



**RAMA
UNIVERSITY**

www.ramauniversity.ac.in

FACULTY OF ENGINEERING AND TECHNOLOGY

Distributed Systems (BCS-701)

LECTURE -15

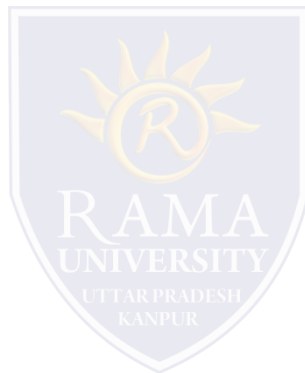
Dr. Hariom Sharan

Professor & Dean

Computer Science & Engineering

OUTLINE

- **Deadlock**
- **Fundamental Causes of Deadlock**
- **Deadlock Handling Strategies**
-
- **MCQ**
- **Reference**



DISTRIBUTED DEADLOCK DETECTION

Deadlock

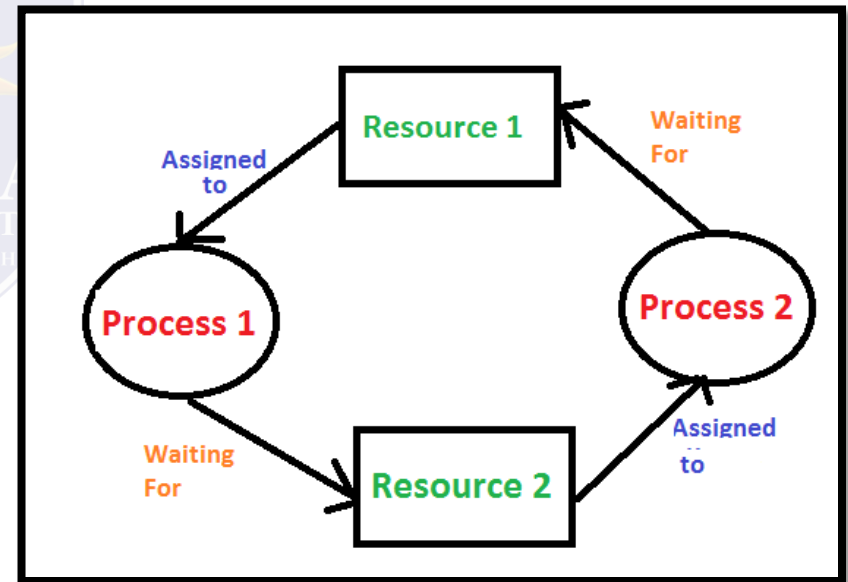
Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Resource deadlock:

each process requests resources held by another process in the set and it must receive all the requested resources before it can become unblocked.

Communication deadlock:

each process is waiting for communication from another process, and will not communicate until it receives the communication for which it is waiting.



DISTRIBUTED DEADLOCK DETECTION

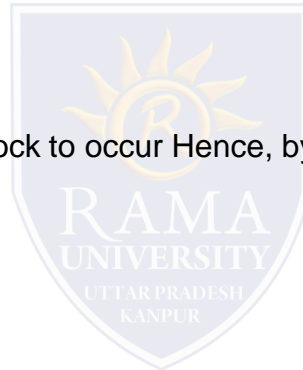
Fundamental Causes of Deadlock

- exclusive access
- wait while holding
- no preemption
- circular wait

All these conditions are necessary for deadlock to occur Hence, by preventing any one of these we prevent deadlock.

Deadlock Handling Strategies

- prevention
- avoidance
- detection



DISTRIBUTED DEADLOCK DETECTION

System Model

- The systems have only reusable resources.
- Processes are allowed only exclusive access to resources.
- There is only one copy of each resource.

Distributed Deadlock Models

Based on WFG (not GRG)

- Nodes are processes
- Resources are located at a site, but may be held by processes at other sites
- Edge (P,Q) indicates P is blocked and waiting for Q to release some resource
- Deadlock exists if there is a directed cycle or knot.



