BITS Pilani
Pilani Campus

CS G623: Advanced Operating Systems

Lecture 14

Amit Dua

Sept 6, 2018

Topics

Deadlocks

What is a deadlock?

- Problem definition
  - Permanent blocking of a set of processes that either compete for system resources or communicate with each other
  - No node has complete and up-to-date knowledge of the entire distributed system
  - Message transfers between processes take unpredictable delays

- Mostly seen on distributed databases (Lock & Unlock)

An example: using locks with transactions

Types of Deadlocks in Distributed Systems

- Communication Deadlock

- Resource Deadlock
  - Fighting for exclusive access to files, I/O devices, locks, or other resources.

Deadlocks in resource allocation

- Conditions for deadlocks in resource allocation
  - Mutual exclusion: can be used by only one process at a time
  - Hold and wait: A process holds a resource while waiting for other resources
  - No preemption: A process cannot be preempted to free up the resource
  - Circular wait: A closed cycle of processes is formed, where each process holds one or more resources needed by the next process in the cycle
- Strategies
  - Prevent the formation of a circular wait
  - Avoid the deadlocks
  - Detect the potential or the actual occurrence of a circular wait

Deadlock Prevention Strategy

1. Prevent by defining a linear ordering of resource types

    - Disadvantages
        - Resources cannot be requested in the order that are needed

2. Let the process acquire all needed resources before starting execution
    - Disadvantages
        – Inefficient use of resources, Reduced concurrency, Process can become deadlocked during the initial resource acquisition, Future needs of a process cannot be always predicted

Continued…

3. Use of time-stamps
   - The circular wait condition is prevented by comparing time-stamps: transaction with an earlier time-stamp always wins
   - "Wait-die" method (A non-preemptive technique)
   - "Wound-wait" method (A preemptive technique)

Deadlock avoidance

- Decision is made dynamically, before allocating a resource, the resulting global system state is checked - if safe, allow allocation

- Disadvantages
  - Every site has to maintain global state of system (extensive overhead in storage and communication)
  - Different sites may determine (concurrently) that state is safe, but global state may be unsafe

- Conclusion: Deadlock avoidance is impractical in distributed systems

Deadlock detection and resolution

- Very Popular and practical
- Done by detecting a cycle in the Wait-For-Graph (WFG)
- Once a cycle is formed, it remains till detected and broken
- Along with Cycle detection, nodes can do their regular activities.
- Resolution means breaking Wait for dependencies to resolve the deadlock.

And, Or Wait-For-Graphs

Cycle vs Knot example

Example continued…

- A strongly connected subgraph of a directed graph, such that starting from any node in the subset it is impossible to leave the knot by following the edges of the graph.

