Distributed Systems
Monsoon 2022
Lecture 20

Kishore Kothapalli

Material from Chapter 10 of the Kshemkalyani book

Deadlocks

- Deadlocks is a fundamental problem in systems.
- A deadlock is a state where a set of processes request resources that are held by other processes in the set.
- A process may request resources in any order, which may not be known a priori and a process can request a resource while holding others.
- If the sequence of the allocations of resources to the processes is not controlled, deadlocks can occur.

Deadlocks

- The problem appears in standard systems too.
- An Operating System can work on this problem at multiple levels:
  - Detect deadlocks
  - Recover from deadlocks
  - Prevent deadlocks
- Various algorithms exist for these problems.

To the Distributed Setting

- A distributed program is composed of a set of n asynchronous processes $p_1, p_2, \ldots, p_i, \ldots, p_n$ that communicate by message passing over the communication network.
- Without loss of generality, we assume that each process is running on a different processor.
- The processors do not share a common global memory and communicate solely by passing messages over the communication network

To The Distributed Setting

- There is no physical global clock in the system to which processes have instantaneous access.
- The communication medium may deliver messages out of order, messages may be lost garbled or duplicated due to timeout and retransmission, processors may fail, and communication links may go down.
- We make the following assumptions:
  - The systems have only reusable resources.
  - Processes are allowed to make only exclusive access to resources.

System Model

- A process can be in two states:
- Running or blocked.
- In the running state (also called active state), a process has all the needed resources and is either executing or is ready for execution.
- In the blocked state, a process is waiting to acquire some resource.

Wait-for-Graph

- The state of the system can be modelled by using the concept of Wait-For-Graph (WFG).
- The WFG is a directed graph on processors as nodes.
- There is an edge from node P_i to node P_j iff P_i is waiting to acquire a resource that is currently held by P_j.
- It is clear that a system has a deadlock iff the WFG has a directed cycle.

Strategies for Handling Deadlocks

- In the single processor setting, any of the three approaches: Prevention, Detection, Recovery, can be used.
- In the distributed setting, several challenges come to the fore.
- For deadlock prevention, complete knowledge at all times is required.
  - No site has complete knowledge of the current system state.

Strategies for Handling Deadlocks

- For deadlock prevention, complete knowledge at all times is required.
  - No site has complete knowledge of the current system state.
  - Resource Request messages may be in transit and face arbitrary delays.
  - Prevention may sometimes require process pre-emption, which is not feasible.
- For deadlock avoidance, the typical approach is to keep track of the safety of the system.
  - Any resources allocated should move the system from one safe state to another.
  - Requires, again, good knowledge of the resource usage across the system.
- Detecting deadlocks is the only hope!

Solution Properties – Deadlock Detection

- Any algorithm for deadlock detection needs to satisfy two properties.
  - Progress: Detect all existing deadlocks
  - Safety: Not report non-existing deadlocks

- Deadlock Resolution
  - Rollback a process that is part of a deadlock.

Models of Deadlocks

- Depending on how processes in a system raise requests, one can study different ways.
- Some of the popular models are:
  - Single Resource Model
  - The AND model
  - The OR Model
  - The AND-OR Model
  - The P out of Q Model.

Models of Deadlocks

- Single Resource Model
  - The simplest of all models.
  - Any process can have at most one outstanding request for only one unit of a resource.
  - In the WFG, the out degree of any node is at most one.
  - A directed cycle in the WFG immediately implies a deadlock.

Models of Deadlocks

- The AND Model
  - Processes can request more than one resource simultaneously
  - The request of a process is satisfied if all its requests are granted.
  - The outdegree in the WFG can be more than 1.
  - A cycle in the WFG implies a deadlock.
  - Surprisingly, a process may not be on a cycle in the WFG but can still be deadlocked.

Models of Deadlocks

- The OR Model
  - Processes can request more than one resource simultaneously
  - The request of a process is satisfied if any of its requests are granted.
  - A cycle in the WFG does not imply a deadlock.

Models of Deadlocks

- The AND-OR Model
  - Combines the prior two models.
  - WFG may not help in detecting a deadlock by using graph theoretic notions.

A Classification of Algorithms

- Distributed deadlock detection algorithms can be divided into four classes:
  - path-pushing
  - edge-chasing
  - diffusion computation
  - global state detection.

