CSE535 Asynchronous Systems
Deadlock Detection

YoungMin Kwon

Deadlock

- A subset of processes are permanently blocked because all processes are waiting for other processes.

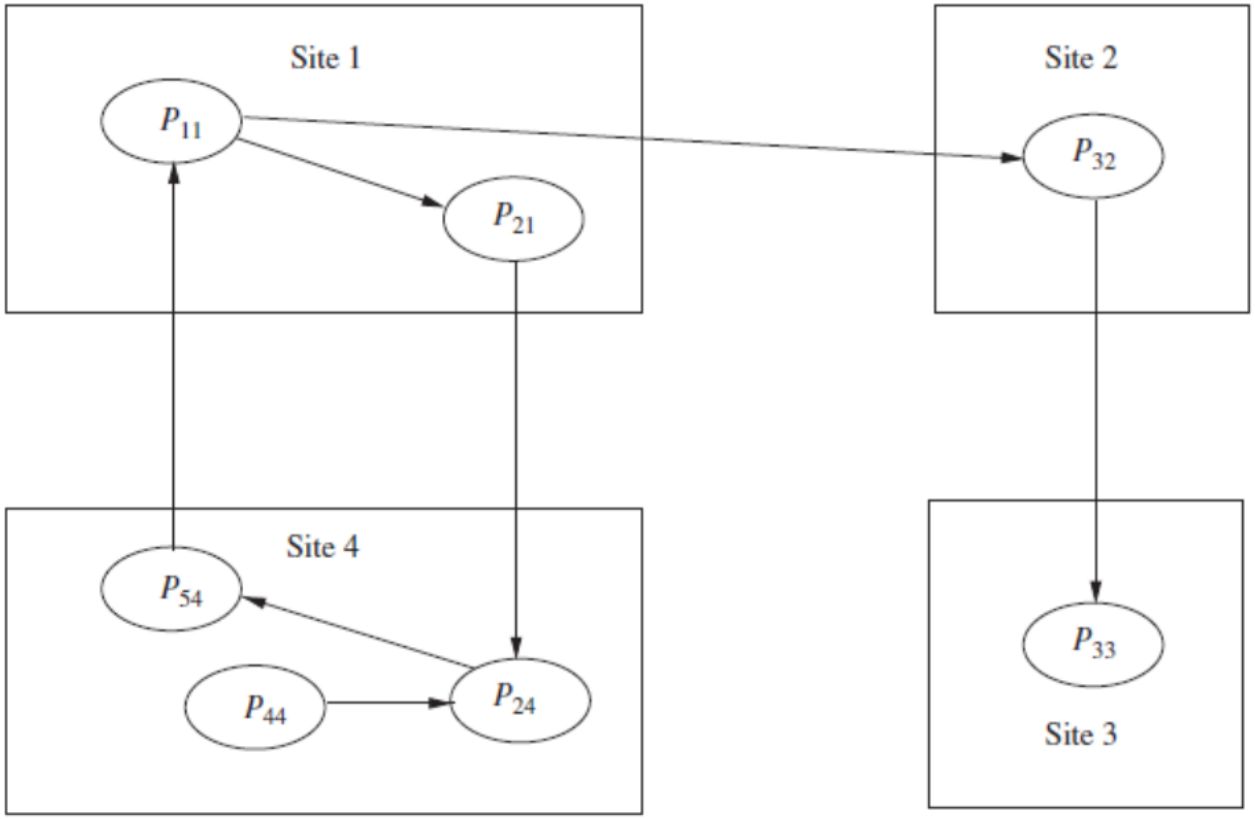
- Deadlock prevention
  - E.g. acquire all the resources simultaneously
- Deadlock avoidance
  - E.g. grant resources only when it is safe
- Deadlock detection
  - E.g. detect a deadlock and abort a deadlocked process

Wait-For Graph (WFG)

- Nodes of a WFG are processes
- There is a directed edge from p_i to p_j if p_i is blocked and is waiting for p_j to release some resource

