the set of nonnegative integer numbers. The marking of a PN indicates the number of tokens in each place. In the manufacturing domain, tokens can be interpreted as resources or system status. Places, transitions, input functions, and output functions constitute the static structure of a PN, which is a bipartite directed graph.

**DEFINITION 2**

A transition $t \in T$ is enabled in marking $M$ iff $p \in P, M_0(p) \geq I(p, t)$. The firing results in removing one token from each of its input places and adding one token to each of its output places.

A marking $M$ is said to be reachable from $M$ if there exists a firing sequence $\sigma$ from $M$ to $M'$. The above statement can be represented in a concise notation: $M[\sigma] \Rightarrow M'$. In addition, in the subsequent discussions of this paper, we use $R(M)$ to denote all the markings that are reachable from $M$, and $M[\sigma]$ to denote that $\sigma$ is a firing sequence from $M$.

There are many ways of using PNs to represent a shared-resource manufacturing system [3,9,6]. In this method, a shared-resource manufac-
turing system is denoted by the processes that control the manufacturing resources and the interactions among the processes. The resource control processes are formulated according to the allocations and de-allocations of the resources regarding the manufacturing operations. In other words, the related manufacturing operations are incorporated into each resource control process. For example, if in a manufacturing process, only operation 1 and operation 2 need resource A, then only operation 1 and operation 2 are considered in modeling the resource control process for A. By this means the synthesis (design) process can be modularized.

4. DEADLOCK AVOIDANCE

Because the reachability graph searching problem is NP-complete, the computational complexity of an algorithm of checking if a given deadlock avoidance condition is satisfied or not, must have the polynomial character.

4.1 Synchronization Zones Based Policy

The concept considered takes advantage of the production sequence information for each job in the system [2], [3]. All active jobs may compete with each other for resources, but there is difference in the resource conflicts between jobs for the same product and jobs for different products. The production route information can be used to advantage by partitioning the production sequence into subsequences, or zones, which are treated as individual “pipes” in the system. The policy is less conservative than existing deadlock avoidance algorithms for computer operating systems because it is based only on resource requirements in the current production sequence zone for each job rather than the entire set of resource requirements.

Within this framework, taking into account the FMC from Fig. 1, we are looking for some satisfactory conditions so their examination will require a reasonable amount of time and their implementation will result in avoidance of firings, leading to the deadlocks.

Deadlock Avoidance Conditions.

In order to develop such conditions, let us consider the following definition of the resource request graph G,

\[ G = \langle RE, \prec \rangle, \quad \prec \subseteq RE \times RE \]  
(2)

Where \((RE_i, RE_j) \in \prec\) if \(RE_i\) and \(RE_j\) belong of the subsets of the two subsequent elements in a production route, i.e. \(RE_i \in crd^k(\{PR_{a_n}\})\) and \(RE_j \in crd^k(\{PR_{a_n}\})\) hold for \(i \neq j\) and \(n \in \{1, 2, ..., v\}, K \in \{1, 2, ..., k\}\).

The resource request graph obtained for the processes specification (1) is shown in Fig. 2 (b). From the analysis of deadlocks it follows that the system state \(S\) is a state of processes blocking if and only if there exists a cycle in the resource request graph G so that all its resource are allocated (possessed) at this state. Its implementation to the systems where every resource capacity is equal to one, leads to the following rule: The state \(S\) is safe if the condition (3) holds

\[ N(S, G) \leq L(G) - 1 \]  
(3)

Where \(N(S, G)\) is the number of parts processed along the production routes \(PR\) at a state \(S\) which encompasses the current resources allocation, \(L(G)\) is the
length of the shortest elementary cycle in the graph \( G \).

Therefore to avoid deadlocks, no more than \( L(G) - 1 \) processes (parts) can be simultaneously processed in the system. The disadvantage of this solution is the low level of system resources utilization observed, for instance, in processes where two or more disjoint cycles exist.