DISTRIBUTED DEADLOCK DETECTION

Distributed Deadlock Handling Strategies

Deadlock prevention

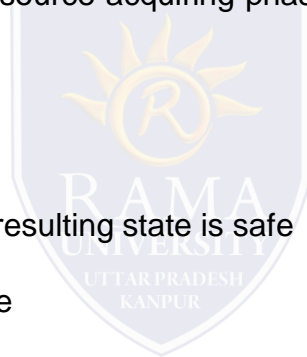
- All resource at once.
- Preventing a process from holding while waiting
- inefficient, can become deadlocked at resource acquiring phase, resource requirements are unpredictable -- not an efficient, universal solution.

Deadlock avoidance

- A resource is granted to a process if the resulting state is safe
- Every site has to maintain the global state
- The checking for a safe state must be done with mutual exclusion
- The number of processes and resources in a distributed system is large
- Not a practical solution

Deadlock detection

- Once deadlock, always deadlock -- detection won't be outdated
- deadlock detection can be preceed concurrently with normal activities
- This is the usual approach



DISTRIBUTED DEADLOCK DETECTION

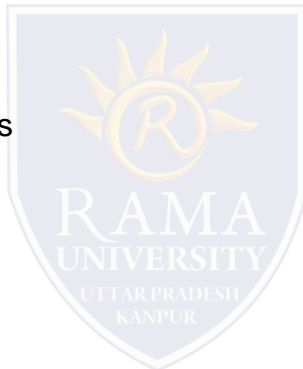
Distributed Deadlock Detection Issues

issues

- maintenance of WFG
- detection of cycles (or knots) in the WFG

requirements

- progress = no undetected deadlocks
- safety = no false deadlocks



Categorization of Methods

- centralized control
- distributed control
- hierarchical control

DISTRIBUTED DEADLOCK DETECTION

Classification of Distributed Detection Algorithms

path-pushing

- ❑ path information transmitted, accumulated

edge-chasing

- ❑ ``I'm waiting for you" probes are sent along edges
- ❑ single returned probe indicates a cycle

diffusion

- ❑ ``Are you blocked?" probes are sent along all edges
- ❑ all queries returned indicates a cycle

global state detection

- ❑ take and use snapshot of system state



DISTRIBUTED DEADLOCK DETECTION

Obermarck's Path-Pushing Algorithm

- ❑ designed for distributed database systems
- ❑ processes are called "transactions" T_1, T_2, \dots, T_n
- ❑ there is special virtual node Ex
- ❑ transactions are totally ordered