- A system is deadlocked iff there exists a directed cycle or knot in the WFG

Wait-For Graph

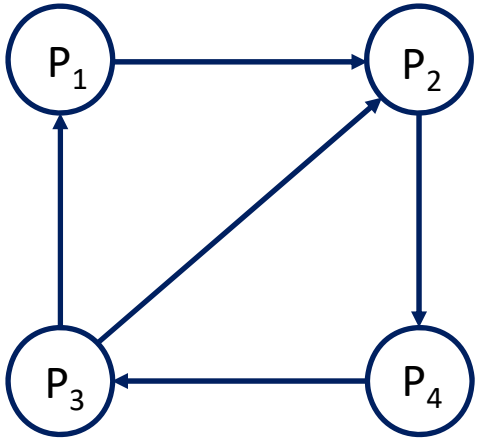


Models of Deadlocks

- Single-resource model
  - A process can have at most one outstanding request for only one unit of resource
  - Any cycle in WFG indicates a deadlock
- And model
  - A process can request more than one resources simultaneously
  - The request is satisfied if all the requested resources are granted
  - Any cycle in WFG indicates a deadlock

Models of Deadlocks

- Or model
  - A process can make a request for multiple resources simultaneously
  - The request is satisfied if any of the requested resources are granted
  - A knot in WFG indicates a deadlock



- Knot
  - A vertex v is in a knot if
    $\forall u_{\text{(u is reachable from v)}}$. v is reachable from u.

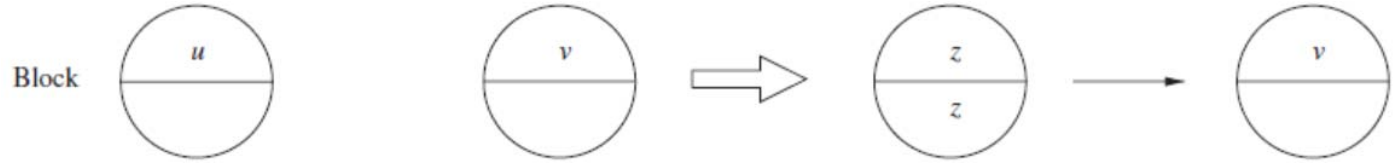
Mitchell and Merritt's Algorithm Single-Resource model

- Algorithm
  - Probes are sent in the opposite direction to the edge of WFG
  - When a probe initiated by a process comes back to it, the process declares deadlock

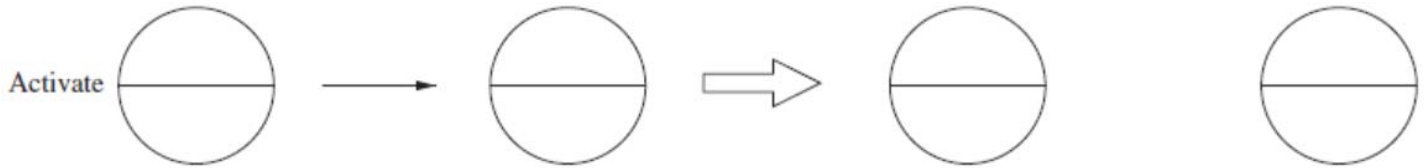
- Each node has two variables called labels
  - A private label: unique to the node
  - A public label: can be read from other processes
  - z = inc(u,v) yields a unique label greater than both u and v

Mitchell and Merritt's Algorithm

- Block: when a p_i requests a resource from p_j
  - p_j replies with its public label