Path-Pushing Algorithms

- In path-pushing algorithms, distributed deadlocks are detected by maintaining an explicit global WFG.
- The basic idea is to build a global WFG for each site of the distributed system.
- In this class of algorithms, at each site whenever deadlock computation is performed, it sends its local WFG to all the neighboring sites.
- The sites receiving these WFG fragments update their local WFG.
- The updated WFG is then passed along subsequently.
- The procedure is repeated until some site has a sufficiently complete picture of the global state to announce deadlock or to establish that no deadlocks are present.
- This feature of sending around the paths of global WFG has led to the term path-pushing algorithms.

Edge-Chasing Algorithms

- The presence of a cycle in a distributed graph structure is verified by propagating special messages called probes, along the edges of the graph.
- These probe messages are different than the request and reply messages.
- The formation of cycle can be deleted by a site if it receives the matching probe sent by it previously.
- Only blocked processes propagate probe messages along their outgoing edges.
- Active processes that receive a probe message discard such messages.
- Advantage:
  - Probes are short, fixed size messages.
  - WFG not built globally

Diffusing Computations Based

- In diffusion computation based distributed deadlock detection algorithms, deadlock detection computation is diffused through the WFG of the system.
- These algorithms make use of echo algorithms to detect deadlocks.
- This computation is superimposed on the underlying distributed computation. If this computation terminates, the initiator declares a deadlock.
- To detect a deadlock, a process sends out query messages along all the outgoing edges in the WFG.
- These queries are successively propagated (i.e., diffused) through the edges of the WFG.

Diffusing Computations Based

- When a blocked process receives first query message for a particular deadlock detection initiation, it does not send a reply message until it has received a reply message for every query it sent.
- For all subsequent queries for this deadlock detection initiation, it immediately sends back a reply message.
- The initiator of a deadlock detection detects a deadlock when it receives reply for every query it had sent out.

Global state detection based

- Global state detection based deadlock detection algorithms exploit the following facts:
- A consistent snapshot of a distributed system can be obtained without freezing the underlying computation and
- If a stable property holds in the system before the snapshot collection is initiated, this property will still hold in the snapshot.
- Therefore, distributed deadlocks can be detected by taking a snapshot of the system and examining it for the condition of a deadlock.

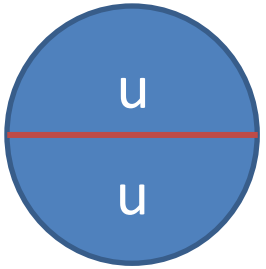
Mitchell and Merritt's Algorithm

- Belongs to the class of edge-chasing algorithms where probes are sent in opposite direction of the edges of WFG.
- When a probe initiated by a process comes back to it, the process declares deadlock.
- Only one process in a cycle detects the deadlock.

Mitchell and Merritt's Algorithm

- Each node of the WFG has two local variables, called labels:
- A private label, which is unique to the node at all times,
- A public label, which can be read by other processes and which may not be unique.
- Each process is represented as `u/v` where `u` and `v` are the public and private labels, respectively.
- Initially, private and public labels are equal for each process.
- A global WFG is maintained and it defines the entire state of the system.

Mitchell and Merritt's Algorithm



Public label

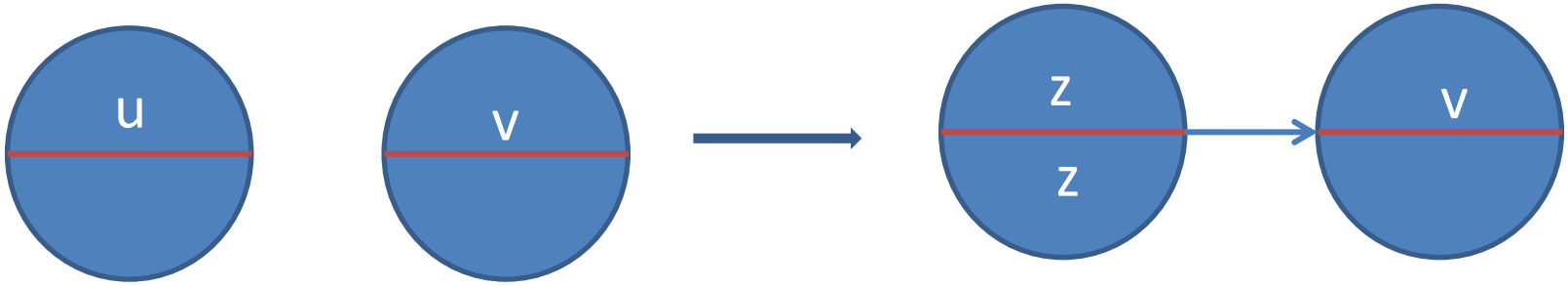
Private label

- The algorithm is defined by the four state transitions.
- The function z = inc(u, v) yields a unique label greater than both u and v.
- Labels that are not shown do not change.