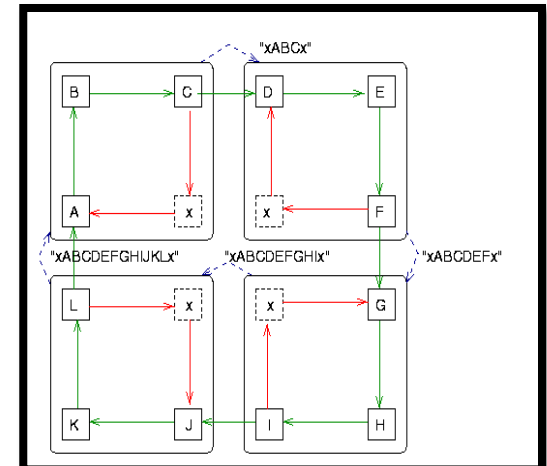
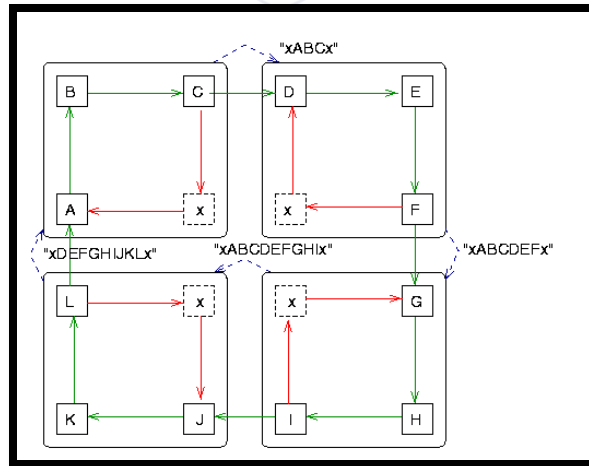
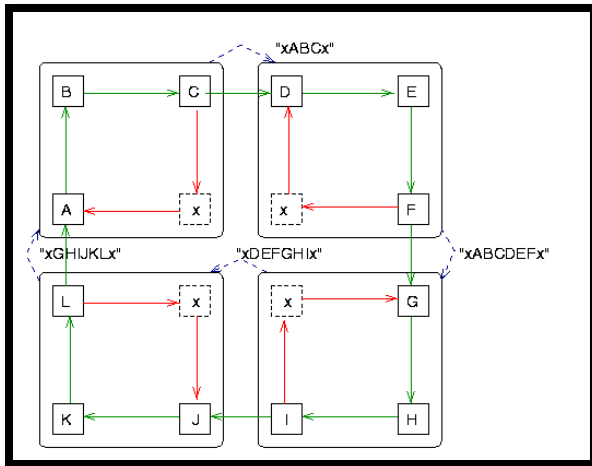
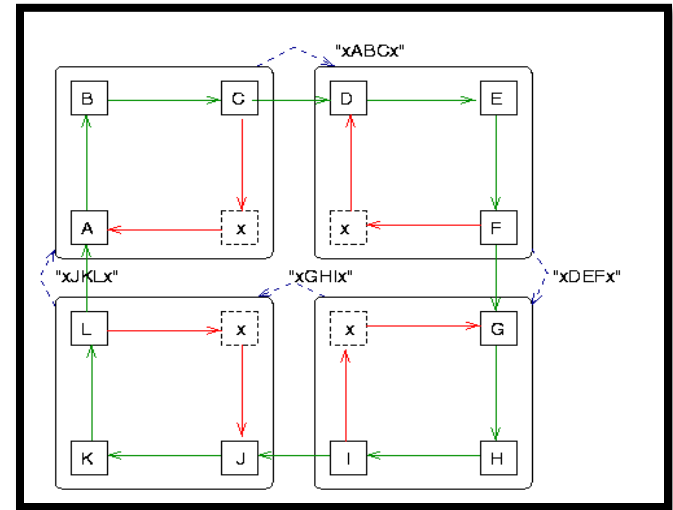
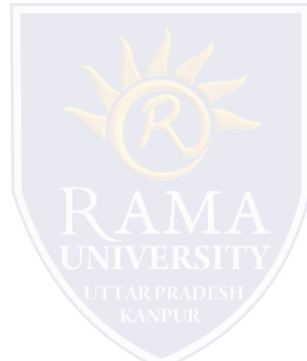
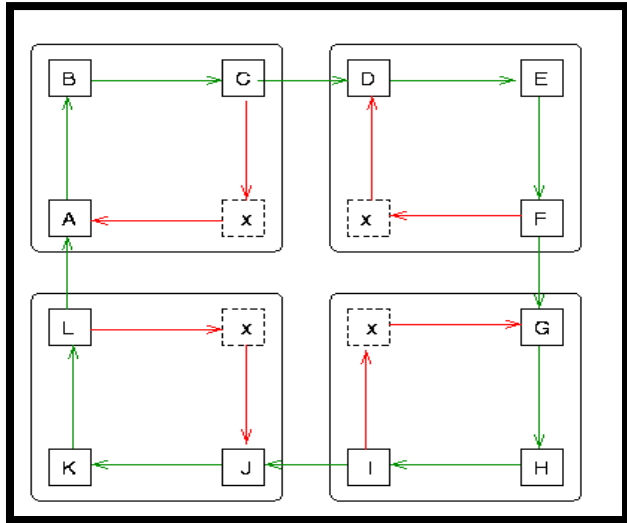
Obermarck's Path-Pushing Algorithm

