Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2020 SCPE. Volume 21, Issues 3, pp. 391–406, DOI 10.12694:/scpe.v21i3.1734

DEADLOCK FREE RESOURCE MANAGEMENT TECHNIQUE FOR IOT-BASED POST DISASTER RECOVERY SYSTEMS

MADHAVI DEVI B[*], SMRITI AGRAWAL[†], AND R. RAJESHWARA RAO[‡]

Abstract. Disasters are inevitable, but their impact can be mitigated with careful planning. An IoT-based network with limited resources can be used in the post-disaster recovery. However, the resource of common interest creates contention among its contenders. This contention leads to tussle which in turn may lead to a deadlock. Some of the existing techniques prevent or avoid deadlock by performing stringent testing with significant testing overhead. While others propose recovery action after the deadlock is detected with significant overhead. A deadlock leads to a breakdown of the post-disaster recovery system while testing overhead implies delayed response either case can lead to catastrophic losses. This paper presents a new class of techniques that do not perform stringent testing before allocating the resources but still ensure that the system is deadlock-free and the overhead is also minimal. The proposed technique suggests reserving a portion of the resources to ensure no deadlock would occur. The correctness of the technique is proved in the form of theorems. The average turnaround time is approximately 18 % lower for the proposed technique over Banker's algorithm and also an optimal overhead of $O(m)$.

Key words: Banker's Algorithm, Deadlock, Deadlock avoidance, Deadlock Recovery, Operating systems, Safety sequence, post disaster management, IoT network, IoT resource scheduling

AMS subject classifications. 68M14

1. Introduction.

The UN Office for Disaster Reduction (UNISDR) [1] survey reveals that natural disasters such as floods, storms, heat-waves, etc., have claimed thousands of human lives, brought misery to millions of people and profoundly affected the economy. The recent pandemic of the corona virus outbreak is no exception. Other than natural disasters caused due to accidents in factories, homes, trains, roads, etc. also have catastrophic effects. In these circumstances, response to the disaster by effective utilization of the available resources is critical for the mitigation of their effects. The management of the limited available resources is essential for various post-disaster activities, such as evacuating or quarantining the people to safe places, providing medical help, etc. A network can connect these resources to find their whereabouts, availability, etc. This network can be established using the Internet of Things (IoT) [2, 3, 4, 5, 6, 7, 8, 9]. The IoT utilizes the internet and can be used in post-disaster recovery [10, 11, 12, 13, 14, 15, 16, 17, 18]. Authors [10, 11, 12, 13, 14, 15], represented the IoT for post-disaster system as an operating system with the set of activities as processes. This paper also considers the same.

Multiple post-disaster recovery activities (processes) of all types and sizes need to share resources for their effective utilization [11].Contention for a resource occurs when two or more processes request for it. This contention increases as the number of processes contending and the set of resources of interest increases. Eventually, some processes hold some resources and wait for others, which are held by another set of processes waiting for some other resources, forming a cycle of wait for. Hence, everybody is waiting for somebody to complete and no one has sufficient resources to complete, hence no process completes. The system utilization tends to become zero as none of the processes has the resources it needs to continue its execution. This situation is referred to as Deadlock [19, 20, 21, 22, 23].

Resource sharing systems such as single processor, multiprocessor systems, IoT based systems, parallel computing,cloud computing and distributed systems [15, 16, 17] with applications such as post-disaster management, distributed systems,automated manufacturing systems, etc. can be designed without keeping deadlock

*Research scholar JNTU Kakinada, India (madhavi.polagangu@gmail.com)
†CBIT, Hyderabad, India (agrawal.smriti@gmail.com)
‡Professor, JNTUUCEV, Vizianagaram, India

in account. The deadlock brings the system into a standby state which may have catastrophic effects. Further, it cannot be resolved on its own and requires external intervention. This is because no process will release the resources it has acquired without completing itself. Thus, efficient resource management[24, 25, 26, 27] is needed to manage the resources for avoiding the deadlock as well as improve the system performance by effective resource utilization and reduce the average turnaround time and increase the throughput of the system.

Coffman [22] suggested that a deadlock can occur only if, necessary conditions namely; Mutual exclusion, Hold and wait, No preemption and Circular wait.Thus, the solution for a deadlock problem lies around these conditions. The first set of the solutions suggests ensuring that one of the necessary conditions does not hold such techniques are referred to as *Deadlock Prevention (DP)*. The second set of techniques is called *Deadlock Avoidance (DA)*, it is a look forward technique to avoid deadlock in future. The next set of solutions advocates reactive approach called *Deadlock Detection and Recovery (DDR)*, where the system is recovered after the deadlock is detected. Most recently, authors [24, 25, 26, 27],suggested a new class of techniques called *Resource Reservation Techniques (RR)* for deadlock handling.

The Resource Reservation (RR) techniques handle deadlock with lower overhead. They supported *reserving a set of resources,* in a reserve pool such that these resources can be used for avoiding deadlock. The remaining unreserved resources were allocated injudiciously to any process requesting it. All these resource reservation techniques[24, 25, 26, 27], reduce the cost for testing the avoidance condition in the deadlock avoidance techniques. They reserve the resources and hence are not completely blind as the reactive technique of deadlock detection and recovery while allocating the resources. Thus, these techniques reduce the overhead of the avoidance techniques and deadlock frequency as compared to deadlock detection and recovery techniques.

In post-disaster recovery systems, deadlock may imply that the recovery activities are held up which may have catastrophic effects. At the same time,a delayed response in resource allocation can also lead to the unavailability of the critical resources when they are most needed. The existing resource reservation techniques present a promising reduction in the computing overhead for resource management. However, none of them is 100 % deadlock-free.Thus, this paper aims to propose a resource handling policy to ensure that the deadlock never occurs while maintaining a low response time. This work aims to re investigate the resource reservation policies to answer the question of;*how many resources must be reserved and how they must be allocated to avoid deadlock completely?* The correctness of the proposed technique is proved in the form of theorems.

The rest of the paper is organized as follows. Section 2 provides a further look at the existing literature. While Section 3 describes the system model, motivational examples are given in Section 4 to illustrate the various existing techniques for deadlock management. Section 5 presents the proposed Deadlock Free Resource Reservation (DFRR) technique, followed by results and analysis in Section 6. Finally, the paper concludes with Section 7.

2. Related Work. The post-disaster response needs dynamic and instantaneous updates for which IoT is one of the best solutions [10, 11, 12, 13]. Zhang et. al. [10] proposed an agent based resource allocation in emergency management systems considering the severity of the disasters at various levels. Authors [11, 12] studied an IoT- based post-disaster response system and estimated the response time of the resources as they wait to be scheduled using Banker's Algorithm. In [33] uses priority basis stable matching algorithm for effective allocation of resources during disasters using IoT. In a post-disaster response system, the response time is critical and can impact lives and property. In such a system there are limited resources and deadlock of these resources can have catastrophic effects. This paper aims to develop a deadlock-free system with a reduced response time of the resource management system and also estimate the resources needed for establishing an optimal emergency response center.

Deadlock is a well-known problem in computing as well as in all those fields where resources are shared. It attributes to major overhead in the system in both cases: disregarded or managed. In case the deadlock is completely disregarded, it may seize the system. However, the deadlock management has the corresponding overhead, it has been studied over the years. One of the earliest studies used a graph based model proposed by Coffman et. al. [22] for inferring the conditions that must simultaneously hold for the deadlock to occur. There are four strategies to handle deadlocks; DDR (deadlock detection and recovery), DA (deadlock avoidance), DP (deadlock prevention) and RR (Resource Reservation).