Mitchell and Merritt's Algorithm

- Four events/messages that affect the WFG.
- Block creates an edge in the WFG.
  - Two messages are needed, one resource request, and
  - One message back to the blocked process to inform it of the public label of the process it is waiting for.
- Activate denotes that a process has acquired the resource from the process it was waiting for.
- Transmit propagates larger labels in the opposite direction of the edges by sending a probe message.
- Detect means that the probe with the private label of some process has returned to it, indicating a deadlock.

Mitchell and Merritt's Algorithm

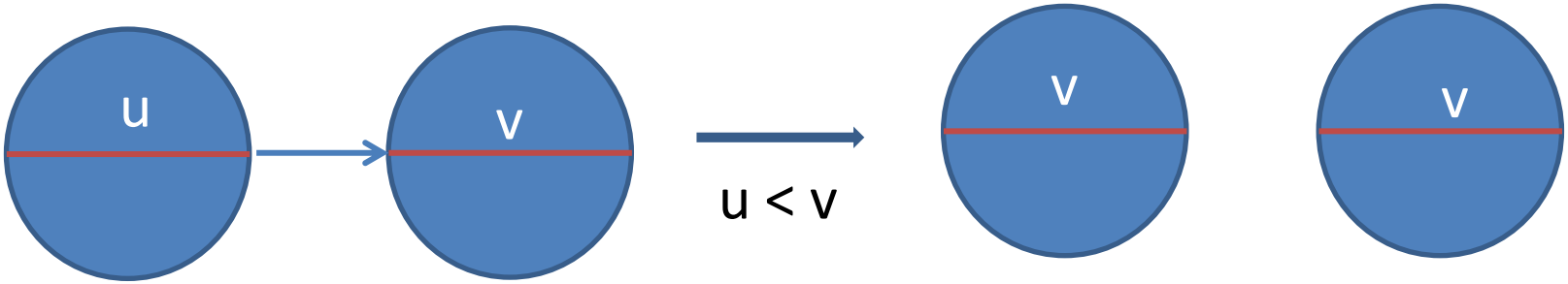


- Block creates an edge in the WFG.
- Two messages are needed, one resource request and one message back to the blocked process to inform it of the public label of the process it is waiting for.

Mitchell and Merritt's Algorithm

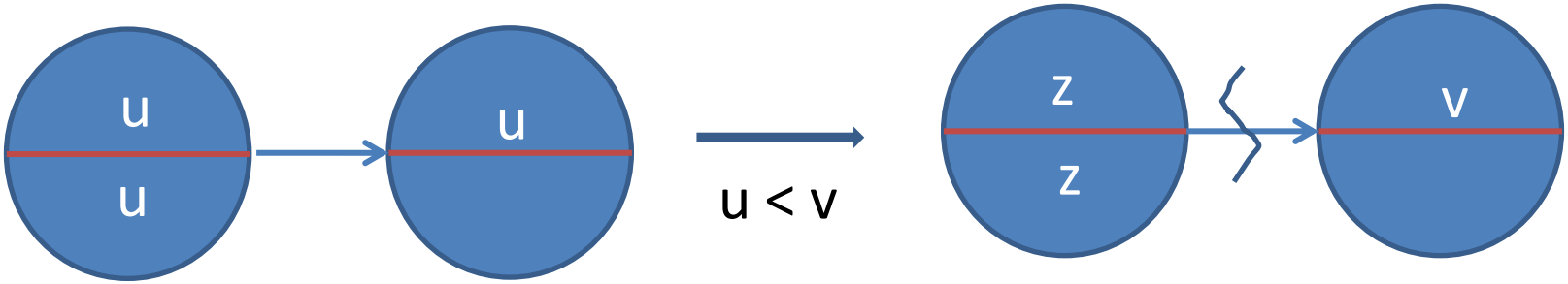
- Activate indicates that a process is able to acquire a resource it is waiting for.
- Results in an edge removed from the WFG.

Mitchell and Merritt's Algorithm



- Transmit propagates larger labels in the direction opposite to that of the probe messages.
- Whenever a process receives a probe which is less then its public label, then it simply ignores that probe.