Deadlock detection requirements

- Progress

- Safety

Control Framework

- Centralized control
- Distributed Control
- Hierarchical Control

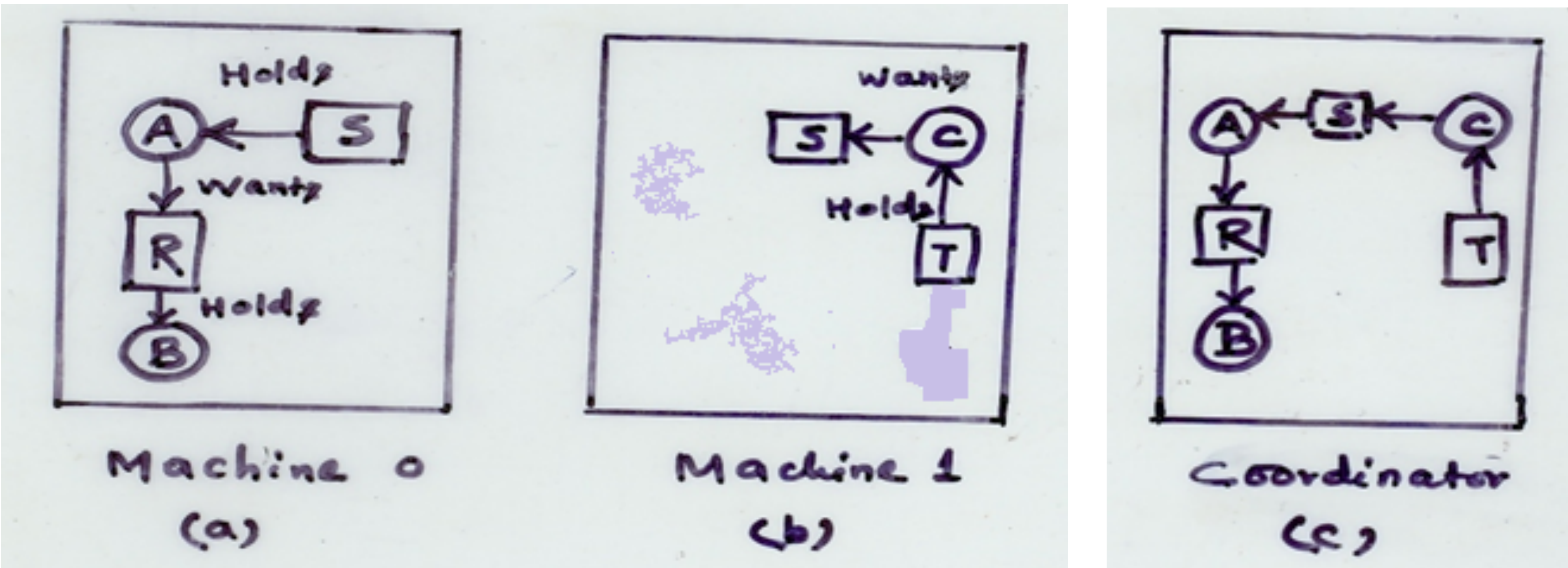
Central control

Continued …

- There are different ways, by which each node may send its' WFG status to the coordinator
  - Whenever an arc is added/deleted at a site, message is sent
  - Periodically, every process can send list of arcs since last update
  - Coordinator can ask the information when required

- None of these work well

False deadlocks



The Ho-Ramamoorthy Algorithms

Two phase (can be for AND or OR model)

- each site has a status table of locked and waited resources
- the control site will periodically ask for this table from each node
- the control node will search for cycles and, if found, will request the table again from each node
- Only the information common in both reports will be analyzed for confirmation of a cycle

One phase (can be for AND or OR model)

- each site keeps 2 tables; process status and resource status
- the control site will periodically ask for these tables (both together in a single message) from each node
- the control site will then build and analyze the WFG, looking for cycles and resolving them when found

- One Phase is faster than two-Phase
- One Phase requires fewer msgs and more storage (i.e 2 tables).

Distributed Deadlock Detection Algorithms

- Responsibility is shared by all
- Not vulnerable to single point of failure
- N/W Congestion is not seen
- Only when there is a suspicion, deadlock detection starts
- Difficult to design, because of no shared memory
- All the processes may be involved in detecting a single/same deadlock
- Types :
  - Path- Pushing , Edge-Pushing, Diffusion Computation and Global State Detection

Distributed Deadlock Detection Algorithms

Path Pushing:

- WFG is disseminated as paths – sequences of edges.
- Each site sends this path info to neighbors.
- Each site updates its local copy.
- Deadlock is seen if a process detects a local cycle.
- Example- Obermarck's algorithm.

Edge-Chasing:

- Probe (Special) messages circulate along the edges.
- Blocked processes forward probe to processes holding requested resources.
- Deadlock if initiator (there may be several) receives it's own probe.
- Example- Chandy-Misra-Haas algorithm.

Continued…

Diffusion Computation:

Query messages sent to dependent set.

Active processes discard query, blocked processes forward query under certain conditions, reply under other conditions.

Deadlock if initiator receives replies to all its queries.

Global State Detection:

If a stable property (deadlock) holds in the system, before a snapshot then it will still hold in the snapshot.

A consistent global wait for graph is sufficient to define this.