DDR is the simplest technique suggesting to 'do nothing', that is till the time the deadlock actually occur do nothing. The DDR technique advocates detecting the deadlock and suggests recovering from it. Holt [21] suggested one of the oldest technique for deadlock detection. This technique represented the resource hold and request as a graph and reduced it for detecting a deadlock.Cheng Xin et.al [30], used threads for deadlock detection in distributed systems. Xiao and Lee [39], suggested a parallel algorithm for deadlock detection on Multi-Processor System-on-a-Chip (MPSoC). Shiu et. al. [40], presented a low cost hardware solution for deadlock detection.DDR techniques though simple yet have the overhead of deadlock detection algorithm running periodically, and may need to restart one or more process for recovering from the deadlock which may not be possible in all the systems. The DP techniques target to prevent one of the necessary conditions from occurring so that the deadlock will not occur.

Deadlock avoidance (DA) methods perform forward calculation based on the knowledge of the resources required by the processes in future to avoid deadlock. Havender [25], suggested to allocate all the resources a process demands in the beginning to avoid deadlock. This method was though effective had serious limitation, because the resources remained underutilized. Other methods suggested by Havender [25] include i) ordering the resources and ii) preempting and re-requesting. The ordering technique ordered the resources in such a fashion that the deadlock would not occur. The resources were preempted and re-requested if sufficient resources were not available when an incremental request was made. Another one of the most renowned DA algorithm is Banker's Algorithm, derived by Dijkstra [19] for a single resource type. Habermann [20] extended this algorithm to multiple resource types. Banker's Algorithm tests all possible allocations and ensures that the request for resources is granted only when no deadlock is foreseen. Lang [28] extended the Banker's algorithm using a control flow graph to determine the resources are released before the control are transferred to the processes in the next region.

Yin et.al. [37] studied the deadlock problem in multithreaded program in a multicore architecture. The authors converted the program source code into a formal model which was used by the discrete control theory to automate the lock acquisition on the resources. They suggested to postpone the resource lock acquisition to avoid deadlock. Wang et. al. [38] used Banker's Algorithm for broadcasting in wireless network, where carrier frequency are one of the resources. Lee and Mooney [31] implemented the DA as a hardware, for a Multi-Processor System-on-a-Chip (MPSoC). Pyla and Varadarajan [42] reiterated the fact that deadlock can be avoided completely only when some prior knowledge of the resource requirement of the processes in the system is available and it is likely to occur if the resources are allocated arbitrarily. They suggested 'Sammati'a tool for threaded applications using POSIX for automatic deadlock detection and recovery. Youming Li et. al. [43] proposed a modification of Banker's algorithm (BA). Permutation matrices were used for each resources and the process was selected greedily. However, their space requirement was more than the original Banker's Algorithm.

Kawadkar et. al. [44] extended the Banker's algorithm by examining the processes in the waiting queue. A process entered the waiting queue when its requested resources are either unavailable or could not be granted.They suggested to pick a process from the waiting queue based on the resources it is holding and needing. However, authors do not mention how selected process from the waiting queue should be allocated the resources such that the deadlock is avoided. Dixit and Khuteta [45] suggested to change the resource requirement at the run-time to prevent a possible deadlock.

All the deadlock avoidance techniques have high computation time as they must perform some test before each allocation to avoid deadlock. The Resource Reservation (RR) techniques eliminated this test for deadlock avoidance and thus, reduce the turnaround time of the processes. The RR techniques suggest to reserve some resources that can be used to avoid the deadlock. Botlagunta et. al. [25] first introduced this idea and suggested to reserve resources based on a threshold. Agrawal et. al. [24] suggested to estimate the resources to be reserved based on the sum of the resources needed further.

Authors [26] suggested another way based on the shortest execution time process. The process with minimum worst case execution time reserved the resources as suggested in [26]. Shubham et.al [36] extended the technique suggested in [26] to reserve the resources based on the remaining resource need. Botlagunta et. al. [27] extended the RR techniques to dynamically allocate a budget of resources to each incoming process and perform resource reservation as per the technique suggested in [25]. These techniques are discussed in detail in the subsequent section.

Table 2.1
Deadlock Handling Methods

Deadlock detection and recovery (DDR)	Deadlock Prevention (DP)	Deadlock Avoidance (DA)	Resource Reservation(RR)
Reactive to deadlock, does nothing to handle it in advance	Prevent one of the four necessary conditions from occurring	Look forward to avoid any deadlock in future. Perform Safety Sequence Test	Reserve some resources for avoiding the deadlock

Both DA and RR techniques presume that the system resource requirement is known in advance. The DA techniques perform some kind of test for ensuring that the deadlock will not occur in the future. However, the RR techniques do not perform any test rather use the knowledge of the resource requirement for estimating the portion of resources that must be reserved to avoid deadlock. The resource reserved by different RR Techniques existing so far is not sufficient for completely avoiding the deadlock. The present work extends the idea of the RR techniques to use the knowledge of the resource requirement by the processes for estimating the reserve pool strength. The proposed technique ensures that all the processes have the opportunity to complete with the reserved resources.

3. System Model. The post-disaster management system resource pool is assumed as a finite set of $m$resource types represented as $R_1,\ R_2,\ R_3 \ldots R_m$, with instances as $\alpha_1,\ \alpha_2,\ \alpha_3 \ldots \alpha_m$ of each. Further, the post-disaster recovery activities are called processes $P_1,\ P_2,\ P_3 \ldots P_n$ with attributes arrival and worst-case execution time represented as $(a_i,\ e_i)$. The state of the system resources is represented in the data structures from Table 3.1.

4. Motivational Example. Motivational examples are presented in this section to analyze the existing techniques.

Table 3.1
Notations used in System

Notations	**Meaning**
P $_i$	ith process
R $_j$	jth resource
alpha$_j$	number of instances of resource type R $_j$
m	Number of resource types
n	Number of processes.
Request[i][j])	A process maximum resource requirement is indicated by Max[i][j]. However, a process does not need all the resources at once but incrementally. The resources requested by the processes at any time instance is represented as a two-dimensional array of size n by m. Thus, Request[i][j]= k, where k= 0, 1,. . . Max[i][j].
Allocation[i][j]	It is a two dimensional n by m array containing the resources assigned to each process.
Need[i][j]	It is a $n \times m$ array, containing the estimated resources further required by processes. Mathematically, $Need\,[i]\,[j] = Max\,[i]\,[j] - Allocation\,[i]\,[j]$
Available[j]	A subset of the resource pool consisting of resources that can be allocated to the requesting process. It is represented by an array of m elements. It is initializes as $Available\,[j] = \alpha_j - Reserve\,[j]\,, \forall R_j j = 1, 2, 3\ \ldots m$
Reserve[j]	A subset of the resource pool is marked as reserved. It contains the quantity of each resource type reserved. It is represented as an array with $m$elements. Thus, $Reserve[j]$ equals k where k indicate that k instance of the resource of type R_j are reserved.
Max[i][j]	Maximum resource requirement
Throughput	the number of processes completed per unit of time.
Turnaround time(TT)	the difference between submission of a process and its completion.
Average turnaround time(ATT)	The total turnaround time divided by the number of processes.
Safety Sequence (SS) [19]	A safety sequence was suggested by Banker's algorithm is an order in which the processes can be completed without deadlock.
Safe State and UnSafe State[19]	A system for which a safety sequence exists ensures that the deadlock will not occur is said to be in a Safe State otherwise in Unsafe state

