International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

Illustration of Safe and Unsafe State Using Transition Table and Java Simulation

October Lambert Kekebou Erefaghe

Email: *octoberlambert[at]gmail.com*

Abstract: ***The burden of allocation and utilization when it comes to system resources lies completely in the domain of the operating system. Giving resources to processes as they require them for execution is a necessary task that demands all conscientiousness as it pushes the system to different states because the limitedness of system resources don't leave processes or programs with ample supply of their desired demand of resources to be allocated when needed, this implies that there is always a wait pull or queue for system resources as long as processes keep requesting for resource. The crucial aspect in allocating or de - allocating resources is the technicalities involved that pertains to ensuring that processes are not holding resources other processes will need without making use of them. When this two task allocation and de - allocation is not sufficiently managed many processes can be resource deficient waiting on the system for resources to be allocated that are kept by processes which are waiting for more resources which leads to concept starving, and the resultant effect of this is called unsafe state in which processes cannot execute to completion for not getting more resources, and it can be worsen a deadlock state which no execution of process is possible. However, with efficient algorithms and techniques a system can allocate its resources to all request and all waiting processes can run to completion. There is always a sequence or order to follow to successfully allocate resources to processes for to avoid unsafe state leading to deadlock state, this sequence is known as safe sequence. This research work is to illustrate safe and unsafe state using transition table, and simulate how a system can be in safe state using the deadlock avoidance algorithm implemented in java programming language.***

Keywords: safe state, unsafe state, deadlock avoidance, processes, resources, transition table

1. Introduction

The operating system is the brain and controller of system resources and executable programs or application, this is seen as the most important task in systems because without which users can system activities. The operating system manages different types of resources which can be semaphores, files, input/output devices, CPU cycles etc. the limitedness of these resources demands proper utilization and management for task that will request for them. Process execution is made happen by resources, hence, the concept resources and processes are inseparable in OS [1]. A process can be defined as an instance of a computer program in execution, which undergoes frequent state and attribute changes. It's an active entity as opposed to program which is considered to be a passive entity. The trajectory wise fashion followed by the operating system for any computational task is properly guided by set of instructions called programs. The actual performing or carrying of the instructions is what we tag process. How and when these set of operations or activities are enforced to execute are part of the sole managerial responsibilities of the operating system. On the other hand, a resource is an inclusive term of all system resources such as Memory space, CPU cycles, I/O devices (like printer, tape drive etc.), files and semaphores. Some resources are used by processes that require uninterrupted resource utilization to be completed while other resources require operating system control for processes to safely exchange the utilization of the resource [1].

A process using a resource goes through three sequences of events Request event, Use event and Release event [6]. When the operating system is tasked to provide resources by a process is called Request event, in an instance where the resource is unavailable the process waits. In the Use event, the process does whatever operation it need to do using the resources that it requested. The usage of resources by a process is characterized with Release event after successfully utilizing the resource which is a returnable.

One necessary condition for all processes to be completed is proper allocation and de - allocation of resources resident in the system. A proportion or whole of the resources requested by a process can be occupied by another process and sometimes the process only keep the resources while waiting for more depending on the need, when this happen processes waits longer periods to get executed or sometimes never executed, this is a conflict that is pertinent when allocating resources. This is where carefulness and proper allocation is applicable and considered to be very important for processes to be completed. A system state can be classified into safe, unsafe and deadlocked through resource allocation and utilization. A resource allocation that will be satisfy or go round all active request in an orderly fashion to avoid plunging the system to deadlock state is called safe state, that is it's a state where processes even being starved a bit will still be served. An unsafe state means that some of the pending processes will not be serviced, and the danger of reaching a deadlock state is imminent [2]. An unsafe state may lead to deadlock but not all unsafe state guarantee deadlock. Since safe state cannot lead to deadlock, we can avoid unsafe state by giving the operating system prior notice of what resources a process may request.

The significance of safe and unsafe state is for handling of deadlock problems because the challenge of deadlock is very serious and as such must be clearly understood and handled properly in different environments cutting across multiprogramming systems, database environments, distributed systems, networking, hence, a clear illustration of safe and unsafe states is needed in which deadlock is categorized as a subset of unsafe state, in this regard this paper uses the concept of transition table to illustrate safe and unsafe state to give explicit comprehension and a

simulation of safe and unsafe state using the Bankers' algorithm in the modern programming language java.

2. Safe State

Safe state is used to describe when the system can be able to allocate requested resources maximally to each process demand that will satisfy all pending processes for complete execution. The logical order followed for all pending processes for resource request to be allocated is known as the safe sequence. Resource request are viewed to know the possibility of safe sequence if positive then the system is safe. This means that all processes including pending ones can get resources for complete execution.

