BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, Pilani Pilani Campus

# The Ho-Ramamoorthy Algorithm

## Instructor :

Dr. Avinash Gautam

## Presented By :

- Keval Kulkarni
- Brijgopal Dixit
- Shouvik Chatterjee

# Introduction

- Centralized deadlock detection in Distributed Systems.
- The **Ho-Ramamoorthy** algorithm based on two centralized deadlock detection approaches –
    1. *One Phase algorithm*
    2. *Two phase algorithm*
- Deadlock is detected by Control site.
- Control site collects process status table from local sites.

# Terminology

I. **Control Site(Controller)**
   - Designated site that maintains the WFG of entire system.
   - Checks it for the existence of deadlock cycles.

II. **Status Table**
   - Contains status of all the process includes resource locked or waited upon.

III. **Resource Status Table**
   - Keeps track of process that have locked or are waiting for resources.

IV. **WFG**
   - Wait for graph
   - Showing dependency between the processes.

# The One-Phase Algorithm

- Each site maintains 2 status tables
  1. *Resource status* table
  2. *Process status* table.
- Control site periodically collects these tables from each site.
- Control site then builds and analyze the WFG.
- Check for the cycles in WFG.
- If cycle is not present then system is free from deadlock
- Otherwise a deadlock is detected in system.

