Volume 3, Issue 3
June 2014

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - -708X

Comparison between Proposed and Existing Algorithms for Deadlock Avoidance and Recovery

[1]**Rabia Shakir**, [2]**Muhammad Ali Khan**, [3]**Muhammad Adnan Khan**

[1, 2]Federal Urdu University of Arts, Science and Technology, Islamabad, Pakistan
[3] Schools of Engineering and Applied Sciences (SEAS), ISRA University, Islamabad, Pakistan
Email: [1]rabi.khan91288@yahoo.com, [2]alikhan09111@gmail.com, [3]adnan_600@yahoo.com

ABSTRACT

The main objective of this paper is to provide concrete solution who constraints the system to move in safe state. It also ensures that the system is not in unsafe state. Deadlock is one of the major disputes encountered in operating system. It occurs due to ineffective allocation of resources to the processes. To prevent the system from deadlock, existing algorithms and techniques for deadlock prevention, recovery and avoidance have been analyzed. This paper offerings an efficient algorithm Shortest Job First with respect to Claim (SJFC) and Suspend Process with Maximum Need (SPMN) strategy to prevent, avoid and recover a system from deadlock. It presents a comparison between suggested and recent algorithms and strategies. Virtues and imperfections have also been designated.

Keywords: *Deadlock Avoidance, Deadlock Recovery, Deadlock Algorithms.*

1. INTRODUCTION

1.1 Deadlock

There are several resources in the system. These resources acquired by many processes. A deadlock is a state in which a process access a resource and waiting for another resource acquired by another process. The system may drives in a block state. This situation of a system is called Deadlock.

Figure 1: Deadlock

In Figure 1, there are two persons that want to hold a single resource (a book). Both cannot access at the same time, so deadlock may occur. “The deadlock conditions are uninvited in any system meanwhile they violence the system’s efficiency” [1].

1.2 System Model

“System model includes determinate states and unallocated resources”[2].
Deadlock occurs between the processes. The set of finite processes are P= {P_0, P_1, P_2… P_n} and a set of finite resources are R = {R_1, R_2… R_n}.

Every process allocates a resource. A process can:

i. Request a resource
ii. Use a resource
iii. Release a resource

1.3 Conditions for deadlock

Deadlock can be considered in four necessary conditions. If they occurs at the same time in the system then deadlock occurs.

i. Mutual Exclusion
ii. Hold and wait
iii. No Preemption
iv. Circular wait

In mutual exclusion, at a time one process utilize one resource .Multiple processes cannot access the same resource at the same time. If a process request for a resource allocated by another process, then the requesting process must be delayed until the resource has been released. There should be at least one non-sharable resource.

In Hold and wait, a process currently capture a resource and waiting for another resource to be held by another process.

In No Preemption, processes allocate the resource and deallocate the resource when its task has been completed.

In circular wait, there is a set of waiting processes { $P_{0,}$ $P_{1,}$ P_2,…,P_n } such that P_0 is waiting for a resource accessed by P_1, where P_1 is waiting for a resource accessed by P_2,…,P_n is waiting for a resource accessed by P_0.

Volume 3, Issue 3
June 2014

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - -708X

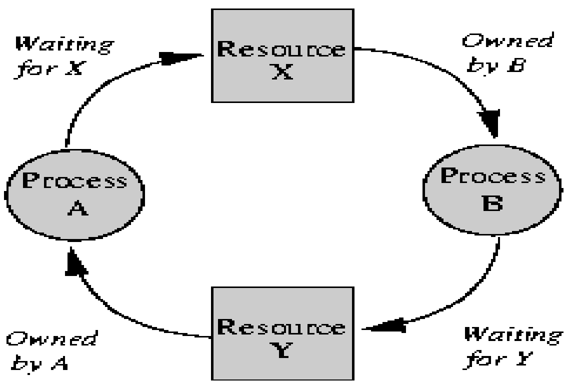

Figure 2: Deadlock Occurrence Conditions

In figure 2, all above mentioned conditions arise. Mutual Exclusion occurred because, at a time a single resource held by a process .process A allocates a resource Y while process B allocates a resource X.