1. wait for info from previous iteration of Step 3
2. combine received info with local TWFG
 - ❑ detect all cycles
 - ❑ break local cycles
3. send each cycle including the node Ex to the external nodes it is waiting for
4. time-saver: only send path to other sites if last transaction is higher in lexical order than the first



DISTRIBUTED DEADLOCK DETECTION

Obermarck's Path-Pushing Algorithm



DISTRIBUTED DEADLOCK DETECTION

Problems with Obermarck's Path-Pushing Algorithm

- ❑ Detects false deadlocks, due to asynchronous snapshots at different sites.
- ❑ Message complexity? Message size? Detection delay?
- ❑ Exactly how are paths combined and checked?

Obermarck's Path-Pushing Algorithm: Performance

- ❑ $O(n(n-1)/2)$ messages
- ❑ $O(n)$ message size
- ❑ $O(n)$ delay to detect deadlock



DISTRIBUTED DEADLOCK DETECTION

Chandy-Misra-Haas Edge-Chasing Algorithm

- ❑ for AND request model
- ❑ probe = (i,j,k) is sent for detection initiated by P_i , by site of P_j to site of P_k
- ❑ deadlock is detected when a probe returns to its initiator

Terminology

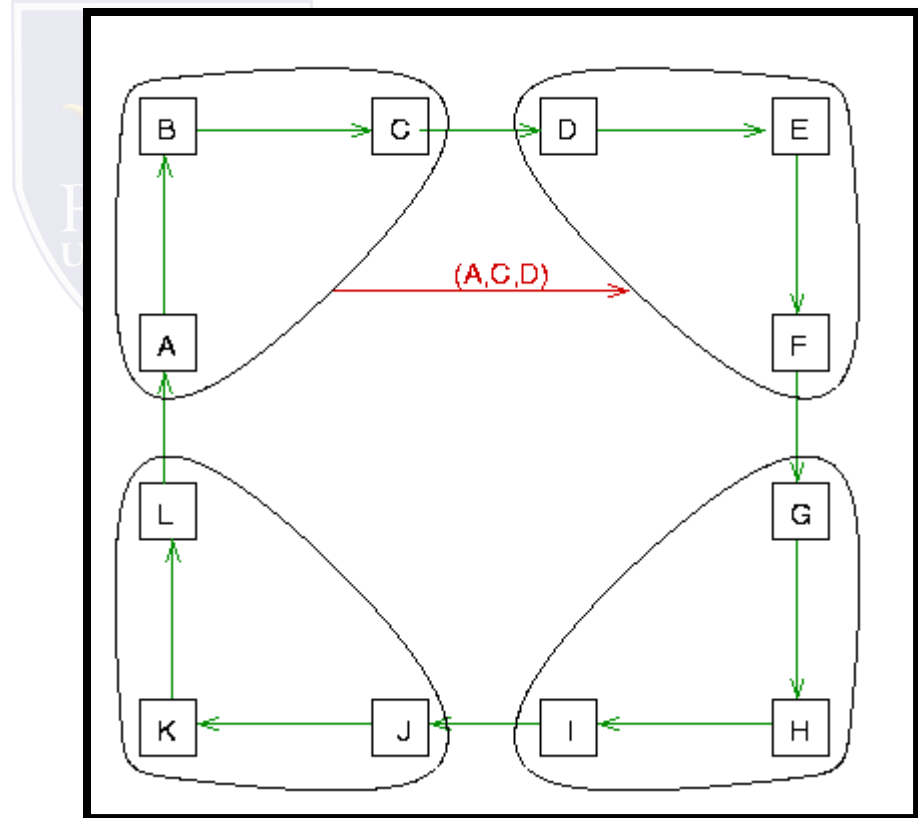
- ❑ P_j is dependent on P_k if there is a sequence $P_j, P_{i1}, P_{i2}, \dots, P_{im}, P_k$ such that each process except P_k is blocked and each process except the first holds a resource for which the previous process is waiting
- ❑ P_j is locally dependent on P_k if it is dependent and both processes are at the same site
- ❑ array dependent $i(j) = \text{true} \hat{=} P_i$ knows that P_j is dependent on it