Table 4.1
System initial state at start time t_0

	Allocation				Maximum				Request			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	0	0	0	0	1	2	0	0	0	0
P_2	0	0	0	0	2	7	5	0	0	2	1	0
P_3	0	0	0	0	6	6	5	6	4	3	1	2
P_4	0	0	0	0	4	3	5	6	1	0	2	4
P_5	0	0	0	0	0	6	5	2	0	2	2	0
Available												
					R_1	R_2	R_3	R_4				
					8	13	11	10				

Table 4.2
System state at time t_1 after allocation of resources requested at time t_0

	Allocation				Need				Request			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	0	0	0	0	1	2	0	0	0	0
P_2	0	2	1	0	2	5	4	0	0	2	1	0
P_3	4	3	1	2	2	3	4	4	0	0	0	0
P_4	1	0	2	4	3	3	3	2	1	0	0	0
P_5	0	2	2	0	0	4	3	2	0	0	0	0
Available												
					R_1	R_2	R_3	R_4				
					3	6	5	4				

Example 1 [9]: Consider a system consisting of resources as $\{R_1,\ R_2,\ R_3, R_4\} = \{8, 13, 11, 10\}$ is to implement processes P_1, P_2, P_3, P_4, P_5 with resource requests as illustrated in Table 4.1 for time t_0. Table 4.2 indicates the system state as the resources are allocated. Table 4.3 enlists a hypothetical sequence of resource requests made by the processes subsequently. The existing techniques discretion on these requests is as follows:

1. ***Deadlock Avoidance Banker's Algorithm(BA) [19]:***
   The requests made as per (refer Table 4.3) are considered by BA by estimating the safety sequence. Thus, request of P_2, $P_4$and P_3 are granted but the last request ofP_2 is not granted.
2. ***Deadlock Detection and Recovery (DDR) [23]:***
   This is a reactive technique where no pretesting is done to ensure that a deadlock will not occur. It will simply grants every possible request if sufficient resources are available as and when they are made. In the present example,all requests as listed in Table 4.3 are granted. The resulting state of the system is shown in Table 4.4. The system does not enter into deadlock immediately as the need of process P_1 can be satisfied. Once the process P_1 completes the state of the system is as shown in Table 4.5. This state is referred to as Unsafe state (defined in Sect.3), as it leads to a deadlock. Substantial overhead is incurred in detecting and recovering from it.
3. ***Worst-Case Execution Time Based Resource Reservation (ETRR) Technique [26]:***
   This technique is motivated by shortest job first scheduling assigning highest priority to a process with smallest the worst case execution time. In this example priorities are assigned as $\langle P_1,\ P_5,\ P_2,\ P_4,\ P_3\rangle$, indicating that the computation time of process P_1 is least. Hence, resources (0, 0, 1, 2) are placed in the reserve pool. All the requests as an when they arrive (refer Table 4.3) are granted by this technique also. Finally, at time t_4 when process P_1 completes, the system state can be seen in Table 4.5. Thus, this technique also fails to avoid the deadlock subsequently.
4. ***Threshold based Resource Allocation (TRA) Technique [25]:***
   The resources are reserved by this technique based on a threshold estimated as $Threshold[j] = \lceil \lfloor Need[i][j]\, \forall i = 1, 2 \ldots n \rfloor,\ 0 \rceil$. Accordingly, for this example, it reserves (2, 3, 1, 2) resources at the onset. At time t_1, when P_2 and P_4 request for additional resources, both the requests are granted. Thereafter, the resources in the reserve pool are (2, 3, 1, 2) while that in the available pool are (0, 1, 3, 2). However, at time t_2 when process P_3 requests for (2, 0, 0, 2) resources, sufficient resources are

TABLE 4.3
Sequence of requests

Time at which request is made	Process requesting for the resources	Request (R_1, R_2, R_3, R_4)
t_1	P_2	(0,2, 1, 0)
t_1	P_4	(1,0, 0, 0)
t_2	P_3	(2,0, 0, 2)
t_3	P_2	(0,2, 1, 0)

TABLE 4.4
Snapshot after allocation to P_2, resources (0, 2, 1, 0) at time t_3

	Allocation				Maximum				Need			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	0	0	0	0	1	2	0	0	1	2
P_2	0	6	3	0	2	7	5	0	2	1	2	0
P_3	6	3	1	4	6	6	5	6	0	3	4	2
P_4	2	0	2	4	4	3	5	6	2	3	3	2
P_5	0	2	2	0	0	6	5	2	0	4	3	2
Available												
				R_1	R_2	R_3	R_4					
				0	2	3	2					

TABLE 4.5
Snapshot at time t_4 after completion of process P_1

	Allocation				Maximum				Need			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	Execution Complete											
P_2	0	6	3	0	2	7	5	0	2	1	2	0
P_3	6	3	1	4	6	6	5	6	0	3	4	2
P_4	2	0	2	4	4	3	5	6	2	3	3	2
P_5	0	2	2	0	0	6	5	2	0	4	3	2
			Available				Reserve					
			R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4		
			0	2	3	2	0	0	0	0		

TABLE 4.6
System initial state at time t_0 for Example 2

	e_i	Allocation		Maximum		Request		Available	
		R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2
P_1	10	0	0	2	4	1	2	4	4
P_2	2	0	0	4	2	2	1		

TABLE 4.7
System state after request granted to process P_2

	Allocation		Maximum		Need		Available	
	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2
P_1	1	2	2	4	1	2	1	1
P_2	2	1	4	2	2	1		

TABLE 4.8
System state after request granted to P_1

	Allocation		Maximum		Need		Available		Reserve	
	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2
P_1	2	4	2	4	0	0	0	0	2	0
P_2	0	0	4	2	4	2				
Request of process P_1 is granted and P_2 is denied. safety sequence is $\langle$ P_1, P_2 $\rangle$										

TABLE 4.9
System state with reserve pool after request granted to process P_2

	Allocation		Maximum		Need		Available		Reserve	
	R₁	R₂	R₁	R₂	R₁	R₂	R₁	R₂	R₁	R₂
P₁	0	0	2	4	0	0	0	2	2	1
P₂	2	1	4	2	2	1				
Request of process P₁ is denied and P₂ is granted. Safety sequence is ⟨ P₂, P₁ ⟩										

TABLE 4.10

a) Process P₁ requests (1, 2) resources							
R_j	Request	Need	Available	Threshold	X	Y	Can_grant
R₁	1	2	4	2	1	1	1
R₂	2	4	4	2	0	1	1
granted							
b) Process P₂ requests (2, 1) resources							
R_j	Request	Need	Available	Threshold	X	Y	Can_grant
R₁	2	4	3	1	0	1	1
R₂	1	2	2	2	1	0	1
granted							

not available in the pool. Hence, the reserve pool resources are used and (2, 3, 4, 4) are granted. No deadlock occurs for this example.

5. ***Total Need Based Resource Reservation (TNRR) Technique:[24]***
The total number of resources neededby requesting processes is considered for reserving resources by this technique. Mathematically, $Reserve\,[j] = \lceil\frac{(Total\_Need-\alpha_j)}{n}\rceil$ where $Total\_Need\,[j] = \sum_{i=1}^{n} Max\,[i]\,[j]$, α_j and $Reserve\,[j]$ are defined in Sect. 3.