The new less restrictive conditions sufficient for deadlock avoidance take into account the information about resources use order have the following form:

(i) \( N(S, G_{i,k}) \leq L(G_{i,k}) - 1 \) for \( (G_{i,k} \neq 1 \) or \( N(S, G_{i,k}) \leq 1 \) for \( L(G_{i,k}) = 1 \) and

(ii) \( N(S, Z_{k+1}) \leq L(Z_{k+1}) \), and

(iii) \( N(S, Z_{k+1}) + L(G_{i,k}) \leq L(G) - 1 \), for all zones \( Z_{i,k} \in \{Z_1, Z_2, ..., Z_k\} \) and

\( K \in \{1,2,3\} \)

where \( G_{i,k} \) is the subgraph composed of resources corresponding to the shared zone \( Z_{i,k} \) which precedes the non-shared \( Z_{i+1} \), \( N(S, G_{i,k}) \) is the number of parts processed by shared operations at a state \( S \), shared (non-shared) zone is a sequence of resources that multiple occur (uniquely occur) in production routes of processes specification, \( N(S, Z_{i,k}) \) is the number of parts processed by the operations in the \( Z_{i,k} \) with non-shared zone at a state \( S \), \( L(Z_{i,k}) \) is the number of operations in the non-shared zone \( Z_{i,k} \).

In the general case, the condition (4) is less restrictive than (3) that leads to the better utilization of systems resources. This is because the resources determined in the non-shared zones (see \( N(S, Z_{i+1}) \leq L(Z_{i+1}) \) and some non-shared resources (see \( N(S, G_{i,k}) \leq L(G_{i,k}) \) can be used simultaneously.

4.2 Request /Allocation Graph Based Policy

A Request/Allocation Graph (RAG) concept of synchronization segments provides the local rules for workflows control. The policy requires determining of a set of synchronization segments in a RAG model of a FMS operation. It assumes that working processes require access to an unit of a specific resource to execute particular technological operation. The efficiency of the procedure is higher than the efficiency of preceding policies proposed in [3] and can be as high as the efficiency of the methods presented in [8].

In order to provide the motivation of this approach let us consider the following specification of the system:

\[ R = \{r_1, r_2, ..., r_5\} \] - a set of resources,

\[ C(r_i) = 1, i = 1, ..., 6 \] - capacities of the resources,

\[ BI = \{b_1, b_2\} \] - a set of input buffers,

\[ BO = \{b_1, b_2\} \] - a set of output buffers.

A set of programs: \( SPS = \{PS(1), PS(2)\} \).

A set of Resource Sequences \( SRS = \{RS(1), RS(2)\} \) corresponding to the programs:

\[ RS(1) = b_1, r_1, r_3, r_2, s_1 \] - the resource sequence #1,

\[ RS(2) = b_2, r_2, r_3, r_1, s_2 \] - the resource sequence #2.

A set of jobs (processes) \( P_j = \{p_1, p_2, p_3, p_4\} \), \( P(1) = \{p_1, p_2\} \) - a set of processes that have to execute program \( PS(1) \).

P(2) = \{p_3, p_4\} - a set of processes that have to execute program \( PS(2) \). A RAG corresponding to the SRS is shown in Fig.3.

A set of all cycles in the RAG:

\[ ZL = \{L_i\} \] where \( L_i = \{r_1, r_2, r_3\} \).
The capacity of $L_4$ can be calculated as follows:

$$c(L_4) = c(r_3) + c(r_5) = 3.$$  

The final resources allocation sequence is as follows:

$$S_0 = (p_1, r_1) S_1 = (p_5, r_1) S_2 = (p_2, r_1) S_3 = (p_3, r_2) S_4 = (p_2, r_2) S_5 = S_7 = (p_1, r_1) S_8 = (p_3, r_1) S_9 = (p_3, r_2) S_{10} = S_{11} = ...$$

At the state $S_{11}$, the processes $P_1, P_3$ have finished their programs while the process $P_2$ is using a unit of $r_1$ and the process $P_4$ a unit of $r_2$. Therefore $S_{11}$ is equivalent to $S_1$.