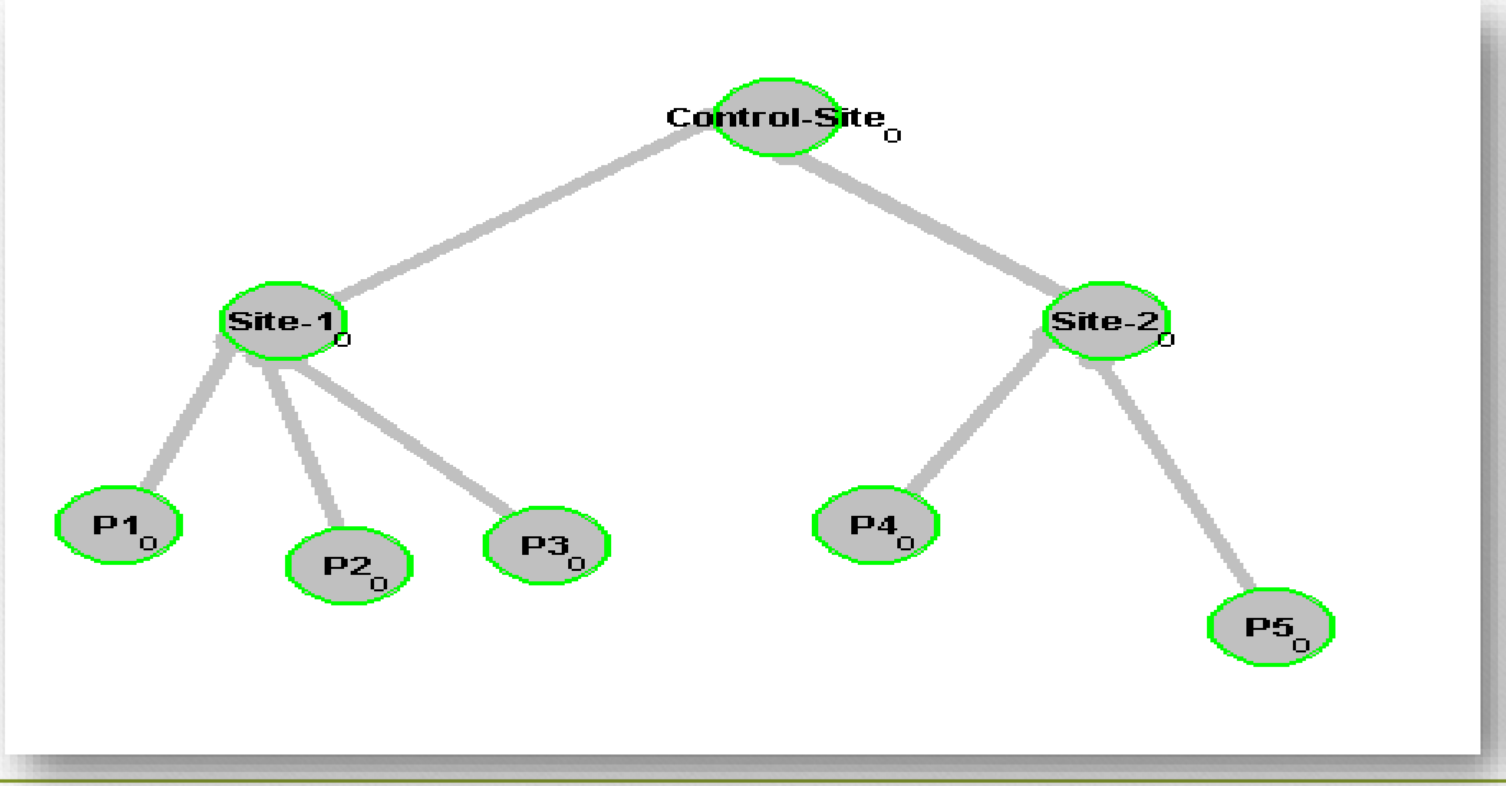
# The Two-Phase Algorithm

- Each site maintains a status table of all processes initiated at that site.
- Controller requests the status table from each site.
- Controller then constructs WFG from these tables, searches for cycles.
- If no cycles, no deadlocks.
- Otherwise, (cycle exists): Request for state tables again.
- Construct WFG based *only* on common transactions in the 2 tables.
- If the same cycle is detected again, system is in deadlock.

# Assumptions

- ❑ **Number of Processes** => 5 (P1-P5)
- ❑ **Number of Resources** => 4 (R1-R4)
- ❑ **Number of Sites** => 3
- ❑ Considering all the resources as global.
- ❑ Resources request generated randomly by Processes.

# Example : One Phase



- ❑ **Processes** : P1,P2,P3,P4,P5
- ❑ **Resources** : R1, R2, R3, R4
- ❑ **Systems** : Site-1, Site-2, Control site

# Example : One-Phase

**STEP-1**

- Site-1 Constructs *Resource status* and *Process status* table

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P1 | 1 | 0 | 0 | 0 |

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P2 | 0 | -1 | -1 | 0 |

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P3 | -1 | 1 | -1 | 0 |

**Site -1 PST**

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P1 | 1 | 0 | 0 | 0 |
| P2 | 0 | -1 | -1 | 0 |
| P3 | -1 | 1 | -1 | 0 |

**Site -1 RST**

| | P1 | P2 | P3 |
|---|---|---|---|
| R1 | 1 | 0 | -1 |
| R2 | 0 | -1 | 1 |
| R3 | 0 | -1 | -1 |
| R4 | 0 | 0 | 0 |

**(-1 : request & 1 : allocated)**

# Example : One-Phase

## STEP-2

- Site-2 Construct *Resource status* and *Process status* table.

|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P4 | -1 | -1 | -1 | 0 |

|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P5 | -1 | -1 | 1 | 1 |

**Site -2 PST**

|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P4 | -1 | -1 | -1 | 0 |
| P5 | -1 | -1 | 1 | 1 |

**Site -2 RST**

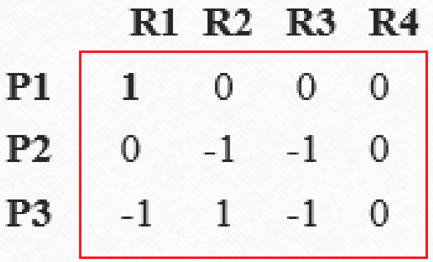
|  | P4 | P5 |
|---|---|---|
| R1 | -1 | -1 |
| R2 | -1 | -1 |
| R3 | -1 | 1 |
| R4 | 0 | 1 |

**(-1 : request & 1 : allocated)**

# Example : One-Phase

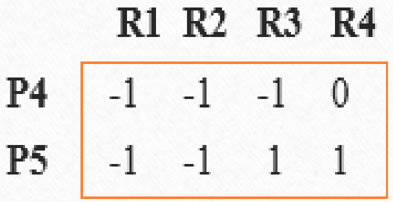
**STEP-3**

- Control site collect tables from all local sites and construct Global process status table.