Hold and wait condition occur because, a process presently captured a resource and waiting for another resource held by another process. Process owned by a resource Y and waiting for a resource X preserved by process B.

No preemption as a process can release a resource when its task has been completed. Process A and process B will release a resource Y and X when they accomplished their jobs.

Circular wait because each process is waiting for a resource occupied by another process.

1.4 Resolution of Deadlock

To resolve deadlock occurrence, the system must not fulfill any one from the four conditions of deadlock occurrence.

2. RESOURCE ALLOCATION GRAPH

"Processes and resources communicated in Resource Allocation Graph"[3].

"The deadlock detection and resolution constantly wishes that processes should be terminated. Due to this purpose numerous issues must be focused.

1) Termination is more exclusive than wait.
2) Unwanted termination result in unused system resources.
3) Quantity of terminated jobs should be minimized"[4].

"System move in unsafe state in deadlock prevention and avoidance"[5][6].

Resource Allocation Graph can be used to identify dead lock in the system.
Identification of deadlock is easy by this Graph.

Resource Allocation Graph includes:

i. No of Processes P={P_1, P_2, ……, P_n}
ii. No of Resources R={R_m, R_{m+1}, …., R_m}
iii. Edges E= Links between processes and resources

2.1. Symbolic Notation in Resource Allocation Graph

Following notations are used for Resource Allocation Graph

i.	Process	◯
ii.	Resource	▭
iii.	Instances/Copies	●

2.2. Arrow Notation of Resource Allocation Graph

i. $P \longrightarrow R$ means [Request for resource]
ii. $R \longrightarrow P$ means [Allocation of resource]

2.3. Resource Allocation Graph Example

Assume there is a set of processes and resources.

Process P = {P_1, P_2, P_3}

Resources R = {R_1, R_2, R_3, R_4}

Edges E = {$P_1 \longrightarrow R_1$, $R_1 \longrightarrow P_2$, $R_4 \longrightarrow P_4$,$P_4 \longrightarrow R_3$, $P_3 \longrightarrow R_2$, $R_3 \longrightarrow P_2$, $P_2 \longrightarrow R_4$ $R_2 \longrightarrow P_1$, $P_3 \longrightarrow R_4$, }

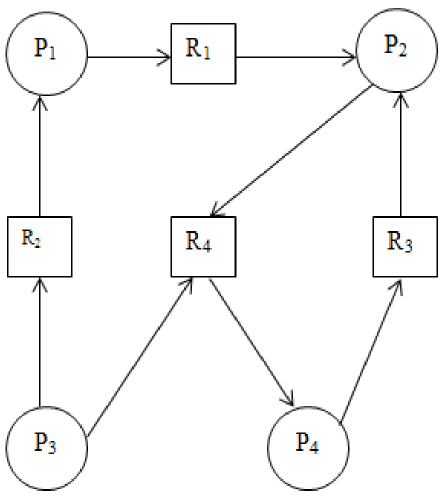

Figure 3: Resource Allocation Graph

2.4. Deadlock Identification in Resource Allocation Graph

If a resource allocation graph founds no cirscular path then there is no deadlock. If it contains a cycle, then the system is in deadlock state.

Volume 3, Issue 3
June 2014

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - -708X

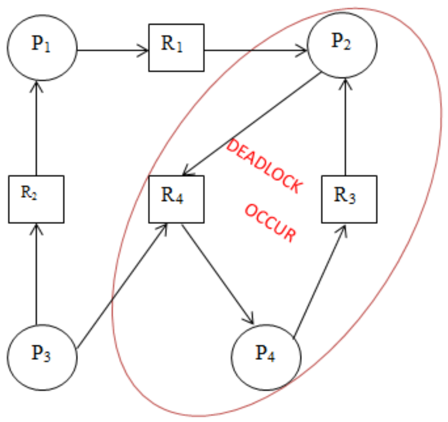

Figure 4: Deadlock Identification in Resource Allocation Graph

In Figure 4, there is only one deadlock is identified because the process $P_2P_4P_2$ creates a cycle.