DISTRIBUTED DEADLOCK DETECTION

Algorithm Initiation by P_i

if P_i is locally dependent on itself then declare a deadlock else send probe (i, j, k) to home site of P_k for each j, k such that all of the following hold

- ❑ P_i is locally dependent on P_j
- ❑ P_j is waiting on P_k
- ❑ P_j and P_k are on different sites



DISTRIBUTED DEADLOCK DETECTION

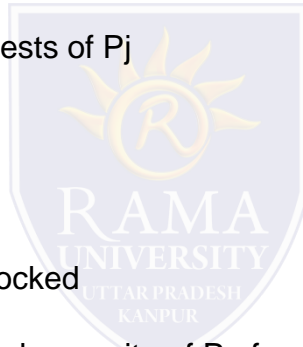
Algorithm on receipt of probe (i,j,k)

check the following conditions

- P_k is blocked
- Dependent $k(i) = \text{false}$
- P_k has not replied to all requests of P_j

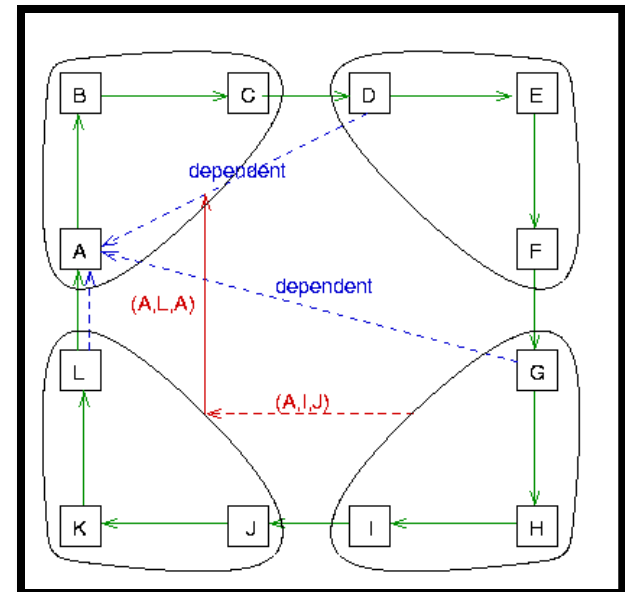
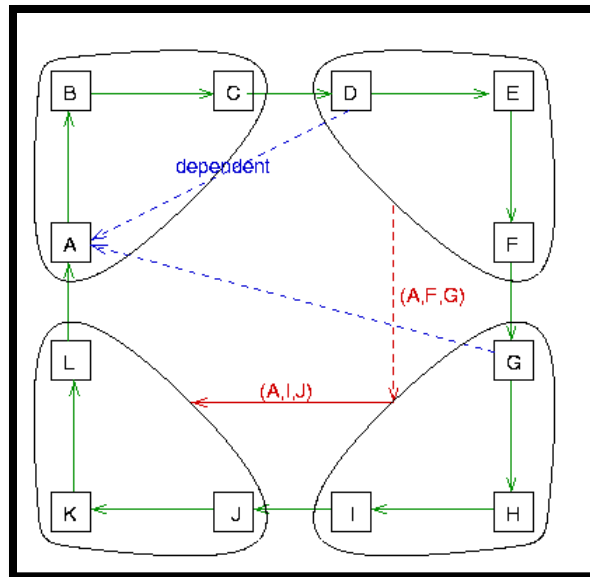
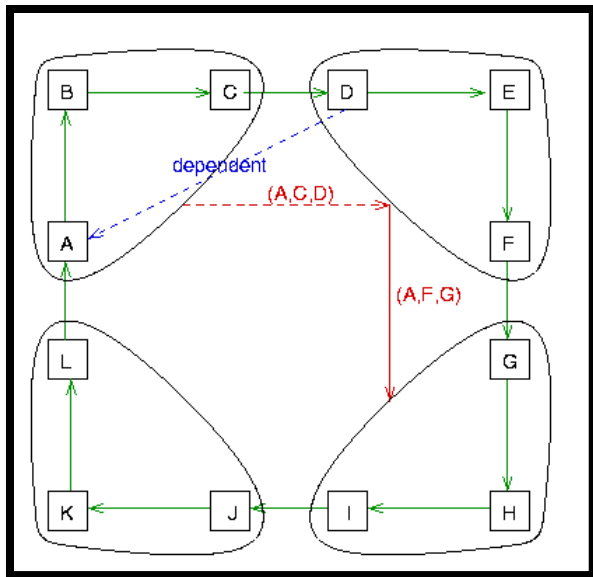
if these are all true, do the following

