# Distributed Deadlock Detection

Y. V. Lokeswari

AP/ CSE

Reference: **Advanced Concepts in OS**

by

**Mukesh Singhal & N.G. Shivaratri**

# Distributed Deadlock Detection

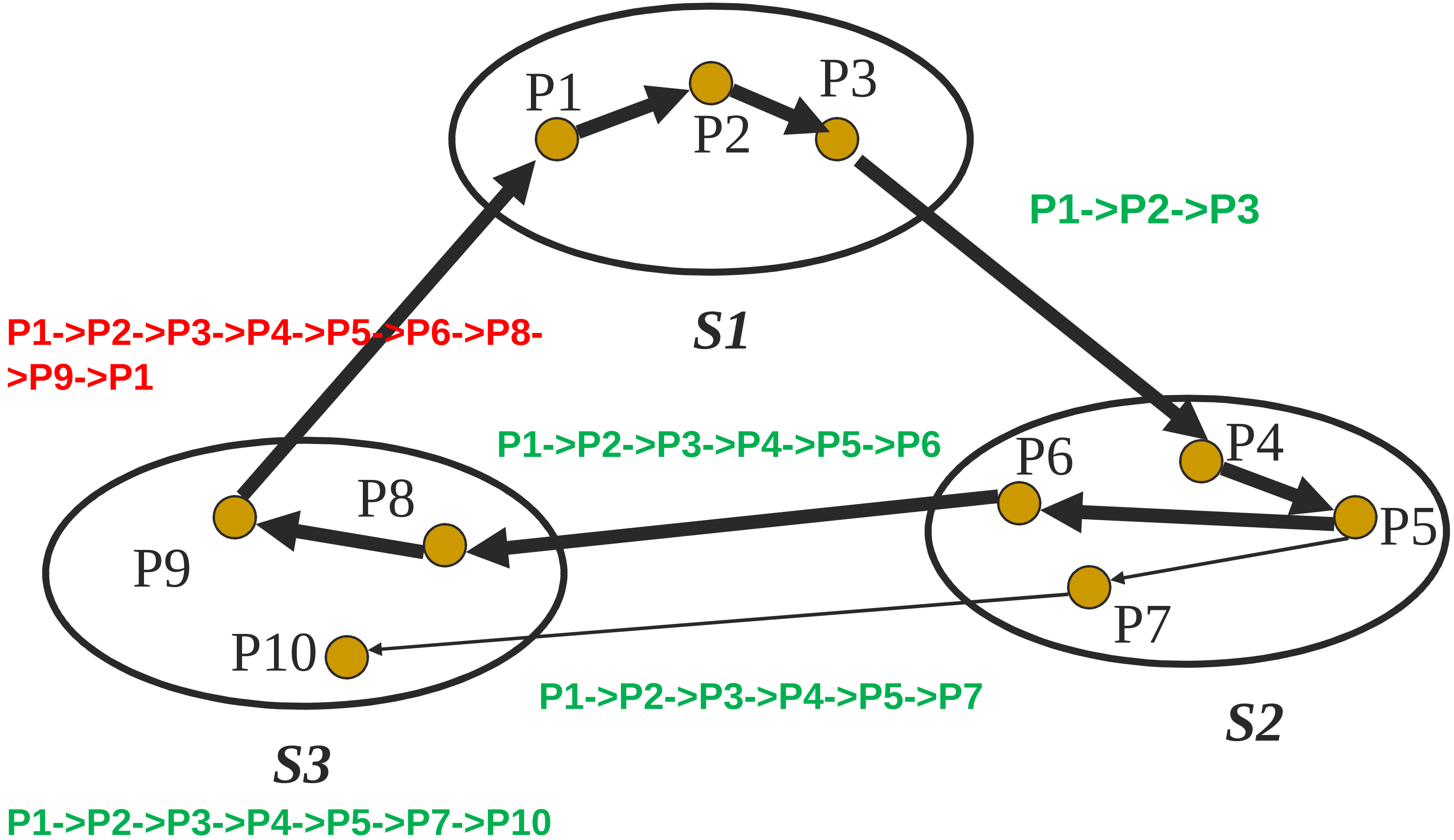
Obermarck's Algorithm

&

Chandy, Misra and Haas Algorithm

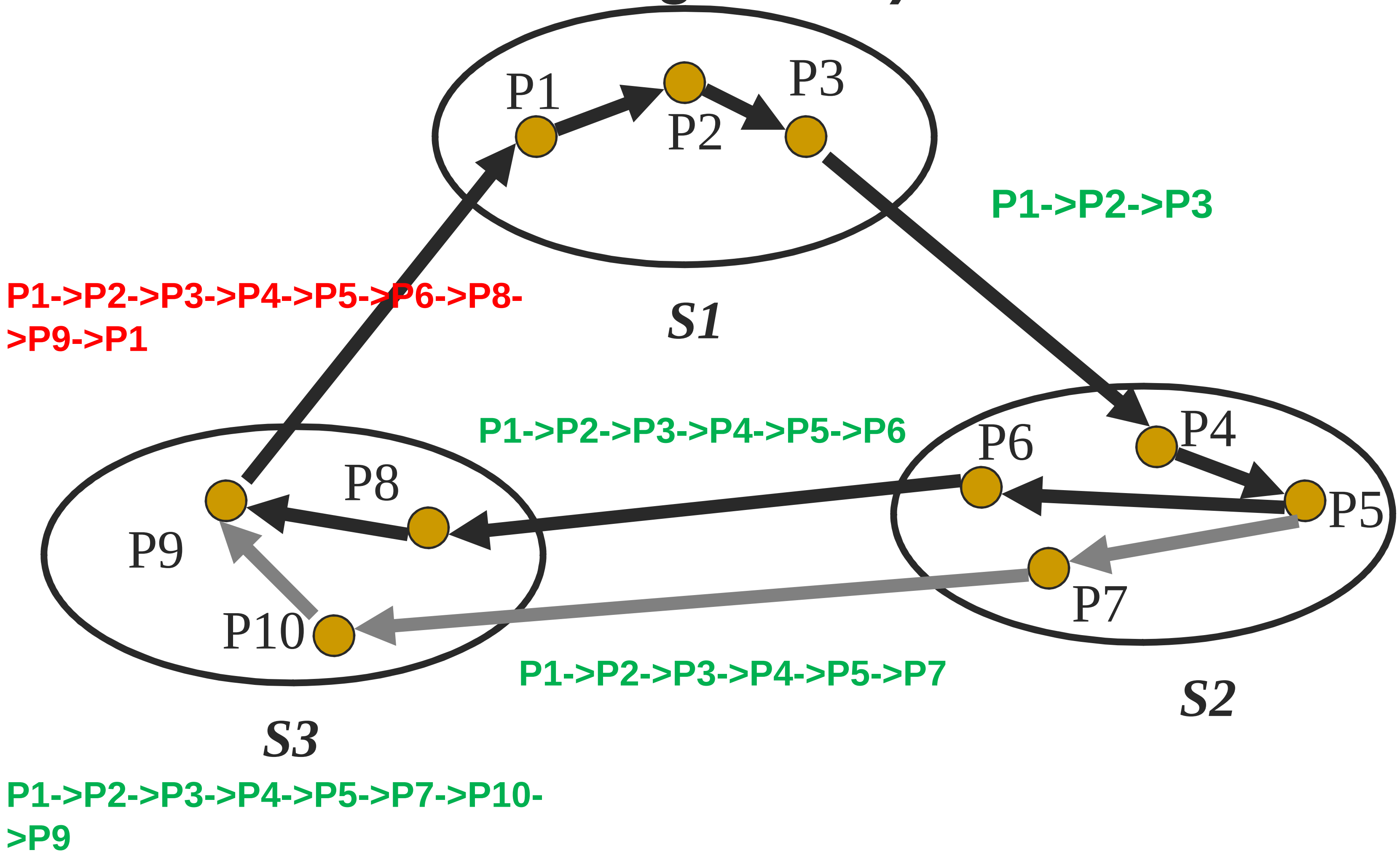
**Deadlock** in the **AND** model; there is a **cycle** but no knot **No Deadlock** in the **OR** model

# Path-Pushing Algorithm (Obermarck's Algorithm)



**Deadlock** in both the **AND** model and the **OR** model; there are **cycles** and a **knot**

# Path-Pushing Algorithm (Obermarck's Algorithm)



# Drawbacks of Path-Pushing Algorithm

- The algorithm **detects Phantom deadlocks**
- The algorithm sends ***n(n-1)/2*** messages to detect deadlock involving ***n*** sites.
- Size of the message is ***O(n)***
- Delay in detecting deadlock is ***O(n)***
- Overhead in sending WFG information to each site via network.
- By the time deadlock is declared, deadlock would have been resolved. (**Phantom deadlocks).**