3. WAIT FOR GRAPH (WFG)

"System moves in unsafe state when processes communicates with each other"[7,8].

"In a Wait-For Graph (WFG)

- Process denoted by a circle;
- An Edge described by a link between nodes" [9].

Wait for Graph can be constructed through Resource Allocation Graph. This graph contains just Processes and Edges. It doesn't contain resources.

Wait for graph is used to detect deadlock & then deadlock can be resolved.

If two or more processes are combined and make a cycle then there must be a deadlock in system.

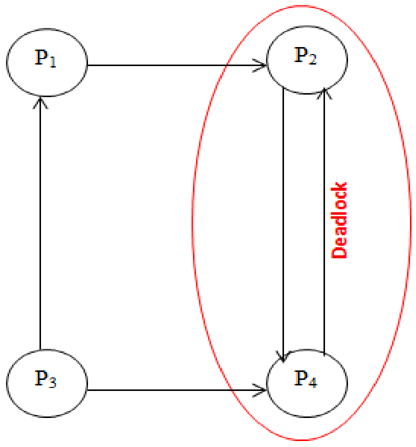

Figure 5: Wait for Graph

There is a circular wait between the processes P_2 and P_4, so deadlock occurs. Deadlock is between $P_2P_4P_2$.

4. RESOLVING DEADLOCK

Methods for deadlock resolution can be described in two main categories:

4.1. Deadlock Prevention

In deadlock prevention, resources can be allocated to the processes before the task starts. During execution resources cannot increased/decreased. Processes describes earlier that which resources they have to allocate in upcoming.

"Through escaping cycles, the system can be eluded form unsafe state"[10].

We assume that any one of the four necessary conditions must not occur. If it occurs then we simply prevent circular wait. To resolve deadlock, apply any one of the two cases:

i. Case-I : Resource Preemption

Resource should be preemptive. Deallocate a resource from a one process and allocate this resource to another process.

For instance; from the wait for graph, deallocate a resource from P_2. In this state only process can exist and links can be removed that comes in and out from the process.

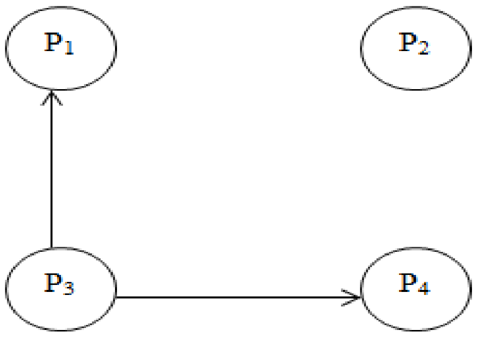

Figure 6: Resource Preemption

Figure 6 illustrates that there is no circular wait because all the resource are preempted from P_2. So there is no deadlock.

Now the question arises, from which process a resource should be preempted. The solution is a process whose burst time is high or captured maximum resources then deallocate some of its resources.

ii. Case-II : No-Preemption

First observe that which the common process among all is. The processes who creates circular wait then try to terminate them one by one.

In process termination, remove the specific process as well as their linking edges. For example; from the wait for graph, terminate P_4. In this situation, a process and its linking edges can be removed.

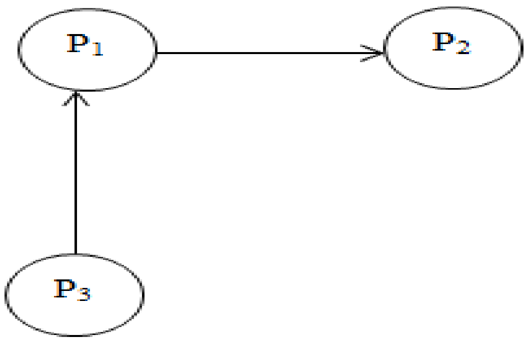

Figure 7: No Preemption

Again the issue is how to terminate a process. The key is try to terminate a low priority, maximum remaining burst time or have maximum resources process.