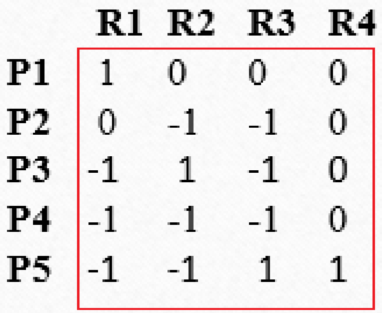
|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P1 | 1 | 0 | 0 | 0 |
| P2 | 0 | -1 | -1 | 0 |
| P3 | -1 | 1 | -1 | 0 |

**Site-1**



|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P4 | -1 | -1 | -1 | 0 |
| P5 | -1 | -1 | 1 | 1 |

**Site-2**



|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P1 | 1 | 0 | 0 | 0 |
| P2 | 0 | -1 | -1 | 0 |
| P3 | -1 | 1 | -1 | 0 |
| P4 | -1 | -1 | -1 | 0 |
| P5 | -1 | -1 | 1 | 1 |

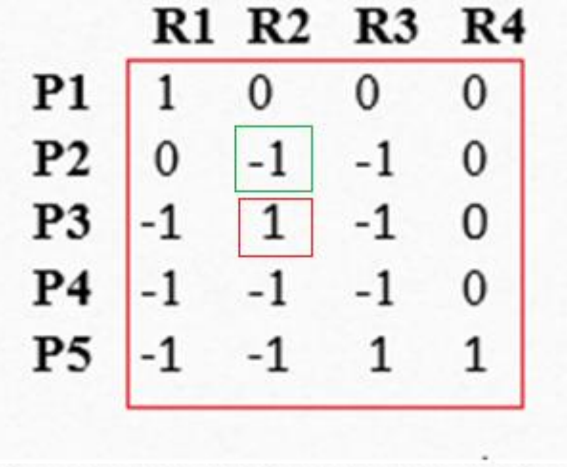
**Control Site**

(-1 : request & 1 : allocated)

# Example : One-Phase

## STEP-3

- Controller constructs the WFG using status table.



|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P1 | 1 | 0 | 0 | 0 |
| P2 | 0 | -1 | -1 | 0 |
| P3 | -1 | 1 | -1 | 0 |
| P4 | -1 | -1 | -1 | 0 |
| P5 | -1 | -1 | 1 | 1 |

P2 → R2 R2 → P3

P2 → P3 P2 → P5

P3 → P1 P3 → P5

P4 → P1 P4 → P3 P4 → P5

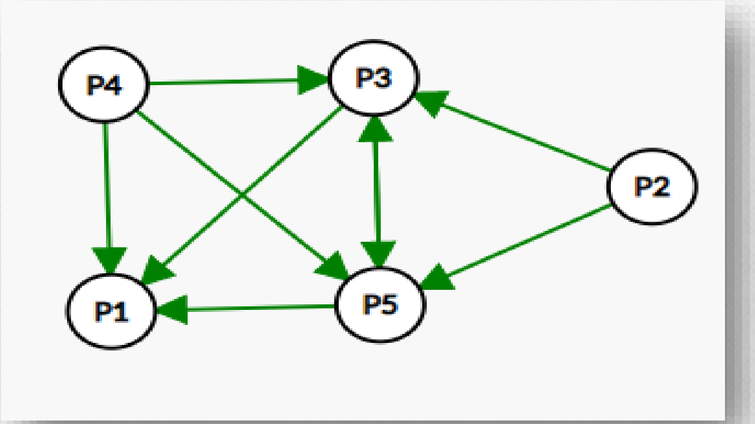
P5 → P1 P5 → P3

(-1 : request & 1 : allocated)

# Example : One-Phase

## STEP-4

- Controller constructs the WFG using status table and check for cycle.
- Nodes are represented as processes