# Chandy Misra Haas's Algorithm

## Edge-Chasing Algorithm

- Instead of sending the WFG from each site, this method sends a Probe message which has a fixed size.
- Probe(i,j,k)

# Chandy Misra Haas's Algorithm

## Edge-Chasing Algorithm

**Controller sending a probe**

```
if Pj is locally dependent on itself
      then declare deadlock
else for all Pj, Pk  such that
      (i) Pi is locally dependent on Pj,
      (ii) Pj is waiting for 'Pk and
      (iii) Pj, Pk are on different controllers.
send probe(i, j, k). to home site of Pk
```

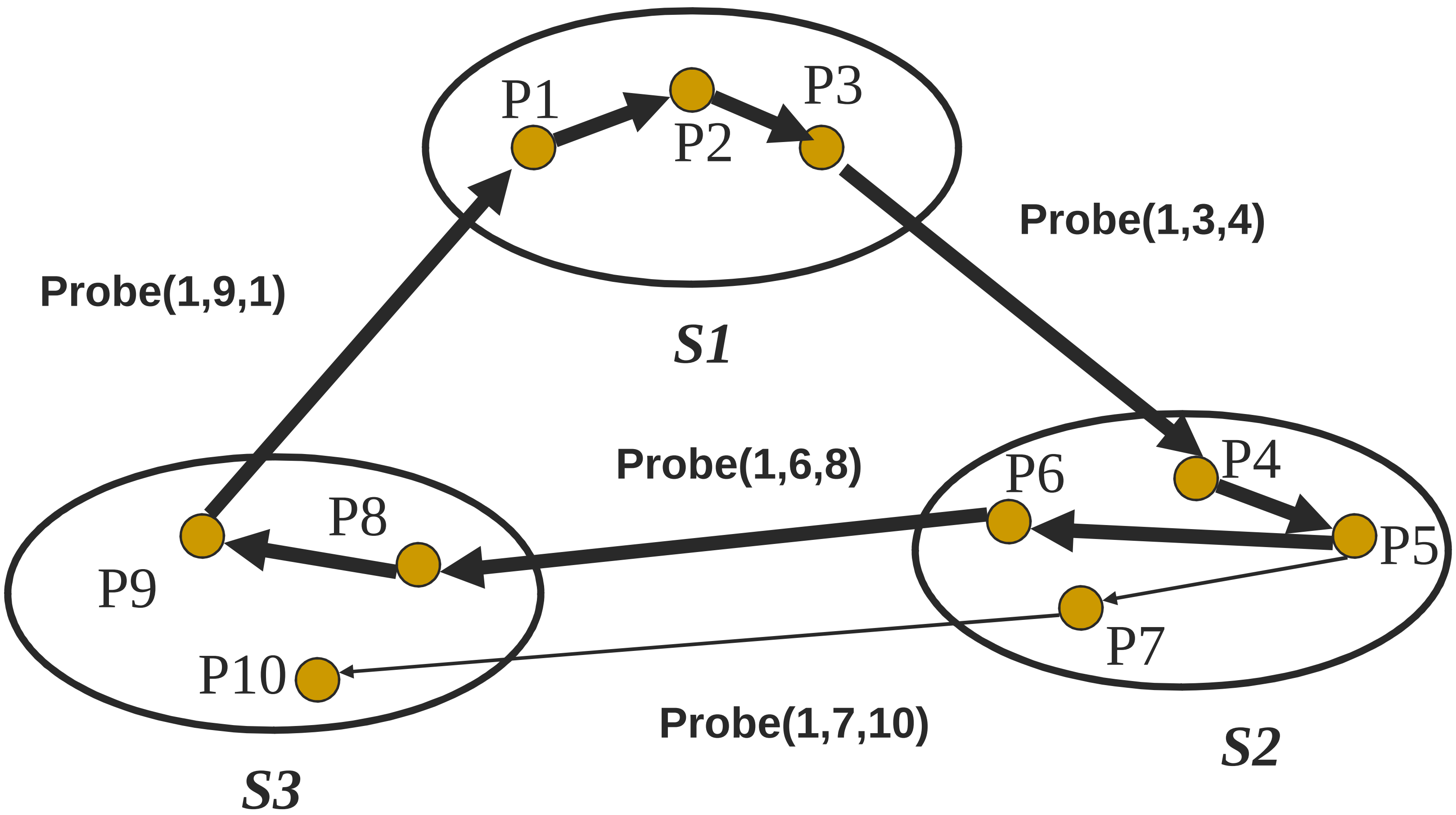
**Controller receiving a probe**

```
if
      (i) Pk is idle / blocked
      (ii) dependentk(i) = false, and
      (iii) Pk has not replied to all requests of to Pj
then begin
      "dependents""k"(i) = true;
      if k == i
      then declare that Pi  is deadlocked
      else for all Pa, Pb  such that
               (i) Pk is locally dependent on Pa,
               (ii) Pa is waiting for 'Pb and
               (iii) Pa, Pb are on different controllers.
      send probe(i, a, b). to home site of Pb
end
```