Mitchell and Merritt's Algorithm



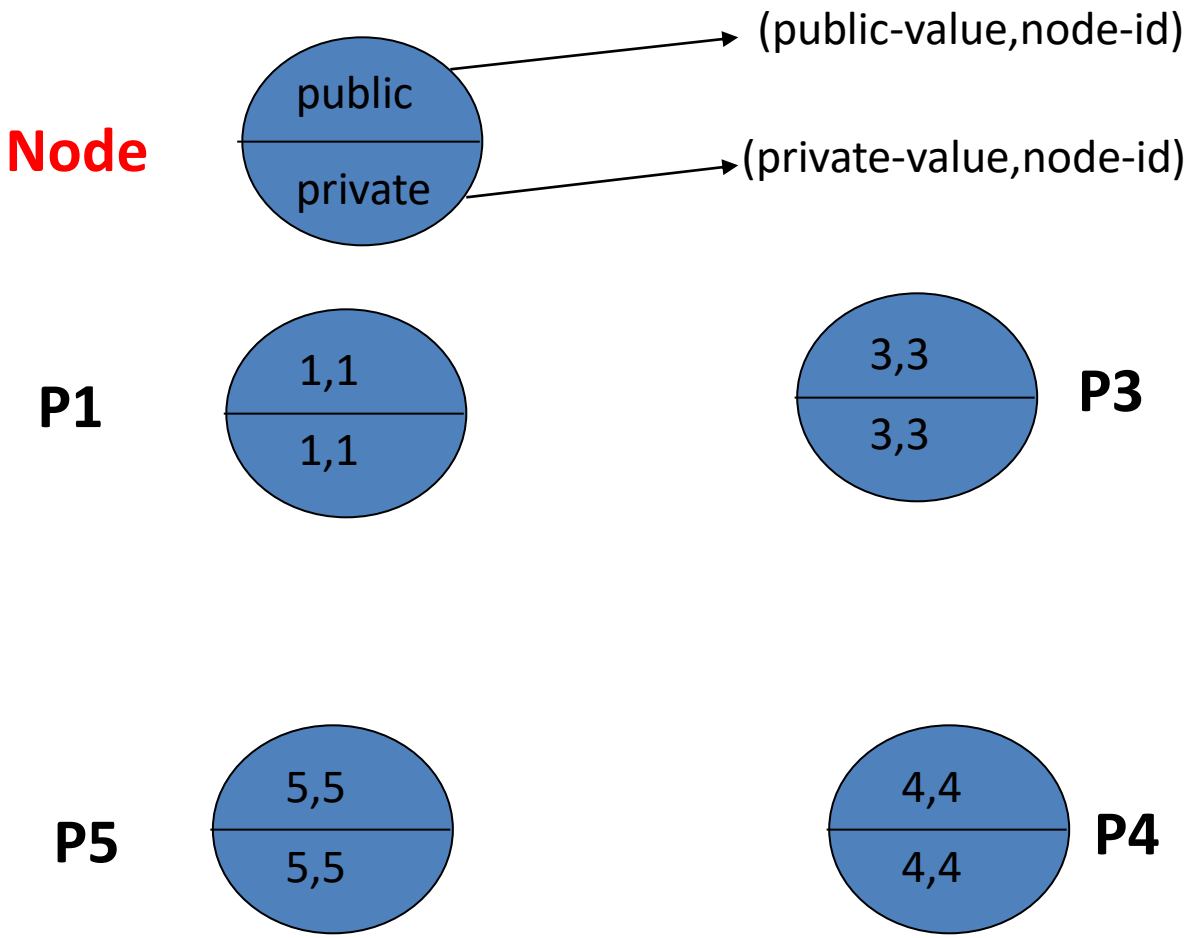
- Detect means that the probe with the private label of some process has returned to it, indicating a deadlock.

Mitchell and Merritt's Algorithm

- The proof that this algorithm works is based on the following claims and invariants.

1. For all processes u/v, v is at most u.
2. For any process u/v , if v < u, then u was set by a Transmit step.
3. If a deadlock is detected, then a cycle of blocked nodes exists.