Obermarack's Algorithm

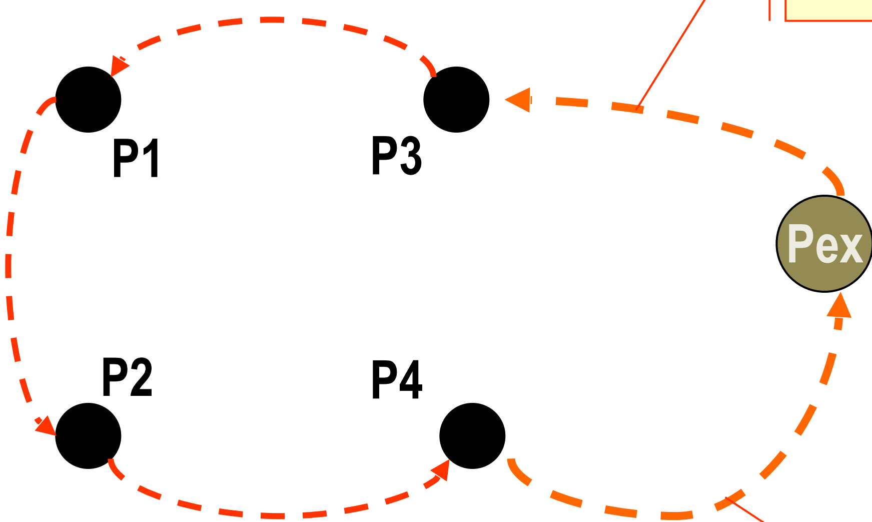
- based on a database model using transaction processing
- Implemented on a distributed database system R*(IBM).
- Sites which detect a cycle in their partial WFG views convey the paths discovered to members of the (totally ordered) transaction
- The highest priority transaction detects the deadlock

  "Ex => T1 => T2 => Ex"
- Algorithm can detect *phantoms* due to its asynchronous snapshot method

Continued…

- Individual sites maintain local WFGs
  - Nodes for local processes
  - Node "Pex" represents external processes
- Deadlock detection:
  - If a site Si finds a cycle that does not involve Pex, it has found a deadlock
  - If a site Si finds a cycle that does involve Pex, there is the possibility of a deadlock
    - It sends a message containing its detected cycle to the sites involved in Pex
    - If site Sj receives such a message, it updates its local WFG graph, and searches it for a cycle
      - If Sj finds a cycle that does not involve its Pex, it has found a deadlock
      - If Sj finds a cycle that does involve its Pex, it sends out a message…
- Can report false deadlock

Situation at node x:



Some process outside node x waits for a resource currently owned by P3

Some process outside of node x holds a resource P4 is waiting for.