**Deadlock** in the **AND** model; there is a **cycle** but no knot

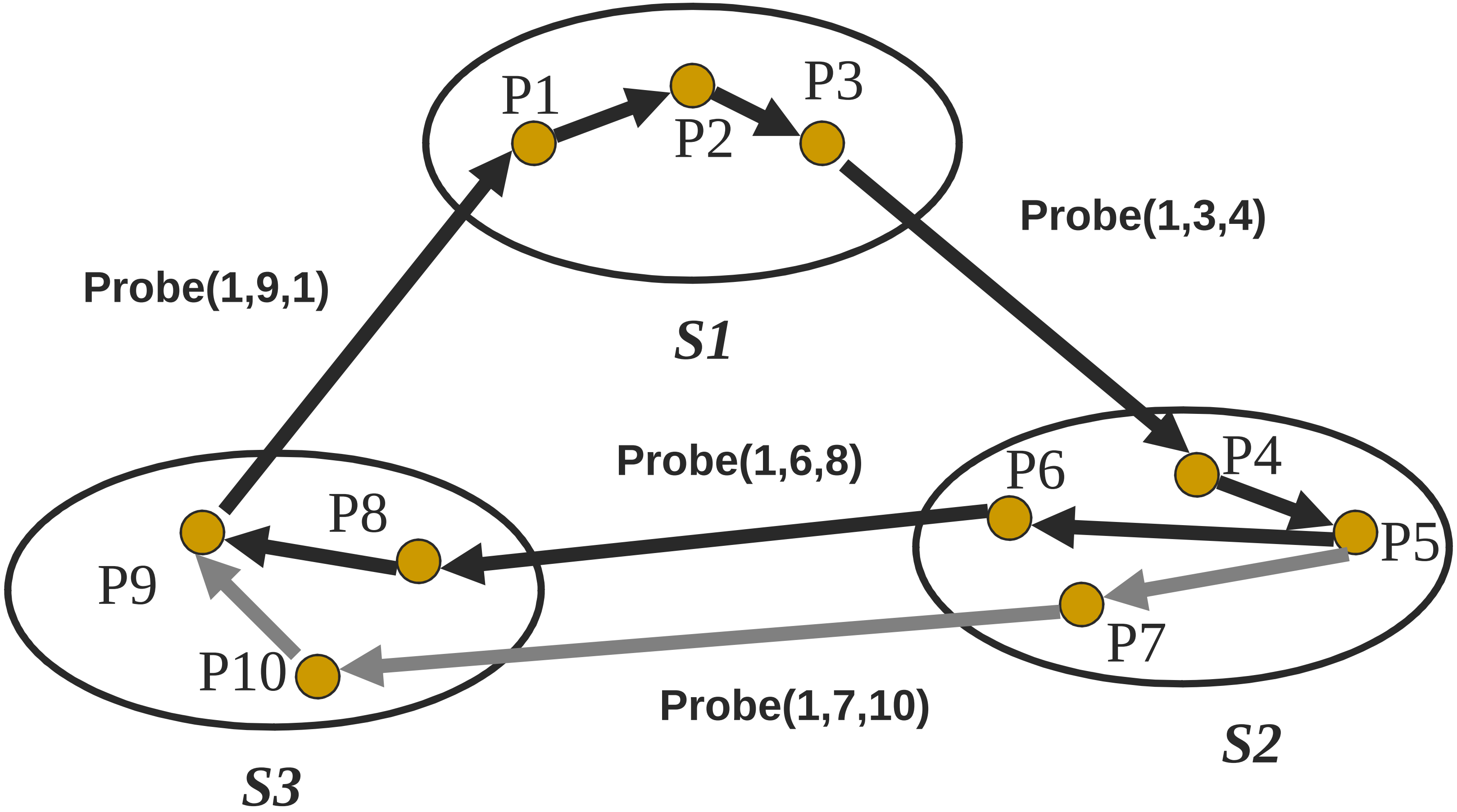
**No Deadlock** in the **OR** model

# Edge-Chasing Algorithm



**Deadlock** in both the **AND** model and the **OR** model;
there are **cycles** and a **knot**

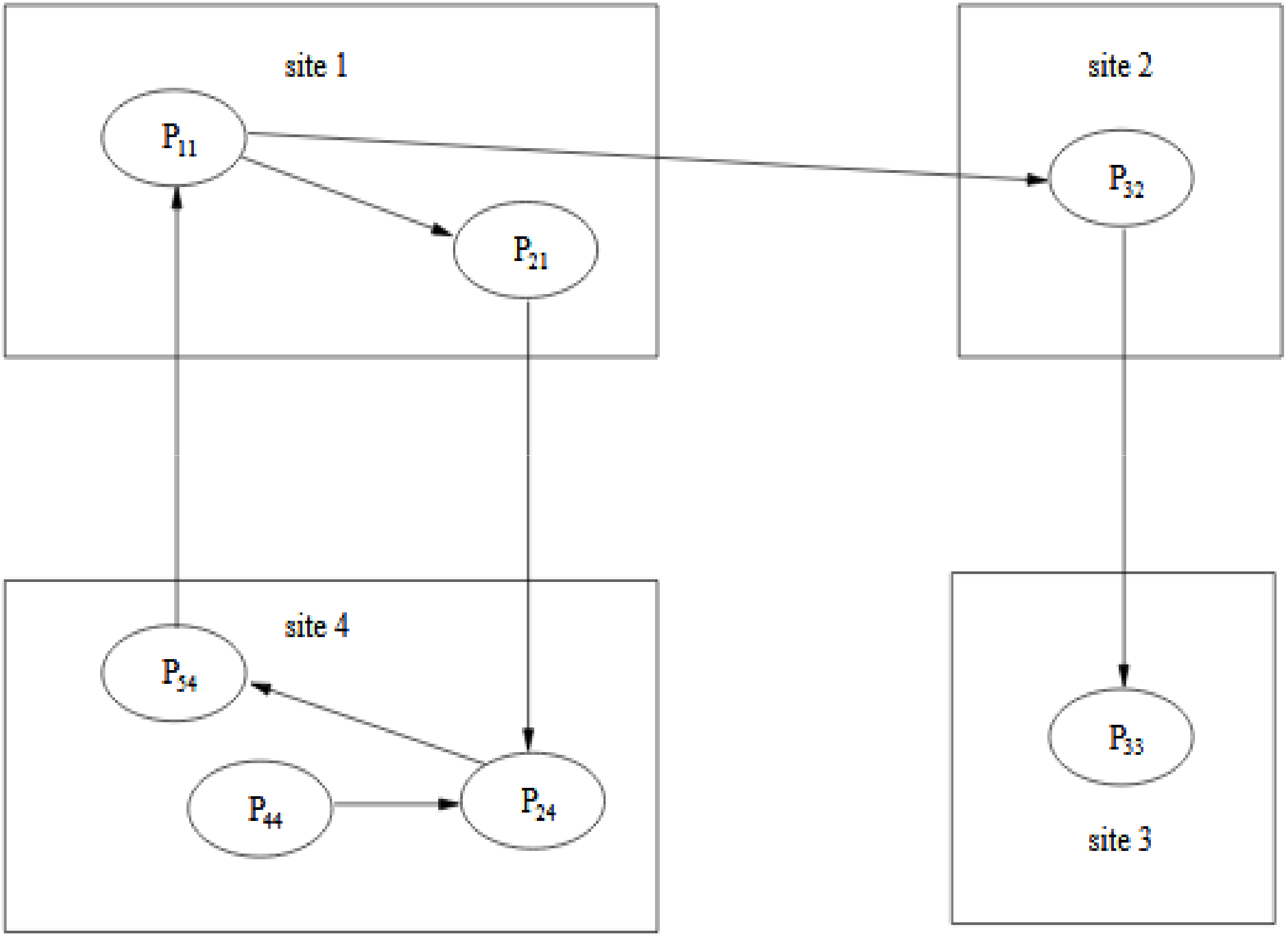
# Edge-Chasing Algorithm



# Edge-Chasing Algorithm

- The algorithm at most exchanges ***m(n-1)/2*** messages to detect deadlock involving ***m*** processes and spans over ***n*** sites.
- Size of the message is fixed. Only 3 integer values.
- Delay in detecting deadlock is ***O(n)***

# Workout Example



# Other Deadlock detection Algorithms

- Diffusion Computation Algorithm
- Global State Detection based Algorithm

## Thank You