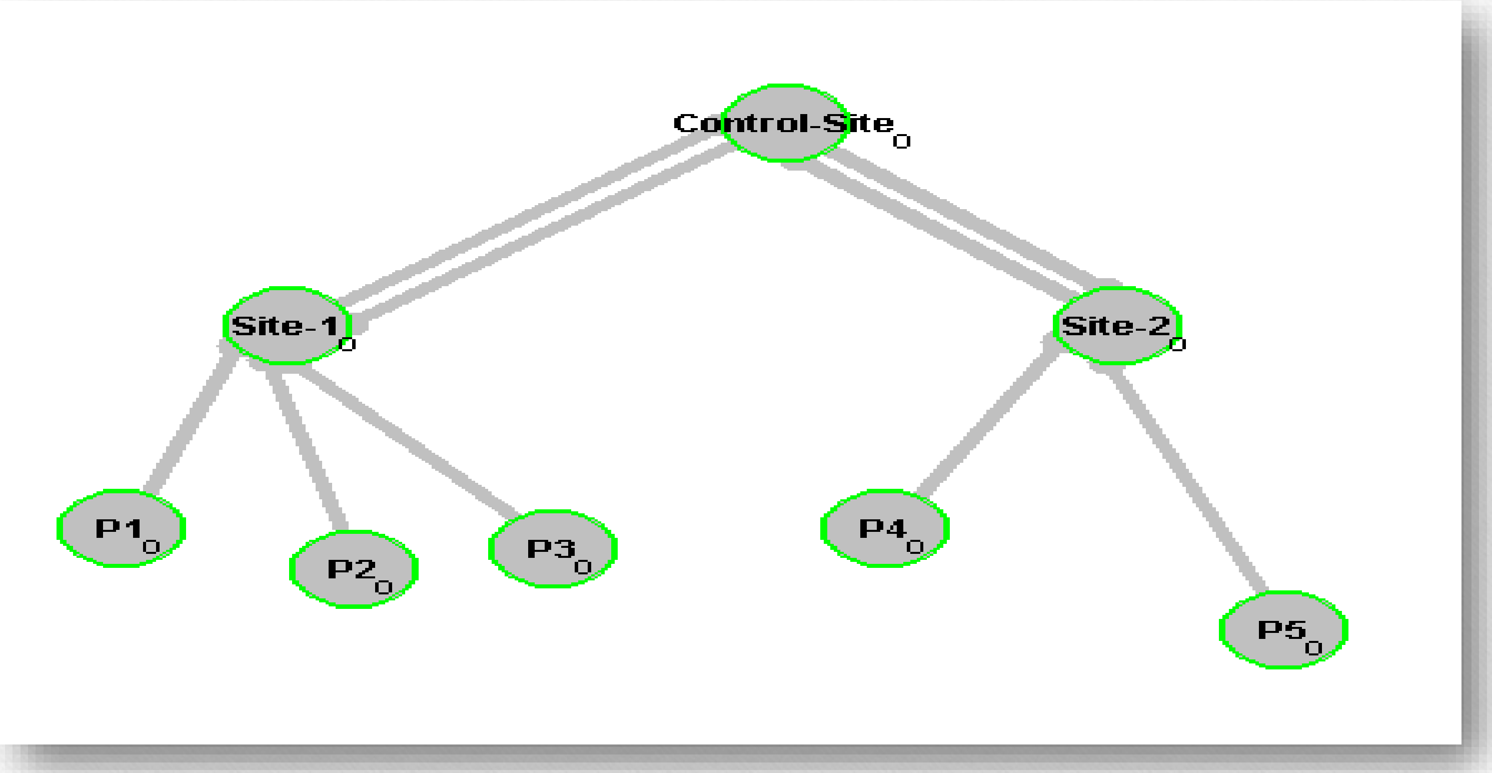
Cycle found between processes : P3 and P5

P3 → P5

P5 → P3

Deadlock Detected

# Example : Two-Phase



- □ **Processes** : P1,P2,P3,P4,P5
- □ **Resources** : R1, R2, R3, R4
- □ **Systems** : Site-1, Site-2, Control site

# Example : Two-Phase

## STEP-1

- Every site maintains status table.

### Site-1

|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| **P1** | 0 | 0 | -1 | -1 |
| **P2** | 0 | -1 | -1 | 0 |
| **P3** | 0 | -1 | -1 | 1 |

| Site id | Process id | Hold resource id | Requesting resource id |
|---|---|---|---|
| S1 | P1 | --- | R3,R4 |
| S1 | P2 | --- | R2,R3 |
| S1 | P3 | R4 | R2,R3 |

**(-1 : request & 1 : allocated)**

# Example : Two-Phase

## STEP-1

- Site -2 maintains status table.

### Site-2

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| **P4** | 1 | -1 | 1 | 0 |
| **P5** | 0 | 1 | 0 | -1 |

| Site id | Process id | Hold resource id | Requesting resource id |
|---|---|---|---|
| S2 | P4 | R1,R3 | R2 |
| S2 | P5 | R2 | R4 |

**(-1 : request & 1 : allocated)**

# Example : Two-Phase

## STEP-2

- Controller requests the status table from each site.