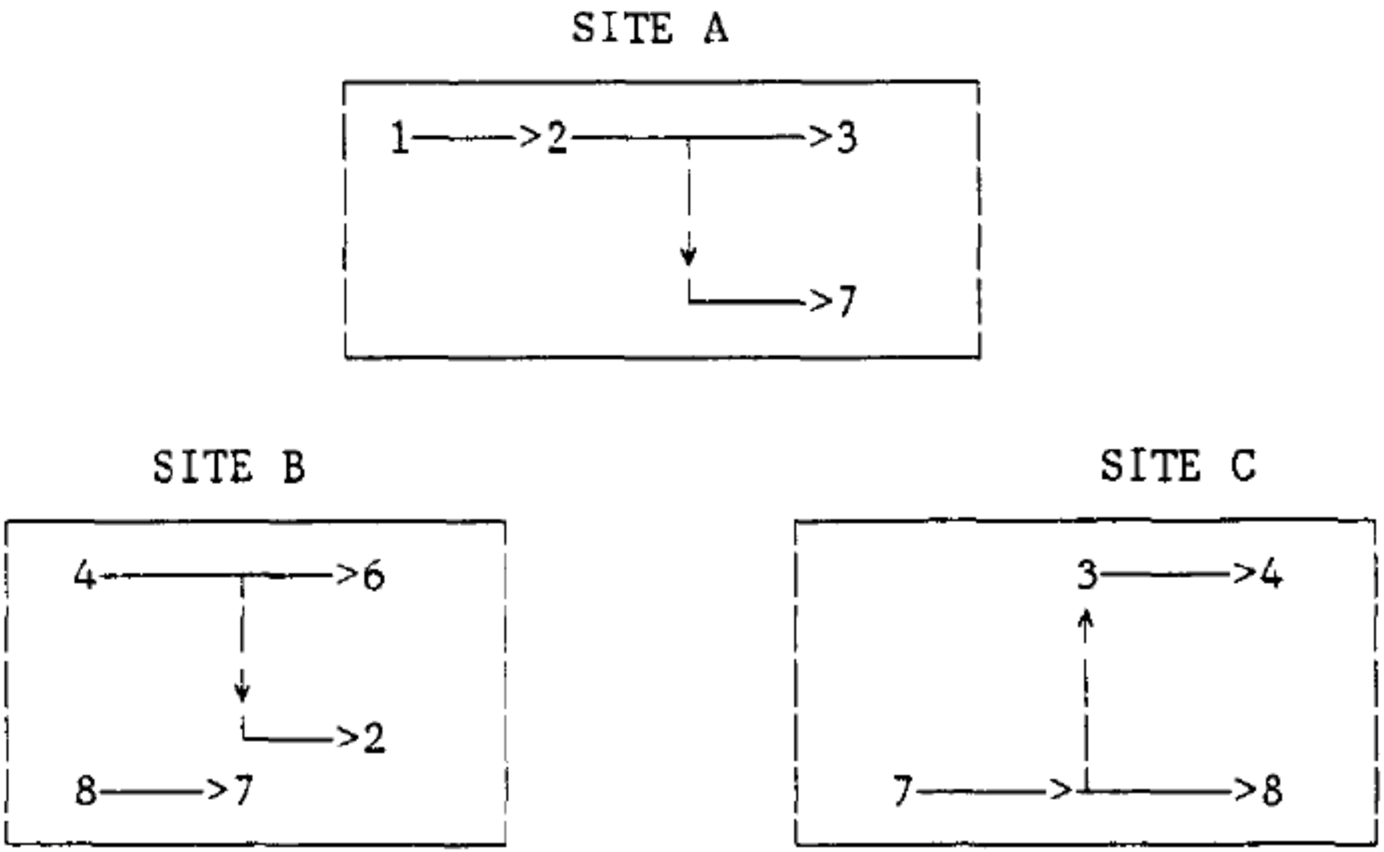
No local deadlock

Example

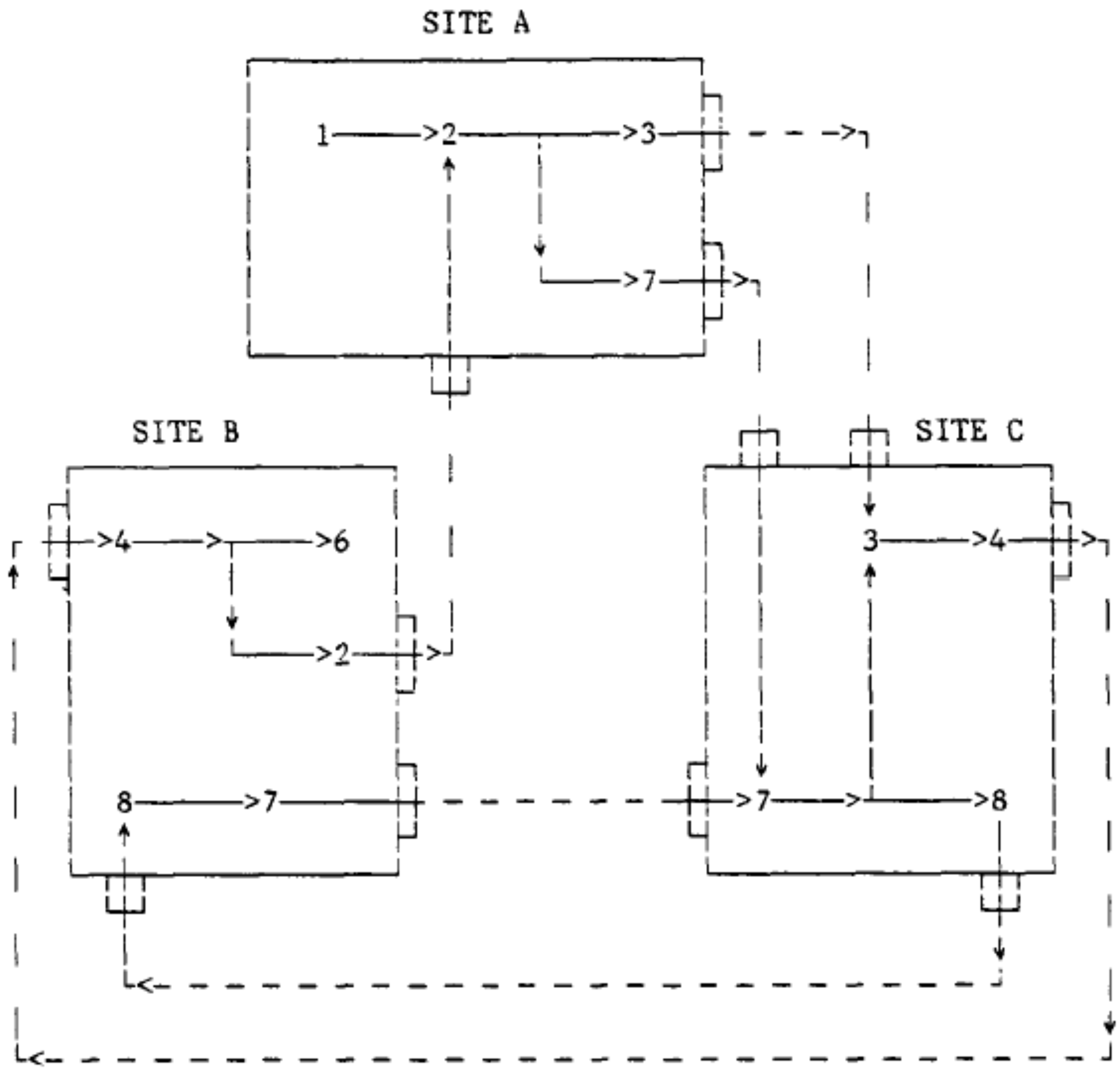
Site A

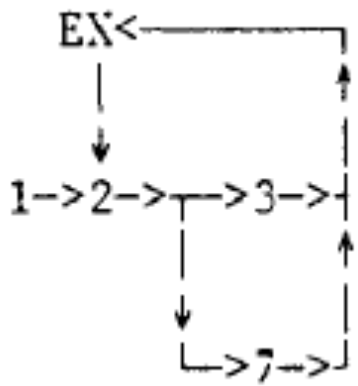
Site B

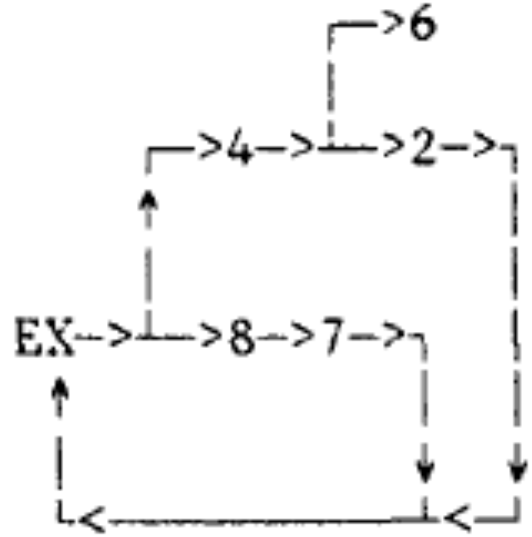
Consider each elementary cycle containing EX. For each such cycle EX → T1→ . . . →Tn → EX compare T1 with Tn. If T1 > Tn, send the cycle to each site, where an agent of Tn is waiting to receive a message from the agent of Tn at this site.