- set $\text{dependent}_k(i) = \text{true}$
- if $k=i$ declare that P_i is deadlocked
- else send probe (i,m,n) to the home site of P_n for every m and n such that the following all hold
 - P_k is locally dependent on P_m
 - P_m is waiting on P_n
 - P_m and P_n are on different sites



DISTRIBUTED DEADLOCK DETECTION

Algorithm on receipt of probe (i,j,k)

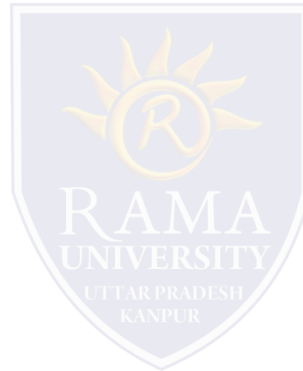


DISTRIBUTED DEADLOCK DETECTION

Algorithm on receipt of probe (i,j,k)

Chandy Misra Haas Complexity

- $m(n-1)/2$ messages for m processes at n sites in the book, is this right?
- 3-word message length
- $O(n)$ delay to detect deadlock



MCQ

1. What is the access point (AP) in a wireless LAN?

- a) device that allows wireless devices to connect to a wired network
- b) wireless devices itself
- c) both device that allows wireless devices to connect to a wired network and wireless devices itself
- d) all the nodes in the network

2. In wireless ad-hoc network _____

- a) access point is not required
- b) access point is must
- c) nodes are not required
- d) all nodes are access points

3. Which multiple access technique is used by IEEE 802.11 standard for wireless LAN?

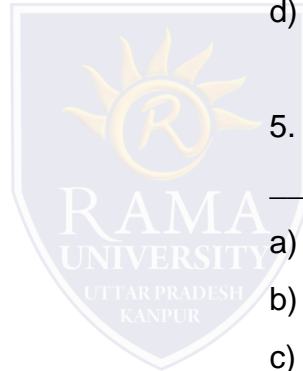
- a) CDMA
- b) CSMA/CA
- c) ALOHA
- d) CSMA/CD

4. In wireless distribution system _____

- a) multiple access point are inter-connected with each other
- b) there is no access point
- c) only one access point exists
- d) access points are not required

5. A wireless network interface controller can work in _____

- a) infrastructure mode
- b) ad-hoc mode
- c) both infrastructure mode and ad-hoc mode
- d) WDS mode



REFERENCES

- ❑ <http://cs-www.cs.yale.edu/homes/aspnes/classes/465/notes.pdf>
- ❑ <https://www.geeksforgeeks.org/mutual-exclusion-in-distributed-system/>
- ❑ <https://www.vidyarthiplus.com/vp/attachment.php?aid=43022>
- ❑ <http://www.cs.fsu.edu/~xyuan/cop5611/lecture9.html>
- ❑ <http://www.cs.fsu.edu/~xyuan/cop5611/lecture10.html>