4.2 Deadlock Avoidance

"When a process request for a resource currently held by another process, the deadlock avoidance algorithm decides whether a requesting process have to wait or one of the waiting process should be terminated"[11].

4.2.1. States

The system may goes from safe to unsafe state and from unsafe to safe state. It totally depends upon carefully allocation of resources between the processes.

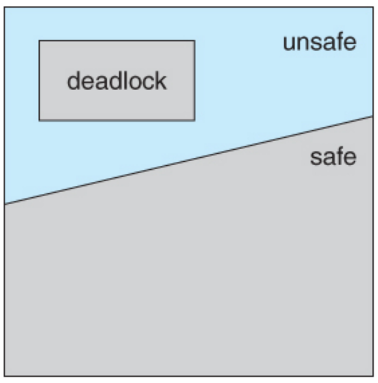

Figure 8: States of Deadlock

i. Safe State: If the system is in safe state , it means that there is no deadlock.

ii. Unsafe State: is the system is in unsafe state, then the system must have deadlock.

4.2.1.1. Safe System Example

This example describes, whether the system is in safe state or not.

Process	Max Need	Allocated	Claim
P_1	10	5	5
P_2	4	2	2
P_3	9	2	7

Total Resources = 12
Total Processes = 3
Total Allocated = 5+2+2 =9
Current Available = Total – Allocated
= 12 – 9
= 3

Then allocate the current available resources carefully.
Allocate 2 from the 3 current available resources to the process P_2.

At this stage Current available = 3-2 =1

After completion the task, P_2 releases 4 resources.

Current Available = 1

Released resources by P_2 = 4

Now, Current Available = 4+1= 5

Now, check the claim of remaining two processes whether it is equal, less or greater than current available resources.
Allocate the Current available resources =5 to the process P_1 to fulfill its need.
P_1 released 10 resources when its task has been completed.

Now current available =10

Finally allocate the resources to P_3.
Its maximum need =9, Allocated = 2, claim=7.
Current available = 10-7=3

Assign 7 from the current available resources then
Max need = Allocate + claim
9 = 2 + 7
9 = 9

After completed its task, P_3 releases 9 resources. Each process successfully completed its task. The carefully allocation of resources to $P_2P_1P_3$ forces the system to move in safe state.

4.2.1.2. Unsafe System Example

Process	Max Need	Allocated	Claim
P_0	10	5	5
P_1	4	2	2
P_2	9	3	6

System will drive in unsafe state if resources allocated to the processes in this manner $P_1P_0P_2$.

4.2.2. Banker's Algorithm

In banker's algorithm no of processes and resources are limited.

No of resources ≤ system's resources

If no of resources allocated are greater than the system's resources, Algorithm will fail.

"Request = Demand for a process P_n. If a process requests m processes then process P_n wishes for m instances of a resource"[12].

Banker's algorithm includes:

i. Mutual exclusion
ii. Hold and Wait
iii. No preemption

"Dijkstra's Banker's algorithm is one of the efficient deadlock avoidance algorithm that is currently used"[13, 14, 15, 16, 17].

4.2.2.1. Banker's Algorithm example

Process	Max Need				Allocated			
	A	B	C	D	A	B	C	D
P_0	1	10	10	3	0	1	2	3
P_1	2	8	7	10	1	0	0	0
P_2	2	14	10	15	1	4	2	3
P_3	0	1	4	2	0	0	3	2
P_4	1	9	7	7	1	7	3	6

Solution:
Matrix Need

Process	Claim			
	A	B	C	D
P_0	1	9	8	0
P_1	1	8	7	10
P_2	1	10	8	12
P_3	0	1	1	0
P_4	0	2	4	1

Claim = Max Need – Allocated
Available =Total – Allocated
= A B C D
= 0 2 2 1

i. First Iteration:

Maximize the possibility that system should be in safe state.