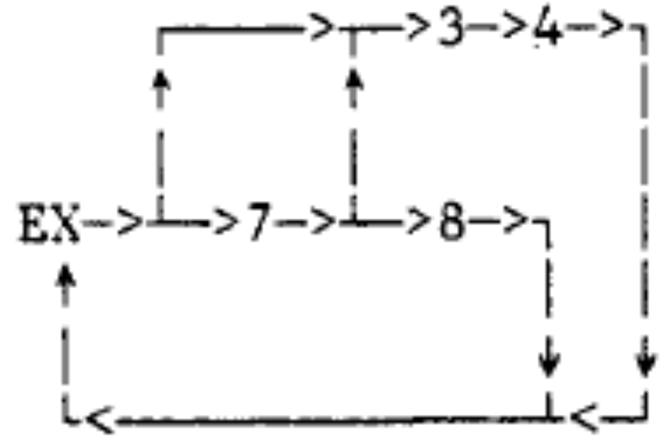

SITE A
1 2 3
7
SITE B
4 6
2
8 7
SITE C
3 4
7 8

- transaction 2 has done work in site B and then migrated to site A, where it is waiting for a resource shared by transactions 3 and 7.
- Transaction 3 has done work in site A and migrated to site C.
- In site C, transaction 3 is waiting for transaction 4.
- Transaction 4 has migrated to site B, where it is waiting for a resource shared by the agents for transactions 2 and 6.