Mitchell-Merritt Algorithm Example

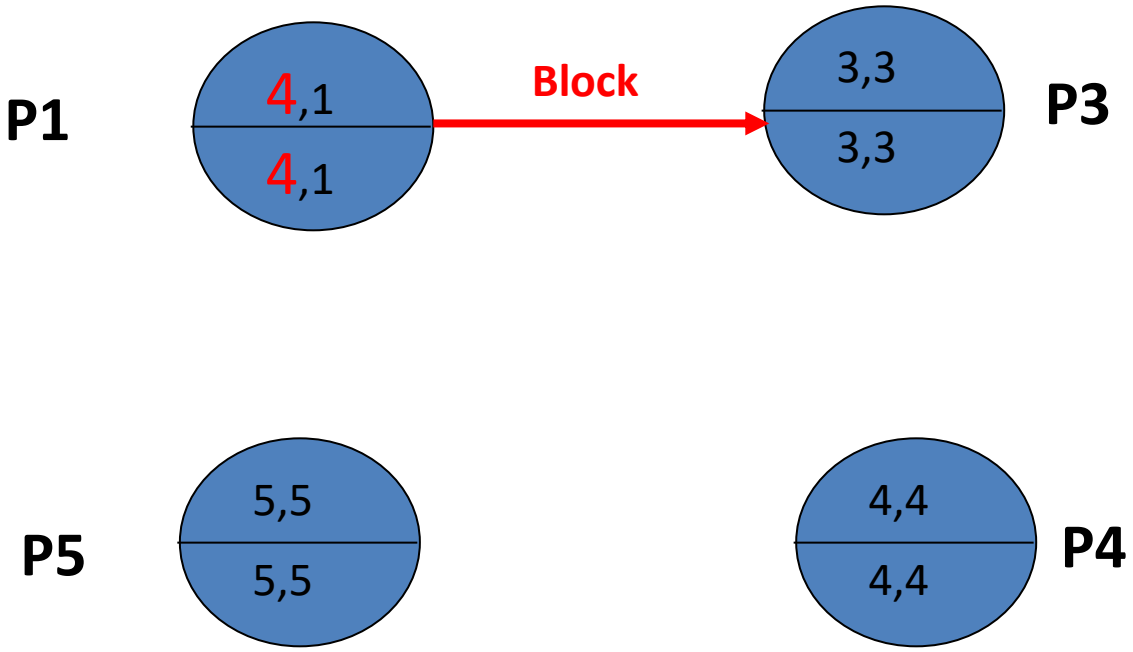


Initially both public and private label values at each node are equal

Mitchell-Merritt Algorithm Example cont…

Now suppose P1 is waiting for P3 (P1 → P3)

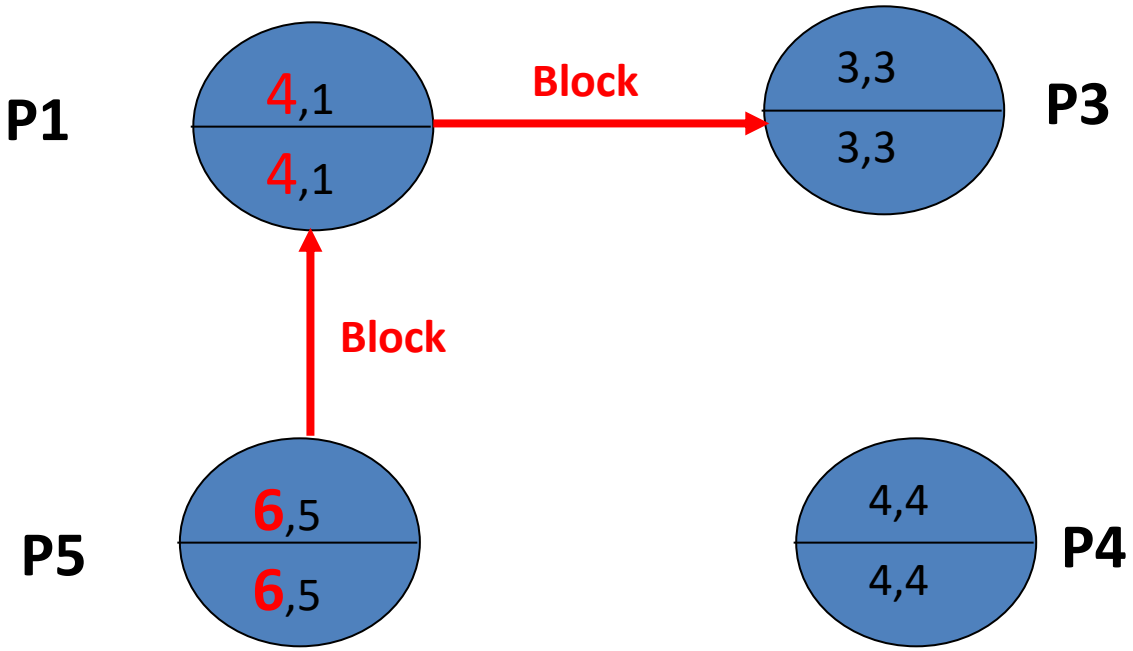
Block state will occur for P1



Mitchell-Merritt Algorithm Example cont…

Now suppose P5 is waiting for P1 (P5 → P1)