Check the process who have claim less than or equal to available.

Allocate to P_3 = 0 1 1 0
Current Available = (0 2 2 1) – (0 1 1 0)
= 0 1 1 1.
P_3 completed its task and releases 0 1 4 2 resources.
P_3 terminated.
Now, current available = (0 1 1 0) + (0 1 4 2)
= 0 2 5 3

ii. Second Iteration:
Allocate the resources to P_4 = 0 2 4 1
Current Available = 0 2 5 3 – 0 2 4 1

Now claim for P_4

After complete P_4, it releases 1 9 7 7 resources.
Current Available = 0 0 1 2 + 1 9 7 7
= 1 9 8 9
P_4 Terminated.

iii. Third Iteration:

Allocate the resources to P_0= 1 9 8 0
Current Available = 1 9 8 9 – 1 9 8 0
= 0 0 0 9

P_0 terminated and released resources = 1 10 10 3
Now,
Current Available = 0 0 0 9 + 1 10 10 3
= 1 10 10 12

iv. Fourth Iteration:

Assign to P_1 = 1 18 7 10
Current Available = 0 2 3 2
P_1 will release resources = 2 8 7 10

Volume 3, Issue 3
June 2014

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - -708X

Now,
Current Available = 2 10 10 12

P_1 Terminated.

v. Fourth Iteration:

Assign to P_2 = 1 10 0 12
Current Available = 1 0 2 0

P_2 released resources = 1 10 8 12
Now,
Current available = 1 0 2 0 + 1 10 8 12
= 3 14 12 13
P_2 terminated.

Solution to be system in safe state is
$P_3P_4P_0P_1P_2$ or $P_3P_4P_0P_2P_1$

5. PROPOSED ALGORITHM

5.1. SJF w.r.t. Claim (SJFC)

This algorithm forces to assign the resource to process who have shortest burst time and claim.

In Shortest Job First w.r.t Claim (SJFC) technique includes

- Arrival time (A.T)
- Burst time claim (B.T)
- Allocated resources
- Maximum need of a process

5.2. Conditions for solving Algorithm

i. Check the process with shortest A.T.
ii. If two or more processes have same A.T then check its Burst Time.
iii. If processes have different A.T then find the process with shortest burst time.
iv. Now assign the resource to the process whose claim is Shortest.

5.3. System is in safe state or not

Process	Max Need	Allocated	Claim	B.T	A.T
P0	11	6	5	0.5	0.1
P1	5	3	2	0.2	0.1
P2	10	7	3	0.3	0.2

Total Resources = 18
Allocated = 16
Current Available = Total resources – Allocated
= 18 – 16
= 2
Check the process with least arrival time:

P0 and P1 have the least arrival time.
Now check the process whose claim ≤ Current Available.

i. Iteration 1

Assign the 2 resources to P1 in order to fulfill its maximum need.
Now, Current Available = 0
P1 completed its task and releases 5 resources.
P1 Terminated.

Now, Current Available = 5.

ii. Iteration 2

As Current Available = 5

There are two processes in memory:

Check the arrival time of P0 and P2.

P2 have highest A.T then P0 but P2 have shortest Burst time.

Finally check the process whose claim is low than current available need. Assign the resource to that process whose claim and burst time is short.

Assign the resource to P2

Maximum need of P2 = 10
Allocated =7
Claim of P2 = 3
Current Available = 5
Assign 2 resources from current available to P2.

Now Current available= 3

Resources released by P2 =10

Now, Current Available = 10 +2 =12

P2 terminated.

iii. Iteration 3

At this stage, there is only one process in system, P0.

Max Need of P0 = 11
Allocated = 6
Claim = 5
Current Available = 12

Assign 5 from current available to process P0.