SITE A
1
2
3
7
SITE B
4
6
2
8
7
SITE C
3
4
7
8


EX
1
2
3
7


6
4
2
EX
8
7


3
4
EX
7
8

Edge-Chasing: Chandy-Misra-Haas

- Processes can request multiple resources at once
  - growing phase of a transaction can be speed up
  - consequence: process may wait on multiple resources
- Some processes wait for local resources
- Some processes wait for resources on other machines
- Algorithm invoked when a process has to wait for a resource
- Uses local WFGs to detect local deadlocks and *probes* to determine the existence of global deadlocks.

Algorithm

- Probe message is generated
  - sent to process(es) holding the needed resources
  - message contains three numbers
    - process that just blocked
    - process sending the message
    - process to whom it is being sent
- when message arrives, recipient checks to see if it is waiting for any processes
  - if so, update message
    - replace second field by its own process number
    - replace third field by the number of the process it is waiting for
    - send messages to each process on which it is blocked
    - if a message goes all the way around and comes back to the original sender, a cycle exists
- *we have deadlock*

Chandy-Misra-Haas's Algorithm

Sending the probe:

if Pi is locally dependent on itself then deadlock.

else for all Pj and Pk such that

(a) Pi is locally dependent upon Pj, and

(b) Pj is waiting on Pk, and

(c) Pj and Pk are on different sites, send probe(i,j,k) to the home site of Pk.

Receiving the probe:

if (d) Pk is blocked, and

(e) *dependentk(i)* is false, and

(f) Pk has not replied to all requests of Pj,

then begin

dependentk(i) := true;

if k = i then Pi is deadlocked

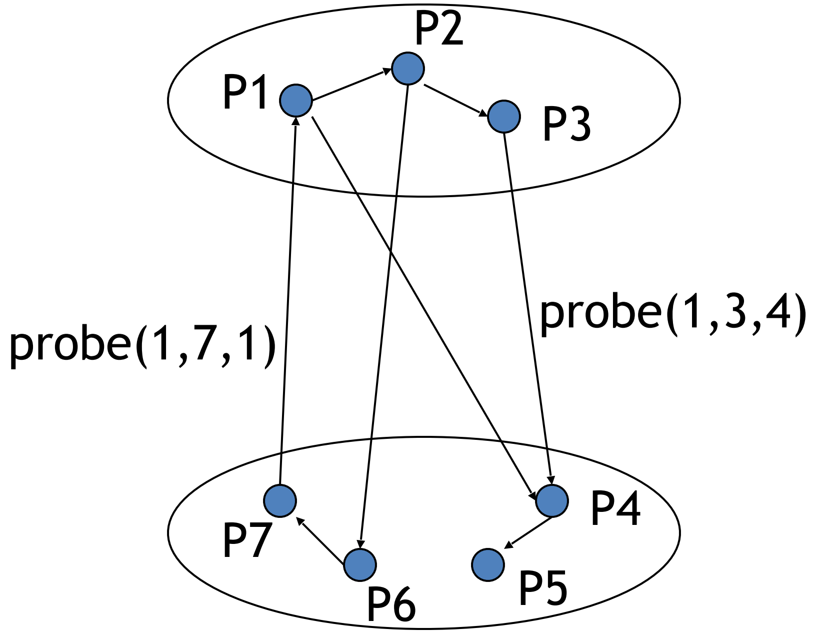
else ...

Continued…

for all Pm and Pn such that
(a’) Pk is locally dependent upon Pm, and
(b’) Pm is waiting on Pn, and
(c’) Pm and Pn are on different sites, send probe(i,m,n) to the home site of Pn.
end.