Safe states are deadlock free, unsafe states leads to deadlock which is a subset of unsafe state, implying not all unsafe states lead to deadlock. Deadlock occurs in multiprocessing and distributed environments, which is prompted by the conditions "hold and wait", "mutual exclusion", "circular wait", and "no preemption", which are necessary conditions for the occurrence of deadlock [3].

3. Handling deadlock

Deadlock can be handled by avoidance, prevention, ignoring and detection, in each case algorithms are used [3]. Indifferent mechanisms employed to proffer solution or handle deadlock, avoidance method proves to be preferable as it tries to only grants resource allocation request that will retain the safe stateof system using efficient algorithms, implying at all cost the deadlock possibility is avoided.

A. Avoiding deadlock

Deadlock is avoided through the usage of banker's algorithm. It consists of resource request algorithm and safety algorithm to request for resources and test for safety by simulating the allocation for predetermined maximum possible amounts of all resources. The applying of the algorithm is relying on the predetermined maxima of each process's demand and the allocation is done in the order of safe sequence.

a) Resource Request Algorithm

The Resource Request Algorithm uses the following variables.

Supposing processes count in the system is denoted as Pand resources count is denoted as R

Available: - denote available resources number of each type.
Maximum: - denotes maximum resources for a process.
Allocation: - denotes resources supplied of each instances.
Need: - denotes the needed total resources for each process [7].

Need [i] = Maximum [i] – Allocation [i]

Let $Request_i$ be the request array for process P_i.

$Request_i$ [j] = k means process P_i wants k instances of resource type R_j. at the demand of P_i one option is invoked for performance

1) If $Request_i <= Need_i$, Goto step (2);
Else, indicate an error since maximum claim has been exceeded.
2) If $Request_i <= Available$, Goto step (3);
Else P must wait as there are no more resources.
3) At this point take that allocation is supplied.
Compute accordingly as below:
$Available = Available – Request_i$
$Allocation_i = Allocation_i + Request_i$
$Need_i = Need_i – Request_i$

b) Safety algorithm

Some parameters used:
Need [i] – denotes the remaining resource need of each process.
Work –denotes resources currently available
Finish [i] – indicator of a process if its analyzed or not.

1) Given Work and Finish be vectors of length 'R' and 'P' respectively.
Initialize: Work = Available
Finish [i] = false; for i=1, 2, 3, 4, …. n
2) Compute an i^{th} index such that both
a) Finish [i] = false
b) Need [i]<= Work
if no such i exists goto step (4)
3) Work = Work + Allocation [i]
Finish [i] = true
goto step (2)
4) if Finish [i] = true for all i, then the system is in a safe state

B. Illustration of Safe and Unsafe States using transition table with single resource type

Supposing a system has ten (10) resources say magnetic tape drive that are been allocated to three (3) processes P_1, P_2, and P_3 as show in Table 1.0.

Table 1

Process	*Allocated*	*Maximum Need*
P_1	3	9
P_2	2	4
P_3	2	7
Free: 3		

For instance, if process P_2 made request for more tape drives and more two (2) tape drive is allocated then process P_2has all its resources and then will be executed completely and release all the four (4) tape drives back to the system which will make the total free tape drives to be five (5). With Available five (5) tape drives process P_3 can get all its needed tape drives then complete its task and return the total seven tape drive to the system, and lastly process P_1 can get its request and finish the task. So the safe sequence for table 1.0 above is $<P_2, P_3, P_1>$. This is illustrated in the transition table diagram in Figure 1.0.

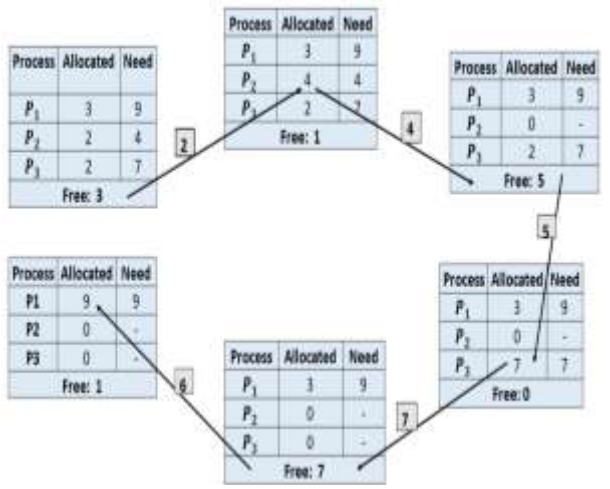


Figure 1: Safe state transition table diagram of a multiple instances of a resource

In the transition table in figure 1.0, the direction of the arrow head denotes allocation of resources and the small box with numbers indicate the number of resources allocated to the process.