Resources Released by P0= Allocated + Claim
= 6 + 5 =11
Now Current available = 12 – 5

=7

P0 terminated

System is in safe state if resources allocated in this manner, P1 P2 P0.

5.4. Key To check system is in Safe State

After termination of all processes:

Total Resources= Current Available + Released Resources

It ensures that system is in safe state.

Now, Total resources =18
Released = 11
Current resources = 7

Total resources = Released + Current Available
18 = 11 +7
18=18

5.5. Benefits of Proposed Algorithm

i. Ensures that the system is in safe state.
ii. Allocation of resources is convenient.

6. WEAKNESS OF CURRENT STRATEGY

Current system has following problems:

i. Problem in No-Preemption

In this strategy

- The process can be removed and the edges associated with it are also removed.
- System stays in safe state but limited process fulfill their needs.
- Process removed who creates deadlock

ii. Problem in Resource Preemption

In this strategy

- Process remains in the system but all edges associated with it are removed.
- Again all the processes unable to fulfill their needs.

7. PROPOSED STRATEGY

This strategy will eliminate the issues addressed in current strategy.

7.1. Suspend Process with Maximum Need (SPMN)

For instance there is a wait for graph; there are two deadlocks in the figure.

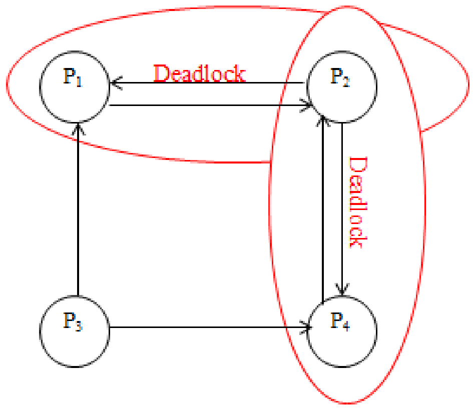

Figure 9: Deadlock

Apply two steps to resolve this deadlock.

i. Step 1

Suggest the process with maximum resources i.e., P_2.

Remove the outgoing edges of P_2.

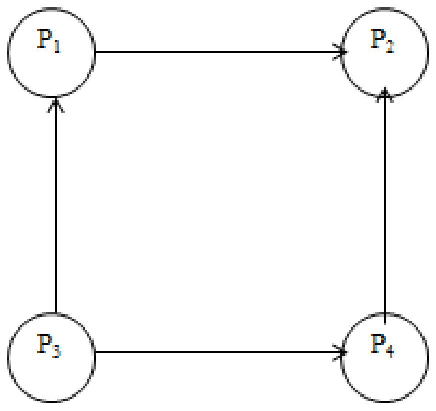

Figure 10: Remove edges from P_2

ii. Step 2

P_1, P_3, P_4 has completed its requirement, but there is still P_2 whose resources associated with P_1 and P_4. P_3 also terminate after completion in step 1.

Remove edges from P_1 to P_2 and P_4 to P_2 because they completed their task in step 1.

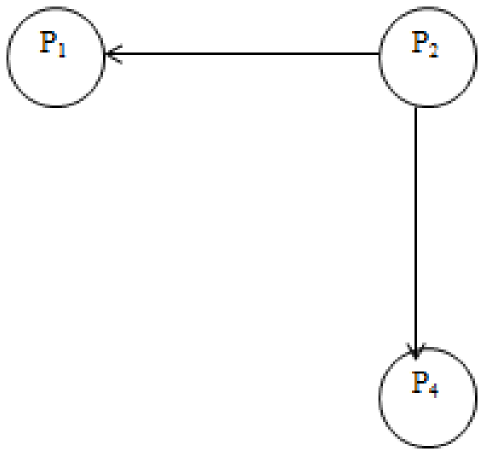

Figure 11: Assign resources to P_2

7.2. Benefits of Proposed strategy

i. Every process in a system fulfills its need.

ii. Deadlock eliminated in the early stage.

iii. Ensures that the system is in safe state.