Example1

Example2



Advantages

1. Popular, Variants of this are used in locking schemes.
2. Easy to implement, as each message is of fixed length and requires few computational steps.
3. No graph constructing and information collection
4. False deadlocks are not detected
5. Does not require a particular structure among processes

Disadvantages

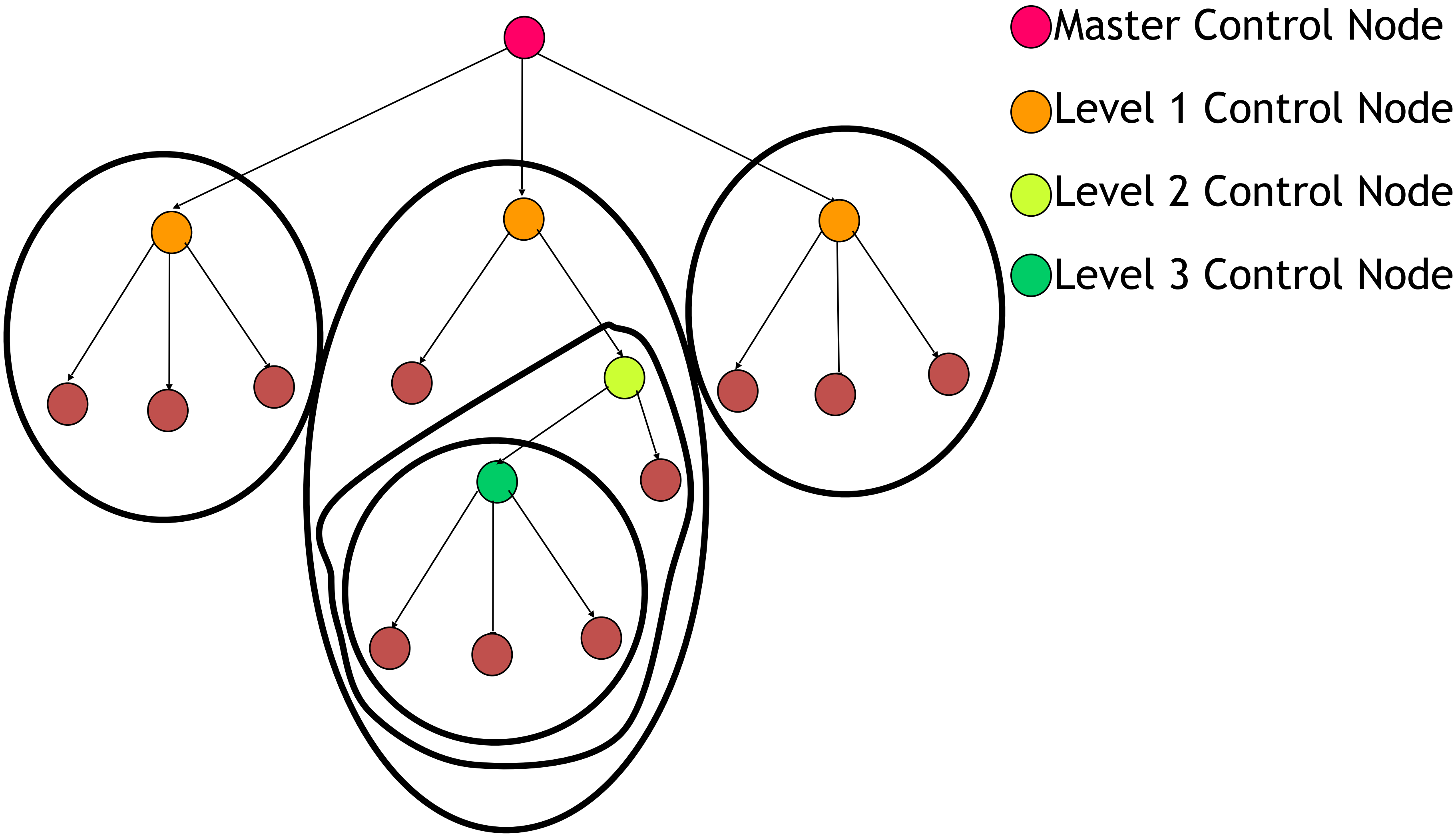
- Two or more processes may independently detect the same deadlock and hence while resolving, several processes will be aborted.
- Even though a process detects a deadlock, it does not know the full cycle