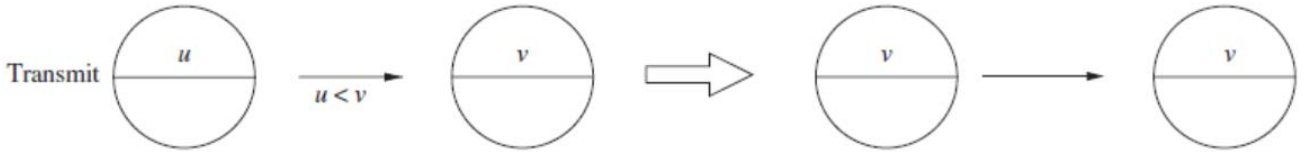


- Activate: a process acquired the resource from the process it was waiting for

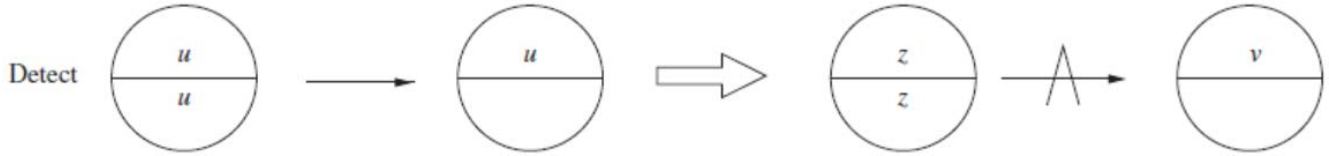


Mitchell and Merritt's Algorithm

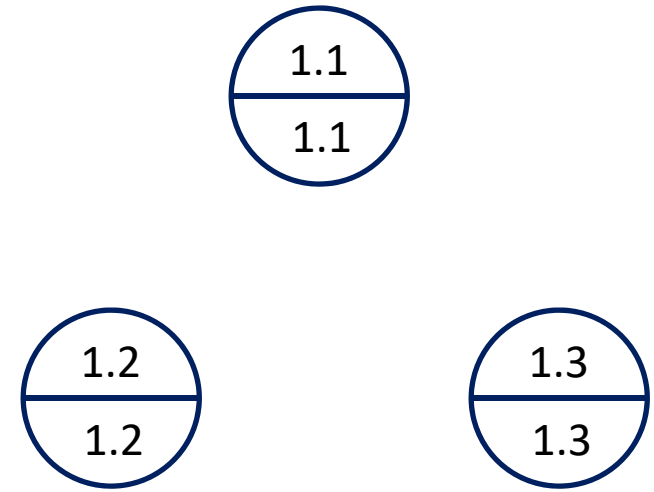
- Transmit:
  - Propagate a public labels (probe message) in the opposite direction to the edge of a WFG
  - When the process' public label is larger than the received label, ignore the message.

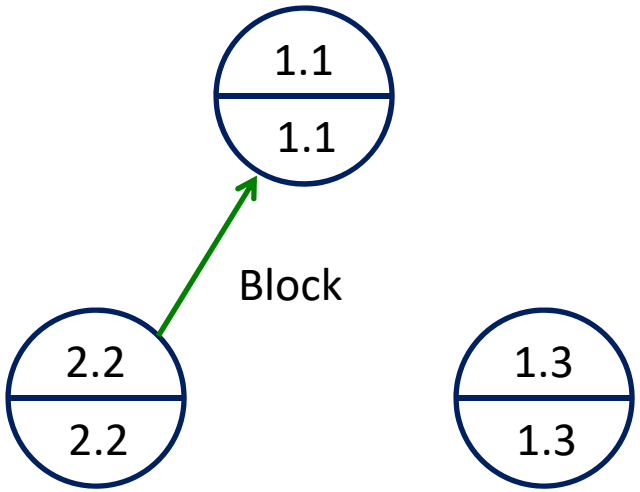


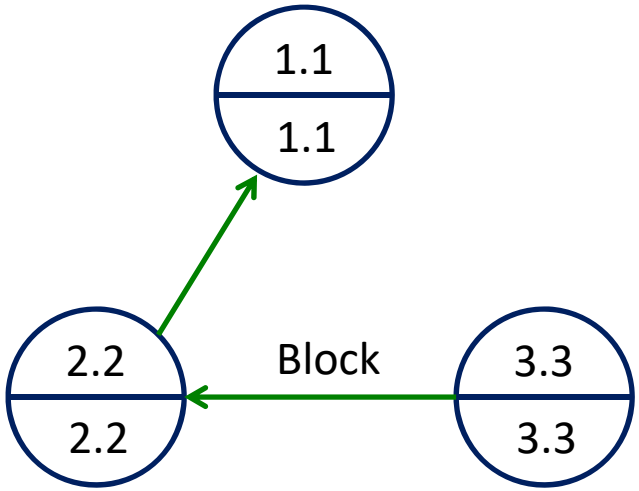
- Detect: if the private label of a process is returned to it, it indicates a deadlock