The concept presented requires a determination of all the cycles in a system's RAG model. Since the problem of cycles determination is NP-complete, the calculations are conducted in an off-line mode while controls are undertaken in an on-line mode due to a particular loop analysis. The effectiveness of the resource allocation policy depends on the RAG's structure. In general case if the loops are greater and the set of shared resources is smaller then the procedure accepts more states. In order to implement the policy in a network of real-time controllers one has to design the protocol that distributes information about the allocation of the cycles between the resource servers. This should be done after each modification of the system's state.

4.3 Capacity / Protocols Allocation Based Policy

A capacity/protocols allocation concept of workflows self-synchronization employs the policy that requires determining of the buffers capacity and dispatching rules allocation due to a given set of production routes [11,12]. The buffer capacity assignment conditions as well as conditions imposed on dispatching rules (protocols) allocation guaranteeing cyclic steady-state processes execution (i.e. starvation-free and deadlock-free processes execution) are of primary role. The protocols can be treated as processes coordinating the processes competing for access to relevant shared resources. The resulting system may be considered as a self-synchronized system which is capable of returning to a unique steady state from any state it was forced into as result of an accidental disturbance.

In order to illustrate the approach considered let us concentrate on a set of sequential cyclic processes specified by the following production routes and interacting as shown in Fig.4:

$$P_1 = M_1, M_2, M_3;$$

$$P_2 = M_4, M_5;$$

$$P_3 = M_1, M_4, M_5;$$

$$P_4 = M_3, M_5;$$

$$P_5 = M_1, M_4,$$  \( ... (5) \)

The state

$$S_0 = \begin{bmatrix} M_1 & M_2 & M_3 & M_4 & M_5 & M_6 \\ 1 & 0 & 2 & 0 & 3 \end{bmatrix}$$

encapsuring the initial processes allocation can be directly changed to the deadlock state $S = (1, 3, 2, 0, 2, 0)$. However, using the following set of local dispatching rules (6) the possible deadlock-free execution (see Table 1) can be guaranteed. It means, that there exists an initial state and a set of dispatching rules guaranteeing the deadlock free processes execution.

$$PR_{11} = \{P_1, P_2\},$$
$$PR_{12} = \{P_1, P_3\},$$
$$PR_{13} = \{P_2, P_3\},$$

$$M_1 = \{ \text{the i-th process allocation to the M_1-th resource} \}.$$  \( ... (6) \)

Moreover, assuming allocation of the extra capacity unit either to resource $M_1$ or $M_2$ or $M_3$, the behaviour of
resultant system in deadlock-free for both: any initial state of processes allocation, and any set of dispatching rules allocation (following however so called flow balancing condition [1,12]).

5. CONCLUSION

Three distributed deadlock avoidance control policies are presented. First one takes advantage of the production sequence information for each job in the system. Jobs for the same product use resources in the same sequence. Due to the second policy, an on-line analysis of all possible cycles included in a RAG model of a FMS operation is required.

However, it should be noted that both procedures do not contain communication protocols (message exchange rules) that allow one to implement them in a network of autonomous controllers. This is still an open problem and should further be investigated.

REFERENCES


Notation of FMC resources:
\( R_{E_j} \) - Requires the subset \( \{ R_{E_j} \mid j \in J, m \} \) of FMC resources,
\( R_{C} \) - is the set of Resources Capacity
\( O_T \) - is Operation Times
A Request/Allocation Graph (RAG)
NP-Complete Problems ("Non-Deterministic Polynomial")

Fig.1. A Flexible Machining Cell.
The petri net model corresponding to the specification (1) has a form shown in Fig.2 (a).
Fig. 2 Graphical representation of:
(a) Petri net model; (b) resource precedence graph determined by (1).

Fig. 3. Circles denote resources $r_1, r_2, ..., r_6$. Double circles denote histories $b_1, b_2, b_3, b_4$. 
Fig. 4. System of sequential cyclic processes

Table 1. One of the possible workflows execution

<table>
<thead>
<tr>
<th>Resource</th>
<th>State</th>
<th>S₀</th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
<th>S₅</th>
<th>S₆</th>
<th>S₇</th>
<th>S₈</th>
<th>S₉</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₁</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>M₂</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>M₃</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
</tr>
<tr>
<td>M₄</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M₅</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>M₆</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
<td>2'</td>
</tr>
</tbody>
</table>