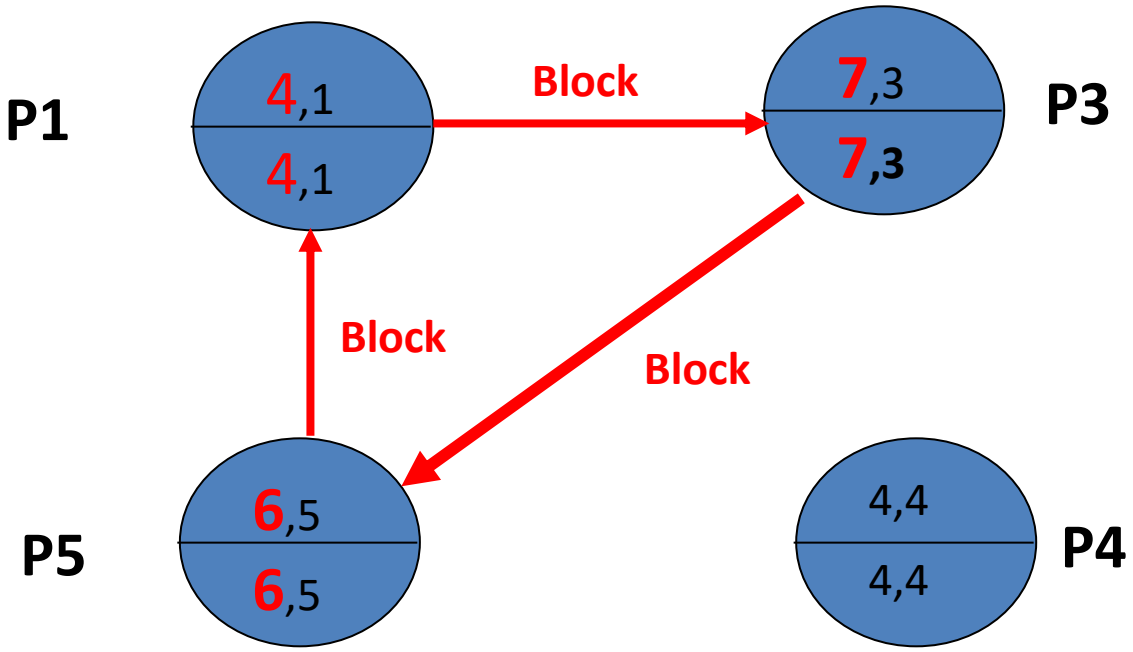
Block state will occur for P5



Mitchell-Merritt Algorithm Example cont…

Now suppose P3 is waiting for P5 (P3 → P5)