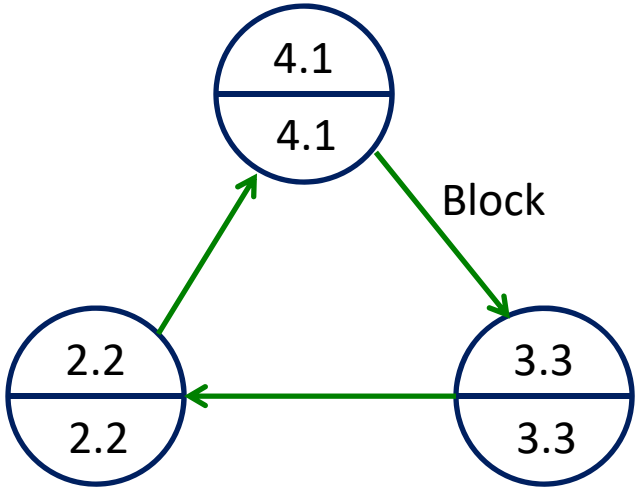


Mitchell and Merritt's Algorithm

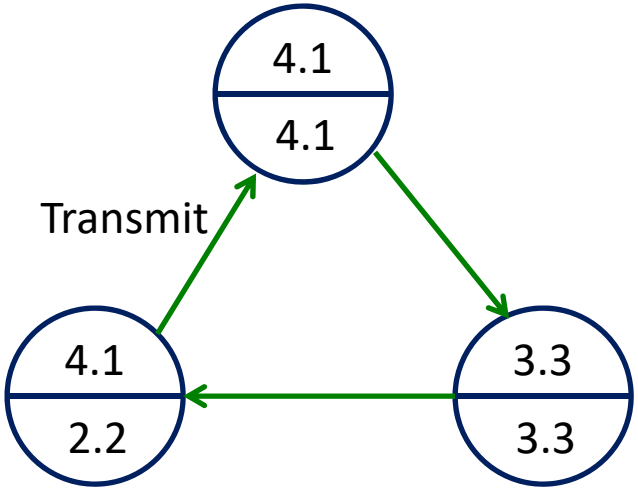


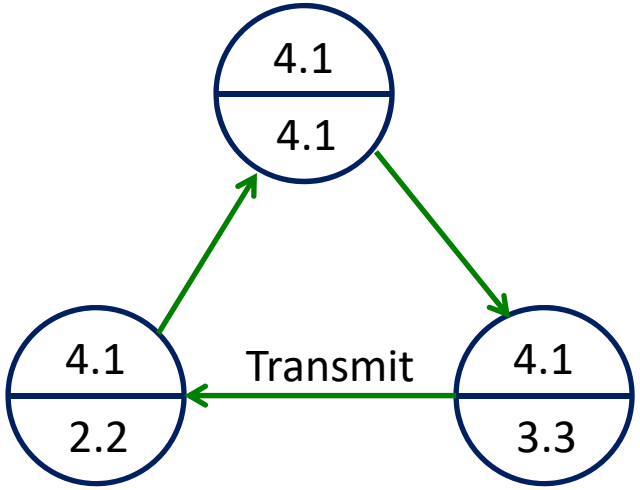


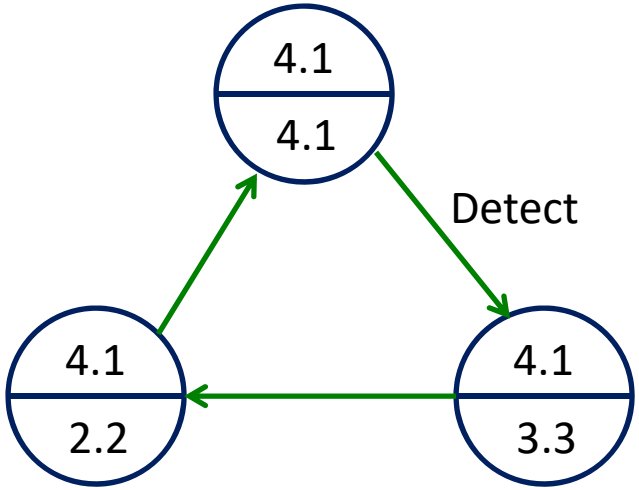




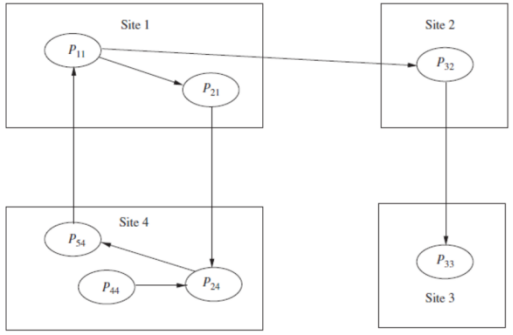
Mitchell and Merritt's Algorithm







Chandy-Misra-Haas Algorithm (And Model)