Global Process Status Table

|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| **P1** | 0 | 0 | -1 | -1 |
| **P2** | 0 | -1 | -1 | 0 |
| **P3** | 0 | -1 | -1 | 1 |
| **P4** | 1 | -1 | 1 | 0 |
| **P5** | 0 | 1 | 0 | -1 |

**Site-1**

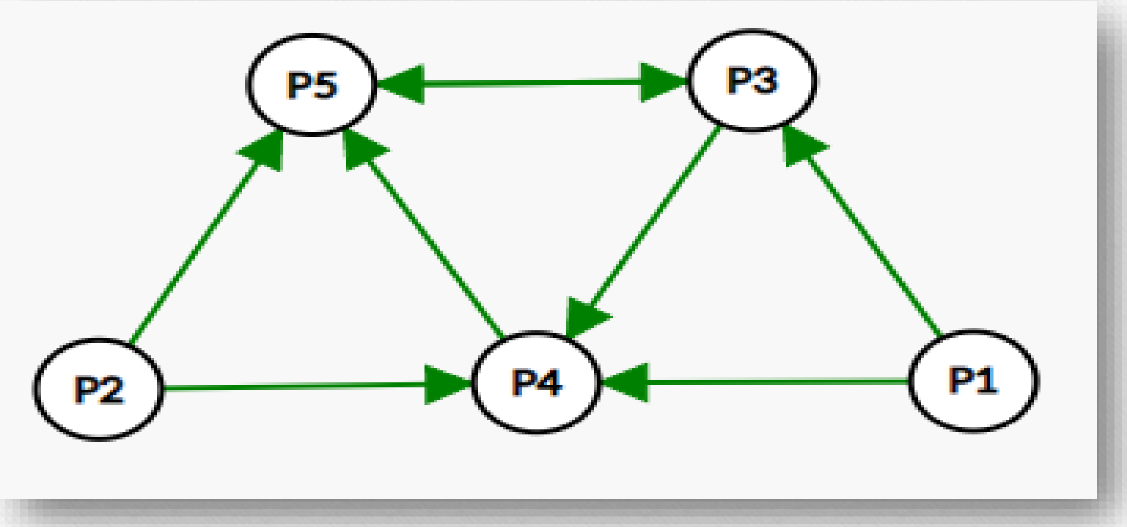
**Site-2**

**(-1 : request & 1 : allocated)**

# Example : Two-Phase

**STEP-3**

- Controller then constructs WFG from these tables, searches for cycle.
- `P1-->P3 i.e P1 is waiting for P3`
- `P1-->P4`
- `P2-->P4`
- `P2-->P5`
- `P3-->P4`
- `P3-->P5`
- `P4-->P5`
- `P5-->P3`



Cycle found between processes : P3 and P5

# Example : Two-Phase

## STEP-4

- Deadlock Detected hence Controller request for state tables again and Construct WFG based *only* on common transactions in the two tables.

**Old Table**

|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| **P1** | 0 | 0 | -1 | -1 |
| **P2** | 0 | -1 | -1 | 0 |
| **P3** | 0 | -1 | -1 | 1 |
| **P4** | 1 | -1 | 1 | 0 |
| **P5** | 0 | 1 | 0 | -1 |

⟷

**New Table**

|  | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| **P1** | 0 | 0 | -1 | -1 |
| **P2** | 0 | -1 | -1 | 0 |
| **P3** | 0 | -1 | -1 | 1 |
| **P4** | 1 | -1 | 1 | 0 |
| **P5** | 0 | 1 | 0 | -1 |

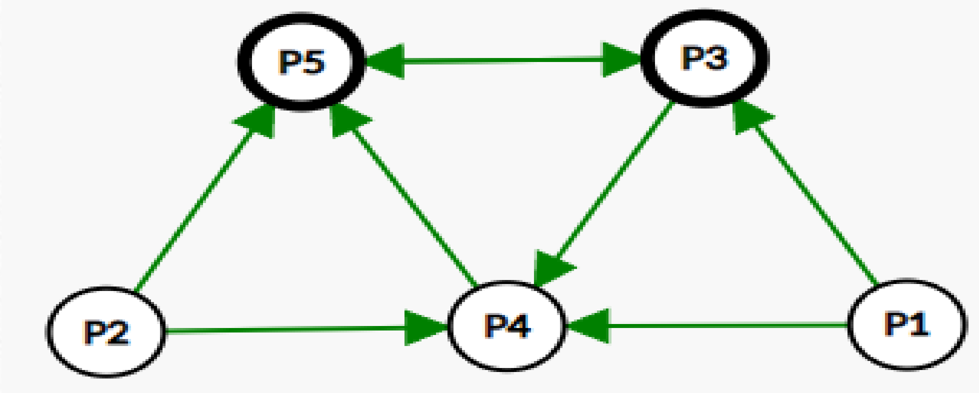
**Site -1**

**Site -2**

# Example : Two-Phase

## STEP-5

- So, same cycle is detected again, system is in deadlock.



Cycle found between processes : P5, P3

**Deadlock Detected**

# Thank You