In the motivational example 1, $Total\_Need\,[j]$ is estimated as 12, 22, 21, 16 for the resources R_1, R_2, R_3 and R_4 respectively. Thus, (2, 3, 2, 2) resources are reserved at the on set. At time t_1, when P_2 and P_4 request for additional resources, both the requests are granted without using any of the reserved resources. However, at time t_2 when process P_3 requests for (2, 0, 0, 2) resources, only (0, 1, 2, 2) resources are in the available pool which are not sufficient. The resources from the reserve pool are used to satisfy need of the process. No deadlock occurs for this example.

This example illustrated scenario where the existing priority based techniques failed to reserve sufficient resources to avoid deadlock. The following example demonstrates the limitation of the other techniques that survived in this example.

Example 2: Suppose in system processes P_1, P_2 are have an upper bound of 4 instances of each resources R_1 and R_2 illustrated in the Table4.6 request for (1, 2) and (2, 1) resources are made by these processes respectively. The assessment on these requests of the existing techniques is as follows:

1. ***Deadlock Avoidance Banker's Algorithm (BA) [19]:***
The Banker's Algorithm calculates the safety sequence for each request. It then grants P_1's request but denies P_2 's request, refer to Table 4.7, to avoid deadlock in future.
2. ***Deadlock Detection and Recovery (DDR) [23]:***
DDR grants every request indiscreetly and does the same in this case too. The resulting state is as in the Table 4.7, which leads to a ***Deadlock eventually occurs***.
3. ***Worst-Case Execution Time Based Resource Reservation (ETRR) Technique[26]***
The ETRR technique reserves the resources for process P_2 , because it is the shortest job(cf.Table 4.6). The reserve and available pools thus, contain (4, 2) and (0, 2) resources respectively. When the process P_1 requests for the resources (1, 2), it is denied by the system. However, the request by process P_2 is granted, refer Table 4.9. The system is in Safe state.

Table 4.11
Analysis of the example 1 and 2

Technique	**Occurrence of deadlock in Example**	**Occurrence of deadlock in Example 2**
BA	Deadlock does not occur	Deadlock does not occur
DDR	Deadlock occurs	Deadlock occurs
ETRR	Deadlock occurs	Deadlock does not occur
TRA	Deadlock does not occur	Deadlock occurs
TNRR	Deadlock does not occur	Deadlock occurs
DFRR(proposed method)	Deadlock does not occur	Deadlock does not occurs

4. ***Threshold based Resource Allocation (TRA) Technique [25]:*** At the inception the threshold is (2, 2) based on which the request of process P_1 is granted shown in4.10 a). It is then updated as (1, 2) (refer to table 10 b)) and the request of P_2 is also granted. The system reaches the state in the 4.7 which indicate that the ***Deadlock will occur***.
5. ***Total Need Based Resource Reservation (TNRR) Technique[24]:*** The resources to be reserved are estimated as $Reserve\,[j] = \lceil\frac{(Total\_Need-\alpha_j)}{n}\rceil$ as $Reserve\ [1] = \lceil\frac{(6-4)}{2}\rceil$ and $Reserve\,[2] = \lceil\frac{(6-4)}{2}\rceil$, i.e., (1, 1). The available pool will contain the remaining (3, 3) resources, sufficient for the process P_1 and P_2 request ((1, 2) and (2, 1) respectively). Thus, the requests are granted leading the system into the unsafe state as shown in 4.7.

Table 4.11 summarizes the effectiveness of the existing techniques in maintaining a deadlock free system for the motivational examples.The DDR technique performs no testing before granting any resource to a requesting process and hence, ends up into a deadlock most often. The overhead saved by eliminating the test before granting of resources to a requesting process is consumed by the periodic testing required for deadlock and recovery whenever it occurs. On the other hand, the BA algorithm which ensures that deadlock will never occur is too costly. The sub optimal techniques (ETRR, TRA and TNRR) are able to prevent the deadlock in some case (table 4.11) but are not full proof. This paper proposes a resources reservation technique which is deadlock free (proved in the form of a theorem). It also lowers the overhead incurred by the BA.

The following section presents the proposed technique for a deadlock free system using resource reservation strategy.

5. Proposed Deadlock Free Resource Reservation Techniques (DFRR). The above section analyzed the performance of the prevailing techniques and summarizes them in the table 4.11. The table 4.11 reveals that the Banker's algorithm calculates the safety sequence before allocating the resources on every request. The cost of such an estimation is $O\left(mn^2\right)$ where m is the different type of resources and n is the number of processes in the system. This overhead is considerable for even a small system. On the other hand, newer RR techniques such as ETRR, TRA and TNRR are cost effective but not 100% deadlock free. A closer look at the motivational example 1, illustrates that ETRR is process oriented, i.e., reserve resources for a process P_1 and guarantee that it will complete successfully and it does. However, process P_1reserves only (0,0, 1, 2) resources, i.e., it reserves only one and two instances of resource type R_3 and R_4 respectively. While all the instances of resources R_1 and $R_2$are unreserved. The remaining resources are claimed by the rest of the processes leading the system into an Unsafe state. The resources released by the process P_1on its completion are not sufficient for any other process and the deadlock will eventually occur. In other words, ETRR although reserve resources for one of the process but they do not restrict the remaining processes from entering into Unsafe state.

The TRA and TNRR techniques are resource oriented, where reservation of resources is done such that there is at least one process requiring that resource. However, a process may require other resources as well that may not be available in sufficient quantity for its successful completion. The example 2, Sect.4, illustrates that sufficient instances of R_1 are available for processes P_1, while for process P_2 ample instances of resource

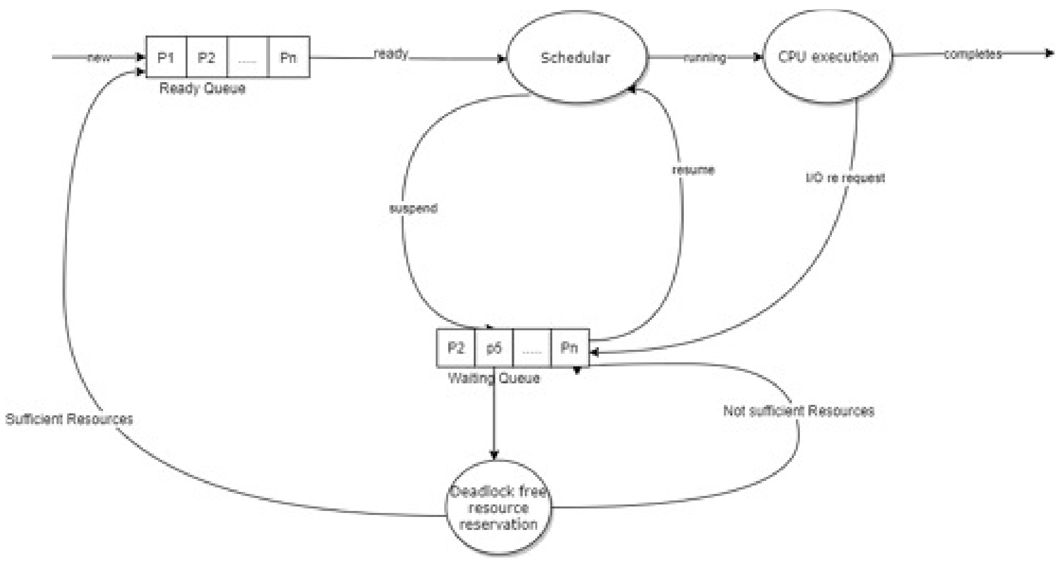

FIG. 5.1. *Proposed System Model.*

R_2 are available. However, enter into a circular wait as they wait for the other to release the resources they are holding.