Diffusion-Computation based Algorithm

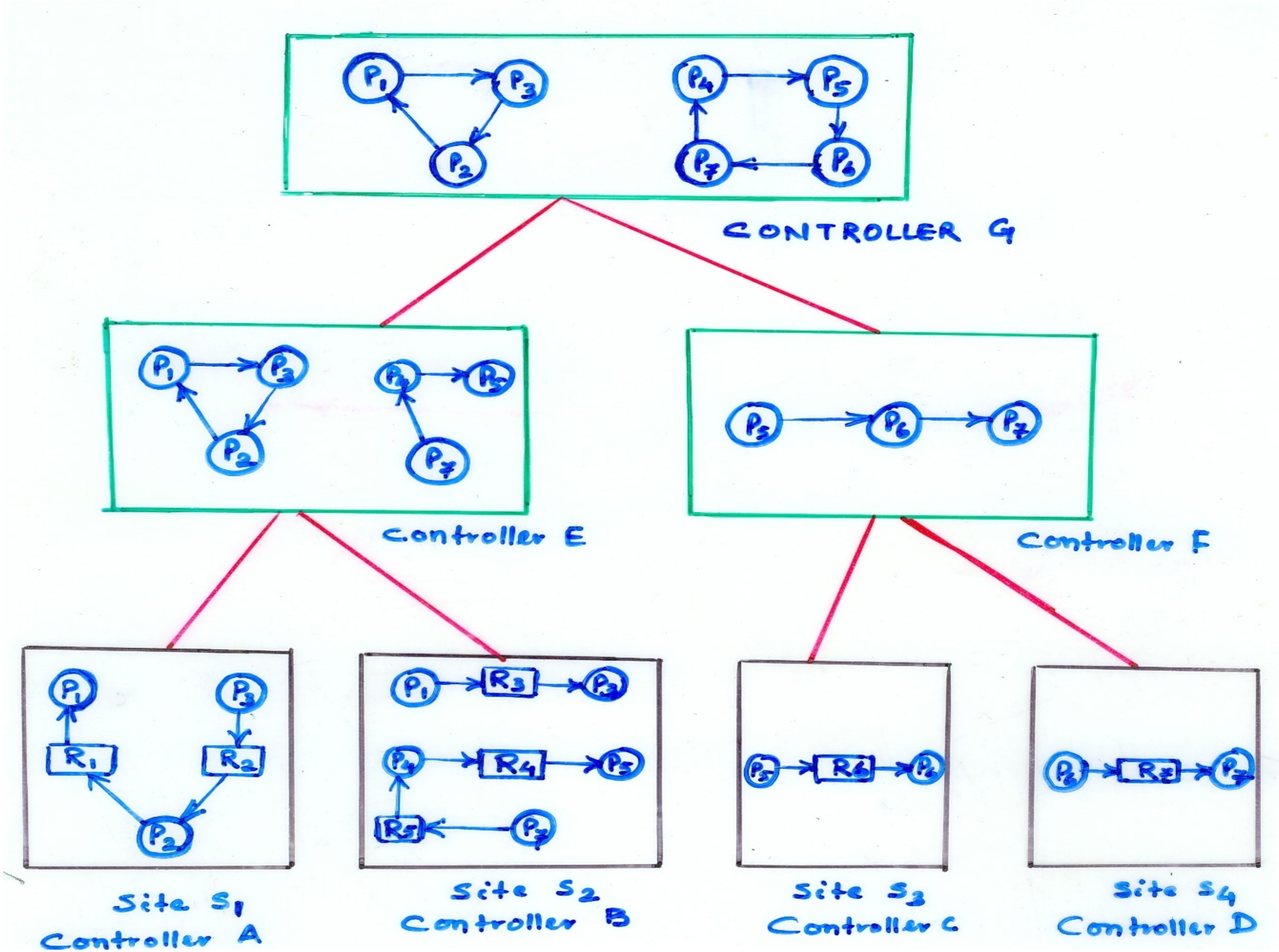
```
Initiation by a blocked process Pi:
    send query(i,i,j) to all processes Pj in the dependent set DSi of Pi;
    num(i) := |DSi|; waiti(i) := true;

Blocked process Pk receiving query(i,j,k):
    if this is engaging query for process Pk /* first query from Pi */
          then send query(i,k,m) to all Pm in DSk;
          numk(i) := |DSk|; waitk(i) := true;
    else if waitk(i) then send  a reply(i,k,j) to Pj.

Process Pk receiving reply(i,j, k)
    if waitk(i) then
          numk(i) := numk(i) - 1;
          if numk(i) = 0 then
               if i = k then declare a deadlock.
               else send reply(i, k, m) to Pm, which sent the engaging query.
```

Example

Hierarchical Deadlock Detection



Menasce-Muntz Algorithm



Ho-Ramamoorthy Hierarchical Algorithm