- Probe message (i, j, k)
  - Initiated by p_i and sent from p_j to p_k
- p_j is said to be dependent on p_k
  - if there exists a sequence of process p_j, p_{i1}, …, p_{im}, p_k such that
  - p_k is blocked and each process, except p_j, holds a resource for which the previous process is waiting
- p_j is said to be locally dependent on p_k if p_j is dependent on p_k and both are on the same site

Chandy-Misra-Haas Algorithm (And Model)

- Each process maintains a Boolean array $dependent_i$, where $dependent_i(j)$ is true if p_i knows that p_j is dependent on it

- Works like a marker flag that prevents multiple responses to the probes with the same initiator probe

Chandy-Misra-Haas Algorithm (And Model)

if P_i is locally dependent on itself
then
    declare a deadlock
else for all P_j and P_k such that
    (a) P_i is locally dependent upon P_j, and
    (b) P_j is waiting on P_k, and
    (c) P_j and P_k are on different sites,
send a probe (i, j, k) to the home site of P_k

Chandy-Misra-Haas Algorithm (And Model)

On the receipt of a probe (i, j, k), the site takes the following actions:

if (d) P_k is blocked, and
(e) $dependent_k(i)$ is false, and
(f) P_k has not replied to all requests 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
(a′) P_k is locally dependent upon P_m, and
(b′) P_m is waiting on P_n, and
(c′) P_m and P_n are on different sites,
send a probe (i, m, n) to the home site of P_n
end.

Chandy-Misra-Haas Algorithm (Or Model)

- Check if there is a knot
- A blocked process p_i initiates deadlock detection by sending queries to all processes in its dependent set DS_i
- If a blocked process receives a query(i, j, k)
  - If this is the first query received by p_k for the detection initiated by p_i (engaging query)
    - Propagate the query to all processes in its dependent set DS_k
    - set a local variable $num_k(i)$ to $|DS_k|$
  - If this is not the engaging query and p_k has been continuously blocked since the engaging query, send a reply message to p_j

Chandy-Misra-Haas Algorithm (Or Model)

- Process p_k maintains an array of boolean variable $wait_k(i)$
  - $wait_k(i)$ is true if p_k has been blocked since the engaging query from p_i
- When p_k receives reply(i, j, k),it decreases $num_k(i)$ only if $wait_k(i)$ holds

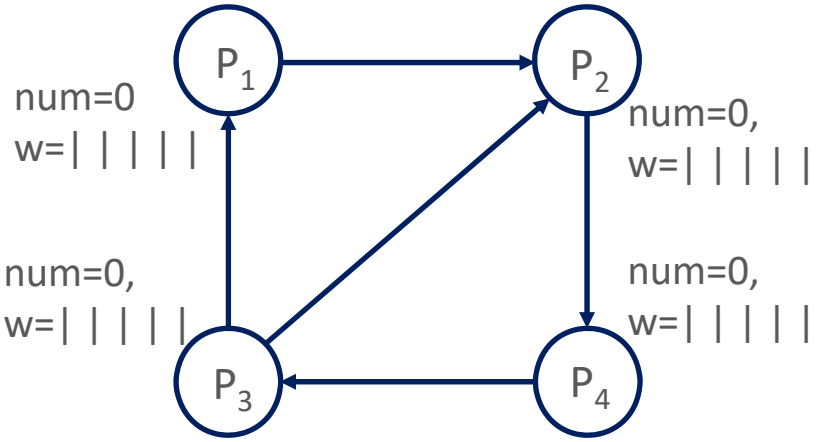
Chandy-Misra-Haas Algorithm (Or Model)