This paper proposes a deadlock free resource reservation (DFRR) technique which ensures that deadlock will never occur. The system model for this DFRR can be seen in the Fig.5.1. It suggests to reserve some resources in the reservation pool as per the theorem 5.2. The remaining resources are made available in the available pool. Whenever a process requests for a set of resources, available pool resources are granted (as per theorem 5.2). In case the request cannot be granted only from the available pool then the reservation pool are also used such that the maximum resources this process can ever want are granted to give this process the chance to complete.

The theorem 5.1,claims that if a process is allocated all the resources it will ever need then the system will be deadlock free again if it was deadlock free before this allocation. Theorem 5.2, proves the deadlock free of the proposed technique.

Theorem 5.1 is stated as follows.

THEOREM 5.1. *A system consisting of set of resource types as $R_1, \quad R_2, \quad R_3 \ldots R_m$ with $\alpha_1, \quad \alpha_2, \quad \alpha_3 \ldots \alpha_m$ instances of each type, when scheduling a set of independent processes $P_1, \quad P_2, \quad P_3 \ldots P_n$, if is in a safe state will remain in safe state even after allocating all the resources ever needed by a requesting process P_{req} from the set.*

Proof. Suppose at any time t the system is in safe state, then the safety sequence can be estimated as

$$\langle P_b, \quad P_{req}, \quad P_a \rangle \tag{5.1}$$

where $P_b = P_{b1}, \ P_{b2} \ldots P_{bx}$ and $P_a = P_{a1}, \ P_{a2} \ldots P_{ay}$ are the set of processes in the safety sequence before and after the requesting process P_{req} .

Let, the resources available at this time t be

$$\omega(t) = Available[j] + Reserve[j] \quad \forall j = 1 \ldots m \tag{5.2}$$

Processes are in the safety sequence, this implies:

$$Need[b1][j] \leq \omega(t); \quad \forall j = 1 \ldots m \tag{5.3}$$

It will complete and release all it's resources. Similarly, the remaining processes will complete. Reconsider if at this time t a process P_{req} needs

$$Need\,[req]\,[j] = Max\,[req]\,[j] - Allocation\,[req]\,[j] \tag{5.4}$$

resources to complete.

In worst case, initially, no resources are allocated to this requesting process P_{req}, i.e.,

$$Allocation\,[req]\,[j] = 0 \quad \forall j = 1 \ldots m \tag{5.5}$$

Therefore, substituting (Eq.5.5) in (Eq.5.4)
$Need\,[req]\,[j] = Max\,[req]\,[j] \quad \forall j = 1 \ldots m$

$$Need\,[req]\,[j] \leq \omega\,(t) \quad \forall j = 1 \ldots m \tag{5.6}$$

If Eq.5.6 holds, then all the needed resources can be granted to this requesting P_{req} process. In worst case (Eq.5.6 will be)

$$Need\,[req]\,[j] = \omega\,(t) \quad \forall j = 1 \ldots m \tag{5.7}$$

indicates system has just sufficient resources needed by the process to complete. These resources are thus allocated to the requesting process P_{req}; mathematically,

$$Allocation\,[req]\,[j] = Allocation\,[req]\,[j] + Need\,[req]\,[j]$$

Substituting (Eq.5.5) and (Eq.5.7)

$$\Rightarrow Allocation\,[req]\,[j] = \omega\,(t) \quad \forall j = 1 \ldots m \tag{5.8}$$

implying that all needed resources are allocated, thus, the need can be updated as

$$Need\,[req]\,[j] = 0, \quad \forall j = 1 \ldots m.$$

The process P_{req} can thus, complete at time t1. It then releases

$$Allocation\,[req]\,[j] = \omega\,(t)\,, \quad \forall j = 1 \ldots m$$

all the resources allocated to it as per equation (Eq.5.8).

Therefore, the resources in the system will again be restored to be $\omega\,(t)$. These are sufficient for process P_{b1} (Eq.5.1) as suggested by equation (Eq.5.3).

The new safety sequence is $\langle P_{req},\ P_b,\ P_a\rangle$.
Thus, system remains in the safe state. Hence proved. $\square$

The theorem 5.1, ensures that if a process P_i is granted all the resources it may further need to complete, i.e. $Need\,[i]\,[j] = Max\,[i]\,[j] - Allocation\,[i]\,[j]$, then the system will remain/return to safe state after it's completion. Now, the question is:*does the system has the Need[i][j] number or resources?* The theorem 5.2, ensures that every process in the system can acquire the needed resources, and ensure the system is deadlock free. The theorem 5.2 can be stated as follows.

Theorem 5.2. *A set of independent processes $P_1,\ P_2,\ P_3 \ldots P_n$ when scheduled on a system consisting of set of resource types as $R_1,\ R_2,\ R_3 \ldots R_m$ with $\alpha_1,\ \alpha_2,\ \alpha_3 \ldots \alpha_m$ instances of each type, with Max[i][j] as the maximum number of resources of type R_j demanded by the i^{th} process P_i during its execution time can be done without deadlock if the following conditions hold:*

1. $\forall R_j$ system reserves

$$Reserve[j] = \begin{cases} 0 & \eta_j \leq 1 \\ 0 & \sum_{i=1}^{n} Max[i][j] \leq \alpha_j \\ \left\lceil \frac{\left(\sum_{i=1}^{n} Max[i][j]\right)-\alpha_j}{\eta_j - 1} \right\rceil & \sum_{i=1}^{n} Max[i][j] \geq \alpha_j \end{cases}$$

instance of its type forming a reserve pool, where $\eta_j = count\,(Max\,[i]\,[j] \neq 0, \forall i = 1, 2 \ldots n)$ is the number of processes requesting for the resource R_j .Leaving the resources in the available pool as $Available\,[j] = \alpha_j - Reserve\,[j]$

2. $\forall P_i,\ \ Max\,[i]\,[j] - Reserve\,[j] \geq 0$

3. A process P_i can acquire either

(a) $Max\,[i]\,[j] - Reserve\,[j]$ resources from the available pool,

(b) $Max\,[i]\,[j]\,\forall R_j$ from available and reserve pool put together such that the available pool is consumed first. That is a process must acquire all the resource it might need.

4. A process P_i on completion relinquishes all the resources, $Max\,[i]\,[j]\,\forall R_j$, that are returned to the reserve and available pools as per condition 1.

Proof. A process P_i can acquire resources either partially as per condition 3.a. or completely as per condition 3.b. Thus, some processes may acquire partial resources ($Allocation\,[i]\,[j] = Max\,[i]\,[j] - Reserve\,[j]$) and some complete set ($Allocation\,[i]\,[j] = Max\,[i]\,[j]$) .

Processes that acquire resources as per condition 3.b.

$$(Allocation\,[i]\,[j] = Max\,[i]\,[j]\,;\ Need\,[i]\,[j] = Max\,[i]\,[j] - Allocation\,[i]\,[j] = 0)$$

have all the requisite resources and will complete and relinquish the resources as per condition 4. The system remains in safe state as per theorem 5.1.