Supposing processP_1request for more tape drives and the system give 1 drive to P_1out of the 3 free drives, process P_1will not still run to completion since it need more 5 drives. Then process P_2request is granted with the remaining 2 free drives which only process P_2 will be completed and release its allocated 4 drives, but neither process P_1nor P_3 can be completed as the free drives cannot meet the maximum needs of either process, hence, the sequence $<P_1, P_2, P_3>$ is an unsafe sequence. The unsafe sequence is represented in the transition table diagram in figure 2.0.

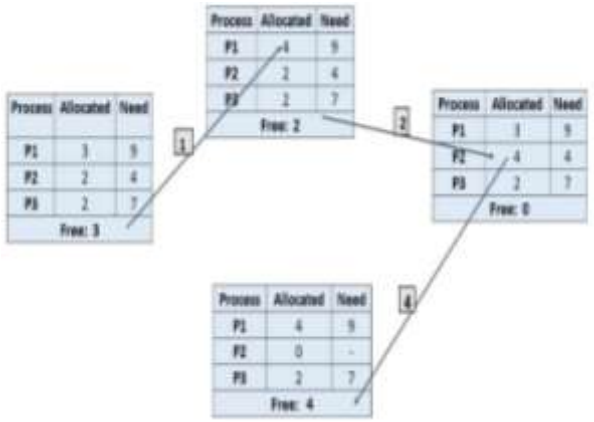


Figure 2: Unsafe state transition table diagram for multiple instances of a resource

For multiple instances of different resource types take a look at Table 2.0.

Table 2

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_1	3	6	2	1	6	8	8	5	4	3	2	2
P_2	5	4	3	1	7	9	5	7				
P_3	2	5	0	4	5	8	4	9				
P_4	1	0	4	2	4	3	5	4				

Table 2.0 contains four (4) processes P_1, P_2, P_3, P_4 and four (4) different types of resources denoted as A, B, C, D. The allocation, maximum need, available of each resource type is given as presented in the table. The need variable is what we have to calculate and it can be done with the formula
Max [4, 4]– Allocation [4, 4] = Need [4, 4]

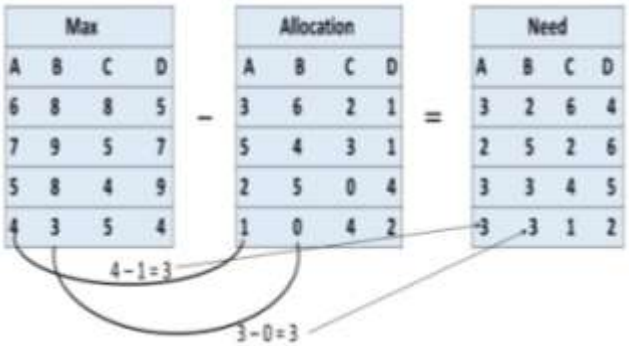


Figure 3

Having gotten the need matrix, the illustration of the safe state is in Figure 3.0.