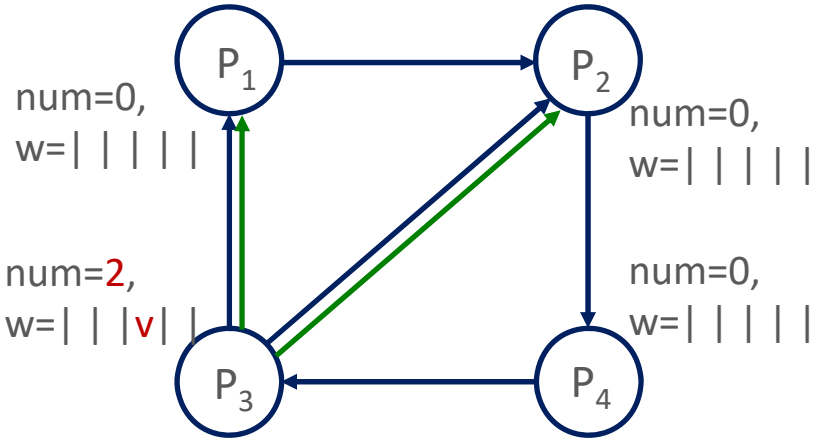
Initiate a diffusion computation for a blocked process P_i:
send *query*(i, i, j) to all processes P_j in the dependent set DS_i of P_i;
$num_i(i) := |DS_i|$; $wait_i(i) := true$;

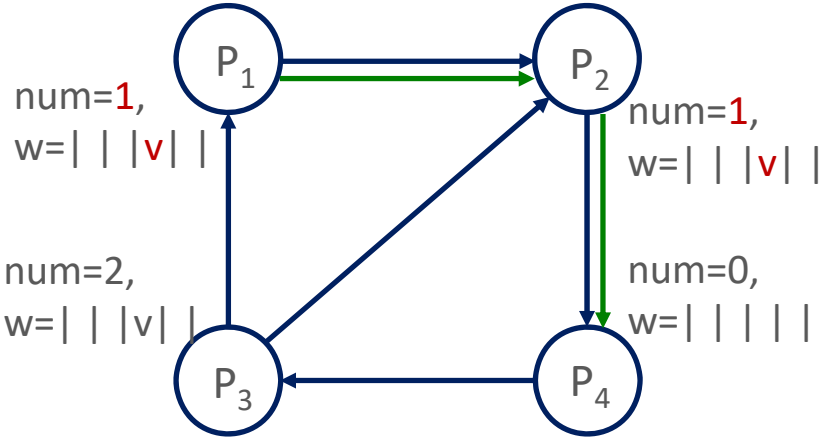
When a blocked process P_k receives a *query*(i, j, k):
if this is the engaging *query* for process P_i then
send *query*(i, k, m) to all P_m in its dependent set DS_k;
$num_k(i) := |DS_k|$; $wait_k(i) := true$
else if $wait_k(i)$ then send a *reply*(i, k, j) to P_j.

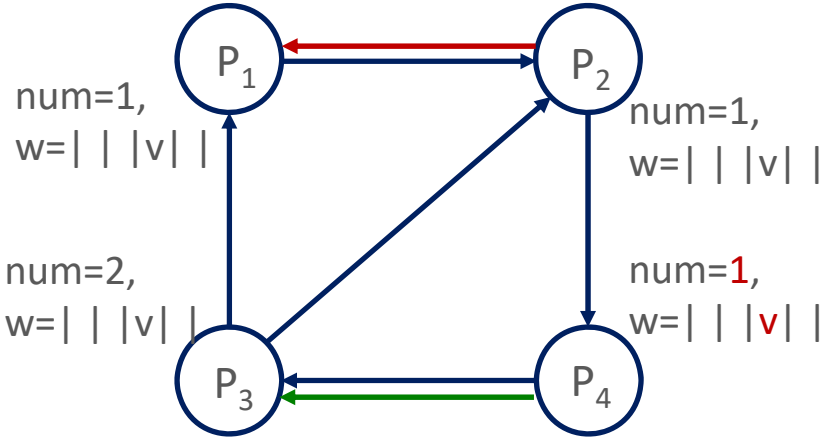
When a process P_k receives a *reply*(i, j, k):
if $wait_k(i)$ then
$num_k(i) := num_k(i) - 1$;
if $num_k(i) = 0$ then
if $i = k$ then **declare a deadlock**
else send *reply*(i, k, m) to the process P_m
which sent the engaging query.

Chandy-Misra-Haas Algorithm (Or Model)









Chandy-Misra-Haas Algorithm (Or Model)