In worst case all the processes acquire resources partially as per the condition 3. a. and none of the process can complete. Mathematically,

$$Allocation\,[i]\,[j] = Max\,[i]\,[j] - Reserve\,[j] \quad \forall P_i \tag{5.9}$$

Applying summation to equation (cf.Eq.5.9); the total number of instances of type R_j , allocated to any process is thus, $\sum_{i=1}^{n} Allocation\,[i]\,[j] = \sum_{i=1}^{n} (Max\,[i]\,[j] - Reserve\,[j])$ where $Max\,[i]\,[j] -$ $Reserve\ [j]$ is the number of resource type R_j allocated to process P_i .

Since, $Reserve\,[j]$ is independent of the process, the total number of instance already allocated are

$$\sum_{i=1}^{n} Allocation\,[i]\,[j] = \left(\sum_{i=1}^{n} Max\,[i]\,[j]\right) - n * Reserve\,[j] \tag{5.10}$$

Consider $Reserve\,[j] = \lceil \frac{\left(\sum_{i=1}^{n} Max[i][j]\right)-\alpha_j}{\eta_j - 1} \rceil$ as per condition 1., here, $\eta_j = n$ in worst case, indicating a resource is demanded by all the processes in the system, leading to higher conflicts. Therefore, $(n-1) * Reserve\,[j] = (\sum_{i=1}^{n} Max\,[i]\,[j]) - \alpha_j$ implies

$$n * Reserve\,[j] = \left(\sum_{i=1}^{n} Max\,[i]\,[j]\right) - \alpha_j + Reserve\,[j] \tag{5.11}$$

Substituting, $Available\,[j] = \alpha_j - Reserve\,[j]$, as per condition 1., in equation (Eq.5.11) implies

$$n * Reserve\,[j] = \left(\sum_{i=1}^{n} Max\,[i]\,[j]\right) - Available\,[j] \tag{5.12}$$

Substituting (Eq.5.12) in (Eq.5.10), the number of instance allocated to all the processes put together is $\sum_{i=1}^{n} Allocation\,[i]\,[j] = (\sum_{i=1}^{n} Max\,[i]\,[j]) - (\sum_{i=1}^{n} Max\,[i]\,[j]) - Available\,[j]$, implying:

$$\sum_{i=1}^{n} Allocation\,[i]\,[j] = Available\,[j]\,, \forall R_j$$

That is the number of instance allocated to all the processes put together are equal to $Available\,[j]\,, \forall R_j$. In other words, at most available pool is emptied by the processes, however, reserve pool resources are still unused. Since, every process has acquired, as per equation (Eq.5.9), $Allocation\,[i]\,[j] = Max\,[i]\,[j] - Reserve\,[j]$, hence,

$$Need\,[i]\,[j] = Max\,[i]\,[j] - Allocation\,[i]\,[j]$$

$$\Rightarrow Need\,[i]\,[j] = Max\,[i]\,[j] - (Max\,[i]\,[j] - Reserve\,[j])$$

$$\Rightarrow Need\,[i]\,[j] = Reserve\,[j]\,, \quad \forall R_j$$

Therefore, every process needs only $Reserve\,[j]\,, \forall R_j$ more resources to complete which are available in the reserve pool. Thus, every process has equal opportunity to complete.Once any process completes it will release any resources it was holding (at least $Reserve\,[j]\,\forall R_j$), which are sufficient for the subsequent processes to complete (Theorem 5.1). Hence, there is no deadlock or starvation. Hence proved. □

The theorem 5.2 guarantees that the deadlock will never occur if the system follows it. The deadlock free resource reservation (DFRR) technique is proposed based on these theorems it is stated in the form of the DFRR algorithm in cf.Alg.1. The time complexity for resource management to decide on the resource allocation as per the request of the proposed algorithm is $O\,(m)$. The resource manager only refers the available pool for the requested resources incurring $O\,(m)$ overhead. In case the sufficient resources are not available in the Available pool the same are granted from the Reserve pool. This resource management policy is optimal and is proved in the form of lemma 5.3.

Lemma 5.3. *No resource management policy can assign resources to a requesting process from a set of independent processes* $P_1, P_2, P_3 \ldots P_n$ *when scheduled on a system consisting of set of resource types as* $R_1, R_2, R_3 \ldots R_m$ *with* $\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_m$ *instances of each type, with an overhead lower that* $O\,(m)$.

Proof. Each Process can request any resource $R_1,\;\; R_2,\;\; R_3 \ldots R_m$ its request vector is thus of length 'm'. Hence, any resource management technique will check for the request and availability of all the 'm' resources at least once for a process. Hence, the overhead will not be less than $O\,(m)$. □

The motivational examples in Sect. 2 are revisited to show the effectiveness of the proposed technique. The resources reserved for the motivational example 1 (Table 4.1) are (2, 3, 3, 2). Requests in Table 4.1 are granted followed by the request made by the processes P_2 and P_4 as at time t_1 as stated in table using resources from the available pool. However, the request of process P_3 and P_2 at time t_2 and t_3 cannot be granted as per the condition 3.a theorem 5.2. The system remains in the safe state.

Deadlock Free Resource Reservation Technique, reserves (2, 2) resources, for the motivational example 2, cf. Table 4.6. Hence, either process P_1 acquires the necessary resources and completes as per Table 4.8 or as per Table 4.9, P_2 completes and relinquishes the resources. Hence, no deadlock.

The following section presents the results obtained on implementation of the proposed technique.

6. Simulations Results. Simulations were performed on process sets to analyze the behavior of the proposed DFRR technique as compared to the existing techniques. The average time of the each process spends in the system from its submission to its completion also known as*Average Turnaround time* is the key parameter of the system.

Algorithm 1: DFRR Algorithm

```
   input: Process Priority Queue
1  begin
2      for j=1 to m do
3          Reserve[j]= $\left\lceil \frac{\left(\sum_{i=1}^{n} Max[i][j]\right)-\alpha_j}{\eta_j - 1} \right\rceil$
4          Available[j]=$\alpha_j$-Reserve[j]
5      end
6      while No new Request do
7          Execute a ready process;
8      end
9      A process $P_i$ request for Request[i][j] resources do
10         if (Allocate[i][j]+Request[i][j])$\leq$(Max[i][j]-Reserve[j])$\forall$j=1,2...m) then
11             Allocate[i][j] = Allocate[i][j] + Request[i][j];
12             Available[j ]= Available[j] - Request[i][j];
13             Need[i][j]=Need[i][j]-Request[i][j];
14         end
15         else if Need[i][j]$\leq$(Available[j]+ Reserve[j])$\forall$j=1,2...m then
16             Allocate[i][j]=Allocate[i][j]+ Need[i][j];
17             if Need[i][j]Available[j] then
18                 Available[j]= Available[j]-Need[i][j];
19             end
20             else
21                 Reserve[j]=Reserve[j]+ Available[j]-Need[i][j];
22                 Available[j]=0;
23             end
24             Need[i][j]=0;
25         end
26     end
27     If a process $P_i$ completes then return the resources and goto to step 5.
28 end
```

All simulations are performed on a 2.0 GHz Processor. With a Resource pool of 10 resources with instances as { 11, 14, 7, 2, 16, 8, 15, 17, 13, 5}. 100 process sets for each point in the graph were generated with each process randomly selects a execution time between 0 to 50.