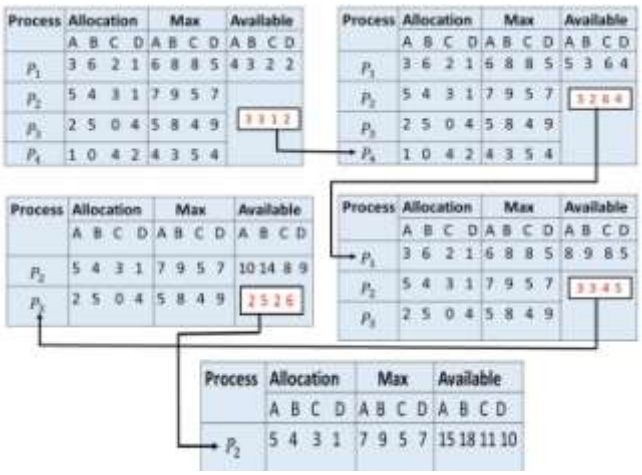


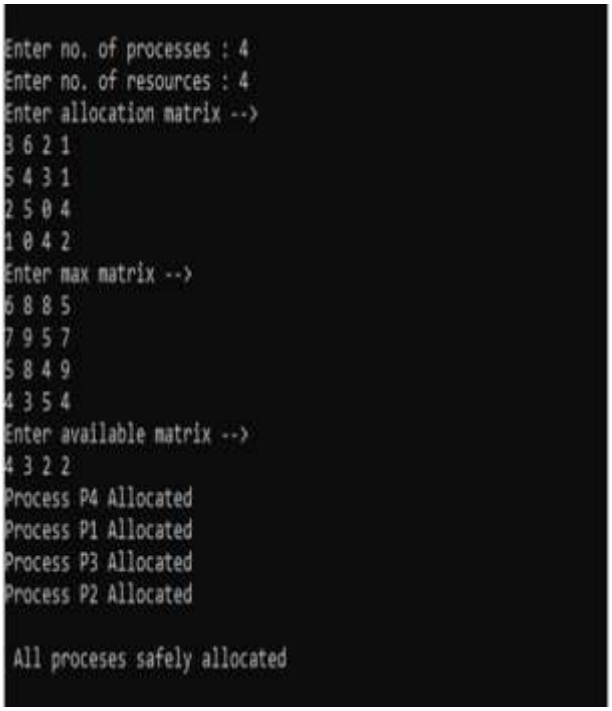
Figure 4: Safe state transition table diagram of multiple instances of multiple resources

In the transition table diagram in figure 4.0, the arrow points to the processes that when its request is granted the system can be in safe state. The new allocation is indicated by the box in the available column for the four different resources and the safe sequence is $<P_4, P_1, P_3, P_2>$ for the Table 2.0

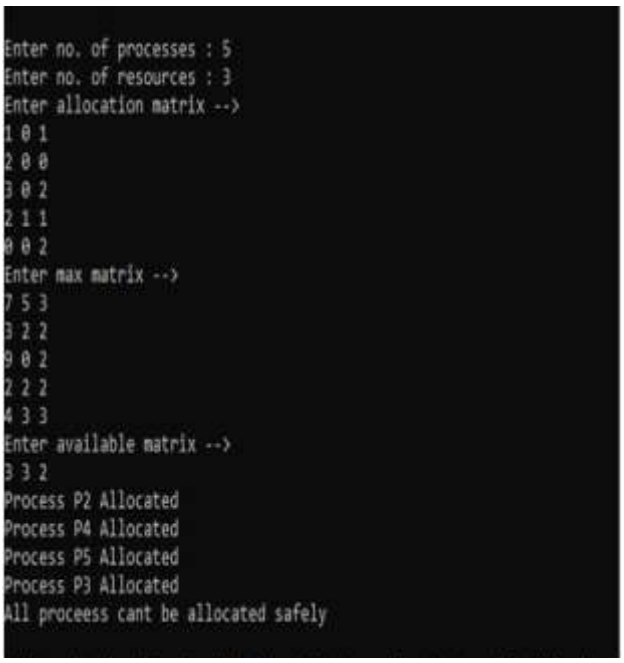
4. Simulation of deadlock avoidance algorithm

A. System details

The coding is done using java programming language. The Operating system used is 64 - bit Windows 10 with 6 GB RAM. The simulation program was test in a personal system and the screenshot of the output is displayed bellow. The simulation was tested with different sizes of windows command line inputs of processes and resources and there were no observable limits of processes and resources.



Screenshot 1.0



Screenshot 2.0

5. Conclusion

The concept of deadlock is very important in this advanced stages of technological development especially for software development, and having a clear understanding is a cutting - edge advantage one can have to apply the principles of how to handle deadlock in different fields of computing, networking and in industries.

References

[1] G. Dimitoglou, 1998, Deadlocks and Methods for their Detection, Prevention and Recovery in Modern Operating Systems, ACM SIGOPS Operating Systems Review available https: //www.researchgate. net/publication/220623929

[2] P Kawadkar, S Prasad, and A D Dwivendi, 2014, Deadlock Avoidance Based on Banker's Alghoritm for Waiting State Processes, Int. Journal of Innovative Science and Modern Engineering (IJISME), vol 2

[3] A. S. Tanenbaum, *Modern Operating Systems.* Englewood Cliffs, NJ: Prentice Hall, 1992, pp.233 - 300.

[4] E. W. Dijkstra, *Co - operating Sequential Processes.* London, UK: Academic Press, 1965, pp.43 - 112.

[5] K. Hameed, et. al.2016, Resource Management in Operating Systems - A Survey of Scheduling Algorithms, Int. Conf. on Innovative Computing (ICIC), (Pakistan: University of Management and Technology) vol 1

[6] M. A. Khan, (2014, july) Comparison between Proposed and Existing Algorithms for Deadlock Avoidance and Recovery available: https: //www.researchgate. net/publication/288592077

[7] https: //www.geeksforgeeks. org/bankers - algorithm - in - operating - system - 2/.