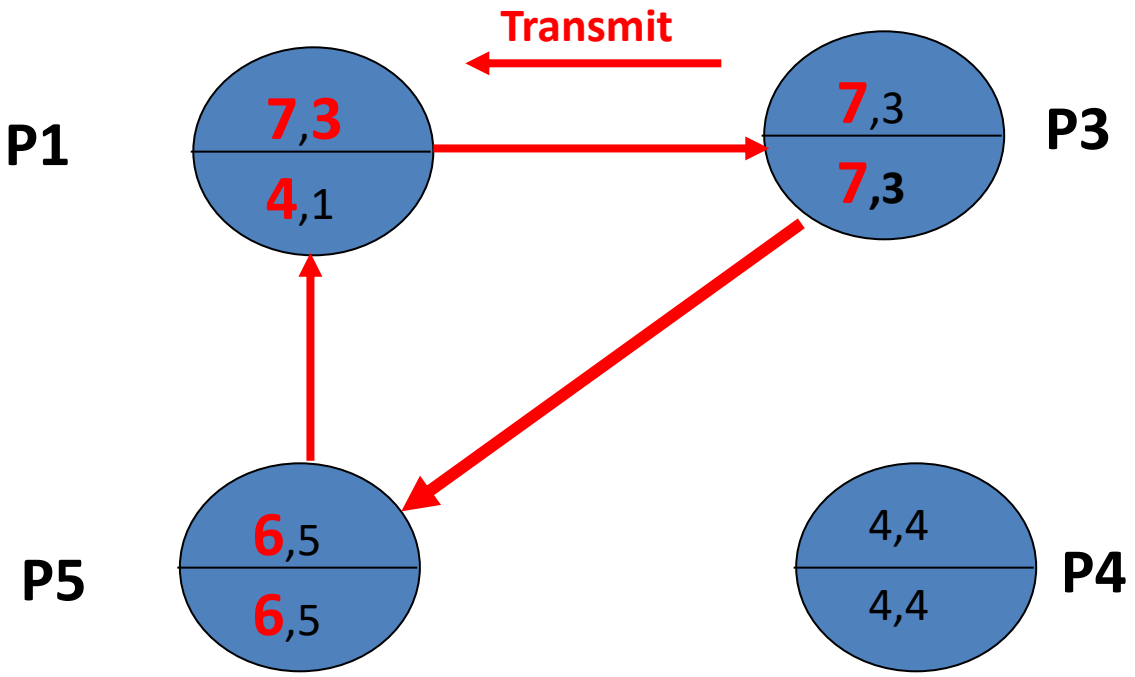
Block state will occur for P3



Mitchell-Merritt Algorithm Example cont…

Now P3 initiates Transmit Phase

P3 will transmit its public label to P1 (Reverse Direction)



Here P1 reads public label of P3

P1's public label =(4,1)

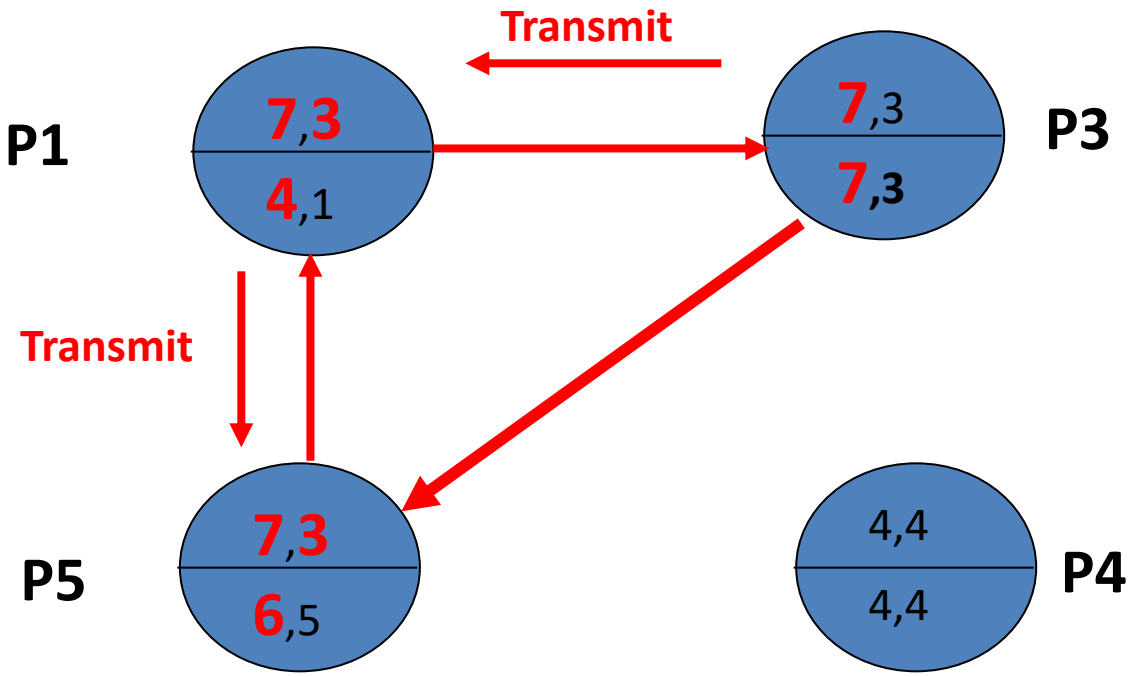
P3's public label =(7,3)

So P1 will change is public label to (7,3)

But No change for private label of P1

Mitchell-Merritt Algorithm Example cont…

P1 will transmit its public label to P5 (Reverse Direction)



P1's public label =(7,3)