Fig.5.1, illustrate the effect on average turnaround time as the process load increases. As process load increases the chances for the resource conflict also increases and hence, the average turnaround time. The analysis of the proposed DFRR reveals that the turnaround time of the processes is relatively better than the existing for all process loads. The improvement in the performance is even more profound for the higher loads. This is because as the load increases so does the resource demand by the processes, leading to frequent resource requests. The Banker's algorithm has a high overhead for serving each request and hence the turnaround time increases as the load increases for it. On the other hand, the existing RR and DDR technique face more frequent deadlocks and spend time in recovering from the deadlock, which in turns increases the turn around time. The proposed DFRR technique has lower overhead for resource management and is deadlock free. The average turnaround time is approximately 18 % lower for the proposed DFRR over Banker's algorithm.

A process can request all the resources it could need at a time or incrementally as its execution processed. Fig.6.1 illustrates the effect of the number of incremental resource requests (Steps) made by the process set in its execution on the average turnaround time of the processes. The process load is fixed to be 0.8 for this figure. The Steps ranges from 1 to 10, where Steps =1 indicates that the processes requests all the resources at the

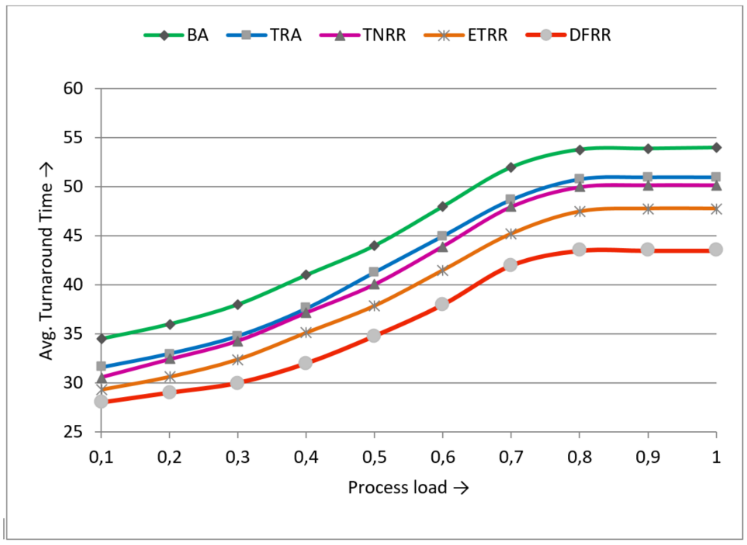

FIG. 6.1. *Process load vs. average turnaround time.*

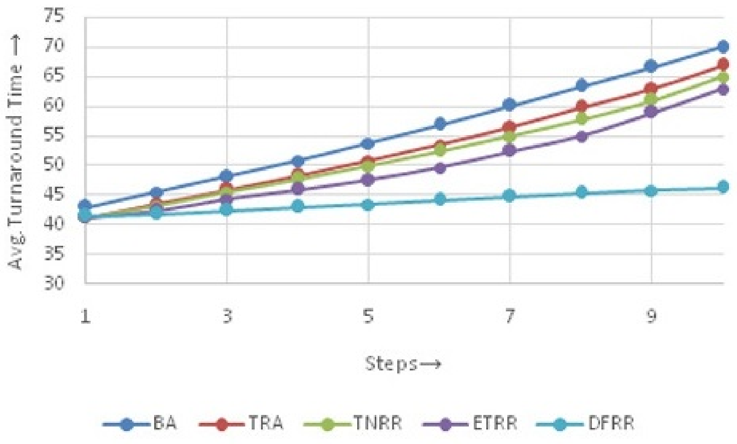

FIG. 6.2. *Steps vs average turnaround time.*

inception while 2, 3....10 indicate that the processes made 2 or 3 ... 10 incremental resource requests. BA has high overhead for resource management. Thus, as the processes make multiple incremental requests the BA spends more and more time in deciding on the resource grant. Moreover, the process whose request is denied by the BA enter into pending state which increases their wait time and hence, the average turnaround time of the set increases. The other TRA, TNRR and ETRR techniques though have lower overhead for resource management are not 100 % deadlock free. As the Steps increases the number of deadlocks encountered by them also increases incurring deadlock resolution overhead. The proposed DFRR technique has low overhead for resource management as well as it is deadlock free, hence the average turnaround time for this technique remains fairly constant.

7. Conclusion. This paper presented an IoT based post-disaster recovery technique for managing the limited resources available such that the system is deadlock free. A new class of deadlock handling technique which handle deadlock by reserving a portion of the resources was proposed. This class eliminates the excessive overhead incurred by the Deadlock Prevention and Avoidance techniques as well as the uncertainty of deadlock occurrence in the deadlock detection and recovery techniques. The proposed Deadlock Free Resource Reservation (DFRR) technique ensures that deadlock will never occur. The correctness of the proposed technique is proved in the form of the theorems. Its effectiveness is shown through motivational examples. The simulation analysis of the proposed DFRR indicate that it has approximately 18 % lower turnaround time than the existing Banker's algorithm. Thus, the proposed technique is a deadlock free technique with optimal overhead.

REFERENCES

[1] M.Wahlstromand D. Guha-Sapir, The Human Cost of Weather-Related Disasters 1995-2015, UNISDR, Geneva, Switzerland,2015.

[2] Solanki, A., Nayyar, A, Green internet of things (G-IoT): ICT technologies, principles, applications, projects, and challenges. In Handbook of Research on Big Data and the IoT (pp. 379-405), IGI Global, 2019

[3] Singh, S. P., Nayyar, A., Kumar, R., Sharma, A. Fog computing: from architecture to edge computing and big data processing. The Journal of Super computing, 75(4), 2019

[4] D.Rathee, K.Ahuja, A.Nayyar, Sustainable future IoT services with touch-enabled handheld devices. Security and Privacy of Electronic Healthcare Records: Concepts, Paradigms and Solutions, 131,2019

[5] Krishnamurthi. R, Nayyar. A, Solanki. A, Innovation Opportunities through Internet of Things (IoT) for Smart Cities. In Green and Smart Technologies for Smart Cities (pp. 261-292). CRC Press,2019

[6] Nayyar, A., Rameshwar, R., Solanki, A. Internet of Things (IoT) and the Digital Business Environment: A Standpoint Inclusive Cyber Space, Cyber Crimes, and Cybersecurity, The Evolution of Business in the Cyber Age, pp.111-152. 2020

[7] Pramanik. P. K. D, Solanki. A, Debnath. A, Nayyar. A, El-Sappagh. S, Kwak. K. S, Advancing Modern Healthcare With Nanotechnology, Nanobiosensors, and Internet of Nano Things: Taxonomies, Applications, Architecture, and Challenges. IEEE Access,vol 8,2020

[8] Balaji. B. S, Raja.P.V, Nayyar.A, Sanjeevikumar. P, Pandiyan.S, Enhancement of Security and Handling the In conspicuousness in IoT Using a Simple Size Extensible Blockchain. Energies, 13(7), 2020

[9] G. M. Lee, N. Crespi, J. K. Choi, and M. Boussard, Internet of things, in Evolution of Telecommunication Services, pp. 257–282,Springer, 2013.

[10] J. Zhang, M. Zhang, F. Ren, and J. Liu, An innovation approach for optimal resource allocation in emergency management, IEEE Transactions on Computers, 2016.