8. CONCLUSION

This research explores an algorithm & strategy for deadlock prevention & recovery. Proposed and existing algorithms and strategies have been examined. Merits of existing and demerits of current algorithm and techniques have also been determined. After investigating the results it is concluded that, the proposed algorithms SJFC for deadlock prevention and SPMN for deadlock recovery is best among existing algorithm.

REFERENCES

[1]. Michal arnay. Comparing Versions of Banker's Algorithm for Deadlock Avoidance in Resource Allocation Systems

[2]. Mark Lawley, Assistant Professor, Deadlock Avoidance for Sequential Resource Allocation Systems. Hard and Easy Cases,School of Industrial Engineering, Purdue University

[3]. J. L. Peterson. Operating system concepts. Addison-Wesley, 1981.

[4]. Pooja Chahar and Surjeet Dalal Deadlock Resolution Techniques, International Journal of Scientific and Research Publications, Volume 3, Issue 7, July 2013.

[5]. Kunwar Singh Vaisla, Menka Goswami, Ajit Singh, “VGS Algorithm an Efficient Deadlock Resolution Method”, International Journal of Computer Applications, Vol.44, April 2012.

[6]. Terekhov, T. Camp, “Time efficient deadlock resolution algorithms”, June 1998.

[7]. http://www.cs.colostate.edu/~cs551/CourseNotes/Bankers.html.

[8]. Elmagarmid A. K. A Survey of Distributed Deadlock Detection Algorithms, sigmod record, Vol. 15, No.3, pp. 37-45.

[9]. http://www.cs.colostate.edu/~cs551/CourseNotes/Deadlock/WFGs.html

[10]. Swati Gupta, Deadlock Detection Techniques in Distributed Database System, International Journal of Computer Applications, Volume 74, July 2013.

[11]. Meenu Vijarania, Swati Gupta,Analysis for Deadlock Detection and Resolution Techniques in Distributed Database, International Journal of Advanced Research in Computer Science and Software Engineering Analysis for Deadlock Detection and Resolution Techniques in Distributed Database , Volume 3,July 2013

[12]. Silberschatz A, Galvin PB, Gagne G. Operating System Concepts 2012, 8th edition, Wiley India.

[13]. E.W. Dijkstra, “Cooperating sequential process,” Technological University, Eindhoven, the Netherlands, Tech.Rep.EWD-123.

[14]. A.N. Habermann, “Prevention of system deadlocks,” communications of the ACM, vol.12, pp.373-377.

[15]. J.W. Havender, “Avoiding deadlock in multitasking systems,” IBM Systems Journal, vol.2, pp.74-84.

[16]. R.C.Holt, “Some deadlock properties of computer systems,” ACM Computing Surveys, vol. 4, pp.179-196.

[17]. T. Araki, Y. Sugiyama, and T. kasami, “Complexity of the deadlock avoidance problem,” 2nd IBM Symposium on Mathematical Foundations of Computer Science, pp.229-257.

AUTHOR PROFILES

1. Rabia Shakir is doing MS (CS) degree from FUUAST, Islamabad, Pakistan. She is serving as a Lecture in the department of Computer Science, Federal Urdu University of Arts, Science and Technology, Islamabad, Pakistan. She is also occupied as a Lecture in the department of Computer Science, Islamabad Model College for Girls (Post Graduate), G-10/4, Islamabad, Pakistan, since 2011. Her research interests include Operating System, DBMS, Software Development/ Engineering, Software Project Management and Data Mining.

2. Muhammad Ali Khan is doing MS (CS) degree from FUUAST, Islamabad, Pakistan. His research interests include Operating System, DBMS, Software Development / Engineering, SPM, and Data Mining.
3. Muhammad Adnan Khan received his MS (Electronic Engineering) degree from IIU, Islamabad, Pakistan, in 2010. He is PhD scholar at ISRA University, Islamabad. He has a number of publications in the field of computational intelligence, receiver optimization, digital signal processing, space time coding, digital communication and operating system.