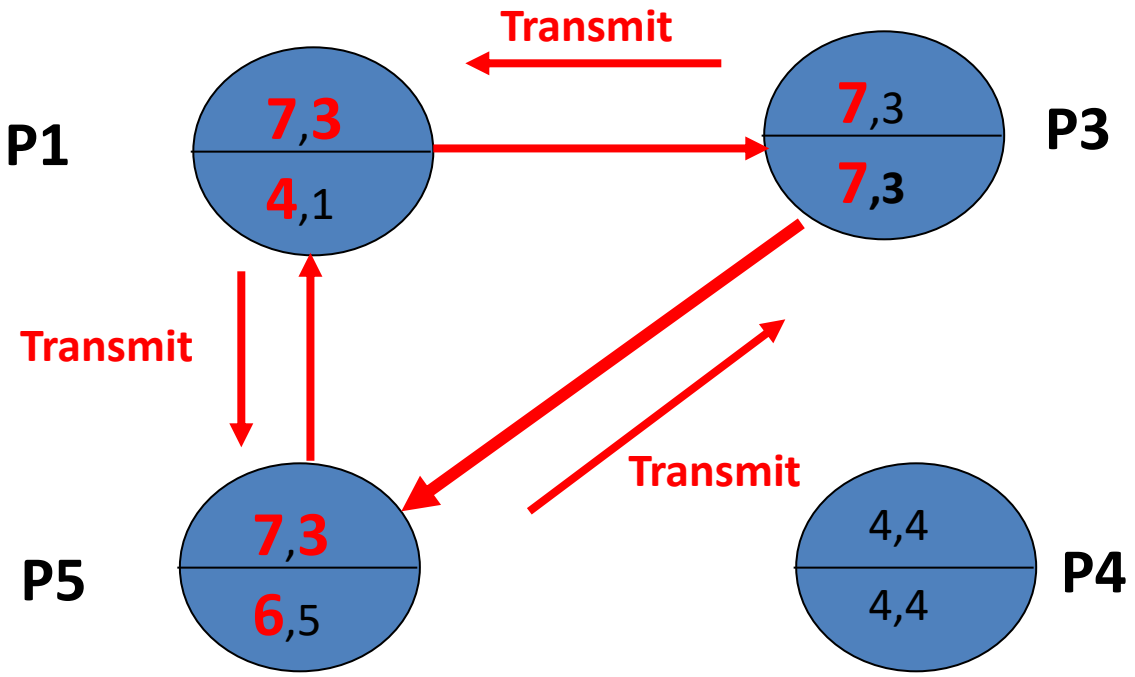
P5's public label =(6,5)

So P5 will change is public label to (7,3)

But No change for private label of P5

Mitchell-Merritt Algorithm Example cont…

P5 will transmit its public label to P3 (Reverse Direction)



P5's public label =7,3

P3's public label =7,3

So P3 Detects Deadlock

An Algorithm for the AND Model

- Chandy-Misra-Haas's distributed deadlock detection algorithm for AND model is based on edge-chasing.
- The algorithm uses a special message called probe , which is a triplet (i, j, k), denoting that it belongs to a deadlock detection initiated for process P_i and it is being sent by the home site of process P_j to the home site of process P_k.
- A probe message travels along the edges of the global WFG graph, and a deadlock is detected when a probe message returns to the process that initiated it.

An Algorithm for the AND Model

- A process P_j is said to be dependent on another process P_k if there exists a sequence of processes P_j, P_{i1}, P_{i2}, , ..., P_{im}, P_k such that each process except P_k in the sequence is blocked and each process, except P_j, holds a resource for which the previous process in the sequence is waiting.
- Process P_j is said to be locally dependent upon process P_k if P_j is dependent upon P_k and both the processes are on the same site.

An Algorithm for the AND Model

- Each process P_i maintains a Boolean array, dependent
  - $dependent_i(j)$ is true only if P_i knows that P_j is dependent on it.
- Initially, $dependent_i(j)$ is false for all i and j.

An Algorithm for the AND Model

- Algorithm : The following algorithm determines if a blocked process is deadlocked:
- If P_i is locally dependent on itself then declare a deadlock
- else for all P_j and P_k such that
  1. P_i is locally dependent upon P_j, and
  2. P_j is waiting on P_k, and
  3. P_j and P_k are on different sites,
- send a probe (i, j, k) to the home site of P_k

An Algorithm for the AND Model

- On the receipt of a probe (i, j, k), the site takes the following actions:
- if
  1. P_k is blocked, and
  2. $dependent_k(i)$ is false, and
  3. P_k has not replied to all requests by P_j then
- begin
  - $dependent_k(i)$ = true;
  - if k = i then declare that P_i is deadlocked
  - else for all P_m and P_n such that
    1. P_k is locally dependent upon P_m, and
    2. P_m is waiting on P_n, and
    3. P_m and P_n are on different sites,
  - send a probe (i, m, n) to the home site of P_n
- end

An Algorithm for the AND Model

- A probe message is continuously circulated along the edges of the global WFG graph and a deadlock is detected when a probe message returns to its initiating process.
- Performance Analysis
- One probe message (per deadlock detection initiation) is sent on every edge of the WFG which connects that two sites.
- Thus, the algorithm exchanges at most $m(n - 1)/2$ messages to detect a deadlock that involves m processes and that spans over n sites.
- The size of messages is fixed and is very small (only 3 integer words).
- Delay in detecting a deadlock is O(n)