[11] J. Satishkumar, Mukesh A.Zaveri, Resource Scheduling for Postdisaster Management in IoT Environment, Hindawi Wireless Communications and Mobile Computing,2019.

[12] L. Yang, S. H. Yang, and L. Plotnick, How the internet of things technology enhances emergency response operations, Technological Forecasting & Social Change, vol. 80, no. 9, pp. 1854–1867, 2013.

[13] L.Khubnani, MS thesis, https://www.cs.rit.edu/usr/local/pub/GraduateProjects/2161/lhk3416/ Report.pdf

[14] A. Meissner, T. Luckenbach, T. Risse, T. Kirste, and H. Kirchner, Design challenges for an integrated disaster management communication and information system, in Proceedings of the First IEEE Workshop on Disaster Recovery Networks (DIREN '02), vol. 24, pp. 1–7, 2002.

[15] J Satishkumar, Mukesh A.Zaveri, Graph-based Resource Allocation for Disaster Management in IoT Environment, ACM, Proc. Advanced Wireless Information, Data, and Communication Technologies, 2017

[16] Sujoy Sahal,Nitin Agarwal, Priyam Dhanuka, Subrata Nandi, Google map based user interface for network resource planning in post disaster management ACM proc. computings of development, 2013

[17] Kaur M, Mohana R.Static load balancing technique for geographically partitioned public cloud. Scalable Computing: Practice and Experience. 2019 May 2;20(2):299-316.

[18] A. Musaddiq, Y. Bin Zikria, O. Hahm, H. Yu, A. K. Bashir, and S. W. Kim, A Survey on Resource Management in IoT Operating Systems, IEEE Access. 2018, doi: 10.1109/ACCESS.2018.2808324.

[19] E.W. Dijkstra, Cooperating Sequential Processes, Programming Languages, F. Genuys, ed., pp. 103-110, New York: Academic Press,1968.

[20] A.N. Habermann, Prevention of System Deadlocks, Comm. ACM, vol. 12, no. 7, pp. 373-377, 385, July 1969.

[21] R.C. Holt, Some Deadlock Properties of Computer Systems , ACM Computing Surveys, vol. 4, no. 3, pp. 179-196, Sept. 1972.

[22] Coffman, Edward G., Jr.; Elphick, Michael J.; Shoshani, Arie, System Deadlocks . ACM Computing Surveys 3 (2): 67–78. 1971

[23] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Principle , Seventh Edition, Wiley India.

[24] Agrawal, Smriti, Madhavi Devi Botlagunta, and Chennupalli Srinivasulu. A total need based resource reservation technique for effective resource management. International Journal of Computer Applications 68, no. 18 (2013).

[25] Devi, Botlagunta Madhavi, Smriti Agrawal, and Chennupalli Srinivasulu. An Efficient Resource Allocation Technique for UniProcessor System. International Journal of Advances in Engineering Technology 6, no. 1 (2013): 353.

[26] Botlagunta, Madhavi Devi, Smriti Agrawal, and R. Rajeshwara Rao. Effective resource management technique using reservation pool. In International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014), pp. 1-7. IEEE, 2014.

[27] Botlagunta, Madhavi Devi, Smriti Agrawal, and R. Rajeshwara Rao. Dynamic Budget-Thershold based Resource Reservation Technique. Compusoft 8, no. 7 (2019): 3242-3249.

[28] Sheau-Dong Lang, An Extended Banker's Algorithm for Deadlock Avoidance ,IEEE Transactions On Software Engineering,

vol. 25, no. 3, May/June 1999

[29] Odun-Ayo, Isaac, Sanjay Misra, Nicholas A. Omoregbe, Emmanuel Onibere, Yusuf Bulama, and Robertas Damasevicius. Cloud-Based Security Driven Human Resource Management System. In ICADIWT, pp. 96-106. 2017.

[30] Cheng Xin, Xiaozong Yang,A concurrent distributed deadlock detection/resolution algorithm for distributed systems, Proceedings of the 5th WSEAS/IASME International Conference on Systems Theory and Scientific Computation, 2005, pp. 336-341.

[31] Jaehwan Lee, Vincent John Mooney,A novel deadlock avoidance algorithm and its hardware implementation, Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software code sign and system synthesis, 2004, pp. 200-205.

[32] A. Chowdhury and S. A. Raut, A survey study on Internet of Things resource management, Journal of Network and Computer Applications. 2018, doi: 10.1016/j.jnca.2018.07.007.

[33] Kumar J.S, Zaveri M.A, Choksi M, Activity Based Resource Allocation in IoT for Disaster Management, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 220. Springer, 2018

[34] H.Wu,WN.Chin,J.Jaffar, An efficient distributed deadlock avoidance for the AND model ,IEEE Trans. Software Engg 28 (2002)18–29.

[35] Y.B.Ling, S.G.Chen, C.Y.Chiang, On optimal deadlock detection scheduling , IEEE Trans on Computers55(9)(2006)1178–1187.

[36] Shubham Kumar and SaravananChandran, Modified Execution Time based Resource Reservation (METRR) Algorithm), 3rd International Conference on Business and Information Management (ICBIM) 2016.

[37] Yin, W., Stephane, L., Terence, K., Manjunath, K., Scott, M.,The Theory of Deadlock Avoidance via Discrete Control , 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages, PoPL,202- 214(2009).

[38] Hongwei Wang; JinfengTian; Mingqi Li; Weixin Mu; XiangchuanGao,Banker's algorithm based resource allocation in next generation broadcasting wireless systems . IEE Proc Int'l Conf Communications and Networking in China (2015)

[39] Xiang Xiao, Jaehwan John Lee. A parallel multi-unit resource deadlock detection algorithm with O(log2(min(m, n))) overall run-time complexity. J. Parallel Distrib. Comput. 71. pp. 938–954. (2011)

[40] P. H. Shiu, Yudong Tan and V. J. Mooney, A novel parallel deadlock detection algorithm and architecture, Ninth International Symposium on Hardware/Software Codesign. CODES 2001 (IEEE Cat. No.01TH8571), Copenhagen, Denmark, 2001, pp. 73-78.

[41] E. E. Ugwuanyi, S. Ghosh, M. Iqbal and T. Dagiuklas, Reliable Resource Provisioning Using Bankers' Deadlock Avoidance Algorithm in MEC for Industrial IoT, in IEEE Access, vol. 6, pp. 43327-43335, 2018, doi: 10.1109/ACCESS.2018.2857726.

[42] H. K. Pyla and S. Varadarajan. Avoiding deadlock avoidance. PACT '10, pages 75–86, New York, NY, USA, 2010. ACM.

[43] Youming Li, A Modified Banker's Algorithm, in Springer Innovations and Advances in Computer, Information, Systems Sciences, and Engineering, pp 277-2819 (2012).

[44] Pankaj Kawadkar, Shiv Prasad, Amiya DharDwivedi, Deadlock Avoidance based on Banker's Algorithm for Waiting State Processes in International Journal of Innovative Science and Modern Engineering (IJISME) ISSN: 2319-6386, Volume-2 Issue-12, November 2014

[45] Dixit, Kshipra and Ajay Khuteta. A Dynamic and Improved Implementation of Banker's Algorithm. International Journal on Recent and Innovation Trends in Computing and Communication 5, no. 8 (2017): 45-49.

Edited by: Anand Nayyar
Received: Apr 4, 2020
Accepted: Jul 1, 2020